8

Model Development Process

Part I of this book described a variety of biologically inspired algorithms, and described how these could be used for modelling purposes. Part II discusses the process of actually developing a high-quality financial model, concentrating on the development of a market trading system.

The construction of any financial model is a multi-stage process, consisting of the determination of the goals for the project followed by data collection, data preprocessing, model construction, data postprocessing, model validation, and finally model implementation. It cannot be overemphasised that the degree of success of any model/trading system is critically impacted by the rigour of all the steps in its development process, not just the sophistication of the biologically inspired algorithm(s) embedded in it.

8.1 Project Goals

The object of a trading system is to act as a screening mechanism, to decide which financial assets to buy (or sell), and when to buy (or sell) them. At the heart of any trading system is a predictive model for the market which is being traded. Having selected the market of interest, the modeller must define:

- what the model will forecast; and
- what performance measure is appropriate for the model.

8.1.1 What to Forecast?

Although it may seem obvious that the common goal in financial prediction is to forecast the future price of an asset, this is not necessarily the case, and, in any event, a raw price will not usually be an easy predictive target. Suppose the S&P 500 index is currently 1200, and that it changes by an average of 10 points per day. If the objective is to forecast the next day's

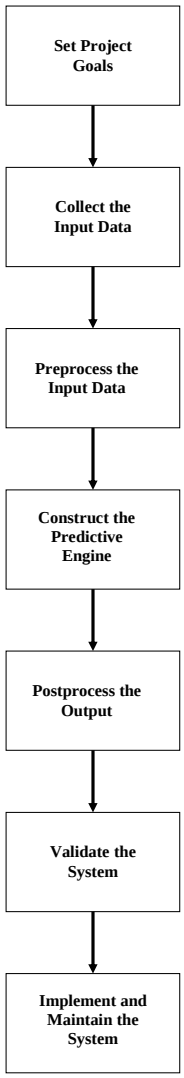

Fig. 8.1. Seven steps in building a trading system

closing value of the index to within 3 points, this corresponds to a required accuracy of approximately 0.25% (3 out of 1200), presenting a difficult task for any predictive model. If the goal of the model is altered to predict the one-day *change* in the index instead of the absolute level of the index, the required accuracy drops to 3 out of 10. Changing the predictive target can substantially impact on the ease of the predictive task. Other predictive targets which can

be used apart from the magnitude of change in price over the next x days include the direction of the change in price (+/-) in the next x days, whether a (for example) 2% move in price will occur in the next x days, or the expected market condition (trending up, down or non-trending) over the next x days. Accurate predictions of any of these would provide a very useful input into a trading system.

8.1.2 What Performance Measure Is Appropriate?

Although many algorithms (including canonical feedforward MLPs) minimise error measures such as mean squared error (MSE) or root mean squared error (RMS), models constructed using these criterion may not perform well when used for trading purposes. *Squared error* based goodness-of-fit criteria will tend to penalise large errors heavily during the model construction process, but this does not guarantee that the final model will be good at correctly identifying and avoiding all large errors. MSE could be low, not because the model makes no large errors, but perhaps because the model is accurate for small changes in price which cannot be profitably traded due to trading costs, while the model misses a substantial number of larger changes which could have been traded profitably if they had been anticipated. Similar problems can arise with the use of an R^2 goodness-of-fit criterion, based on the correlation of the model's prediction and actual market changes.

Metrics such as MSE or R^2 assume that the costs of predictive errors are symmetric. This is not the case in financial prediction. The cost of an error to a trader depends on both its magnitude and direction. If a model generates a buy signal, but underestimates the size of the upward price movement, the trader makes a profit despite the prediction error. A numerical prediction error of similar magnitude but in the other direction (the model predicts that a share's price will increase but in fact it decreases) means the trader makes a loss. This problem can be alleviated by using an *asymmetric error function* to more heavily penalise errors where the direction of the predicted price change is wrong.

The choice of error measure depends on the use to which the model will be put. If the model is to be used for trading purposes, the most appropriate measures of performance are trading returns, scaled by a measure of risk. The trading characteristics of the developed system will depend critically on how returns and risk are defined. The risk associated with a trading system can be measured in a large number of ways, including its *drawdown*, which is the maximum cumulative trading loss of a system during its training or testing period. Hence a performance metric such as the *Stirling ratio*:

$$\frac{\text{Return}}{\text{Drawdown}} \tag{8.1}$$

could be used to evaluate a trading rule. A variant on this is the *modified Stirling ratio* [61]:

$$\frac{\text{Return}}{\text{1+Modified drawdown}} \tag{8.2}$$

where the modified drawdown is defined as being the *max*(drawdown, or 2% of the current position). One advantage of the modified form of the ratio is that it is more robust to minor changes in the value of drawdown, when the absolute size of the drawdown is small. Another common risk measure is the *Sharpe ratio* which compares the level of excess returns (defined as the trading returns less the risk-free returns generated by a trading system over a period of time) with the volatility of those returns:

$$\frac{\text{Trading profit - Risk free return}}{\text{Standard deviation of trading profit}} \tag{8.3}$$

As for the Stirling rations, high values of the Sharpe ratio are preferred. The choice of performance (or fitness) function will determine how often the system trades and what percentage of its trades are winning trades. For example, a trading system could be biased to:

i. maximise the ratio of average trade profit to maximum drawdown,
ii. maximise the Sharpe ratio, or
iii. minimise the volatility of trading returns.

