# A Sarsa based Autonomous Stock Trading Agent

Achal Augustine

The University of Texas at Austin
Department of Computer Science
Austin, TX 78712 USA
achal@cs.utexas.edu

**Abstract**

This paper describes an autonomous stock trading agent designed based on the classical reinforcement learning algorithm Sarsa λ [1]. The Sarsa based agent, named Sarsa-trader, is compared with an agent with a successful hand coded strategy known as the Reverse Strategy [4] using a simulated stock market. Experimental results on the simulated stock market show that Sarsa-trader outperforms the hand coded agent under several different market conditions.

## 1 Introduction

Stocks worth billions of dollars are traded on electronic stock markets every day. A large portion of such trades are executed by day traders. Day trading is the practice of buying and selling financial instruments within the same trading day such that all positions are closed before the market close of the trading day. Day Traders provide liquidity in the market by acting as intermediaries between investors who want to buy stocks and others who want to sell stocks. Investors generally buy or sell stocks by estimating the inherent value of the stocks while day traders follow short term price movements of the stocks to make trading decisions. With the advent of internet commerce, the design of agents that can be deployed in realistic electronic markets has become crucial to many financial institutions for successfully participating in electronic trading.

A survey of foreign exchange traders in London estimates that up to 90% of them use some form of trading rules in daily practice. The huge volume and complexity of the electronic markets makes it imperative that these trading rules are learned automatically based on trading experience. Many successful agents in the financial trading domain are based on the use of machine learning techniques [2] [3] [4] [5]. Reinforcement learning, being a sub-area of machine learning which focuses on how an agent should to take actions in an environment so as to maximize some notion of long-term reward is very well suited for learning trading rules by interacting with an electronic market. This paper describes the design and implementation of an autonomous trading agent based on a classical reinforcement learning algorithm Sarsa λ [1] with replacing eligibility traces [6]. The reinforcement learning based agent, named Sarsa-trader is compared with an agent with a successful hand coded strategy known as the Reverse strategy [3] or the Trading Against Trend strategy using a simulated stock market. Experimental results on a

simulated stock market show that Sarsa-trader outperforms the hand coded agent under several different market conditions.

The rest of the paper is organized as follows, Section 2 describes the implementation of the stock market simulation used for the experimental evaluation. Section 3 provides the design and implementation details of the Sarsa based autonomous stock trading agent. Section 4 presents the hand coded agent with the Reverse strategy. Section 5 analyses the experimental results by comparing the performance of the Sarsa-trader with the hand coded agent and Section 6 concludes the paper with discussion on strengths and limitations of the Sarsa-trader and directions for future work.

## 2 Stock Market Simulation

The stock market consists of only one stock, the price of which is varied at each trading round in a random manner depending on the market condition. The stock price is simulated using a random number generator and a base stock price. At any given time the price of the stock is generated by adding the base price of the stock with random fluctuation generated by the random number generator.

The stock market can simulate three basic market conditions a steady price figure 1, a rising price figure 2 and a falling price figure 3. These three basic market conditions can be combined in arbitrary order to generate a wide range of market conditions. Figure 4 shows such a combined market condition in which the price of the stock is increased with fluctuations for the first half of the trading day using the basic rising price generator and the stock price is decreased with fluctuations for the rest of the trading day using the basic falling price generator.

