Reinforcement Learning for Trading Systems and Portfolios

John Moody and Matthew Saffell*
Oregon Graduate Institute, CSE Dept.
P.O. Box 91000, Portland, OR 97291-1000
{moody, saffell}@cse.ogi.edu

Abstract

We propose to train trading systems by optimizing financial objective functions via reinforcement learning. The performance functions that we consider as value functions are profit or wealth, the Sharpe ratio and our recently proposed *differential Sharpe ratio* for online learning. In Moody & Wu (1997), we presented empirical results in controlled experiments that demonstrated the advantages of reinforcement learning relative to supervised learning. Here we extend our previous work to compare Q-Learning to a reinforcement learning technique based on real-time recurrent learning (RTRL) that maximizes immediate reward.

Our simulation results include a spectacular demonstration of the presence of predictability in the monthly Standard and Poors 500 stock index for the 25 year period 1970 through 1994. Our reinforcement trader achieves a simulated out-of-sample profit of over 4000% for this period, compared to the return for a buy and hold strategy of about 1300% (with dividends reinvested). This superior result is achieved with substantially lower risk.

Introduction: Reinforcement Learning for Trading

The investor's or trader's ultimate goal is to optimize some relevant measure of trading system performance, such as profit, economic utility or risk-adjusted return. In this paper, we propose to use reinforcement learning to directly optimize such trading system performance functions, and we compare two different reinforcement learning methods. The first uses immediate rewards to train the trading systems, while the second (Q-Learning (Watkins)) approximates discounted future rewards. These methodologies can be applied to optimizing systems designed to trade a single security or to trade a portfolio of securities. In addition, we propose a novel value function for risk adjusted return suitable for online learning: the *differential Sharpe ratio*.

Trading system profits depend upon sequences of interdependent decisions, and are thus path-dependent. Optimal trading decisions when the effects of transactions costs, market impact and taxes are included require knowledge of the current system state. Reinforcement learning provides a more elegant means for training trading systems when state-dependent transaction costs are included, than do more standard supervised approaches (Moody, Wu, Liao & Saffell). The reinforcement learning algorithms used here include maximizing immediate reward and Q-Learning (Watkins).

Though much theoretical progress has been made in recent years in the area of reinforcement learning, there have been relatively few successful, practical applications of the techniques. Notable examples include Neuro-gammon (Tesauro), the asset trader of Neuneier (1996), an elevator scheduler (Crites & Barto) and a space-shuttle payload scheduler (Zhang & Dietterich). In this paper we present results for reinforcement learning trading systems that outperform the S&P 500 Stock Index over a 25-year test period, thus demonstrating the presence of predictable structure in US stock prices.

Structure of Trading Systems and Portfolios

Traders: Single Asset with Discrete Position Size

In this section, we consider performance functions for systems that trade a single security with price series z_t. The trader is assumed to take only long, neutral or short positions $F_t \in \{-1, 0, 1\}$ of constant magnitude. The constant magnitude assumption can be easily relaxed to enable better risk control. The position F_t is established or maintained at the end of each time interval t, and is re-assessed at the end of period $t+1$. A trade is thus possible at the end of each time period, although nonzero trading costs will discourage excessive trading. A trading system return R_t is realized at the end of the time interval $(t-1, t]$ and includes the profit or loss resulting from the position F_{t-1} held during that interval and any transaction cost incurred at time t due to a difference in the positions F_{t-1} and F_t.

In order to properly incorporate the effects of

* The authors are also with Nonlinear Prediction Systems, Tel: (503)531-2024.

transactions costs, market impact and taxes in a trader's decision making, the trader must have internal state information and must therefore be recurrent. An example of a single asset trading system that could take into account transactions costs and market impact would be one with the following decision function: $F_t = F(\theta_t; F_{t-1}, I_t)$ with $I_t = \{z_t, z_{t-1}, z_{t-2}, \ldots; y_t, y_{t-1}, y_{t-2}, \ldots\}$ where θ_t denotes the (learned) system parameters at time t and I_t denotes the information set at time t, which includes present and past values of the price series z_t and an arbitrary number of other external variables denoted y_t.