A particular advantage of using a methodology such as GE to construct a trading system is that there are no requirements that the performance measure is differentiable. Hence, the evolution of trading systems can be biased towards whatever risk/return relationship is preferred by the trader. The evolutionary process could be biased to favour trading rules which produce good returns with low drawdown, or a constraint could be placed that only trading rules which produce a drawdown of less than $\$x$ during the training period will be permitted to evolve.

8.2 Data Collection

No matter how sophisticated the biologically inspired algorithm, the old adage *garbage-in-garbage-out* (GIGO) applies. The success of a modelling effort is largely determined by the quality of the data collected, its preprocessing, and the postprocessing of the resulting outputs.

8.2.1 Trading Philosophy

Before detailed consideration can be given to selection of variables for inclusion in a trading system, the *trading intent* and *trading time-horizon* of the system must be defined. The trading intent arises from the strategy the trader intends to adopt in identifying which financial assets to buy and sell, and must be

based on an underlying hypothesis as to how the market behaves (see Chap. 15) for an example of how trading patterns may be influenced by the time horizon adopted by an investor). In essence, the process of constructing a trading system consists of formulating a hypothesis of how the market works, and then testing whether the hypothesis holds up when tested on real data.

The selection of potentially useful explanatory variables is also impacted by the intended trading time-horizon. If a model is being constructed with a view to assisting with long-term equity investment decisions, the relevant variables will be those which help assess the long-term prospects for a firm's shares, and the investor may focus on variables which may indicate that the value of a share has moved out of alignment with its fundamental value. The trading intent is therefore to identify under-(or over) valued shares, and hold the trading position for a period of time. Therefore, minor intra-day movements in share price will not be important.

If attention is placed on short-term trading, short-term price changes will be relevant. Is the intent to hold positions for a few days, or is the intention to build a system for intra-day trading? For either of these time-horizons, the focus will shift from long-term indicators of value, which will be invariant in the short term, to daily flows of demand and supply in the market. Therefore, the relevant explanatory variables may include technical indicators of the underlying forces of supply and demand for the asset and inter-market data concerning the values of related financial assets in other markets.

Whatever the trading intent and horizon, there must be a plausible relationship between the selected inputs and the predictive target. The trader cannot simply throw a group of input variables into a biologically inspired search engine and expect it to automatically uncover something interesting. Throwing 'everything' in and hoping something useful comes out will most likely result in the production of a spurious model.

What Variables?

The range of potentially useful variables will vary depending on the market the trading system is being constructed for. Is the model intended to trade equities, foreign exchange, commodities or financial derivative products? Taking equity markets as an example, three primary sources of information exist:

- technical indicators,
- intermarket indicators, and
- fundamental indicators.

It is not possible to give a complete discussion of each of these sources of information in a single text and only a brief introduction to them is provided. Technical indicators are explanatory variables formed from various combinations of current and historic asset price and transaction volume information. They are widely used in short-term trading systems, and the language of

technical analysis permeates the financial press. Technical indicators are underpinned by the concept of technical analysis. Advocates of technical analysis consider that it can be used to preprocess historic price/volume information to uncover patterns, which when they recur can be recognised and traded on. The next chapter discusses technical analysis, and hence only intermarket and fundamental indicators are considered here.

Intermarket Indicators

Analysis of intermarket indicators attempts to highlight when divergences from long-standing relationships between markets are emerging, possibly suggesting that a particular market is overbought or oversold. There may also be subtle interactions between markets whereby certain markets may lead other markets [40]. Globalisation of companies and capital markets increases the links between the performance of individual equity markets, with many smaller equity markets taking general direction from major markets such as the US. Examples of intermarket indicators which can be relevant in predicting the value of an equity market index include bond prices, stock indices in other countries, and commodity prices such as oil.

When incorporating intermarket data into a trading model, it is important to ensure that future data is not accidentally supplied to the model. Consequently, care must be taken when using data drawn from markets in different time zones. Although the date attribute of discrete pieces of input data might appear to agree, a model may actually be including future information from a later time zone in making its predictions, thus biasing the performance of the developed trading system.

Fundamental Indicators

The choice of relevant fundamental indicators will depend on the financial market which is to be traded. Fundamental analysis can be applied to individual firms in an attempt to assess whether their share prices are currently under or over-valued, or it can be applied at a macro-economic level to assess the likely performance of, for example, the equity market in its entirety.[1]

If the intention is to assess individual firms, useful fundamental indicators will include current and historic information on dividends, profits, sales, assets, debt levels, and liquidity. These factors, along with non-financial information

[1] An interesting parallel can be drawn here with horse racing. Some gamblers analyse fundamental factors such as a horse's past racing history and information concerning jockeys, trainers, ground conditions in order to determine whether the odds offered on a particular horse in a race are mis-priced. Such mispricings can give rise to good-value betting opportunities. Other gamblers concentrate their attention on what they observe in the betting markets (technical analysis). A good discussion of both these groupings of strategies is provided in [69].

about the firm, could be compared with similar information on the firm's competitors to assess the firm's competitive position versus its peers. This information can be combined with data from a sectoral and macro-economic analysis to form an assessment of the likely future growth potential for the firm's profits and dividends.

