Learning Trading Rules with Inductive Logic Programming

Liviu Badea

AI Lab, National Institute for Research and Development in Informatics
8-10 Averescu Blvd., Bucharest, Romania
badea@ici.ro

Abstract. We apply Inductive Logic Programming (ILP) for inducing trading rules formed out of combinations of technical indicators from historical market data. To do this, we first identify ideal trading opportunities in the historical data, and then feed these as examples to an ILP learner, which will try to induce a description of them in terms of a given set of indicators. The main contributions of this paper are twofold. Conceptually, we are learning strategies in a chaotic domain in which learning a predictive model is impossible. Technically, we show a way of dealing with *disjunctive* positive examples, which create significant problems for most inductive learners.

1 Introduction and Motivation

Stock market prices are inherently chaotic and unpredictable. They are generated by a large number of time-dependent processes and are therefore non-stationary. Trying to induce a predictive model of the market evolution is thus bound to failure, mostly because any regularity would be immediately exploited and broken.

As long as the evolution of the market cannot be predicted, we cannot directly apply an inductive learning algorithm on the market data and hope to obtain a predictive model. However, although we are incapable of prediction, we may interact with the market (using suitable trading rules) and still make money.

We should therefore concentrate on inducing trading rules rather than predictive models. It is exactly this aspect that makes learning trading rules a challenging domain. Viewed more abstractly, we are dealing with an agent involved in an interaction (a sort of game) with an environment whose evolution cannot be predicted from past records. The challenge consists in devising (inducing) a winning strategy, despite the fact that the environment is unpredictable. For example, a profitable trading strategy is to buy at (local) minima and sell at (local) maxima. The difficulty consists in detecting such extrema by looking only into the past.

But what is the difference between predicting the next value of the price time-series and finding a profitable strategy? For one thing, the ability to predict (the next value) entails a profitable strategy, as follows. Roughly speaking, if we can predict the next value to go up (down), then it is profitable to BUY (respectively SELL). If we do not expect a significant increase or decrease, we should simply wait.

On the other hand, a strategy saying to buy (sell) will probably predict an increase (decrease). The key difference manifests itself when the strategy says to wait, case in which it either predicts a more or less stationary value, or it is *unable to predict*.

Strategies are therefore *partial* predictive models. They can be profitable although at times they may be unable to predict.

R. López de Mántaras, E. Plaza (Eds.): ECML 2000, LNAI 1810, pp. 39-46, 2000.

We mentioned the fact that trading at (local) extrema is a profitable strategy. But how do we detect such extrema only from past data? A large number of so called *technical analysis indicators* [1] are usually employed for this purpose. Roughly speaking, there are two categories of such technical indicators: trend-following indicators, as well as indicators employed in choppy and sideways-moving (non-trending) markets. *Trend-following indicators*, such as moving averages, aim at detecting longer (or shorter) term trends in the price time series, usually at the expense of a longer (respectively shorter) response delay. Therefore they are called "lagging indicators". We can detect local extrema in trending markets by studying the crossings of two moving averages of different averaging lengths. In the case of sideways moving markets, trend-following indicators will usually produce losses. Other indicators, like stochastic oscillators for example, are used instead. If we could discriminate trending markets from non-trending ones, we could apply indicators suited to the specific market conditions. Discriminating between trending and non-trending markets is however difficult. Indicators like the average directional movement index (ADX) are sometimes employed for this purpose.

Using the indicators appropriate for the specific market conditions, possibly as filters, has proved a crucial but extremely difficult task, which has been approached mostly by empirical means. While tuning the numerical parameters of the indicators occurring in trading rules can be done automatically, this is only a first step in the process of adapting a set of indicators to a given market. Finding the most appropriate *combinations* of indicators is in certain ways more interesting, although it is also more complicated due to possible combinatorial explosions.

This paper applies Inductive Logic Programming (ILP) for inducing trading rules formed out of combinations of technical indicators from historical market data. To do this, we first identify buy and sell opportunities in the historical data (by looking not only into the past, but also at future time points).[1] These buy/sell opportunities are then given as examples to an ILP learner which will try to induce a description of these trading opportunities in terms of a given set of technical indicators. For an appropriate set of indicators, the induced trading rules may be profitable in the future as well.

Our main departure from other approaches to the problem of inducing trading rules consists in the fact that whereas other approaches simply test the profitability of syntactically generated strategy variants, we are focusing on recognising the ideal trading opportunities (such as local extrema). Our strategy may prove more reliable, since it may be less influenced by contingent fluctuations in the historical data used for training (because we are explicitly concentrating on the key aspects of a successful strategy, such as recognising extrema). On the other hand, looking just at the profit of a particular candidate strategy on the historical data without analysing its behaviour in more detail may not represent a guarantee for its profitability in the future.