Portfolios: Continuous Quantities of Multiple Assets

For trading multiple assets in general (typically including a risk-free instrument), a multiple output trading system is required. Denoting a set of m markets with price series $\{\{z_t^a\} : a = 1, \ldots, m\}$, the market return r_t^a for price series z_t^a for the period ending at time t is defined as $((z_t^a/z_{t-1}^a) - 1)$. Defining portfolio weights of the a^{th} asset as $F^a()$, a trader that takes only long positions must have portfolio weights that satisfy: $F^a \geq 0$ and $\sum_{a=1}^m F^a = 1$.

One approach to imposing the constraints on the portfolio weights without requiring that a constrained optimization be performed is to use a trading system that has softmax outputs: $F^a() = \{\exp[f^a()]\}/\{\sum_{b=1}^m \exp[f^b()]\}$ for $a = 1, \ldots, m$. Here, the $f^a()$ could be linear or more complex functions of the inputs, such as a two layer neural network with sigmoidal internal units and linear outputs. Such a trading system can be optimized using unconstrained optimization methods. Denoting the sets of raw and normalized outputs collectively as vectors $\mathbf{f}()$ and $\mathbf{F}()$ respectively, a recursive trader will have structure $\mathbf{F}_t = \text{softmax}\{\mathbf{f}_t(\theta_{t-1}; \mathbf{F}_{t-1}, I_t)\}$.

Financial Performance Functions

Profit and Wealth for Traders and Portfolios

Trading systems can be optimized by maximizing performance functions $U()$ such as profit, wealth, utility functions of wealth or performance ratios like the Sharpe ratio. The simplest and most natural performance function for a risk-insensitive trader is profit. We consider two cases: additive and multiplicative profits. The transactions cost rate is denoted δ.

Additive profits are appropriate to consider if each trade is for a fixed number of shares or contracts of security z_t. This is often the case, for example, when trading small futures accounts or when trading standard US$ FX contracts in dollar-denominated foreign currencies. With the definitions $r_t = z_t - z_{t-1}$ and $r_t^f = z_t^f - z_{t-1}^f$ for the price returns of a risky (traded) asset and a risk-free asset (like T-Bills) respectively, the additive profit accumulated over T time periods with trading position size $\mu > 0$ is then defined as:

$$\begin{aligned} P_T &= \mu \sum_{t=1}^{T} R_t \qquad (1) \\ &= \mu \sum_{t=1}^{T} \left\{ r_t^f + F_{t-1}(r_t - r_t^f) - \delta |F_t - F_{t-1}| \right\} \end{aligned}$$

with $P_0 = 0$ and typically $F_T = F_0 = 0$. Equation (2) holds for continuous quantities also. The wealth is defined as $W_T = W_0 + P_T$.

Multiplicative profits are appropriate when a fixed fraction of accumulated wealth $\nu > 0$ is invested in each long or short trade. Here, $r_t = (z_t/z_{t-1} - 1)$ and $r_t^f = (z_t^f/z_{t-1}^f - 1)$. If no short sales are allowed and the leverage factor is set fixed at $\nu = 1$, the wealth at time T is:

$$\begin{aligned} W_T &= W_0 \prod_{t=1}^{T} \{1 + R_t\} \qquad (2) \\ &= W_0 \prod_{t=1}^{T} \left(1 + (1 - F_{t-1}) r_t^f + F_{t-1} r_t\right) \\ &\qquad \cdot \left(1 - \delta |F_t - F_{t-1}|\right). \end{aligned}$$

When multiple assets are considered, the effective portfolio weightings change with each time step due to price movements. Thus, maintaining constant or desired portfolio weights requires that adjustments in positions be made at each time step. The wealth after T periods for a portfolio trading system is

$$W_T = W_0 \prod_{t=1}^{T} \{1 + R_t\} = W_0 \prod_{t=1}^{T} \left(\sum_{a=1}^{m} F_{t-1}^a \frac{z_t^a}{z_{t-1}^a} \right) \qquad (3)$$
$$\cdot \left(1 - \delta \sum_{a=1}^{m} |F_t^a - \tilde{F}_t^a| \right),$$

where $\tilde{F}_t^a$ is the effective portfolio weight of asset a before readjusting, defined as $\tilde{F}_t^a = \{F_{t-1}^a (z_t^a/z_{t-1}^a)\}/\{\sum_{b=1}^m F_{t-1}^b (z_t^b/z_{t-1}^b)\}$. In (4), the first factor in the curly brackets is the increase in wealth over the time interval t prior to rebalancing to achieve the newly specified weights F_t^a. The second factor is the reduction in wealth due to the rebalancing costs.

