



Ernest Chan

The concept of regimes – such as bull and bear markets - is elemental to financial markets. The desire to predict regime switches, commonly known as turning points, is similarly elemental. Ernest Chan, CEO of E. P. Chan & Associates, examines a possible technique for this most demanding of tasks.

Machine Learning + Regime Switching = Profitability?

If attempts to predict the switching from a bull to a bear market were even slightly successful, one could focus on just this one type of switching and call it a day. If only it were that easy! In fact, the difficulty with predicting this type of bull/bear switching has encouraged researchers to examine other types of financial markets regime switching instead, in the hope of finding some that may be more amenable to existing statistical tools.

Some of the other common financial or economic regimes studied are inflationary vs. recessionary regimes, high vs. low volatility regimes, and mean reverting vs. trending regimes. Among these, volatility regime switching seems to be the most popular. Hardly surprising, given the long history of success among financial economists in modelling volatilities, as opposed to the underlying stock prices. Unfortunately, while such predictions of volatility regime switches can be of great value to options traders, they are of no help to stock traders.

Academic attempts to model regime switches in stock prices generally proceed along the following lines:

1. Propose that the two (or more) regimes are characterised by different price probability

distributions. In the simplest cases, the logarithms of the prices of both regimes may be represented by normal distributions, except that they have different means and/or standard deviations.

2. Assume that there is some kind of probability of transition among the regimes.
3. Determine the exact parameters that specify the regime probability distributions and the probability of transition by fitting the model to past prices, using standard statistical methods.
4. Based on the fitted model above, find out the expected regime of the next time step and more importantly, the expected stock price.

Despite the elegant theoretical framework, such regime-switching models are generally quite useless for actual trading purposes. The reason is that they assume a constant probability of transition among regimes at all times. In practice, this means that at any time, there is always a very small probability of the stock transitioning from a normal, quiescent regime to a volatile regime. But this is useless to traders who want to know when — and under what precise conditions — the probability of transition suddenly peaks. This question can be tackled by the use of turning points models.

Turning points models take a data mining approach and enter all possible variables that might predict a turning point or regime switch. Variables such as current volatility, last-period return, changes in macroeconomic numbers such as consumer confidence, oil price changes, bond price changes etc., can all be part of this input. In fact, in a highly topical article about turning points in the real estate market by the noted Yale economist Robert Shiller, it was suggested that the crescendo of media chatter about impending boom or bust might actually be a good predictor of a coming turning point. (See 'All the News that's Fit to Trade' – p. 40, Automated Trader 2008)

Example

The following example illustrates how it might be possible to detect turning points using a data mining approach with just simple technical indicators built on stock price series as inputs, while using stock returns of multiple holding periods as outputs. A prominent brokerage stock - Goldman Sachs (GS) - is used as a proxy for the financial sector. The objective is to detect the turning points of this sector where it goes from bull to bear and back. The initial hypothesis is that major shifts of interest rates, a release of government macroeconomic data, or earnings announcements are likely triggers of turning points. In this example, a large percentage change in GS is used as a proxy for such news releases. Furthermore, it is assumed that whenever GS reaches an N-day high or low just before this large drop or rise in prices occurs this represents a good signal that the previous regime is close to an end. So this condition is used as an additional input. The search problem has a number of different