2 Identifying Positive and Negative Examples

Most approaches to learning trading rules guide the syntactical generation of candidate rule sets by global performance criteria, such as profit or risk. Our approach, on the other

[1] If, at a given time point we knew not only the past, but also the future, then we could easily make money by buying at minima and selling at maxima. The future is normally unpredictable and therefore we cannot detect extrema without a delay. But we can easily do this for a given historical time series.

hand is more selective: we first label the historical time series with ideal BUY/SELL opportunities by taking into account both the past *and the future*. These will be subsequently given as examples to an inductive learner. This approach allows us to have a tighter control on the performance of the rules and to avoid selecting rules just because they are profitable on the historical data. Their profitability may depend on contingencies of the historical data and may not guarantee the profitability in the future. Trying to recognise ideal trading opportunities (determined in advance by the initial labelling process) may be more selective, and therefore have better chances to generalise to unseen data. Let us describe the labelling process in more detail.

2.1 Positive Examples

Since we cannot predict *global* extrema just from past data, we shall consider the *local* extrema as ideal trading opportunities. But unfortunately, even these can be too difficult targets for trading strategies based on predetermined sets of technical indicators. Small deviations from the local extrema (in terms of price and time) are not critical from the point of view of profitability and can be tolerated. Therefore we shall consider the points "around" the local extrema as positive examples of trading opportunities. Additionally, we need to avoid declaring the local extrema at the finest time-scales as potential learning targets. We could deal with eliminating such small-scale fluctuations by not distinguishing among the points in a price band of a given width ε. More precisely, we define the ε-band of a given time point t to be the set of time points preceding t that all stay within a band of height ε: $\varepsilon\text{-band}(t) = \{t' \le t \mid \forall t''.\ t' \le t'' \le t \rightarrow |y(t') - y(t'')| < \varepsilon\}$.

Instead of targeting the precise local extremum t_{extr}, we shall target one of the points in its ε-band: $t \in \varepsilon\text{-band}(t_{extr})$. If one such t is covered, then no other time point of the ε-band needs to be covered any more. Thus, instead of a conjunctive set of examples of the form $buy(t_{\min}^{(1)}) \wedge sell(t_{\max}^{(2)}) \wedge buy(t_{\min}^{(3)}) \wedge \ldots$ (where $t^{(i)}$ are the *exact* local extrema), we will end up with disjunctive examples like $(buy(t_1^{(1)}) \vee buy(t_2^{(1)}) \vee \ldots) \wedge (sell(t_1^{(2)}) \vee sell(t_2^{(2)}) \vee \ldots) \wedge \ldots$ where $t_j^{(i)}$ are the time points in the ε-band of the local extremum $t^{(i)}$.

This *disjunctive* nature of the positive examples creates significant problems for most inductive learners. There seems to be no direct way of dealing with such examples in decision tree learning. ILP has also problems when faced with such examples, unless we are learning general clausal theories (i.e. we are not confined to Horn clauses).

The greater flexibility of ILP (as compared to ID3) allows us to represent such examples as $do(buy, [t_1^{(i)}, t_2^{(i)}, \ldots])$ and to look for candidate hypotheses of the form

```
do(buy, TimeList) :- member(T1, TimeList), indicator1(T1),          (1)
                     member(T2, TimeList), indicator2(T2), ...
```

where *TimeList* is an ε-band of the current time point. Such a rule is unfortunately useless by itself when it comes to suggesting the next action at a given time point *T*, because it does not explicitly refer *T*. Fortunately, we can easily compute the ε-band of a given time point *T* (since the ε-band refers only past data) and use it as an argument to the above *do/2* predicate:

```
do_at(Action, T) :- epsilon_band(T, TimeList), do(Action, TimeList).
```

There is another subtle point regarding the precise form of the candidate hypotheses (1). Roughly speaking, they activate a signal if a *combination* of indicators gets activated in the ε-band of the current time point. The indicators need *not* be activated all at *exactly* the same time point *T*, as they are in the following more restrictive rule:

do(buy, TimeList) :- member(T, TimeList), indicator1(T), indicator2(T), ... (2)

For example it may be that a certain combination of two crossover systems most reliably signals a trading opportunity. It is however extremely unlikely that those crossovers happen both *exactly at the same time point T*. For all practical purposes, it is sufficient if they both happen in the same ε-band, even if not at exactly the same time.

