Expert Systems with Applications 30 (2006) 543–552

Expert Systems with Applications

www.elsevier.com/locate/eswa

An automated FX trading system using adaptive reinforcement learning

M.A.H. Dempster [*], V. Leemans

Centre for Financial Research, Judge Business School, University of Cambridge & Cambridge Systems Associates Limited, Cambridge, UK

Abstract

This paper introduces *adaptive reinforcement learning* (ARL) as the basis for a fully automated trading system application. The system is designed to trade *foreign exchange* (FX) markets and relies on a layered structure consisting of a machine learning algorithm, a risk management overlay and a dynamic utility optimization layer. An existing machine-learning method called *recurrent reinforcement learning* (RRL) was chosen as the underlying algorithm for ARL. One of the strengths of our approach is that the dynamic optimization layer makes a fixed choice of model tuning parameters unnecessary. It also allows for a risk-return trade-off to be made by the user within the system. The trading system is able to make consistent gains out-of-sample while avoiding large draw-downs.

1. Introduction to trading systems

An active subject of research among practitioners as well as academics over the past decade has been the construction of systems that autonomously trade in securities or currencies. A summary of earlier currency work can be found in (Austin, Bates, Dempster, Leemans, & Williams, 2004). In the quest for a trading algorithm, artificial intelligence methods have been employed to construct systems that are better than or at least as good as their human counterparts in timing trade entry and exit opportunities. Often such a system takes as inputs past market prices (or some transformation thereof) and attempts to return a trading signal indicating the preferred position to hold in a given currency relative to another (i.e. long or short). The majority of systems described in the literature aim to maximize trading profits or some risk-adjusted measure such as the Sharpe ratio.

Many attempts have been made to come up with a consistently profitable system and inspiration has come from different fields ranging from fundamental analysis, econometric modelling of financial markets, to machine-learning (Dempster & Romahi, 2002; Moody & Saffell, 1999). Few attempts were successful and those that seemed most promising often could not be used to trade actual markets because of associated practical disadvantages. Among others these included large draw-downs in profits and excessive switching behaviour resulting in very high transaction costs. Professional traders have generally regarded those automated systems as far too risky in comparison to the returns they were themselves able to deliver. Even if a trading model was shown to produce an acceptable risk-return profile on historical data there was no guarantee that the system would keep on working in the future. It would cease working precisely at the moment it became unable to adapt to changing market conditions.

This paper aims to deal with the above problems to obtain a usable, fully automated and intelligent currency trading system. To accomplish this a risk management layer and a dynamic optimization layer are added to a known machine-learning algorithm. The middle layer manages risk in an intelligent manner so that it protects past gains and avoids losses by restraining or even shutting down trading activity in times of high uncertainty. The top layer dynamically optimizes the global trading performance in terms of a trader's risk preferences by automatically tuning the system's hyper-parameters. While the machine-learning system is designed to learn from its past trading experiences, the optimization overlay is an attempt to adapt the evolutionary behaviour of the system and its perception of risk to the evolution of the market itself. In the past an automated trading system based on 2 superimposed artificial intelligence algorithms was proposed (Dempster & Romahi, 2002). This research departs from a similar principle by developing a fully layered system where risk management, automatic parameter tuning and dynamic utility optimization are combined. (For an earlier attempt in this direction see Venturi, 2003). The machine-learning algorithm combined with the dynamic optimization is termed *adaptive reinforcement learning*.

[*] Corresponding author. Address: Centre for Financial Research, Judge Business School, University of Cambridge, Cambridge, UK

E-mail addresses: mahd2@cam.ac.uk (M.A.H. Dempster), vl227@cam.ac.uk (V. Leemans).

URL: www-cfr.jims.cam.ac.uk.

0957-4174/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2005.10.012

Section 2 of this paper briefly discusses the RRL machine-learning algorithm underlying the trading system. Section 3 looks at the different layers of the trading system individually. In Section 3.1 the modifications to the standard algorithm are set out and in Sections 3.2 and 3.3 the risk management and optimization layers are explained. The impressive performance of the trading system is demonstrated in Section 4 and Section 5 concludes.

2. The machine-learning algorithm

The basic model from which we depart was originally applied with some success to FX trading by Moody and Saffell (Moody & Saffell, 1999). This section briefly reviews the mathematical model specification and estimation procedure as explained in (Moody & Saffell, 2001). The trading model takes as inputs standardized past time series changes and outputs the preferred position (long or short) in one of the two currencies involved. Further, the model is specified as a recurrent single layer neural network of the price of one currency in terms of another, i.e.

$$F_t = \operatorname{sign}\left(\sum_{i=0}^{M} w_{i,t} r_{t-i} + w_{M+1,t} F_{t-1} + v_t\right) \tag{1}$$

where $F_t \in \{-1,1\}$ is the position to take at time t, w_t and v_t are, respectively, the weight vector and threshold of the neural network at time t and $r_t := p_t - p_{t-1}$ are the returns in terms of price changes of the FX time series. For simplicity of exposition we assume here that the discrete time index t refers to a basic data frequency. We also assume in this section only that the trading system always takes one position or another. For a more detailed discussion of the actual continuous time situation see (Austin et al., 2004; Bates, Dempster, & Romahi, 2003; Dempster & Jones, 2001; Dempster, Payne, Romahi, & Thompson, 2001; Dempster & Romahi, 2002).

At every trade exactly 1 unit of a certain currency is bought or sold and hence the profit at time T can be calculated (ignoring interest rates) as:

$$P_T = \sum_{t=0}^{T} R_t \tag{2}$$

$$R_t := F_{t-1} r_t - \delta |F_t - F_{t-1}|, \tag{3}$$

where δ is the transaction cost per trade.