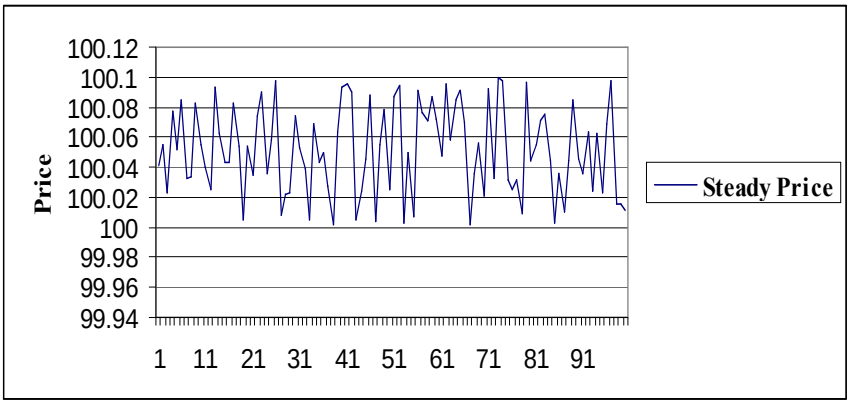


**Figure 1:** A steady price market condition with stock price fluctuating over its base price

The steady price is generated using the following rule to ensure that the fluctuations are not very large but at the same time remained random.

*Current Price = Base Price + Random Number * .001 * Base Price*

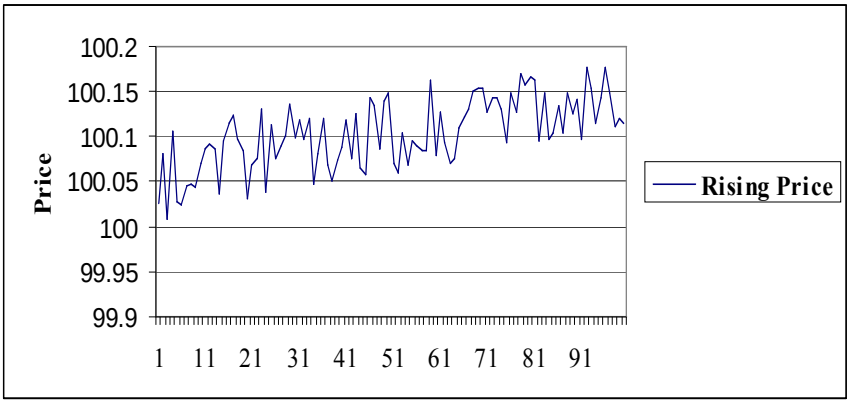


**Figure 2:** A rising price market condition with stock price rising during the day

A rising price is generated as follows to ensure that the price doesn't grow exponentially during any trading day but at the same time remained random.

```
a = Random Number;
if (a < .01) {
        Base Price = Base Price + a * .001 * Base Price;
}
return Base Price + a * .001 * Base Price;
```

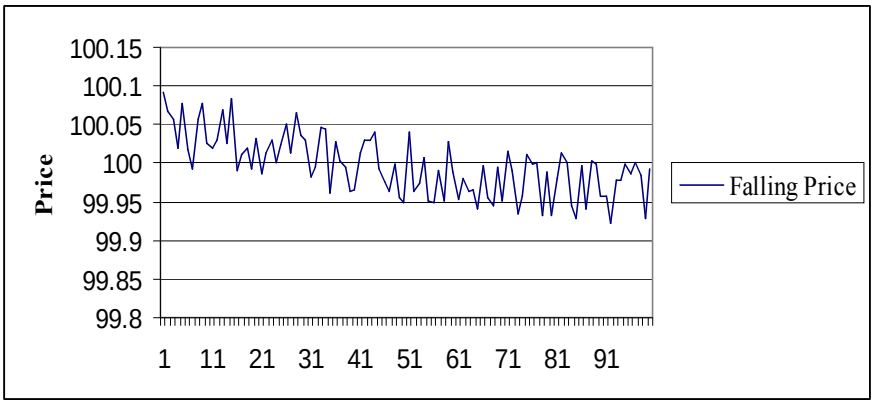


**Figure 3:** A falling price market condition with stock price falling during the day

The falling price can be generated similarly by ensuring that the price is decreased fractionally at random intervals. The price of the stock is independent of the trading behavior of the agents participating in the markets and is completely determined by the combination of market conditions used during any given trading day. This separation of stock price from the trading activity of the agents enabled the Sarsa agent to focus entirely on making the optimal trading decision without having to model the behavior of other agents and the effect of their behavior on the stock price. This model is also a close approximation of the environment in which small volume traders operate on a real electronic market where their actions have almost no direct effect on the price of the stock.