Risk Adjusted Return: The Sharpe and *Differential* Sharpe Ratios

Rather than maximizing profits, most modern fund managers attempt to maximize risk-adjusted return as advocated by Modern Portfolio Theory. The Sharpe ratio is the most widely-used measure of risk-adjusted return (Sharpe). Denoting as before the trading system returns for period t (including transactions costs) as R_t, the Sharpe ratio is defined to be

$$S_T = \frac{\text{Average}(R_t)}{\text{Standard Deviation}(R_t)} \qquad (4)$$

where the average and standard deviation are estimated for periods $t = \{1, \ldots, T\}$.

Proper on-line learning requires that we compute the influence on the Sharpe ratio of the return at time t. To accomplish this, we have derived a new objective function called the *differential Sharpe ratio* for on-line optimization of trading system performance (Moody, Wu, Liao & Saffell). It is obtained by considering exponential moving averages of the returns and standard deviation of returns in (4), and expanding to first order in the decay rate η: $S_t \approx S_{t-1} + \eta \frac{dS_t}{d\eta}|_{\eta=0} + O(\eta^2)$. Noting that only the first order term in this expansion depends upon the return R_t at time t, we define the *differential Sharpe ratio* as:

$$D_t \equiv \frac{dS_t}{d\eta} = \frac{B_{t-1}\Delta A_t - \frac{1}{2}A_{t-1}\Delta B_t}{(B_{t-1} - A_{t-1}^2)^{3/2}} \quad . \tag{5}$$

where the quantities A_t and B_t are exponential moving estimates of the first and second moments of R_t:

$$\begin{aligned} A_t &= A_{t-1} + \eta \Delta A_t = A_{t-1} + \eta(R_t - A_{t-1}) \\ B_t &= B_{t-1} + \eta \Delta B_t = B_{t-1} + \eta(R_t^2 - B_{t-1}) \end{aligned} \quad . (6)$$

Treating A_{t-1} and B_{t-1} as numerical constants, note that η in the update equations controls the magnitude of the influence of the return R_t on the Sharpe ratio S_t. Hence, the *differential Sharpe ratio* represents the influence of the return R_t realized at time t on S_t.

Reinforcement Learning for Trading Systems and Portfolios

The goal in using reinforcement learning to adjust the parameters of a system is to maximize the expected payoff or reward that is generated due to the actions of the system. This is accomplished through trial and error exploration of the environment. The system receives a reinforcement signal from its environment (a *reward*) that provides information on whether its actions are good or bad. The performance functions that we consider are functions of profit or wealth $U(W_T)$ after a sequence of T time steps, or more generally of the whole time sequence of trades $U(W_1, W_2, \ldots, W_T)$ as is the case for a path-dependent performance function like the Sharpe ratio. In either case, the performance function at time T can be expressed as a function of the sequence of trading returns $U(R_1, R_2, \ldots, R_T)$. We denote this by U_T in the rest of this section.

Maximizing Immediate Utility

Given a trading system model $F_t(\theta)$, the goal is to adjust the parameters θ in order to maximize U_T. This maximization for a complete sequence of T trades can be done off-line using dynamic programming or batch versions of recurrent reinforcement learning algorithms. Here we do the optimization on-line using a standard reinforcement learning technique. This reinforcement learning algorithm is based on stochastic gradient ascent. The gradient of U_T with respect to the parameters θ of the system after a sequence of T trades is

$$\frac{dU_T(\theta)}{d\theta} = \sum_{t=1}^{T} \frac{dU_T}{dR_t} \left\{ \frac{dR_t}{dF_t}\frac{dF_t}{d\theta} + \frac{dR_t}{dF_{t-1}}\frac{dF_{t-1}}{d\theta} \right\} \quad . \tag{7}$$

The above expression as written with scalar F_i applies to the traders of a single risky asset, but can be trivially generalized to the vector case for portfolios.