For computational efficiency (Moody & Saffell, 1999; Moody & Saffell, 2001), choose the differential Sharpe ratio as the risk-adjusted return measure to be optimized. It is derived by considering a moving average version of the classical Sharpe ratio given by:

$$\hat{S}(t) := \frac{A_t}{B_t} \tag{4}$$

$$A_t = A_{t-1} + \eta(R_t - A_{t-1}) \tag{5}$$

$$B_t = B_{t-1} + \eta(R_t^2 - B_{t-1}). \tag{6}$$

When $\hat{S}(t)$ is expanded as a Taylor series in the adaptation parameter η, the first derivative term can be regarded as an instantaneous performance measure

$$D_t := \frac{d\hat{S}(t)}{d\eta}\Big|_{\eta=0} \tag{7}$$

$$= \frac{B_{t-1}\Delta A_t - \frac{1}{2} A_{t-1} \Delta B_t}{(B_{t-1} - A_{t-1}^2)^{\frac{3}{2}}} \tag{8}$$

Because of its simplicity and tractability, a simple gradient ascent is used as optimization algorithm to gradually improve the model parameters by means of the recursion:

$$w_{i,t} = w_{i,t-1} + \rho \Delta w_{i,t}. \tag{9}$$

Δw can then be approximated for an on-line update by considering only the term that depends on the most recent trading return R_t, viz

$$\Delta w_{i,t} = \frac{\mathrm{d}D_t}{\mathrm{d}w_i} \tag{10}$$

$$\approx \frac{\mathrm{d}D_t}{\mathrm{d}R_t}\left\{\frac{\mathrm{d}R_t}{\mathrm{d}F_t}\frac{\mathrm{d}F_t}{\mathrm{d}w_{i,t}} + \frac{\mathrm{d}R_t}{\mathrm{d}F_{t-1}}\frac{\mathrm{d}F_{t-1}}{\mathrm{d}w_{i,t-1}}\right\}. \tag{11}$$

Since, the neural network is recurrent, an updating scheme similar to back-propagation through time can be applied with:

$$\frac{\mathrm{d}F_t}{\mathrm{d}w_{i,t}} \approx \frac{\partial F_t}{\partial w_{i,t}} + \frac{\partial F_t}{\partial F_{t-1}}\frac{\mathrm{d}F_{t-1}}{\mathrm{d}w_{i,t-1}}. \tag{12}$$

All terms and factors in (11) and (12) can be calculated from the previous expressions. Together, they form the set of equations necessary for updating the model weights. This machine-learning technique has been termed *recurrent reinforcement learning* (RRL) (Moody & Saffell, 2001).

As demonstrated by Gold (Gold, 2003), RRL lends itself perfectly to a rolling-window approach (Dempster & Jones, 2001). Here, the model is trained for a number of epochs n_e on a training set of length L_{train} and then applied to a test set of length L_{test}. Next, the training window is advanced by L_{test} time points and the whole procedure is repeated. The out-of-sample performance of the model is the sum of the performances on the individual non-overlapping test sets. We found that the optimal values of L_{train} and L_{test} for our FX (midpoint quote) datasets were around 2000 and 500 ticks, respectively, which are the values chosen in (Gold, 2003). The number of recursive training epochs n_e was set to 10. The optimal values of these three parameters did not change significantly when different FX data sets were used.

3. The trading system

The trading system discussed in this paper consists of three layers: the basic trading system, the risk and performance management layer, and the parameter optimization layer as shown in Fig. 1. Each of these layers is discussed in more detail in the sequel, starting with the bottom-most layer.

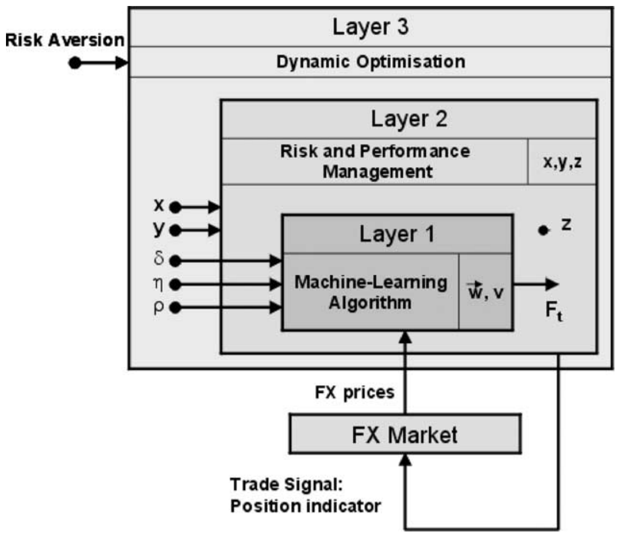


Fig. 1. Schematic illustration of the automated trading system consisting of three main layers: the machine-learning algorithm, the risk and performance management layer and the dynamic optimization layer. The combination of the first and third layer is termed adaptive reinforcement learning (ARL). The trader's risk aversion ν is an exogenous parameter to the system. Layer 3 optimizes the trailing stop-loss level x, the trading threshold y, the trading cost δ, the adaptation parameter η and the learning rate ρ. The auto-shut-down critical loss parameter z was fixed.

3.1. Layer 1: extensions to the machine-learning algorithm

The core of the trading system discussed here is the RRL algorithm (Moody & Saffell, 1999) discussed in the previous section. A number of changes to this basic algorithm were necessary however to improve its performance and to make it suitable for a structural risk management approach.

RRL was extended to take account of other inputs than past returns r_{t-i} and the current position F_t. Experiments were conducted to include signals originating from 14 popular technical indicators as first suggested in (Dempster et al., 2001). Performance did not increase by adding indicators to current data, except when a low number of past returns M was fed into the system. The filtering of the price data timeseries offered by the technical indicators thus did not contribute to increased performance on current data. This indicates that the RRL algorithm is able to efficiently exploit the information in past returns timeseries.

During the training phase, the transaction cost factor δ was not fixed to the actual bid-ask spread as in (Gold, 2003), but was instead left as a tuning parameter. Setting a higher cost factor necessitates a higher expected raw profit before engaging in a trade. Consequently the parameter δ will influence the performance and risk profile of the resulting trading strategy.

