ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5. Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

Application of stochastic recurrent reinforcement learning to index trading

Denise Gorse[1]

1- University College London - Dept of Computer Science
Gower Street, London WC1E 6BT - UK

Abstract. A novel stochastic adaptation of the recurrent reinforcement learning (RRL) methodology is applied to daily, weekly, and monthly stock index data, and compared to results obtained elsewhere using genetic programming (GP). The data sets used have been a considered a challenging test for algorithmic trading. It is demonstrated that RRL can reliably outperform buy-and-hold for the higher frequency data, in contrast to GP which performed best for monthly data.

1 Introduction

In a pioneering work Allen and Karjalainen [1] used genetic programming (GP) to evolve trading rules that were profitable in their own terms but unable to consistently outperform buy-and-hold in the presence of transactions costs, results that were taken as supporting evidence for then widely held academic beliefs about market efficiency. However these results were challenged in later GP work by Becker and Sashadri [2] whose evolved rules based on the same Standard and Poors 500 (S&P 500) data sets did in contrast succeed in outperforming buy-and-hold, though it was not clear to what extent the improved performance was due to a decision to adopt monthly rather than daily trading. Most recently Lohpetch and Corne [3] have revisited this data and in a thorough comparative study demonstrated that it is indeed the use of lower frequency data that allows GP-induced trading rules to gain traction in this market.

These results for monthly data were very encouraging, but do not necessarily mean other learning methods may not also be able to discover exploitable structure in the higher frequency data GP found problematical. Reinforcement learning (RL) is one such alternative, being a form of machine learning that has shown considerable promise in trading and asset allocation. In particular Moody and co-workers have proposed the method of *recurrent reinforcement learning* (RRL) [4,5], a technique that has been used successfully by later workers for stock index [6] and currency [7,8] data, though mixed results in the latter case led Gold [8] to suggest that it might be beneficial to adapt RRL to use forms of learning other than gradient ascent.

The current work follows this suggestion in using a learning procedure based on associative reward-penalty (A_{RP}) learning [9] but with the elaboration of extended bitstreams so that multiple trial-and-error experiments can be carried out at each time step. The method is applied to trading the S&P 500 using the same data as in [1—3]. To facilitate comparison with the GP work it utilises an online learning adaptation of the performance measure first proposed as a fitness function in [1], demonstrating that the RRL methodology can be successfully adapted to use a wider range of performance measures than have been generally explored.

2 The stochastic RRL model

In its original form [4,5] RRL was a gradient-based method. Outputs were derived from a *tanh* unit and thresholded to give trading decisions. In the stochastic version developed here the *tanh* output function is modified to

$$y_t = sig\left(\sum_{i=0}^{m} w_i r_{t-i} + w_{m+1} \mathrm{y}_{\mathrm{t-1}} + w_{\mathrm{m+2}} \right)$$

where $sig(x) = 1/(1+\exp(-x))$, y is an output probability used both to determine trading positions during performance assessment (outputs ≥ 0.5 leading to funds being invested in the risky asset, outputs < 0.5 leading to funds being invested in a competing risk-free asset) and to generate bitwise outputs in {0,1} during the learning process. Following [1—3] we here use as external inputs at each of m+1 previous time steps $r_t = \log(p_t) - \log(p_{t-1})$ (indicating the continuously compounded return, with p_t the price at time t), which together with the feedback weight w_{m+1} and adaptive threshold w_{m+2} gives a total of m+3 parameters overall.

2.1 Learning rule

At each time step t a set of k=1..K binary trading decisions b_t^k (we define $\bar{a} \equiv (1-a)$ for any variable a in [0,1]) are made with probability y_t, at each later time t+1 being assessed and allocated retrospective reinforcement in the form of reward (rwd_{t+1}^k) and penalty (pty_{t+1}^k) signals. The weights are then updated using the A_{RP}-based rule

$$\Delta w_i(t+1) = \frac{\eta}{K} \sum_{k=1}^{K} [(b_t^k - y_t) \times rwd_{t+1}^k + \lambda (\bar{b}_t^k - y_t) \times pty_{t+1}^k] x_i^k(t)$$

where η is a training rate, λ is a parameter controlling the amount of exploration when a penalty is received, and the inputs are given by

$$x_i^k(t) = \begin{cases} r_{t-i} & i = 0..\, m \\ b_{t-1}^k & i = m+1 \\ 1 & i = m+2 \end{cases}$$