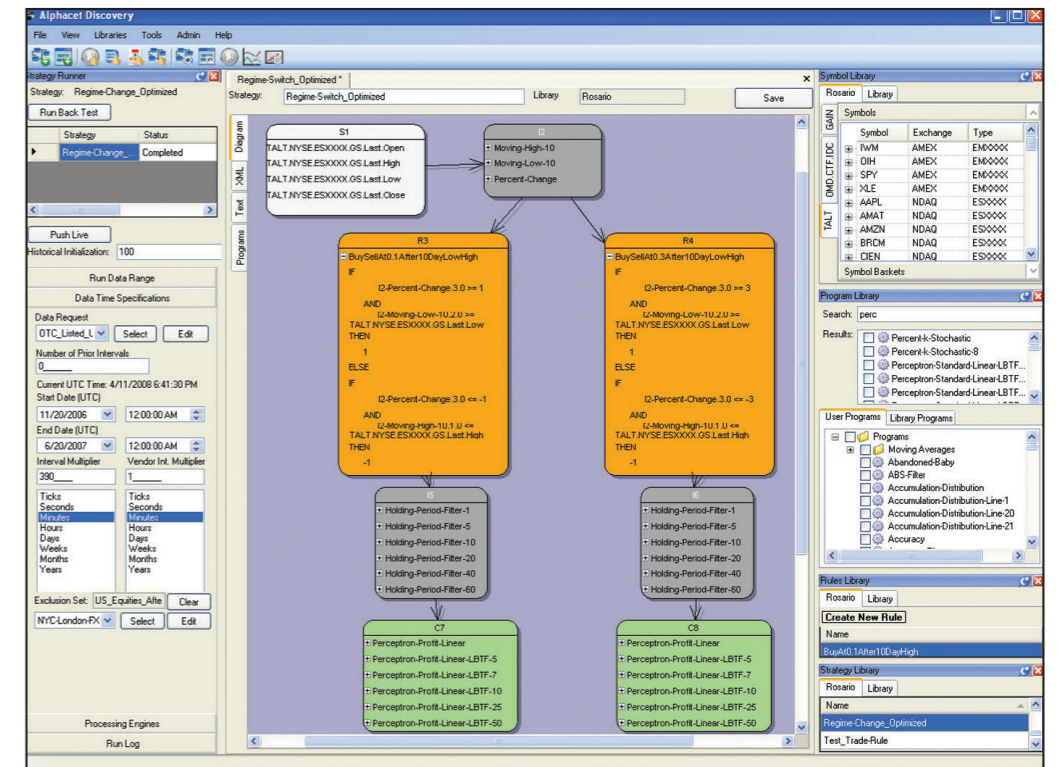


Figure 1

elements. How large a percentage change is sufficient to trigger a regime switch? What should N be in the N-day high/low condition? And how long does the new regime generally last? (In other words, what is the optimal holding period?) To answer these questions in an old-fashioned, manual way would be extremely time consuming, as it would be necessary to run multiple simulations with different thresholds for the independent variables, and multiple return horizons for the dependent variables.

The independent variables of the model are just the one-day returns of GS. The dependent variables are the future returns of GS with various holding periods. The objective is to find an optimal rule, or an optimal combination of rules, which will lead to the best backtest performance. In this case, each percent-change threshold can be encapsulated as a rule. Two thresholds are used for buys and 2 for shorts: -1%, -3%, +1%, +3%. Similarly, each holding-period can be encapsulated as a rule, and in this case six such periods are used: 1, 5, 10, 20, 40 and 60 days.

Separate time series are generated for price, percent-change, and 10-day High/Low time series for the search. (For the sake of simplicity, N is fixed to 10 for the strategy, but this parameter could of course

Machine Learning + Regime Switching = Profitability?

```

R3
BuySellAt0.1After10DayLowHigh
IF
  I2-Percent-Change.3.0 >= 1
  AND
  I2-Moving-Low-10.2.0 >= TALT.NYSE.ESXXXX.GS.Last.Low
THEN
  1
ELSE
  IF
    I2-Percent-Change.3.0 <= -1
    AND
    I2-Moving-High-10.1.0 <= TALT.NYSE.ESXXXX.GS.Last.High
  THEN
    -1
  
```

Figure 2

be optimised as well.) The price series for the test is shown in box S1 of Figure 1. Several pre-packaged rules are then created that compute the 1-day percent-change, and the 10-day moving highs and lows of the time series.

The entry rules are then created; Figure 2 shows rule box R3 for the buy and sell rule based on a change of $\pm 1\%$. A similar rule is created for $\pm 3\%$ in a further R3 box. Note that by default, subsequent entry signals will override positions established by previous signals.