Large jumps in FX returns, as, e.g. when central banks intervene, can introduce instability into the underlying RRL algorithm. To prevent the weights from taking unreasonably large values, all weights were rescaled by a factor $f<1$ as soon as one of the weights hits a certain threshold value. This also prevents the weights from drifting slowly upwards without bound. This phenomenon is undesirable since the user may get incorrect model evaluations at some point because of the numerical instabilities associated with very large weights.

We also improved the performance of the RRL trading model by implementing a better position-updating scheme. The traditional update scheme sometimes causes indecisive switching between different positions from one tick to the next. This can generate huge transaction costs and is highly undesirable. This behaviour is reduced by recalculating the output F_t of the trading model in (1) twice. The first time is as before: after the new inputs have been received a new trading signal is generated. This trading signal is in fact based on the weights $w_{i,t-1}$ that were calculated in the previous time period. Here a second evaluation of the model is added *after* the weights are updated for the current period. Since the weight updating is designed to improve the model at each step, it makes sense to recalculate the trading decision with the most up-to-date version of the parameters $w_{i,t}$. This recalculation may result in

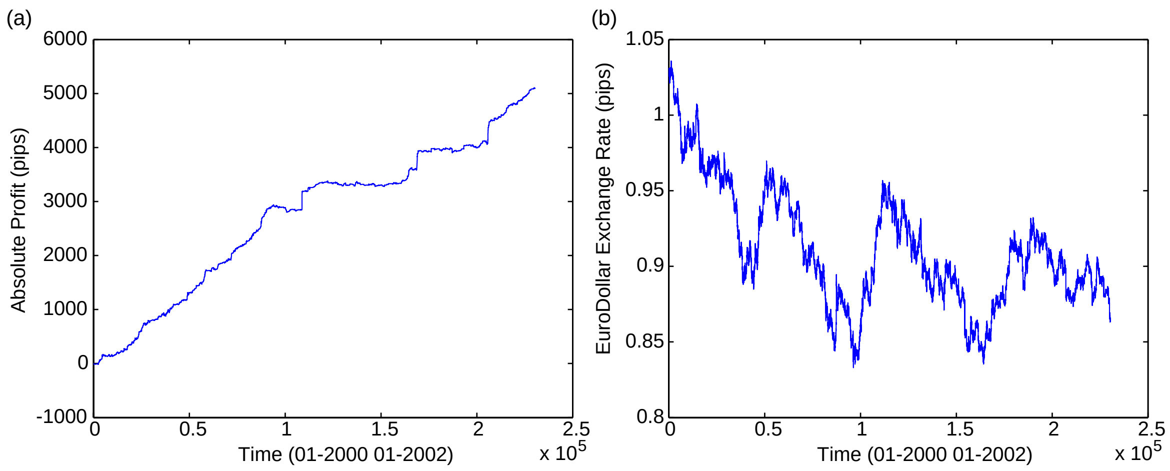


Fig. 2. (a) Out-of-sample cumulative profit measured in basis points for trading exactly one Euro against Dollar from January 2000 until January 2002. The trading system was dynamically optimized for a risk-aversion parameter of 0.5. The automated trading system earned a profit of 5104 pips or approximately 26% p.a., whereas a buy and hold strategy would have resulted in a 1636 pips loss or approximately an 8% p.a. loss. (b) The Euro–Dollar exchange rate over the period considered.

Table 1
Out-of-sample heuristics on the trading simulations: risk aversion ν, cumulative profit W_N, risk Σ, total number of trades N, average profit per trade R, percentage of trades in the right direction (after deduction of costs), utility with dynamic layer 3 optimization and the corresponding utility without layer 3 optimization turned on

ν	W_N (pips)	Σ	N	R (pips)	Direction correct (%)	U with layer 3	U without layer 3
0	4779	0.5334	2854	1.67	62	2.07	1.92
0.1	4717	0.5369	2852	1.65	62	1.79	1.67
0.2	4717	0.5369	2852	1.65	62	1.53	1.43
0.3	4663	0.5596	3004	1.55	62	1.25	1.18
0.4	5144	0.5225	3353	1.53	62	1.13	0.93
0.5	5104	0.4880	3337	1.53	62	0.86	0.69
0.6	5083	0.4962	3220	1.58	62	0.58	0.44
0.7	4780	0.4254	3129	1.53	61	0.32	0.19
0.8	4635	0.3901	2657	1.74	62	0.09	−0.05
0.9	4669	0.3761	2639	1.77	63	−0.14	−0.30

While trading, the spread was fixed at two pips and the system was restricted to only trade during hours when the true spread was not higher than two pips.

a different trading signal than was previously output by the model. This final trading signal is used for effective decision making by the risk and performance control layer.

3.2. Layer 2: risk and performance management

One of the strengths of the layered structure is that the decision to actually enter a trade is separated from the trade recommendations made by the basic structure in Layer 1. While Layer 1 outputs the preferred position to hold in the model and thus idealized world, Layer 2 evaluates this trade recommendation by considering additional risk factors relating to the real world before taking a decision to trade. These risk factors impose requirements on real world trading systems that are hard to include in the trading model itself, but can more easily be treated in the risk management overlay structure.

More than half a century ago J.M. Keynes, the famous economist, noted that: “The markets can stay irrational longer than you can stay solvent.” When a trade goes bad, a psychological tendency exists to keep the position open in the hope that the market will reverse itself and the trade will again turn profitable. This implies a risk-seeking attitude towards losses as opposed to risk-aversion with regard to profits. However, the market could very well not reverse itself and eventually force the close out of the position at a huge loss. This characteristic of human psychology needs to be avoided by every successful automated trading system. Layer 2 avoids this pitfall by building in a trailing stop-loss for every trade. A stop-loss is set and adjusted so that it is always x basis points under or above the best price ever reached during the life of the position. At this point x is an unknown parameter that will influence the performance. A numerical value for x will be obtained via the optimization in Layer 3.