If fundamental analysis is being performed for the equity market as a whole, the primary indicators will include the broad drivers of supply and demand in the economy. These impact on the earnings and dividend potential of firms, therefore impacting on financial asset prices. Examples of these indicators include:

- commodity prices,
- term structure of interest rates,
- foreign exchange rates,
- GNP,
- inflation,
- rate of unemployment, and
- budget/trade deficits.

These factors have a non-linear and time-lagged impact on each other, on the value of individual shares, and on the equity market as a whole. In the latter case, the impact of a change in a fundamental indicator on a market index will vary over time as the firms comprising the market index change.

Several practical problems can arise when using fundamental indicators drawn from macro-economic data. Fundamental indicators by their nature are time-lagged. Hence, the unemployment rate for March will not be known until figures are compiled in April or May, depending on the speed of collection of the data. In using information drawn from historical databases, the modeller must ensure that data is only presented as input to a model when it was actually available; for example if unemployment data was an input to a model, the input for April would actually be the lagged rate of unemployment in (perhaps) March. If government statistics are being included as fundamental indicators in a model, their definition and measurement should be consistent over the period of interest. This is not always the case. Another problem that can arise when using such data is how to deal with subsequent revisions of the data. This can lead to a problem when using historical databases which consist of clean (post-revision) information. A model constructed using clean data may prove brittle when exposed to poorer-quality real-world data.

EC Approaches to Using Fundamental Indicators in a Trading System

In the earlier discussion of the GA (Chap. 3), an illustration was provided of how screening rules to determine which financial assets should be purchased (or sold) could be evolved from collections of fundamental or other indicators.

An alternative way of constructing a stock-picking rule is to encode 'levels' of a group of indicators on a string. For example, suppose the intent is to encode different combinations of fundamental indicators on a binary string. The first bit could encode whether the debt level of a firm was high or low relative to the industry average, the second bit could indicate whether the firm had experienced above-average industry sales growth over the past three years and so on for other fundamental indicators.

High sales growth relative to industry average?	High debt level relative to industry average?	High level of cash flow from operations relative to industry average?	High level of liquidity relative to industry average?	High profit level relative to industry average?

Fig. 8.2. String encoding of a number of fundamental indicators. Each indicator can be coded as a 0 (no) or 1 (yes)

If we restrict attention to a case where the rule consists of, for example, 25 binary decision variables, the total number of possible rule combinations is 2^{25} or 33,554,432. It is clearly difficult to exhaustively examine all of these, hence an evolutionary algorithm like the GA can be used to determine a good stock screening or trading rule which combines these fundamental indicators. A population of binary strings each representing a specific trading rule could be created randomly, with the GA then selecting which of these represent a good *screening rule* for investment purposes, and then applying crossover and mutation to uncover yet better trading rules. The fitness of each rule could be tested on historical data: if rule x had been applied over the past y time periods, what risk-adjusted performance would it have generated?

8.2.2 How Much Data Is Enough?

There is no simple answer to this question. Generally, the more relevant data the better. Use of small datasets increases the risk of model overfit on the training data, with poor generalisation out-of-sample. The issue of how much data is required is bound up with the number of parameters being estimated in the model. An old saw in statistics is that a modeller should have at least 10 data observations for each included parameter. Hence, if a complex model is being constructed, for example a MLP with many weights, a considerable quantity of data may be required to robustly train the model. The *curse of dimensionality* points out that the amount of data required to construct a model increases exponentially with the number of parameters in the model. To gain intuition on this point, consider what happens if a modeller increases

the number of explanatory variables in a linear regression model for a fixed-size dataset. As the dimensionality of a model increases, the coverage of the data space by the fixed-size dataset is reduced. The data points separate further from each other in the expanded data space.

In financial applications, the quantity of data available will vary depending on whether the model is being constructed with daily data or with intra-day data. If daily data from a stock market is being used, approximately 250 values will be available each year. If intra-day tick-by-tick data is being used the number of data values available for a calendar year will be considerably greater. A problem with using daily or lower-frequency data is that if data is drawn from a long time period, say 15 years, it is questionable as to whether the market has remained unchanged over that period. The underlying data-generating process for market data, unlike that for physical processes, is not stationary. If a modeller believes that only recent data is likely to be useful, perhaps because of significant recent changes in market regulation, but is concerned that this leaves a small dataset, one way to overcome the problem is to create a larger *synthetic* data series from the data available. On the assumption that small changes in the inputs should produce relatively small changes in the value of outputs, new (synthetic) data can be created by taking existing input-output data vectors, making small random changes to the inputs, keeping the same output value as the original data vector, and adding these new data vectors to the original dataset.

The intended lifespan of the model before it is retrained will also have an impact on the quantity and form of input data required. If a model is only to be used for a short period before it is replaced, it will not need to be able to detect long-term market trends, therefore simplifying the data preprocessing requirements. However, the risk of such a model is that it will be fragile with respect to changing market conditions.

Penalising Model Complexity

In considering issues of data sufficiency and the number of explanatory variables to include, the lesson of *Occam's razor*[2] should be borne in mind. If there are two competing explanations for an event, all other things being equal, the simpler one is to be preferred. In order to control model size when building a model, the error criterion can be adapted to incorporate a penalty term based on the model's complexity. This term acts to reduce or penalise the model's performance as the fitted model becomes more complex. A wide variety of metrics from the traditional statistical literature on model selection can be employed to manage the trade-off between model fit and model complexity, including Akaike's Information Criterion (AIC), minimum description length (MDL), and Schwarz's Bayesian Criterion (SBC).