2.2 Allocating reinforcement

The GP fitness function used in [1—3] is

$$R = \sum_{t=1}^{T} I_b(t) r_t + \sum_{t=1}^{T} I_s(t) \log(1+\rho_t) + n \log\left(\frac{1-\delta}{1+\delta} \right)$$

in which the binary variables $I_b(t)$, $I_s(t)$ represent the trading position at time t (in or out of the market respectively), ρ_t is the interest earned over a time interval [t-1,t) from investment in a risk-free asset, δ is a transactions cost, and n is the number of completed trades over T time intervals. This performance measure can be used as a reward/penalty signal generator by re-expressing it as a sum of terms R_t, for t=1..T, where

$$R_t = y_{t-1} r_t + \bar{y}_{t-1} \log(1+\rho_t) + y_{t-1} \bar{y}_t \log(1-\delta) - \bar{y}_{t-1} y_t \log(1+\delta)$$

Since y_t influences returns both at times t and t+1 it can be seen that

$$\frac{dR}{dy_t} = \frac{dR_{t+1}}{dy_t} + \frac{dy_{t+1}}{dy_t} \cdot \frac{dR_{t+1}}{dy_{t+1}} + \frac{dR_t}{dy_t}$$

Replacing the derivative dy_{t+1}/dy_t by the cross-correlation $(2b_{t+1}^k - 1)(2b_t^k - 1)$ to facilitate bitwise computation, the above gradient can be approximated by

$$dRdb_t^k = r_{t+1} - \log(1+\rho_t) + \log(1-\delta)[\bar{b}_{t+1}^k - (2b_{t+1}^k - 1)b_t^k - b_{t-1}^k]$$
$$+ \log(1+\delta)[b_{t+1}^k + (2b_{t+1}^k - 1)\bar{b}_t^k - \bar{b}_{t-1}^k]$$

and used to generate reinforcement signals

$$rwd_{t+1}^k = \begin{cases} 1 & if\ dRdb_t^k \times (2b_t^k - 1) > 0 \\ 0 & otherwise \end{cases} \quad , \quad pty_{t+1}^k = 1 - rwd_{t+1}^k$$

at time t+1 for trial actions b_t^k taken at the previous time.

3 Data set

The data used here are as in Lohpetch and Corne [3] the opening prices of the S&P 500 taken over a range of timescales (monthly, weekly, daily) from the years 1960 to 2008, with corresponding risk-free returns derived from three-month US Treasury Bill rates. Data are as in [3] additionally divided into the subsets set out in Table 1:

Data split	**Training period**	**Test period 1**	**Test period 2**
MonthlySplit1	31 years from 1960	next 12 years	next 5 years
MonthlySplit2	31 years from 1960	next 8 years	next 8 years
MonthlySplit3	31 years from 1960	next 9 years	next 9 years
MonthlySplit4	25 years from 1960	next 12 years	next 12 years
WeeklySplit1	366 wks from 1/01/60	next 158 wks	next 157 wks
WeeklySplit2	366 wks from 1/01/72	next 158 wks	next 158 wks
WeeklySplit3	367 wks from 1/01/84	next 157 wks	next 158 wks
WeeklySplit4	366 wks from 1/01/96	next 157 wks	next 158 wks
DailySplit1	378 days from 1/01/60	next 126 days	next 127 days
DailySplit2	380 days from 1/01/75	next 127 days	next 127 days
DailySplit3	379 days from 1/01/90	next 128 days	next 127 days
DailySplit4	376 days from 1/01/06	next 128 days	next 126 days

Table 1: Monthly, weekly, and daily data splits.

Two training/testing regimes were considered in [3]: in Regime 1 (no validation) the test period was that immediately following the training period (period 1), while in Regime 2 the first test period was used for validation and the second period for out of sample testing. Both regimes are also considered here.