If a position is closed out before a trade exit signal is given by the system, the current market behaviour was clearly not expected by the RRL trading model. Otherwise it could have foreseen the situation and would not have suffered a loss. If the market behaves ‘irrationally’ according to the model, it is likely that its deviant behaviour will persist for some time. For this reason a cool-down period is included in Layer 2. After the

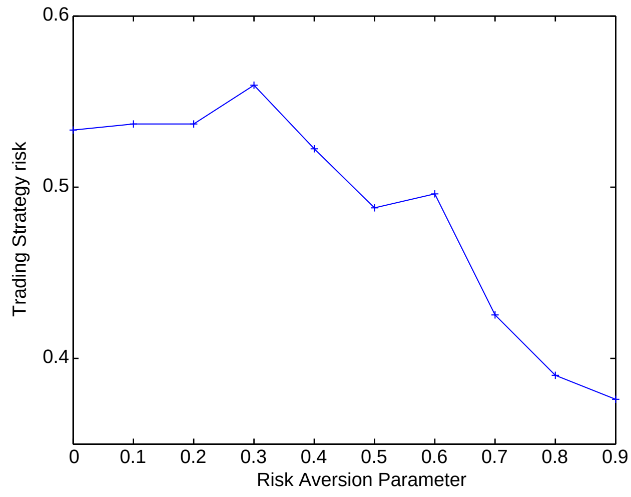


Fig. 3. Out-of-sample trading risk vs. risk aversion parameter. The resulting performance of the trading system becomes less risky as the risk aversion parameter in the layer 3 optimization is increased. For very low risk aversion parameters, the impact of risk in the utility function is insufficient to make the strategy less risky.

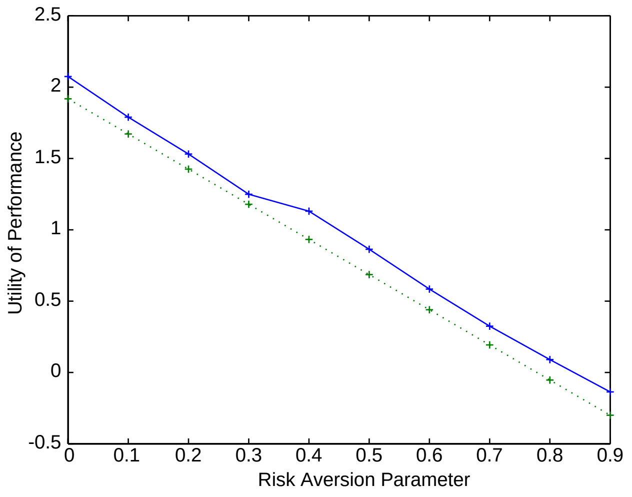
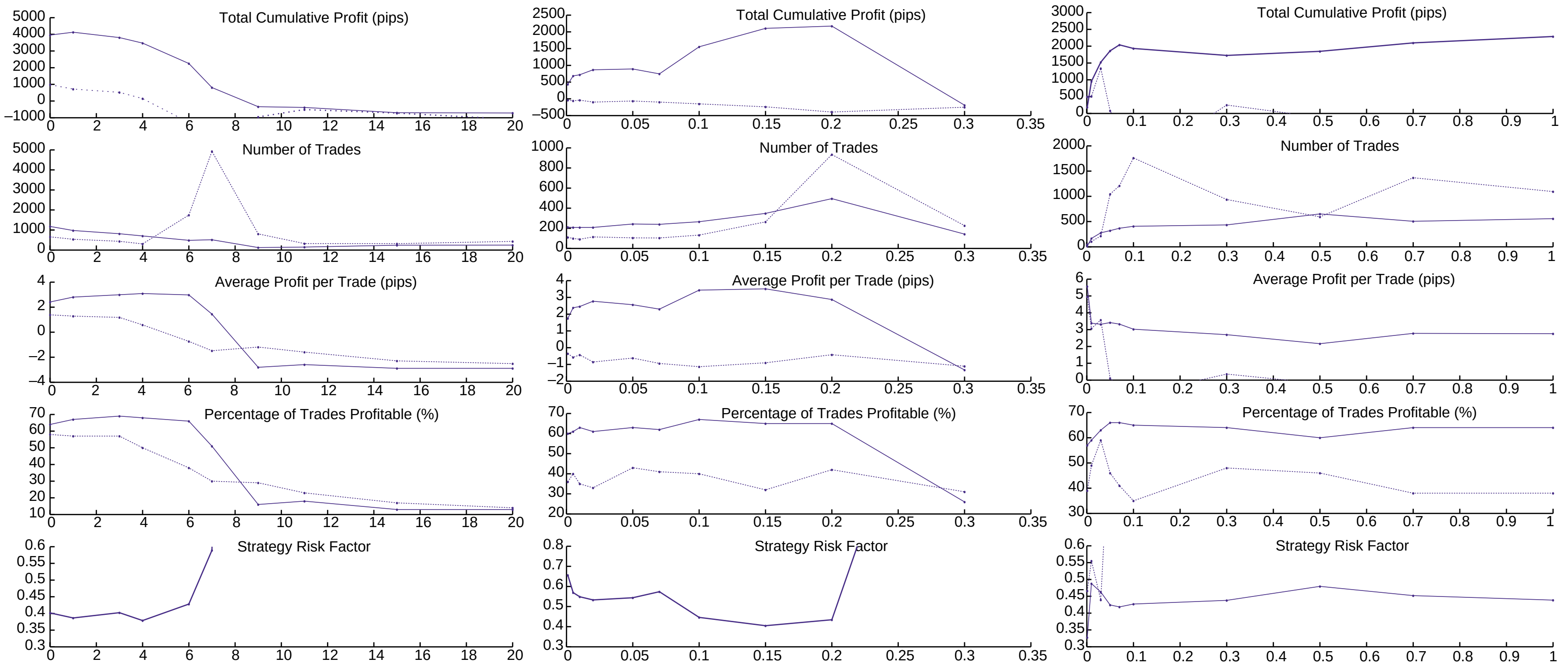


Fig. 4. Utility of out-of-sample trading performance vs. risk aversion parameter. The dotted line represents utility when Layer 3 is deactivated and the parameters are fixed to their optimal static values. The solid line represents utility when dynamic optimization in Layer 3 is activated. For every risk aversion setting, applying dynamic optimization is clearly beneficial.