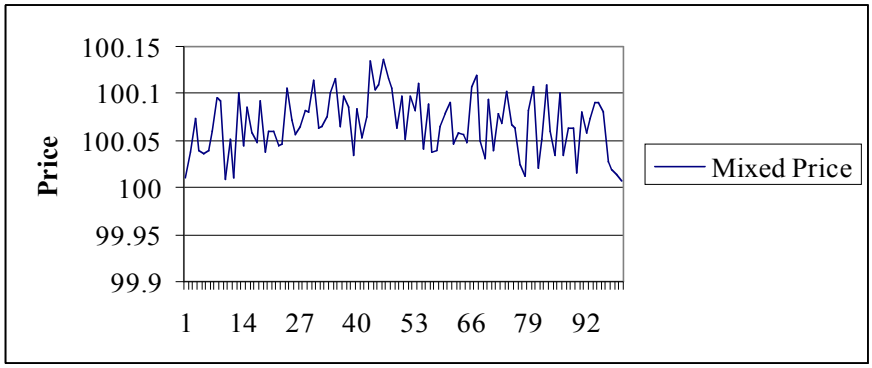


**Figure 4:** A mixed price market condition with stock price rising in the first half of the day and falling during the rest of the day

## 3 The Sarsa λ based Agent

The Sarsa [1] algorithm is an On-Policy algorithm for TD-Learning. The major difference between it and Q-Learning [7] is that the maximum reward for the next state is not necessarily used for updating the Q-values. Instead, a new action, and therefore reward, is selected using the same policy that determined the original action. The name Sarsa actually comes from the fact that the updates are done using the quintuple **Q(s, a, r, s', a').** Where: **s, a** are the original state and action, **r** is the reward observed in the following state and **s', a'** are the new state-action pair.

The Stock trading agent uses Sarsa algorithm because it incorporates the cost of exploration into the estimated Q values and therefore learns the best possible policy given the current rate of exploration while Q-Learning would learn the optimal policy when there is no exploration but may not find the policy that would provide maximum reward given the current rate of exploration. The autonomous stock trading agent operates in non-stationary environment and it is necessary to continue to explore and learn throughout its operation and Sarsa algorithm ensures that agents learns the policy that would give maximum reward possible with the current rate of exploration.

### 3.1 The Agent Specification

In its basic form, a reinforcement learning problem is given by a 4-tuple *{S,A,T,R}*, where *S* is a finite set of the environment's states; *A* is a finite set of actions available to the agent as a means of extracting an economic benefit from the environment, referred to as *reward*, and possibly of altering the environment state; $T : S \times A \rightarrow S$ is a state transition function; and $R : S \times A \rightarrow \mathrm{R}$ is a reward function. The state transition and reward functions *T* and *R* are possibly stochastic and unknown to the agent. The objective is to develop a *policy*, i.e., a mapping from environment states to actions, which maximizes the long-term return. A common definition of return, and one used in this agent, is the discounted sum of rewards: $\sum_{t=0}^{\infty} \gamma^t r_t$ where $0 < \gamma < 1$ is a discount factor and $r_t$ is the reward received at time *t*.

**State space.** The state of the environment is observed by the stock trading agent through the price of the stock. The deviation of the current price from the estimated real value of

the stock determines if the current price is a good price for buying the stock or a good price for selling the stock and therefore the state value relevant to the current problem is percentage deviation of the current price from the estimated real value of the stock. This measure is independent of the absolute value of the stock but at the same time reflect the value relevant to the agent's trading decision. Such a price deviation measure is continuous and in order to allow for generalization to unseen instances of continuous state space the state space is divided into several hundred linear tiles. The agent uses states 0 - 99 for negative deviations, 100 for no deviation and 101-200 for representing positive deviations from the estimated value.