The holding periods are then specified and can be seen in boxes I5 and I6 of Figure 1. the $\pm 2\%$ rule separately). The output of the two R3 boxes is then fed to box I5 the output from box R4 to box I9.

Finally, a perceptron learning algorithm is run on the outputs of I7 and I9. This algorithm will find out the best weights for the

different rules with different holding periods (among other parameters) based on a moving window of historical training data with the objective of maximising the total profit in this window. Based on these optimised weights, the perceptron will trigger a buy and sell decision at the end of each period. Examples of other algorithms that can be selected are a genetic algorithm and a K-nearest-neighbour clustering technique (see sidebar).

Interestingly, the perceptron will not force us to hold a position for exactly N days, even though the component rule was constructed to do this in the moving window. Every day the strategy will decide whether to buy, sell or do nothing, based on the latest parameter-optimisation using the latest data in the moving window and the resulting linearly weighted decisions from the different rules.

Performance results

Figure 3 shows three of the best equity curves that come from the perceptron optimisation. The best curve belongs to a model using a 50-day moving window for the optimisation. (The length of the moving window can of course itself also be optimised,



Figure 3

but in the interests of simplicity that step is omitted here.) The sidebar of the chart shows that this strategy has a 37.93% gross cumulative return over a 6-month backtest period, with 89 round-trip trades. (Compared with the 15.77% return of a buy-and-hold strategy on GS, with a 14% drawdown.) Also shown is the best equity curve from the various holding period routines, which quantifies the improvement from the optimisation (holding period of 10 days in I5 on the 1% rule at R3), which has a gross cumulative return of 18.55% over the period.

Though the backtest period is short, this return nevertheless looks very impressive. So could anything be amiss? In particular, what about the data-snooping or data-mining bias that always seems to creep into every strategy that is based on machine learning or artificial intelligence? One possible solution is to optimise all rules and all parameters in a backward looking moving window, so that absolutely no unseen future data is used for the backtest. Of course, data snooping bias can still creep in because we can abandon a whole category of models when the performance is poor and try one new category after another until it improves. But then, this is unavoidable whenever we are in the business of backtesting.

As can be seen, it is possible to create a credible regime-switching model with the simplest of technical indicators as long as it is possible to optimise efficiently over a large number of parameters using a backward looking moving window. The performance of the model could possibly be further enhanced if the price moves were confirmed with macroeconomic or company-specific news. The same general technique is of course potentially applicable to instruments other than stocks, such as ETF's, futures, or currency pairs.

Perceptron

Essentially a perceptron is a type of neural network in that it operates in a similar manner to the human mind and is capable of learning. A single perceptron receives inputs (data) to which it applies weightings in order to produce an output (a prediction). If that predicted output is then compared with known values for the output, the degree of error can be used to adjust the weightings in order to improve the quality of the next iteration of the prediction.

Genetic algorithm

Genetic algorithms are so called because they tackle problems using a process similar to the controlled evolution of a species. The starting point is a set of random candidate solutions to a problem. To follow the Darwinian analogy, the problem is the environment in which a living species must thrive, while the candidate solutions to the problem represent the species itself. The genetic algorithm software forces the solutions to evolve and reproduce over many generations, with unsuccessful solutions being killed off. Those that represent the better solutions to the problem are then 'mated' in the expectation that the next generation will be even more effective. This process continues until the 'fittest' life form for the environment – or solution to the problem – evolves. In financial markets, genetic algorithms have been used for solving a broad selection of problems, ranging from predicting movements in currency values to optimising equity portfolio composition.

K-nearest-neighbour clustering

Data points near each other in a multi-dimensional space – such as data sets relating to a financial market – share many properties. As a result, the behaviour of these neighbouring records can be used to predict the behaviour of the record under investigation. (In the context of a regime switching model, the point of inflexion where the market moves from a bull to a bear regime, or vice versa.) By calculating the distance of near neighbours, a measure for the strength of the relationship between these and the target variable can be found, which can be used to predict the behaviour of that variable. k refers to the number of cluster points identified.