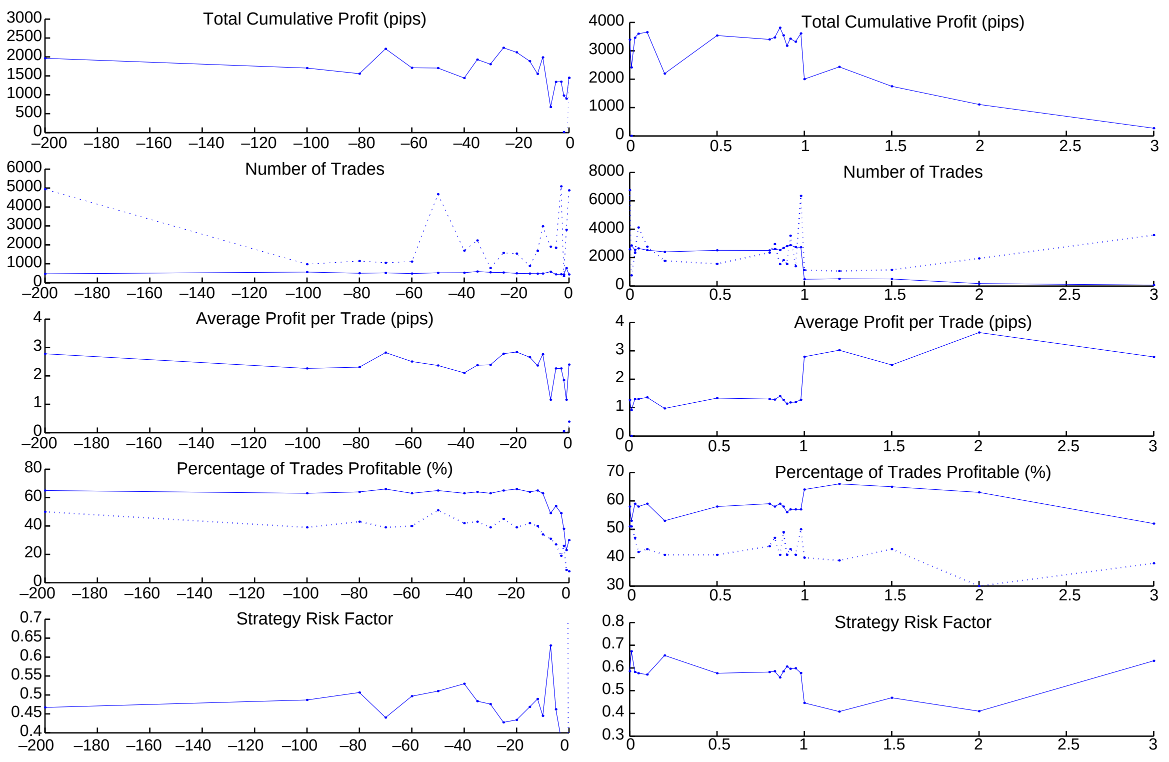


(a) Varying the trading cost δ.

(b) Varying the adaptation parameter η.

(c) Varying the learning rate ρ.

Fig. 5. Local performance behaviour for different values of the parameters. Dotted lines represent a system fed with 14 technical indicators as extra inputs.



(a)

Varying the stop-loss depth x

(b)

Varying the trading treshold y.

Fig. 6. Local performance behaviour for different values of the parameters. Dotted lines represent a system fed with 14 technical indicators as extra inputs.

position has been closed out the system needs to wait a while before trading can be resumed. The length of this optimal period of non-activity was determined experimentally as a fixed number of basic time intervals (here 1 min).

An important feature of the risk management layer is that it allows the evaluation of the strength of the trading signal given by Layer 0. This is accomplished by looking at the unthresholded output generated by (1). For example, a very strong buy signal corresponds to an output close to $+1$. The performance management system offers the possibility of validating a trading signal only when the output exceeds a specified non-zero threshold instead of just applying a sign-function. This threshold is a parameter y that is not fixed in advance but that can be set by the optimization in Layer 3 to influence the risk-return trade-off made by the trading system as a whole.

It is common knowledge in the trading community that automated trading systems often work for a certain while until they suddenly stop being profitable. At that point, the system must be shut down to avoid further losses and should be redesigned and thoroughly tested before making it operational again. The performance management layer makes sure that anomalous performance is detected in an early stage to protect the profits already earned. In that case, the system is automatically shut down and a warning is issued. The detection module employs a measure known as *maximum draw-down*. If at a certain point the draw-down in cumulative profits from its maximum proves larger than an amount z, the auto-shut-down procedure is initialized. This number z can be determined as a Value at Risk by using a fitted draw-down distribution, or it can be set more pragmatically to a fixed value.