[2]The philosopher William of Occam (approx. 1280-1347) is reputed to have said 'Entia non sunt multiplicanda praeter necessitatem' ('Entities should not be multiplied more than necessary').

8.3 Selecting and Preprocessing the Data

Once a plausible set of explanatory inputs has been selected based on financial theory and intuition, the task facing a modeller is to determine which of these inputs should be incorporated into the final model, and how the included inputs should be preprocessed to extract the maximal useful information content. The steps of input selection and preprocessing are intimately interlinked in practice but are separated below for ease of discussion.

8.3.1 Selection

In selecting the inputs, the first step is to perform a *data audit* to identify missing data, and to filter incorrect data before data analysis starts. Methods for this include graphically examining the data series, checking for logical inconsistencies in the dataset (including cases where the closing price for a period is higher than the high-price for the same period, or where the opening price for a period is less than the low-price for that period, or where a zero price is recorded), and the calculation of simple descriptive statistics for each series (mean, maximum, minimum, standard deviation, and the number of data items). These steps will help identify possible outliers in the dataset, and will also help build the intuition that the modeller has concerning the raw data that is being analysed.

Once the data has been cleaned, traditional statistical techniques can help in deciding which data series are suitable candidates for inclusion in the model. Basic tools such as examining the correlation of potential inputs with each other and with the target output can help rule out inputs with little information content, for example data which is invariant with respect to the predictive target, thereby focusing attention on plausible useful inputs. Linear regression models can be constructed between sets of inputs and the target output to identify useful inputs (do the regressions yield coefficients which are significantly different from zero?). Factor analysis techniques can be applied to reduce the number of inputs supplied to the model, by compressing multiple inputs into a limited number of principal components. A drawback of these techniques is that they will not uncover non-linear relationships between inputs and outputs, and hence they can only provide suggestive rather than definitive guides as to which inputs should be eliminated as having little apparent information content.

It may also be useful to create new, additional data series for possible inclusion in the predictive model. For example if the financial market of interest is known to be seasonal (as are many commodity markets), it is often useful to add a seasonality variable as an input.

8.3.2 Preprocessing

One of the most time-consuming aspects of financial modelling is the preprocessing of model inputs in order to make patterns in the data easier for the

predictive engine to detect. Preprocessing has two main stages, the transformation of inputs and their normalisation.

Transformations

Data transformation can be described as the redefining of data using a predefined rule. Common reasons for transforming data include the compression of multiple raw inputs, in order to produce a single model input, and the removal of some aspect of the data (for example a long-term trend) in order to concentrate attention on another characteristic of the data instead.

Transformations to compress the data allow a reduction in the number of inputs presented to the model. A simple example of such a transformation is to use the *ratio* of two pieces of data (rather than the raw data). For example the ratio of the number of advancing versus the number of declining shares on the stock market can provide a more useful indicator of market sentiment than would either measure on its own. Other common transforms are to use *differences* such as today's value less the value of x periods ago, the percentage change in price between two dates, or moving averages which act to compress time-series information in order to smooth out noise and uncover longer-term trends in a data series.

An example of a transformation which concentrates attention on one aspect of the data rather than another, is the removal of a long-term trend in a time-series. If the intention is to forecast over a short-run horizon, it is often useful to transform the time series of interest by subtracting a y-period moving averages of their values from its current values (Fig. 8.3). If a long-term moving average is subtracted from the current price of a financial asset, it eliminates the longer-term trend and emphasises shorter-term swings in the data.

More complex transformations of raw data in a time-series of price information can be undertaken, such as the calculation of the rate of change of a moving average or the use of first-order log difference of the price changes. The rate of change of a moving average can be obtained from the regression:

$$Y_{t-x} = \alpha + \beta P_t \tag{8.4}$$

where P_t is today's price and Y_{t-x} is the moving average over the last x days. The value of β represents the sensitivity of the moving average to a change in today's price. The first-order log difference of the price changes is given by:

$$O_i = \ln\left(\frac{P_t}{P_{t-x}}\right) \tag{8.5}$$

where P_t is today's price, P_{t-x} is the price x days ago, and O_i is the first-order log difference. Technical indicators provide other examples of input data transformations.

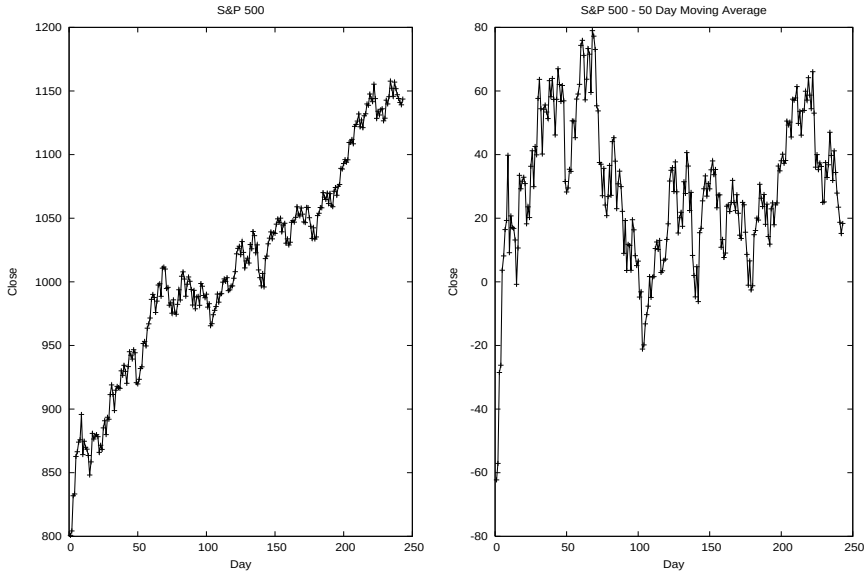

Fig. 8.3. S&P 500 index for 11 March 2003 to 25 February 2004 (left) vs S&P 500 index less a 50-day moving average of the index for the same period (right)