The system can be optimized in batch mode by repeatedly computing the value of U_T on forward passes through the data and adjusting the trading system parameters by using gradient ascent (with learning rate ρ) $\Delta\theta = \rho dU_T(\theta)/d\theta$ or some other optimization method. A simple on-line stochastic optimization can be obtained by considering only the term in (7) that depends on the most recently realized return R_t during a forward pass through the data:

$$\frac{dU_t(\theta)}{d\theta} = \frac{dU_t}{dR_t} \left\{ \frac{dR_t}{dF_t}\frac{dF_t}{d\theta} + \frac{dR_t}{dF_{t-1}}\frac{dF_{t-1}}{d\theta} \right\} \quad . \tag{8}$$

The parameters are then updated on-line using $\Delta\theta_t = \rho dU_t(\theta_t)/d\theta_t$. Such an algorithm performs a stochastic optimization (since the system parameters θ_t are varied *during* each forward pass through the training data), and is an example of *immediate reward* reinforcement learning. This approach is described in (Moody, Wu, Liao & Saffell) along with extensive simulation results.

Q-Learning

Besides explicitly training a trader to take actions, we can also implicitly learn correct actions through the technique of *value iteration*. In value iteration, an estimate of future costs or rewards is made for a given state, and the action is chosen that minimizes future costs or maximizes future rewards. Here we consider the specific technique named Q-Learning (Watkins), which estimates future rewards based on the current state and the current action taken. We can write the Q-function version of Bellman's equation as

$$Q^*(x,a) = U(x,a) + \gamma \sum_{y=0}^{n} p_{xy}(a) \left\{ \max_b Q^*(y,b) \right\} \quad , \tag{9}$$

where there are n states in the system and $p_{xy}(a)$ is the probability of transitioning from state x to state y given action a. The advantage of using the Q-function is that there is no need to know the system model $p_{xy}(a)$ in order to choose the best action. One simply calculates the best action as $a^* = \arg\max_a(Q^*(x,a))$. The update rule for training a function approximator is $Q(x,a) = Q(x,a) + \rho(U(x,a) + \gamma \max_b Q^*(y,b))$, where ρ is a learning rate.

Empirical Results

Long/Short Trader of a Single Security

We have tested techniques for optimizing both profit and the Sharpe ratio in a variety of settings. We present

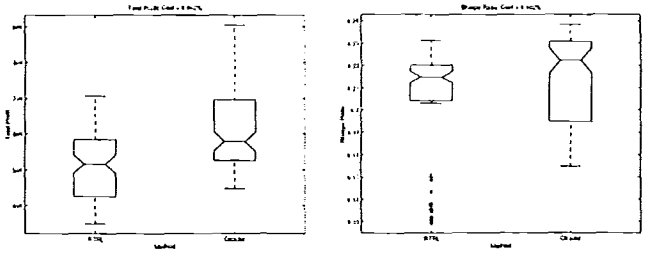

Figure 1: Boxplot for ensembles of 100 experiments comparing the performance of the "RTRL" and "Qtrader" trading systems on artificial price series. "Qtrader" outperforms the "RTRL" system in terms of both (a) profit and (b) risk-adjusted returns (Sharpe ratio). The differences are statistically significant.

results here on artificial data comparing the performance of a trading system, "RTRL", trained using real time recurrent learning with the performance of a trading system, "Qtrader", implemented using Q-Learning. We generate log price series of length 10,000 as random walks with autoregressive trend processes. These series are trending on short time scales and have a high level of noise. The results of our simulations indicate that the Q-Learning trading system outperforms the "RTRL" trader that is trained to maximize immediate reward.

The "RTRL" system is initialized randomly at the beginning, trained for a while using labelled data to initialize the weights, and then adapted using real-time recurrent learning to optimize the *differential Sharpe ratio* (5). The neural net that is used in the "Qtrader" system starts from random initial weights. It is trained repeatedly on the first 1000 data points with the discount parameter γ set to 0 to allow the network to learn immediate reward. Then γ is set equal to 0.9 and training continues with the second thousand data points being used as a validation set. The value function used here is profit.