3.3. Layer 3: dynamic optimization of utility

The trading system up to Layer 2 is able to trade autonomously and manage risk appropriately. However, the system's risk-return profile on a given data history depends on a number of parameters δ, η, ρ, x, y and z discussed above. It is the task of Layer 3 to search for optimal values of these parameters so that some utility of the resulting risk-return profile is maximized. Therefore, it is key to find a suitable utility function that is dependent on the cumulative profit, on a measure of trading risk and on the supervising trader's personal risk aversion. We define the *risk measure* Σ and *utility function* U as follows:

$$\Sigma := \frac{\sum_{i=0}^{N} (R_i)^2 I(R_i < 0)}{\sum_{i=0}^{N} (R_i)^2 I(R_i > 0)} \tag{13}$$

$$U(\bar{R}, \Sigma, \nu) := a(1-\nu)\bar{R} - \nu \cdot \Sigma, \tag{14}$$

where the strategy's raw return at time i is $R_i := W_i - W_{i-1}$, W_i is the wealth or cumulative profit at time i, $\bar{R} = W_N/N$ is the average profit per time interval, ν is the trader's personal risk aversion, a is a constant and N is the number of out-of-sample time

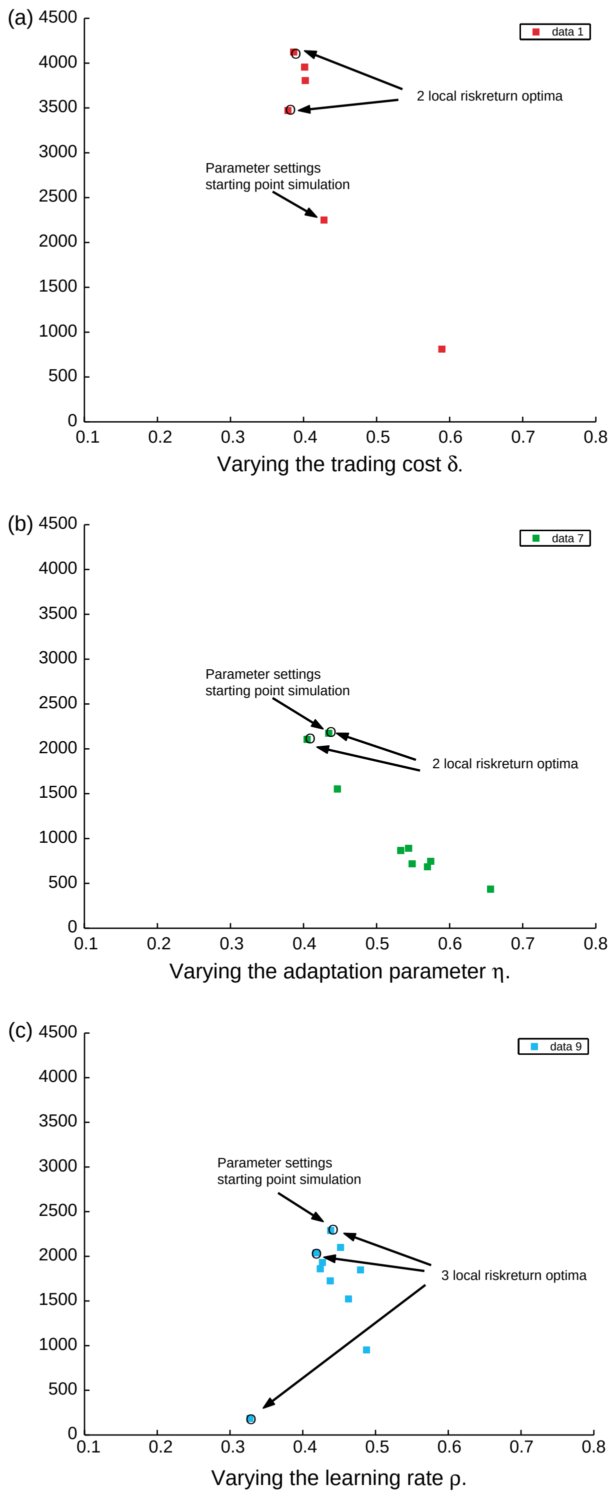


Fig. 7. Local performance behaviour for different values of the parameters, plotted in risk-return space.

intervals. A trader will get a higher utility from a larger average profit $\bar{R}$ and a lower risk factor Σ. A higher risk aversion parameter ν induces the optimization to find a performance profile with lower risk. Often this also implies a lower return.

The risk measure in (13) was constructed in such a way that it satisfies the following conditions:

1. Large negative strategy returns imply a higher risk value than several smaller ones that result in the same total negative return. This is because we want to avoid large sudden draw-downs that could, for example, lead to margin calls with a leveraged position.
2. A larger total impact of losing trades (as measured by the cumulative sum of squares of the negative strategy returns $\sum_{i=0}^{N}(R_i)^2 I(R_i < 0)$ versus the total impact of winning trades (as measured by the cumulative sum of squares of the positive strategy returns $\sum_{i=0}^{N}(R_i)^2 I(R_i > 0)$ implies a higher risk value even if the total profit remains the same.

   The risk measure Σ essentially implies that systems with monitonically increasing profits should be favoured over ones with whipsawing profits.
3. Comparing the total impact of losing versus winning trades is done in a relative way, i.e. by using a ratio, so that strategies that produce returns which only differ by a constant scaling factor result in the same risk value. Since the positive mean return would be multiplied by the scaling factor, the scaled up (leveraged) strategy would be preferred in a risk-return framework.
4. The risk measure Σ penalizes only *negative* strategy returns, and not positive returns smaller than the mean. Using a *centered* semivariance measure would perversely assign a higher risk value to a strategy with infrequent very profitable trades as these shifts the mean upwards while leaving the lower part of the distribution unaltered.

It should be noted that due to the non-standard from of our chosen risk measure Σ, we can not rely on earlier theoretical work by Ruszczyński et al. (Ogryczak & Ruszczyński, 2002; Ruszczyński & Vanderbei, 2003) that determines bounds for the risk-aversion parameter in order for efficient strategies in terms of risk-return utility to be second order stochastically dominant. It is left to further research to extend to our risk measure Σ proofs of consistency between the volatility risk measure and second order stochastic dominance as in Ogryczak and Ruszczyński (Ogryczak & Ruszczyński, 1999).