Transformations can also be applied to the predictive target and this step will be required if the range of output from the modelling methodology is constrained. For example if an MLP is being used, the form of transfer function at the output node will determine the numerical range of output which the network can generate. If a logistic or tanh transfer function is used, the outputs are limited to the ranges (0,1) and (-1, +1) respectively. However, neither function is sensitive at the extreme lower or upper limits of its output range, hence it is usually appropriate to rescale target outputs into a narrower range such as (0.1, 0.9) for the logistic function, or (-0.9, +0.9) for the tanh function (Fig. 8.4).

Normalisation

Normalisation is carried out to distribute the raw input data more evenly across its range of variation, and, if necessary, to scale it into an acceptable range for the modelling technique being utilised. An initial step may be to examine the data in order to determine which ranges of it the modeller wishes to focus attention on. If a data series has a small number of outlier (extreme) observations it may be appropriate to clip the values at +/- 2 standard deviations above and below the mean of the series. The object of clipping the data is to focus attention on the range of typical values that the input assumes most of the time, thereby making the model more sensitive to changes in the inputs in their usual range. If, after the outliers have been removed, the remaining

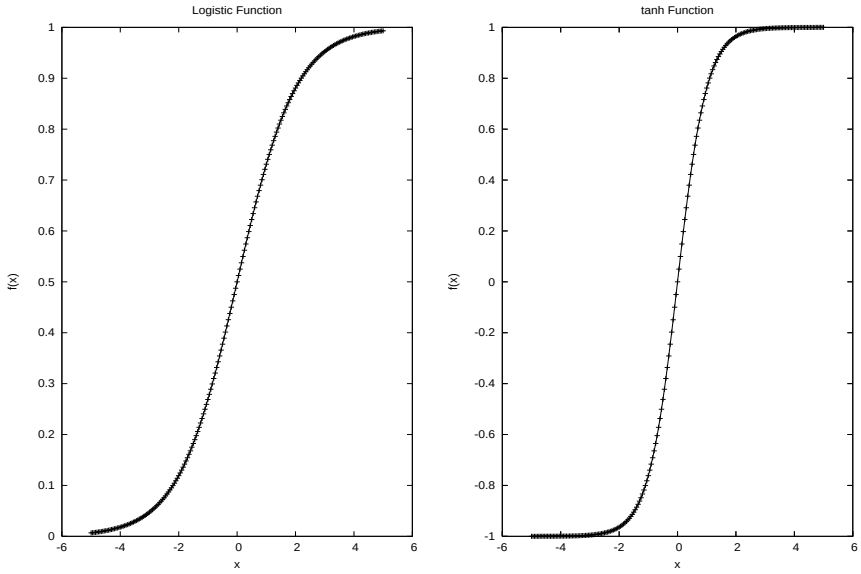

Fig. 8.4. Logistic and tanh functions

data is concentrated across a narrow range of values it is usually useful to rescale it so as to spread out the data to allow the modelling technique to more easily detect differences in the input. The object of this step is to make the distribution of the input data series more uniform. Simple mathematical tricks which can help in this task (depending on the nature of the raw data) include applying the log or exponential function, squaring the data, or taking the square root of the data.

Rescaling each input into a standard range can help ensure that each input has an equal chance to influence the output of the model. The initially selected inputs for the model may have widely differing magnitudes; the current value of the S&P 500 may be 1250, whereas the US$-Euro exchange rate may be 1.2. By rescaling all the inputs into a fixed range, (0,1) or (-1, +1), the model can more easily place appropriate weight on each. A simple method of carrying out this step is to apply linear scaling. For example, to rescale a value at time t into the range (0→1), using data from the last n periods:

$$x_{rescale} = \left(\frac{\text{value}_t \text{ - lowest}_n}{\text{highest}_n\text{-lowest}_n} \right) \tag{8.6}$$

To rescale into the range (-1 +1), the above formula is adapted by subtracting 0.5 and multiplying the result by 2. An alternative method of normalisation is to rescale based on the mean and standard deviation of the data over the last n periods:

$$x_{rescale} = \left(\frac{\text{value}_t \text{ - mean}_n}{\sigma_n} \right) \tag{8.7}$$

The selection of the form of normalisation is influenced by the modelling methodology. Taking the MLP as an example, the most common transfer functions (sigmoid and tanh) are most sensitive to changes in input values around -1 to +1. Hence, rescaling of inputs into this range will tend to increase the sensitivity of the MLP. Whatever the form of transformation or normalisation applied to the input data during the model training process, obviously the same steps must be applied to preprocess live data when the trading system is actually in use.

8.4 Postprocessing the Output

The ultimate output of a trading system is a signal corresponding to the action the investor should take. Should a stock be bought or sold, or should the investor stay out of the market altogether? Therefore it is necessary to postprocess the output of the predictive engine component of the trading system to produce a trading signal. The trading signal generated for a given output from the predictive engine depends on the *entry*, *exit* and *money management* strategies selected by the trader.