4 Results

Results are tabulated below for each type of data split, showing the comparative performance of RRL and the GP-induced trading rules of [3] in relation to buy-and-hold over the relevant test periods. GP results are as quoted in [3] for a Performance Consistency parameter equal to 12. RRL results are for training parameters η=0.05, λ=0.01, a bitstream length K=8, and input window size m=20. The system was not found to be overly sensitive to the values chosen for η and λ, while the effects of changes to m and K are explored below in Figures 1 and 2 respectively. As described above, in Regime 1 the net is trained until performance on the training set exceeds buy-and-hold while in Regime 2 the same test is applied to the validation set.

Data split	**Trials outperforming buy-and-hold for regimes 1 (2)**	
	RRL-A$_{RP}$ (100 trials)	**GP (Lohpetch & Corne [3])**
MonthlySplit 1	99 (0) %	10 (10) out of 10
MonthlySplit 2	7 (94) %	4 (8) out of 10
MonthlySplit 3	4 (97) %	10 (8) out of 10
MonthlySplit 4	0 (72) %	9 (10) out of 10
Monthly average	**27.50 (65.75) %**	**82.5 (90.0) %**
WeeklySplit 1	53 (16) %	6 (2) out of 10
WeeklySplit 2	100 (0) %	10 (10) out of 10
WeeklySplit 3	8 (98) %	4 (4) out of 10
WeeklySplit 4	98 (100) %	10 (10) out of 10
Weekly average	**64.75 (53.50) %**	**75.0 (65.0) %**
DailySplit 1	100 (94) %	0 (0) out of 10
DailySplit 2	100 (100) %	0 (0) out of 10
DailySplit 3	100 (100) %	10 (10) out of 10
DailySplit 4	100 (100) %	2 (2) out of 10
Daily average	**100 (98.50) %**	**30.0 (30.0) %**

Table 2: Summary of comparative results for monthly, weekly, and daily trading, with bracketed figures referring to results found for training/testing regime 2.

It can be seen that in contrast to GP, RRL finds the daily data more tractable and the monthly data less so. Both methods agree in finding the weekly data to be of intermediate difficulty. With respect to the difference between Regimes 1 and 2, there again appears to be agreement between the methods in that results are better in Regime 2 for monthly data but worse for weekly data, with the quality of the daily results about the same. It is surprising that the use of a validation set appears to degrade performance in the case of weekly data. However though fewer Regime 2 trials exceed buy-and-hold profit Figure 1 shows that the average excess profit nevertheless exceeds that for Regime 1 over a range of input window sizes.

It is also clear from this figure that profits can be affected by window size and that the optimal value for this parameter may depend on the data set. Preferred values

ESANN 2011 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2011, i6doc.com publ., ISBN 978-2-87419-044-5. Available from http://www.i6doc.com/en/livre/?GCOI=28001100817300.

appear quite large, with as may be expected less evidence of an overtraining effect in Regime 2. Gradient-based RRL has typically used smaller windows for both stock index and currency data; however it should be noted that not only the learning method but also the performance measure used to provide reinforcement are different in the present case.

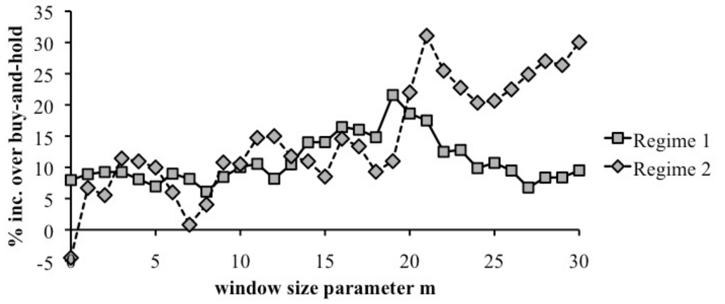

Fig. 1: Weekly data: split- and trial-averaged percentage profit in excess of buy-and-hold as a function of RRL window size parameter *m*.