Figure 1 shows box plots summarizing test performances for ensembles of 100 experiments. In these simulations, the data are partitioned into a training set consisting of the first 2,000 samples and a test set containing the last 8,000 samples. Each trial has different realizations of the artificial price process and different random initial parameter values. The transaction costs are set at 0.2%, and we observe the cumulative profit and Sharpe ratio over the test data set. We find that the "Qtrader" system, which looks at future profits, significantly outperforms the "RTRL" system which looks to maximize immediate rewards because it is effectively looking farther into the future when making decisions.

S&P 500 / TBill Asset Allocation System

Long/Short Asset Allocation System and Data

A long/short trading system is trained on monthly S&P 500 stock index and 3-month TBill data to maximize the differential Sharpe ratio. The S&P 500 target series is the total return index computed by reinvesting dividends. The 84 input series used in the trading systems include both financial and macroeconomic data. All data are obtained from Citibase, and the macroeconomic series are lagged by one month to reflect reporting delays.

A total of 45 years of monthly data are used, from January 1950 through December 1994. The first 20 years of data are used only for the initial training of the system. The test period is the 25 year period from January 1970 through December 1994. The experimental results for the 25 year test period are true *ex ante* simulated trading results.

For each year during 1970 through 1994, the system is trained on a moving window of the previous 20 years of data. For 1970, the system is initialized with random parameters. For the 24 subsequent years, the previously learned parameters are used to initialize the training. In this way, the system is able to adapt to changing market and economic conditions. Within the moving training window, the "RTRL" systems use the first 10 years for stochastic optimization of system parameters, and the subsequent 10 years for validating early stopping of training. The networks are linear, and are regularized using quadratic weight decay during training with a regularization parameter of 0.01. The "Qtrader" systems use a bootstrap sample of the 20 year training window for training, and the final 10 years of the training window are used for validating early stopping of training. The networks are two-layer feedforward networks with 30 tanh units in the hidden layer.

Experimental Results The left panel in Figure 2 shows box plots summarizing the test performance for the full 25 year test period of the trading systems with various realizations of the initial system parameters over 30 trials for the "RTRL" system, and 10 trials for the "Qtrader" system[1]. The transaction cost is set at 0.5%. Profits are reinvested during trading, and multiplicative profits are used when calculating the wealth. The notches in the box plots indicate robust estimates of the 95% confidence intervals on the hypothesis that the median is equal to the performance of the buy and hold strategy. The horizontal lines show the performance of the "RTRL" voting, "Qtrader" voting and buy and hold strategies for the same test period. The annualized monthly Sharpe ratios of the buy and hold strategy, the "Qtrader" voting strategy and the "RTRL" voting strategy are 0.34, 0.63 and 0.83 respectively. The Sharpe ratios calculated here are for the excess returns of the strategies over the 3-month treasury bill rate.

Figure 2 shows results for following the strategy of taking positions based on a majority vote of the ensembles of trading systems compared with the buy and hold strategy. We can see that the trading systems go short the S&P 500 during critical periods, such as the oil price

[1]Ten trials were done for the "Qtrader" system due to the amount of computation required in training the systems