8.4.1 Entry Strategy

An entry strategy determines what output is required from the predictive engine before the system generates a buy (or short sell) signal, enters the market and takes a position. An example of a simple entry strategy would be to buy a share once a trading system predicts it will rise by 3% or more in the next ten days. More sophisticated entry strategies could vary the amount invested depending on the strength of the trading signal produced. Entry filters could be built into the trading system to ensure that trades are in round-lots and of a minimum size, to avoid small cost-inefficient purchases being made.

8.4.2 Exit Strategy

The exit strategy determines when trading positions should be closed out. A position may be closed out in order to capture gains on a trade or to protect against excessive losses when a trade goes wrong, or a trade may be exited because the market has turned. An investor can reduce trading risks by using *stop-loss* and *take-profit* triggers. Under a stop-loss trigger, a position is sold out once a loss of $x\%$ or of $\$x$ occurs.

The selection of a suitable stop-loss trigger can be undertaken judgementally[3] or by simulation. In the latter case, the trader could simulate the performance of the trading system on training data with no stop-loss trigger

[3] As an example of how judgement could be applied, Osler [179] provides evidence for the clustering of stop-loss and take-profit orders in foreign-exchange markets. The

included, and produce a graph of maximum drawdown for each trade versus the final profit outcome on that trade. This will allow estimation of the frequency that a drawdown of $\$x$ or more was followed by a final trading profit, and this can provide an input into the selection of a stop-loss trigger for the system. Under a take-profit trigger, a gain is realised once a profit of $y\%$ or $\$y$ occurs.

Many variants on the simple take-profit trigger exist, including a *trailing stop* where a position is exited once a given level of profit has been achieved, and a price fall then occurs (the market reverses). Alternatively, an exit can be triggered when a subsequent sell signal is indicated by the trading system, or after a set period of time if the take-profit trigger has not been hit.

Although the use of stop-losses can protect a trader, they will sometimes fail. Consider the case where a company announces bad news just before the market opens. The effect of this could be to cause the share price to *gap* substantially downward at the market open, causing the price to fall below the stop-loss trigger before it can be activated (Fig. 15.3 for an illustration of an intra-day gap). A stop-loss is a risk-management, not a risk-insurance tool. Other risk-management tools include the use of put and call options.

8.4.3 Money Management

Money management strategies include limiting the amount of money risked on a single trade, a single stock/sector, or indeed on a single trading model. Investors do not have access to infinite funds, so the drawdown characteristics of trading systems is important. Generally, successful trading strategies should produce good returns, a smooth increase in the *equity curve* (the cumulative profits generated by the trading system over a time period), and little clustering of losses (limited drawdowns) [87]. An example of an equity curve is provided in Fig. 8.5.

8.5 Validating the System

Once a prototype trading system has been developed using training data, it must be rigourously validated before going live. Trading models are typically *back-tested*. They are constructed using historical market data and the

affect of clustering of these orders is that trends in exchange rates tend to behave predictably at these cluster levels. If a stop-loss level is hit, the downward trend in price will tend to accelerate as many stop-losses are hit simultaneously. Similarly, if an upward trending price hits a cluster of take-profits, the upward trend will tend to halt as many investors take profits at this point. Testing this hypothesis, Osler found clusters of take-profit orders at exchange rates characterised by 'round numbers' (ending in 00) with stop-loss orders clustering just beyond round numbers. It is suggested that equity-market stops should be set just above round-number prices [119].

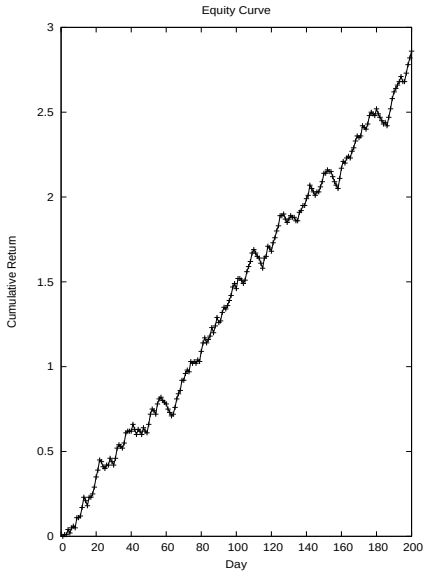

Fig. 8.5. An example of a smooth equity curve from a trading system. Returns are scaled in $000

risk/return characteristics of the trading system are examined to determine whether they are in line with the trader's preferences. Once the model has been constructed, it is further tested using historical data which was not used to train the model. The purpose of the out-of-sample testing is to determine how robust the model is. Do the training period results generalise well to the out-of-sample period? When splitting data into training and out-of-sample datasets, several methods can be adopted. The simplest is a fixed split, for example 60% of available data is used to train the model and the remaining 40% is used to validate it.

An alternative to back-testing which can be used to validate a trading system, particularly one which trades on a short-term horizon, is to perform shadow walk-forward testing. The system is provided with a real-time data feed, and executes simulated trades based on this live information. The performance of the system in shadowing the current market is then assessed.