A further parameter that might be expected to affect performance is *K*, the number of sampling bits in the weight update rule at each time step. Figure 2 shows how performance depends on *K* for an RRL net with window size *m*=20. While overly small values do not give optimal performance there appears to be little benefit in values larger than *K*=8. Provided excessively small values are not used, unlike the input window size the bitstream length does not appear to be a critical parameter

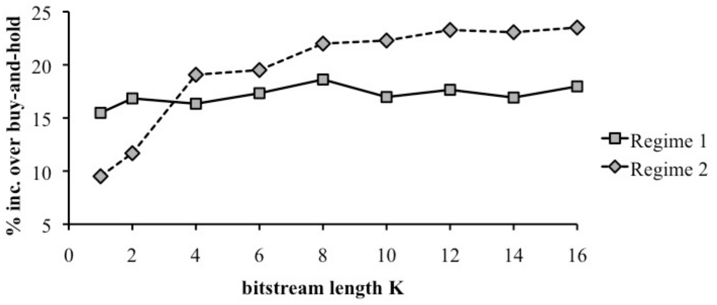

Fig. 2: Weekly data: split- and trial-averaged percentage profit in excess of buy-and-hold as a function of bitstream length *K*, for window size *m*=20.

5 Discussion

The current work has supported that of [2,3] in demonstrating that a trading model can be developed that is able to reliably outperform buy-and-hold on a data set considered challenging in this respect. Results here however differ from the GP-based work of [2,3] in that for RRL it is the higher frequency daily data that is the most tractable. These contrasting results may give insight into the forces that drive markets over different time scales. The rules induced by Lohpetch and Corne [3] are quite complex and utilise as terminal nodes quantities such as moving averages and moving average maxima. However it has been noted in [7] that the inclusion of such derived quantities as additional inputs is not helpful to RRL, for which it appears all relevant information has already been captured by the raw data. The most successful rules for daily trading may be the simplest ones, possibly reflecting both the psychology and preferred tools of human traders operating at these time scales.

As noted in the Results section performance here depends on the size of the past-returns input window. Dependence on a parameter that could easily be over-optimised is always a potential problem. In this context Dempster and Leemans [7] have advocated online adaptation of various model hyperparameters, and this approach could certainly be applied to input window size in the present case.

The use of multilayer networks in RRL was explored by Gold [8] but did not improve performance (this was also found to be the case here). It seems unlikely however that the optimal trading model for the majority of data sets will be a linear one. Maringer and Ramtohul [6] have recently shown that an RRL system that switches between its two specialist units in response to data volatility performs much better than a single-unit system, suggesting that a more effective way to introduce nonlinearity may be via an ensemble of separately trained linear models.

Acknowledgement

The author would like to thank David Corne and Dome Lohpetch for the use of the data investigated herein, and for insightful and helpful discussions.

References

[1] F. Allen and R. Karjalainen, Using genetic algorithms to find technical trading rules, *Journal of Financial Economics*, 51:245-271, Elsevier, 1999.

[2] L. A. Becker and M. Sashadri, Comprehensibility and overfitting avoidance in genetic programming for technical trading rules. Technical Report WPI-CS-TR-03-09, Computer Science Department, Worcester Polytechnic Institute, Worcester, MA 01609, USA, September 2003.

[3] D. Lohpetch and D. Corne, Outperforming buy-and-hold with evolved technical trading rules: daily, weekly and monthly trading, submitted to *EvoApplications 2011*, 10 pages, Springer LNCS, 2011.

[4] J. Moody, L. Wu, Y. Liao and M. Saffell, Performance functions and reinforcement learning for trading systems and portfolios, *Journal of Forecasting*, 17:441-470, Wiley, 1998.

[5] J. Moody and M. Saffell, Learning to trade via direct reinforcement, *IEEE Transactions on Neural Networks*, 12:876-889, IEEE Press, 2001.

[6] D. Maringer and T. Ramtohul, Threshold recurrent reinforcement learning for automated trading. In C. Di Chio et al., editors, *EvoApplications 2010*, Lecture Notes in Computer Science 6025, pages 212-221, Springer-Verlag, 2010.

[7] M. Dempster and V. Leemans, An automated FX trading system using adaptive reinforcement learning, *Expert systems with applications*, 30:543-552, Elsevier, 2006.

[8] C. Gold, FX trading via recurrent reinforcement learning. In *Proceedings of the IEEE International Conference on Financial Engineering*, IEEE Press, pages 363-370, March 20-23, Hong Kong (People's Republic of China), 2003.

[9] A. G. Barto and P. Anandan, Pattern recognising stochastic learning automata, *IEEE Transactions on Systems, Man, and Cybernetics,* 15:360-375, IEEE Press, 1983.