The algorithm for computing the ε-bands of the local local extrema (which will be used as positive examples) dynamically maintains an ε-band. When a "break-out above" occurs, then the current ε-band is marked as a minimum band only if we have entered it from above (i.e. from a previous maximum). Otherwise, it will be treated as an intermediate band (see Figure 1). "Break-outs below" are treated in a similar fashion.

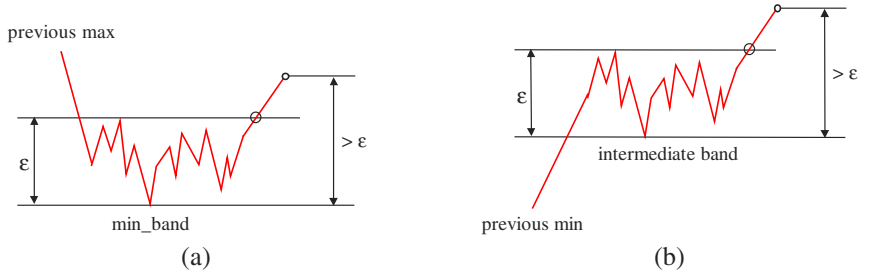

Fig. 1. An ε-band "break-out above" can lead to a minimum (a) or an intermediate band (b) depending on the type of the previous extremum: maximum (a) or minimum (b)

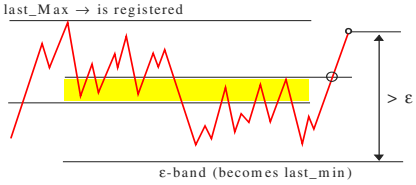

Fig. 2. A break-out above from a minimum-band partially overlapping the previous max-band. The max-band is registered (as a positive example for SELL), but the current min-band is not, because of potential overlaps with the next band (not shown)

Since successive extrema-band price ranges can overlap (leading to potential BUY operations at prices higher than some adjacent SELL), we need to eliminate such overlaps before registering the corresponding extrema bands as positive examples. We also need to delay the registration process of an extremum band *B1* until the next extremum band *B2* is computed, because *B1*'s overlap with *B2* cannot be computed before computing *B2* (see Figure 2). Additionally, we avoid classifying the very first ε-band as either a minimum, or a maximum band.

If the next price y(*t*) stays within the current ε-band (there is no "break-out"), then we simply add it to the current ε-band.

2.2 Negative Examples

Learning from positive examples only [4] does not prevent the learner from obtaining bad candidate rules, which can produce significant losses. We therefore also need negative examples, to avoid obtaining overly general rules that might produce such failures. Adopting all intermediate points[2] as negative examples may be too strict[3]: a point very close to a maximum band should maybe not be considered a negative example for SELL (unless it is also very close to an adjacent minimum-band). It should however be considered a negative example for BUY, since we are too close to the SELL points in the maximum band (see Figure 3). Although such a point will not be necessarily considered a negative example for SELL (thereby tolerating an induced rule that covers it), the point is not a positive example either, so it will not *trigger* the induction of a rule covering it.

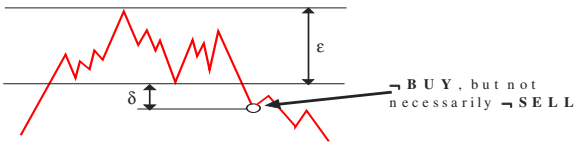

Fig. 3. Points close to a max-band should be negative examples for BUY, but not necessarily for SELL

We generate negative examples according to the following rules

$\neg$ do(buy, [*t*]) if $\min(y[\text{Max\_band}]) - y(t) < \delta$
$\neg$ do(sell, [*t*]) if $y(t) - \min(y[\text{Max\_band}]) < \delta$

for some parameter δ (see also Figure 3). Note that we can have "grey areas" for BUY (SELL), i.e. areas in which we have neither BUY (SELL) nor ¬BUY (respectively ¬SELL).

Besides the above "normal" situation in which the δ-bands are separated, we can have overlapping δ-bands (when the ε-bands are close). In such cases, the intermediate points are negative examples for both BUY and SELL. Of course, we can have not only overlapping δ-bands, but also overlapping ε-bands. As previously discussed in more detail, such overlaps are also considered negative examples for both BUY and SELL.

3 Using ILP for Learning from Disjunctive Examples

The first, 'labelling' phase of our algorithm has set up a learning problem with positive examples of the form

do(buy, [3,4,6,7]). ... (3)
do(sell, [10,11,14]). ...

negative examples like

:- do(sell, [8]). ...
:- do(buy, [15]). ...