Limitations of Back-testing

Although back-testing is an essential validation tool, the limitations of the technique must be remembered. Any trading model constructed and tested using historic data will tend to perform less well in a live environment than in a back-test period for a number of reasons. Live markets have attendant problems of delay in executing trades, illiquidity, interrupted/corrupted data and interrupted markets. The impact of these issues is to raise trading costs and

consequently to reduce trading profitability. In addition, markets are competitive and represent a *Red Queen*: as one market participant introduces new computational technologies in an attempt to gain a trading edge, other traders rapidly imitate the technique to erode its profit potential [88]. Hence, estimates of trading performance based on historical data may not be replicated in live trading as other market participants will apply similar technologies. Also, if the particular trading strategy had actually been implemented in the back-testing period, the trading activity and consequently the prices for securities could have been affected, reducing the actual profitability of the system.

Examining the Test Results

Whether the system is tested using historical or current market data, the characteristics and results of its trades must be carefully examined. The goals for a trading system are usually a balance of generating good returns at acceptable risk. Once the system is constructed, its performance and the robustness of its performance in both the training and out-of-sample periods can be evaluated across several metrics. Measures of return which could be used to evaluate the model's performance include the:

- total profit over a specified period,
- number of trades,
- win-ratio (percentage of profitable trades),
- average profit per trade,
- average profit per successful trade,
- average loss per losing trade,
- the profit factor $\left(\frac{\text{total profit on winning trades}}{\text{total loss on losing trades}}\right)$.

Measures of the robustness of the trading system include:

- standard deviation of return per trade,
- Sharpe ratio,
- modified Stirling ratio,
- maximum drawdown, and
- maximum profit (loss) on a single trade.

Although risk-adjusted trading profits will be a prime metric, the modeller will be keen to see how these profits are generated. A visual examination of the equity curve will help reveal when profits and losses are generated by the trading system (under what market conditions do each occur?). For example, Fig. 8.6 suggests a case where the trading system worked well for the first 100 days trading, but its performance deteriorated notably after that suggesting that the system needs retraining.

The modeller will also need to determine whether the profits are generated over many trades or are generated from a very small number of big wins?

What about the losses? Any big losses? A histogram of the distribution of trade profits and losses will highlight any unusual distribution of these items. If large profits or losses occurred, can the modeller determine why they might have occurred? For example, was company-specific news released that day?

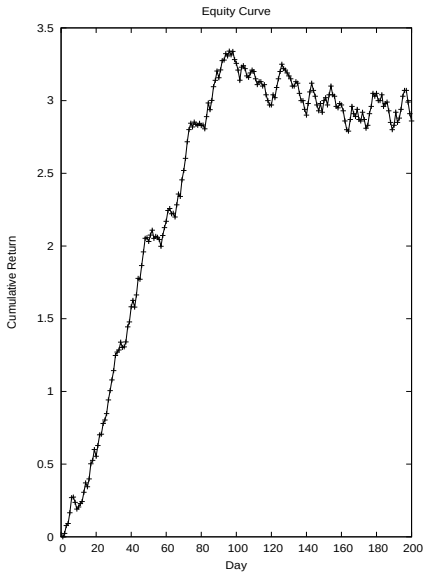

Fig. 8.6. An equity curve which suggests that the model needs retraining. Return axis is scaled in $000

In looking at the profits (losses) from individual trades, some rough statistics can be calculated to help assess whether the trading system produces results which are significantly better than chance alone. For a given sequence of trading results, a t-test can be performed, to determine whether the average profit per trade (assuming there is one!) is significantly better than zero. However, simple statistics should be used with caution. There is likely to be serial dependence in the sequence of trading results as all the data is drawn from a single time period, not randomly, which reduces the effective sample size. Another problem is that if a number of trading systems are created, and the developer is trying to choose the best trading system amongst them, it is likely that one or more trading systems will exhibit results which produce a high t-value through chance alone, and multiple-comparison statistical methods should be used.

Another issue in validating the model will be to test whether the exit strategies for the system could be altered to filter out any major losses which have occurred? Did the 'take-profit' trigger leave substantial profits on the table on several occasions? If so, should the trigger point be raised?

The characteristics of the trading system must be similar on both the training and out-of-sample datasets before they are considered to be robust. Once the system goes live, the above diagnostic metrics should be captured on a regular basis and compared with the results from both the training period and earlier live trading periods, to highlight any degradation in the system's characteristics. Shifts in these metrics can provide an indication that the system should be replaced.

When powerful modelling technologies are applied to a dataset, there is always the danger that they could uncover a spurious pattern in the data which just happens to work well for a time. Apart from using extensive out-of-sample testing to reduce this possibility, the system should pass an *intuition test*. Although not all biologically inspired methodologies, for example MLPs, are amenable to easy deconstruction, rules generated by GA or GE models can be examined. If the rules make no sense to a domain expert, this suggests that they are likely to be spurious. The sensitivity of the model to small alterations in its parameters (for example, its lag periods) should also be checked. If small changes in these parameters result in large changes in the model's performance, the model's results should be viewed with suspicion.

In the above discussion of model validation there has been an assumption that a single (static) trading model has been constructed and then traded for a period of time. An alternative (and common) methodology is to use a *moving window* approach, where a model only predicts one step (or a small number of steps ahead) at a time (Fig. 8.7). The model is continually retrained as new data becomes available. The use of a moving window training implies that the trading model being used changes or adapts over time (see Chap. 14 for an example of this approach).