Our utility function depends on the average profit $\bar{R}$ and the risk measure Σ, which in turn are controlled by the values of five parameters: *trading cost* δ, *adaptation parameter* η, *learning rate* ρ, *stop-loss parameter* x and *trading threshold* y. Hence the parameter optimization problem becomes formally:

$$\max_{\delta,\eta,\rho,x,y} U(\bar{R},\Sigma : \delta,\eta,\rho,x,y;\nu). \tag{15}$$

Unfortunately, (15) cannot be solved analytically since we use an online updating scheme. Moreover, numerical optimization in a five-dimensional space requires too much computing power and time. Therefore, we implement a one-at-a-time random search optimization in which each parameter is approximately optimized individually while

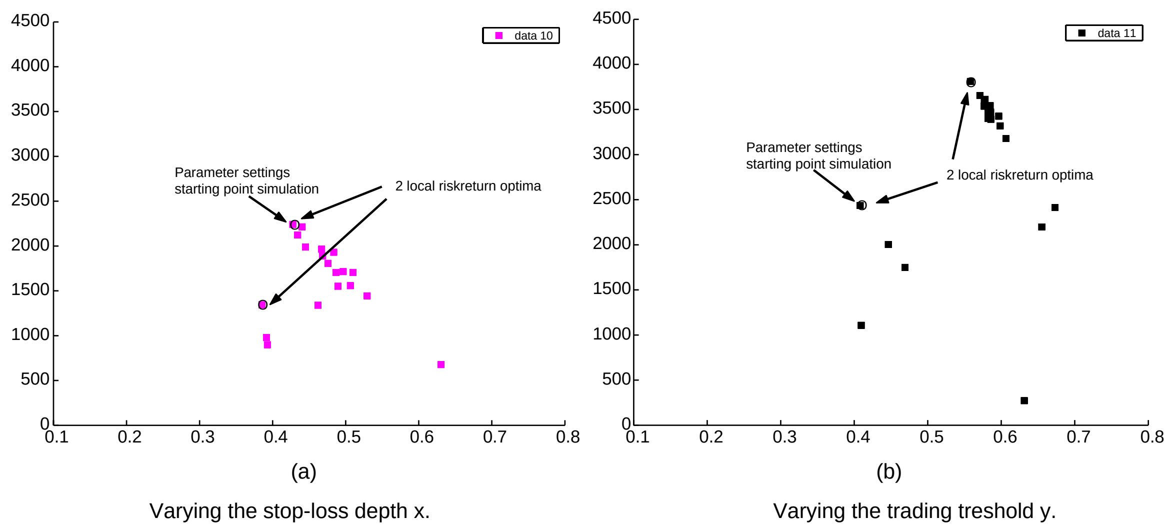


Fig. 8. Local performance behaviour for different values of the parameters, plotted in risk-return space.

keeping the others constant:

$$\max_{\delta}\max_{\eta}\max_{\rho}\max_{x}\max_{y} U(\bar{R},\Sigma : \delta,\eta,\rho,x,y;\nu). \quad (16)$$

The individual optimizations were performed by evaluating 15 random values distributed normally around the starting value and picking the best. This saves function evaluations and we do not have to estimate numerical derivatives of a utility function that is not smooth in the optimized parameters. Of course, other more sophisticated optimization schemes could be applied.

One-at-a-time optimization only works if the objective function is sufficiently well-behaved. To make a preliminary qualitative assessment of the response behaviour of the system when its five tuning parameters are adjusted, two numerical initialization vectors for these five parameters were chosen and the local performance behaviour monitored. Each of the five parameters were modified in turn while keeping the other parameters constant. Figs. 5 and 6 show the local performance behaviour for one of the initialization vectors chosen. The other vector gave very similar results. In Figs. 7 and 8 the associated performance profiles are plotted in risk-return space to illustrate the impact on performance. The main empirical observation is that performance measured in each of the dimensions is generally well-behaved when each of the five parameters lies within vague boundaries that can easily be read from the graphs. Allowing values outside these boundaries will generally increase the search time of the optimization.

The length of the time for this top-layer optimization interval is set to a multiple of the training interval length L_{train} so that it is safe to superimpose it on the previous layers. It is sensible to choose a relatively lower frequency for this optimization as the parameters being optimized determine the risk management characteristics as well as the adaptive behaviour of the underlying learning algorithm. As these are both aspects of the system that typically span multiple time periods or even multiple trades, more time is required to see the impact of these hyper-parameters on trading performance. At the start, the five parameters δ, η, ρ, x and y are initialized with five randomly chosen values because there is no information available about what values result in good performance. The optimization then searches for values that result in optimal utility of performance over some past period of time. The ARL framework not only eliminates the problem of having to choose and fix hyper-parameters a priori, but also allows the trading system to dynamically adapt to changing market conditions and even to take into account changing risk preferences of the user over time.

4. Application to FX trading

This section reports the use of the ARL trading system to trade the foreign exchange (FX) markets based on historical data. For this purpose, the system was fed historical FX data and its out-of-sample performance on this data recorded and analyzed. A few more experiments were conducted to test the impact of the dynamic optimization of utility.

Fig. 2 shows the out-of-sample cumulative profits when the system was used to trade the Euro–Dollar currency pair.[1] Trading was only allowed during the active hours that correspond to sufficient liquidity: from 9 a.m. London time until 5 p.m. London time. During these hours, the average bid-ask spread in the inter-dealer FX market was lower than two pips. As execution speeds and liquidity are usually very high on interdealer electronic trading platforms like EBS and Reuters 3000, no further slippage was assumed. The ARL system was able to earn 5104 pips over 2 years, or more than 26% per

[1] Historical FX data for Euro–Dollar was kindly supplied by HSBC Bank, London, for which we are grateful. The frequency is 1 min and the data spans a period from January 2000 until January 2002.

annum. Given the absence of any serious draw-dawns in the profit graph, this is not a bad performance. Preliminary results on recent FX market data covering a period up to January 2004 indicates that the profit curve exhibits a diminishing slope. This could signify that markets are becoming more and more efficient and that it is becoming more difficult to make profits when only price information is fed into the system.