[2] i.e. all points that are neither in minimum, nor in maximum bands.

[3] in the same way in which marking the local *exact* extrema as positive examples (targets) for buy/sell actions was too strict. Indeed, it is unlikely not only that an induced rule will cover the local extrema *exactly*, but also that it will succeed avoiding *all* intermediate points.

and a "background theory" containing the historical price time series, as well as predicates for computing various technical indicators and trading strategies. In our experiments we have used the following *technical indicators* on daily closing prices of the USD-DEM exchange rate: moving averages, the Relative Strength Index (RSI), the Average Directional Movement Index (ADX), the stochastic oscillators SlowK and SlowD with an internal averaging period of 3 days, all of these computed for time windows of 5, 10, 15, 25, 40, 65 trading days. These indicators have been used for implementing the following *trading strategies*, which are to be used as "building blocks" in inductive hypotheses:

- a moving average crossover system
- an RSI oscillator system (with buy and sell zones)
- an ADX system triggered by crossing over a given threshold after two successive up-movements
- a stochastic SlowK-SlowD crossover system
- a stochastic oscillator system (with buy and sell zones).

Practically all existing propositional learning algorithms cannot directly deal with disjunctive examples of the form (3). Being a first-order learning technique, Inductive Logic Programming (ILP) can deal with such examples by searching for hypotheses of the form

do(buy, TimeList) :- member(T1, TimeList), indicator1(T1),
member(T2, TimeList), indicator2(T2), ...

As previously mentioned, the various indicators occurring in such a hypothesis need not all refer the same time point *T*. (Since *TimeList* is the ε-band of the current time point, they can be triggered at different time points of the ε-band.)

The following simplified Progol description[4] illustrates our approach to learning trading rules using ILP.

```
% Mode declarations
:- modeh(*, do(buy,+time_list))?          :- modeh(*, do(sell,+time_list))?
:- modeb(*, member(-time,+time_list))?
:- modeb(*, indicator_sell1(+time))?      :- modeb(*, indicator_sell2(+time))?
:- modeb(*, indicator_buy1(+time))?       :- modeb(*, indicator_buy2(+time))?

% Type declarations
time(T) :- number(T).
time_list([]).       time_list([X|L]) :- time(X), time_list(L).

% Background knowledge
member(X,[X|_]).     member(X, [_|L]) :- member(X,L).

indicator_sell1(2).  indicator_sell1(6). indicator_sell1(11). indicator_sell1(16).
indicator_sell2(3).  indicator_sell2(7). indicator_sell2(13). indicator_sell2(17).
indicator_buy1(5).   indicator_buy2(9).  indicator_buy1(15).  indicator_buy2(20).
indicator_sell1(21). indicator_sell2(22).

% Positive examples
do(sell,[1,2,3]).  do(sell,[6,7]).       do(sell,[11,12,13]). do(sell,[16,17]).
do(buy, [4,5]).    do(buy, [8,9,10]).    do(buy, [14,15]).    do(buy, [18,19,20]).

% Negative examples
:- do(_,[21]).     :- do(_,[22]).        :- do(_,[23]).
```

The *modeh* declarations describe the atoms allowed in the head of the induced rules, while *modeb* describe the atoms allowed in the body. It is essential that the recall of the member body atom be unlimited ('*'), because the lengths of the time lists in the positive examples

[4] the simplification amounts to explicitly enumerating the indicators at the various time points instead of computing them using background clauses.

can be arbitrary. (If the recall would be set to 1, then only the indicators triggered at the *first* time point of each time-list (of each positive example) would be included in the most specific clause and the search would be incomplete.)

Running Progol 4.4 [3] on this example produces the following rules:

```
do(sell, TimeList)  :- member(T1, TimeList), indicator_sell1(T1),        (5)
                       member(T2, TimeList), indicator_sell2(T2).
do(buy, TimeList)   :- member(T, TimeList), indicator_buy1(T).
do(buy, TimeList)   :- member(T, TimeList), indicator_buy2(T).
```

Note that we have learned a (single) rule for SELL involving a combination of two indicators, although each of these indicators, taken separately, covers at least a negative example. Thus, since none of these indicators has enough discrimination power to avoid negative examples, the simpler rule

```
do(sell, TimeList) :- member(T, TimeList), indicator_sell1(T).
```

will not work (similarly for *indicator_sell2*). Also note that the rule (5) holds although the two indicators are never triggered simultaneously - they are just triggered in the same ε-band. Insisting that both indicators be triggered simultaneously would produce no results at all.