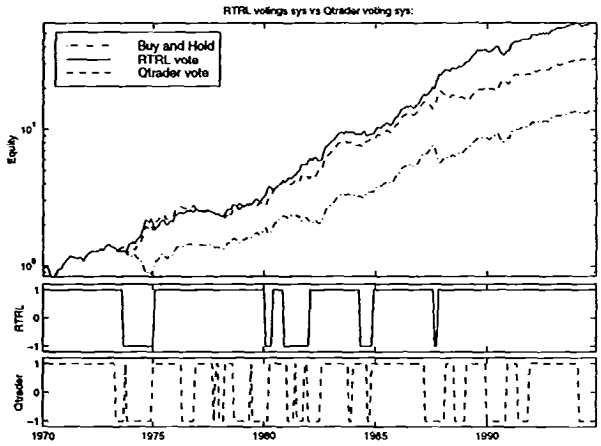

Figure 2: Test results for ensembles of simulations using the S&P 500 stock index and 3-month Treasury Bill data over the 1970-1994 time period. The figure shows the equity curves associated with the voting systems and the buy and hold strategy, as well as the voting trading signals produced by the systems. The solid curves correspond to the "RTRL" voting system performance, dashed curves to the "Qtrader" voting system and the dashed and dotted curves indicate the buy and hold performance. Initial equity is set to 1, and transaction costs are set at 0.5%. In both simulations, the traders avoid the dramatic losses that the buy and hold strategy incurred during 1974. In addition, the "RTRL" trader makes money during the crash of 1987, while the "Qtrader" system avoids the large losses associated with the buy and hold strategy during the same period.

shock of 1974, the tight money (high interest rate) periods of the early 1980's, the market correction of 1984 and the 1987 crash. This ability to take advantage of high treasury bill rates or to avoid periods of substantial stock market loss is the major factor in the long term success of these trading models. One exception is that the "RTRL" trading system remains long during the 1991 stock market correction associated with the Persian Gulf war, though the "Qtrader" system does identify the correction. On the whole though, the "Qtrader" system trades much more frequently than the "RTRL" system, and in the end does not perform as well on this data set.

From these results we find that both trading systems outperform the buy and hold strategy, as measured by both accumulated wealth and Sharpe ratio. These differences are statistically significant and support the proposition that there is predictability in the U.S. stock and treasury bill markets during the 25 year period 1970 through 1994. A more detailed presentation of the "RTRL" results is presented in (Moody, Wu, Liao & Saffell).

Conclusions and Extensions

In this paper, we have trained trading systems via reinforcement learning to optimize financial objective functions including our recently proposed *differential Sharpe ratio* for online learning. We have also provided simulation results that demonstrate the presence of predictability in the monthly S&P 500 Stock Index for the 25 year period 1970 through 1994. We have previously shown with extensive simulation results (Moody, Wu, Liao & Saffell) that the "RTRL" trading system significantly outperforms systems trained using supervised methods for traders of both single securities and portfolios. The superiority of reinforcement learning over supervised learning is most striking when state-dependent transaction costs are taken into account. Here we show that the Q-Learning approach can significantly improve on the "RTRL" method when trading single securities, as it does for our artificial data set. However, the "Qtrader" system does not perform as well as the "RTRL" system on the S&P 500 / TBill asset allocation problem, possibly due to its more frequent trading. This effect deserves further exploration.

Acknowledgements

We acknowledge support for this work from Nonlinear Prediction Systems and from DARPA under contract DAAH01-96-C-R026 and AASERT grant DAAH04-95-1-0485.

References

Crites, R. H. & Barto, A. G. (1996), Improving elevator performance using reinforcement learning, *in* D. S. Touretzky, M. C. Mozer & M. E. Hasselmo, eds, 'Advances in NIPS', Vol. 8, pp. 1017–1023.

Moody, J. & Wu, L. (1997), Optimization of trading systems and portfolios, *in* Y. Abu-Mostafa, A. N. Refenes & A. S. Weigend, eds, 'Neural Networks in the Capital Markets', World Scientific, London.

Moody, J., Wu, L., Liao, Y. & Saffell, M. (1998), 'Performance functions and reinforcement learning for trading systems and portfolios', *Journal of Forecasting* 17. To appear.

Neuneier, R. (1996), Optimal asset allocation using adaptive dynamic programming, *in* D. S. Touretzky, M. C. Mozer & M. E. Hasselmo, eds, 'Advances in NIPS', Vol. 8, pp. 952–958.

Sharpe, W. F. (1966), 'Mutual fund performance', *Journal of Business* pp. 119–138.

Tesauro, G. (1989), 'Neurogammon wins the computer olympiad', *Neural Computation* 1, 321–323.

Watkins, C. J. C. H. (1989), Learning with Delayed Rewards, PhD thesis, Cambridge University, Psychology Department.

Zhang, W. & Dietterich, T. G. (1996), High-performance job-shop scheduling with a time-delay td(λ) network, *in* D. S. Touretzky, M. C. Mozer & M. E. Hasselmo, eds, 'Advances in NIPS', Vol. 8, pp. 1024–1030.