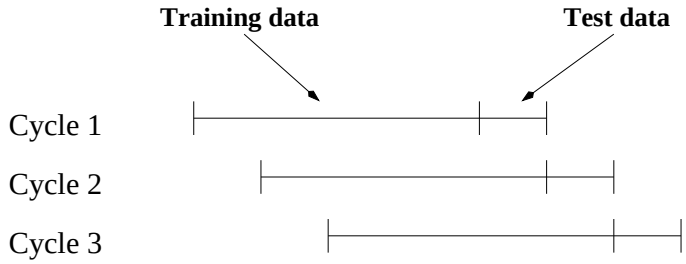

Fig. 8.7. A moving window approach. The system is trained and then tested. After a period of N days (here the length of the test data period), the system is retrained and the training data window is advanced by N days

Iterating the Model Development Steps

The validation and earlier model development steps will be undertaken iteratively. It will usually be possible to improve the performance of the first trading model by careful diagnosis of its errors. Are all the inputs actually influencing the model's decision? Do the results degrade substantially if any of the inputs are dropped? Can the results be improved by altering the predictive target, the input data, or the data pre-/postprocessing steps? A considerable time may be spent iterating between model validation and model development.

8.6 Implementation and Maintenance

Once the model has gone live, a further decision faces the modeller. How long should the model be used before being scrapped? There is no easy answer to this question. Obviously, if a major market event occurs, for example a war, it is quite possible that this will render the assumptions underpinning existing trading systems redundant. The development of new financial products can also produce a change in market structure. For example, significant changes were noted when listed options trading began in the US in 1973, and when stock index futures trading began in 1982. A change in the regulatory environment governing the market of interest can also have implications for the performance of trading systems. Examples of regulatory changes include the introduction of *circuit breakers* on the NYSE, and the move to full decimal pricing on the NYSE on 29 Jan 2001. A short discussion of each of these events is provided to illustrate how rule changes can impact on trading systems.

Circuit Breakers

In response to the market crashes in October 1987 and October 1989, the NYSE instituted several circuit breakers to reduce market volatility. Under Rule 80A, when the Dow Jones Industrial Average (DJIA) moves 180 points or more (the size of this collar is defined each quarter, based on 2% of the average closing of the DJIA for the last month of the previous quarter) from the previous day's close, index arbitrage orders in component stocks of the S&P 500 stock index are subject to a 'tick' test. In up-markets, buy orders may only be executed on a minus or zero-minus tick, in down-markets sell orders may be executed only on a plus or zero-plus tick. Once activated, the rule applies for the rest of the day, unless the DJIA moves back within 90 points of the previous day's close. Rule 80A was activated 51 times on 47 days in 2001. It has been widely credited for helping to reduce market volatility. Under Rule 80B, if the market falls by 10, 20 or 30%, (based on the average closing of the DJIA for the last month of the previous quarter), a market-wide trading halt is activated. The length of the halt varies from 1 hour for a 10% decline to the remainder of the day for a 30% decline. Trading systems

which took no account of market regulations when they were back-tested, or which were developed before significant rule changes occurred, could perform unexpectedly when faced with real-time effects of these rules.

Decimal Pricing

Historically, shares on the NYSE were priced in eighths (of a dollar), and in sixteenths since 1997 [9]. The move to full decimal pricing was completed in 2001. Studies undertaken after this pricing change occurred indicated that it had resulted in a tightening of the bid-ask spread (the difference between the buyer's bidding price and the seller's asking price), from a trade-weighted average of 17c per share in 2000 to approximately 8c in 2001. This change in market structure could clearly impact on the profitability of a trading system constructed before the pricing rule alteration.

Monitoring the System

Even leaving obvious shocks aside, market conditions and the utility of any trading system will change over time. All trading systems embed a simplified representation of the real-data generation process of financial markets, and omit many relevant variables. The significance of the omitted variables will change over time. One rule of thumb is that a short-term trading system should be retrained when 10-15% of the training data can be replaced with new data. Another approach is to construct a monitoring system which looks for degradation in the trading system's performance. A simple monitoring system would track recent trading performance and compare it with the performance in earlier time periods. A change in the trading characteristics of the system, even if the returns from the system are still satisfactory, may provide an early-warning that current market conditions are diverging from those on which the model was trained.

Diversification

A tried-and-tested strategy to deal with market risk is to diversify investment funds across trading systems and across markets. No single trading system can be expected to work well under all market conditions. To overcome this problem multiple systems could be constructed, each of which impounds a different representation of the market. A multi-stage trading system could be constructed, where the first stage (a *gating mechanism*) classifies the 'type' of market that exists at present, and then uses this to select which of a portfolio of trading systems to use. For example, moving average techniques work poorly in non-trending or *choppy* markets. Hence the system could turn off moving average-based components of the trading system if such market conditions are detected. Alternatively, rather than using a hard gating mechanism where a

model is either used in full (100%) or not at all (0%), a soft gating mechanism could be applied which weights the output of each individual model depending on the current market type, and combines their individual outputs to create a 'community' trading decision.

Of course, there is no requirement that a trading system should seek to invest in all conditions. Under the *coherent market hypothesis* [216] markets may be more predictable in certain phases of the business cycle than others, and sometimes it may be better to abandon trading efforts and await more benign conditions.

8.7 Summary

The selection and implementation of a specific biologically inspired algorithm is only one component in the process of developing a complete trading system. No algorithm can compensate for poor-quality data or a poor trading system design. Earlier in the chapter, one of the major sources of information for short-term trading systems (technical indicators) was introduced. The next chapter discusses these in more detail.