The main limitations of the ILP approach is the sheer size of the hypotheses space. For each positive example, such as $do(buy, [t_1, t_2, ..., t_k])$, Progol constructs a Most Specific Clause (MSC) which bounds from below the search space of hypotheses. For each time point t_i in the example, the MSC will contain a *member/2* literal, as well as an indicator literal for every indicator that is triggered at t_i. Since hypotheses roughly correspond to subsets of the MSC, the size of the search space will be exponential in the number of MSC literals.

Running Progol on the USD-DEM historical data[5] has produced trading rules like the following:

```
do(buy, TimeList)  :- member(T,TimeList),
                      mov_avg_xover(T,10,15,buy),  stochastic_xover(T,15,buy).
do(buy, TimeList)  :- member(T,TimeList), stochastic_xover(T,5,buy),
                      stochastic_xover(T,10,buy),  stochastic_xover(T,15,buy).
```

The first rule is particularly interesting since it combines a lagging indicator (moving average crossover) with a stochastic indicator in a single strategy. The second is also interesting since it generates a buy signal only when 3 stochastic crossover systems of increasing lengths agree.

On a different run, Progol 4.2.1 also obtained the following *buy* rules:

```
do(buy,TimeList)  :- member(T1, TimeList), member(T2, TimeList),
                     mov_avg_xover(T2,10,15,buy),  stochastic_xover(T1,15,buy).
do(buy,TimeList)  :- member(T1,TimeList), member(T2,TimeList), member(T3,TimeList),
                     mov_avg_xover(T3,5,10,buy), stochastic_xover(T1,5,buy),
                     stochastic_xover(T2,25,buy).
```

The first is very similar to the first rule of the previous run. The difference is that it doesn't require the two indicators (moving average crossover and stochastic crossover) to be

[5] The noise parameter of Progol has been set to 100% since we are dealing with an extremely noisy domain. The number of nodes explored per seed example was limited to 1000.

triggered at exactly the same time point. As argued before, this makes the rule more general and thereby applicable in more situations than the previous one. The second rule activates a buy signal only if a moving average and two stochastic crossover systems all agree - at least in the same ε-band.

The profitability of our ILP generated trading rules is about 80% of the profitability of an over-optimized moving average crossover system[6], which is encouraging for an initial experiment. (The profitability of such over-optimized strategies doesn't usually extrapolate to unseen time series data.) The percentage of actual trades as compared to the total number of ideal trading opportunities was about 25-30%, which is also encouraging considering the chaotic nature of such financial time series.

4 Conclusions

We present an original approach to the problem of inducing symbolic trading rules consisting in first labelling the historical data with ideal trading opportunities, and then using ILP to induce rules based on a given set of technical indicators.[7]

As opposed to other approaches which essentially produce "black-box strategies", ours produces understandable rules, especially since the technical indicators used as "building blocks" have an intuitive reading for the human trader.

Most existing approaches to learning trading strategies are confined to the optimisation of the various numerical parameters of a fixed strategy. Our approach goes beyond these simple approaches by automatically discovering significant combinations of indicators that are capable of recognising ideal trading opportunities. Compared to simple profit-driven rule discovery, our rules have better chances for generalising to unseen data (since the profit of rules evolved on a particular data series can be due to certain accidental features in the data and may not extrapolate to new data). The main utility of our approach is in the case of atypical markets when a trader would like to test his intuition that a certain set of indicators may be helpful, without being able to pinpoint the exact combinations.

References

[1] S.B. Achelis. *Technical Analysis from A to Z*, Irwin Professional Publishing, 2nd ed., 1995.
[2] Dietterich T.G., R.H. Lathrop, T. Lozano-Perez, Solving the multiple-instance problem with axis-parallel rectangles, Artificial Intelligence, 89 (1-2), pp. 31-71, 1997.
[3] S. Muggleton. *Inverse Entailment and Progol*, New Generation Computing Journal, 13:245-286, 1995.
[4] S. Muggleton. *Learning from Positive Data*, Proc. of the 6th Int. workshop on Inductive Logic programming ILP-96, Stockholm, 1996.
[5] Omega Research. *Easy Language User's Manual*, 1996.

[6] obtained using the Omega TradeStation [5].

[7] As far as we know, there are no other approaches to learning trading rules using ILP. The main advantage of ILP w.r.t. other methods in this area is the capability of dealing with *disjunctive* examples. I am grateful to the anonymous reviewers who pointed out the connection with multiple-instance learning [2].