To illustrate the benefits of our structural approach and the top-level utility optimization, a series of experiments was run to compare performance and utility of performance with and without dynamic optimization turned on. Table 1 shows the resulting performance statistics for traders with different risk aversions. From these experiments we see that as risk aversion increases, utility generally goes down because of heavier penalization of the risk factor. This makes the system less free to chose a utility maximizing strategy. When very low risk aversion parameters of 0 until 0.2 are used, the impact of risk in the utility function is insufficient to obtain a less risky strategy. When risk aversion is increased until 0.4, the cumulative profit and total number of trades go up, but the average profit made per trade decreases. The reason why the system is choosing less profitable trades on average is most likely because it needs to select less risky trades. Fortunately at this point the system adopts a new strategy that takes on a larger number of trades so that the cumulative profit still ends up high, despite the average profit and the utility going down. When the risk aversion is further increased until 0.9, the system needs to select only the most profitable trades and thus the average profit increases again. The total number of trades drops and unfortunately so does the cumulative profit.

Fig. 3 demonstrates that when realistic risk aversion parameters larger than 0.3 are used, the optimization layer makes sure that the resulting system becomes less and less risky as the risk aversion parameter is increased. The average net profit made per trade ranged from 1.53 pips up to 1.77 pips, depending on the trader's risk aversion.

To demonstrate further that the dynamical optimization performed in layer 3 adds to an increased trading performance, we compared the utilities of performance for different risk aversions with the corresponding utilities when no dynamic optimization was performed. In the second case, the values of the five hyper-parameters were set so that they gave optimal performance on the whole dataset. To assure that this static parameter setting is the best one possible, and thus provides a reliable benchmark, static optimal values were calculated based on the training as well as the test sets. It should be noted that the results with those static parameters are only meant as an upper bound on performance without optimization and not as a reflection of real performance. Indeed, when optimizing over both training and test sets, knowledge of all future prices is assumed at any point in time. Hence, these static parameter results are better than what can be achieved in out-of-sample trading. Nevertheless, they provide us with a best-case scenario for static parameter settings. Ideally the ARL system should be able to deliver an even better performance compared to this hard-to-beat benchmark. Fig. 4 shows the utilities that were obtained for several risk aversions of the trading system with and without Layer 3 activated. Clearly the ARL system performs consistently better than the benchmark, which illustrates that the addition of Layer 3 to the trading system contributes significantly to better performance (Figs. 5–8).

5. Conclusions

In this paper we have developed an automated trading system based on adaptive reinforcement learning. The parameters that govern the learning behaviour of the machine-learning algorithm and the risk management layer are dynamically optimized to maximize a trader's utility. A risk-aversion parameter allows control of the system's trade-off between strategy risk and return. ARL automatically tunes the hyper-parameters and thus eliminates the need for manual model calibration or fitting and hence greatly reduces the risk of data-snooping. The system performed very well on the high-frequency historical FX dataset that was examined. A first possible avenue for future research is to examine whether the trading performance of this system can be further improved by feeding other information than price data into the system. In earlier work (Bates et al., 2003; Leemans, 2003) we have demonstrated that order book or order flow information is able to enhance the performance of automated trading systems. Secondly, the risk management layer could be extended to control several automated FX trading systems that trade different currencies, in an attempt to emulate the actions of a human FX trader.

References

Austin, M. P., Bates, R. G., Dempster, M. A. H., Leemans, V., & Williams, S. N. (2004). Adaptive systems for foreign exchange trading. *Quantitative Finance*, *4*(4), 37–45.

Bates, R. G., Dempster, M. A. H., & Romahi, Y. S. (2003). Evolutionary reinforcement learning in FX order book and order flow analysis. *Proceedings of the 2003 IEEE International Conference on Computational Intelligence for Financial Engineering, Hong Kong, March 2003*, 355–362.

Dempster, M. A. H., & Jones, C. M. (2001). A real-time adaptive trading system using genetic programming. *Quantitative Finance*, *1*, 397–413.

Dempster, M. A. H., Payne, T. W., Romahi, Y. S., & Thompson, G. W. P. (2001). Computational learning techniques for intraday FX trading using popular technical indicators. Special issue on Computational Finance, *IEEE Transactions on Neural Networks*, *12*, 744–754.

Dempster, M. A. H., & Romahi, Y. S. (2002). *Intraday FX trading: An evolutionary reinforcement learning approach. Intelligent Data Engineering and Automated Learning 2002,* Lecture Notes in Computer Science. Berlin: Springer pp. 347–358.

Gold, C. (2003). FX trading via recurrent reinforcement learning. *Proceedings of the IEEE International Conference on Computational Intelligence in Financial Engineering 2003*, op.cit., 363–371.

Leemans, V. (2003). *Real time trading systems. MPhil dissertation,* Centre for Financial Research, Judge Institute of Management. Cambridge: University of Cambridge.

Moody, J., & Saffell, M. (1999). Minimising downside risk via stochastic dynamic programming. In Y. S. Abu-Mostafa, B. LeBaron, A. W. Lo, &

A. S. Weigend (Eds.), *Sixth International conference computational finance 1999* (pp. 403–415). Cambridge, MA: MIT Press.

Moody, J., & Saffell, M. (2001). Learning to trade via direct reinforcement. *IEEE Transactions on Neural Networks*, *12*(4), 875–889.

Ogryczak, W., & Ruszczyński, A. (1999). From stochastic dominance to mean-risk models: Semideviations as risk measures. *European Journal of Operations Research*, *116*, 33–50.

Ogryczak, W., & Ruszczyński, A. (2002). Dual stochastic dominance and related mean-risk models. *SIAM Journal on Optimization*, *13*(1), 60–78.

Ruszczyński, A., & Vanderbei, R. J. (2003). Frontiers of stochastically nondominated portfolios. *Econometrica*, *71*(4), 1287–1297.

Venturi S. (2003). Evolutionary algorithms for currency trading. PhD thesis, Centre for Financial Research, Judge Institute of Management, University of Cambridge.