The estimated real value of the stock is its steady state value which is calculated by using linear regression on the history of its price. Figure 5 shows one price estimation which uses last 10 prices of the stock and another estimation which uses last 100 prices of the stock. Although the former gives closer approximation to the actual price movement of the stock the latter predicts the long term stable price of the price more accurately and is used by the agent for approximating the current state of the enviornment.

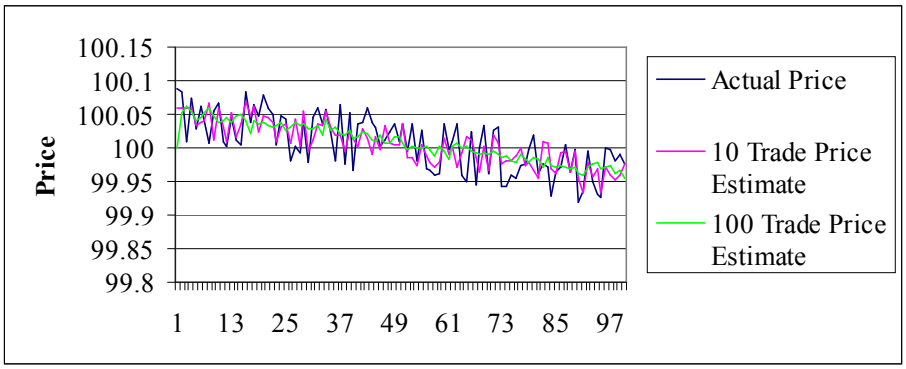


**Figure 5:** The stock price computed using last 10 trade prices is shown in pink while the green graph shows the stock price computed by using the last 100 trade prices.

The state space is limited to the price parameter because it is the only value which relevant to the trading action. The ratio of the cash and stock holding does not determine the expected return from the trade or affected the trade decision and hence was omitted. The unwinding option also does not affect the trading decision as the price of the stock is independent of the volume of the stock traded and therefore unwinding is achieved by not making any new purchases in the last session of the day. If the stock price was affected by the volume of the stock traded then unwinding would have to be completed at measured speed and it would be necessary to keep track of the time of the day as another state parameter.

**Action space.** The possible actions the stock trading agent can take at any state are sell stocks, buy stocks or hold positions and execute no trade. The stock market is designed such that the agent can sell or buy any amount of stocks at the current price generated by the stock market simulation and therefore the price of the stock sold or bought buy the agent is defined by the stock market simulator. When the agent was given an option to sell or buy varying amounts of its stock the action space became very large and the agent couldn't converge to any good policy even after several million trades and therefore the

Sarsa-trader trades stocks worth a fixed amount money at each transaction when it wants to buy or sell. Action values 0, 1, 2 are used for no trade, buy and sell actions respectively. The agent's trade volume can be maximized by setting this value to a very large number. As we shall later the agent uses the state space to manage risk without relying on the quantity of the stocks traded.

**Reward function.** The goal of the trading agent is to maximize the profits it accumulates during a trading day. To make this possible the agent was given a positive reward after each action equal to the percentage increase in the total asset of the agent after the action. The total asset at any time is the sum of cash and stock holdings where the stocks are valued at their estimated real values. This ensures that the agent receives a positive reward for buying the stocks at a price lower than their estimated price and also for selling them at a price higher than the estimated price of the stock.

**Transition function.** The transition from current state to the next state was determined by the change in the price of the stocks and therefore was modeled by the stock market simulator. After each action the agent would ask the simulator for the next price and use that as an observation to estimate the next state.

*Initialize Q(s, a) arbitrarily*
*Repeat (for each episode):*
*e(s, a) = 0 for all s, a.*
*Initialize s*
*Choose a from s using policy derived from Q*
*Repeat (for each step of episode):*
*Take action a, observe reward r, s'*
*Choose a' from s' using policy derived from Q*
$\delta = r + \gamma Q(s', a') - Q(s, a)$
*e(s, a) = 1 (replacing traces)*
*For all s, a:*
$Q(s, a) = Q(s, a) + \alpha\delta e(s, a)$
$e(s, a) = \gamma\lambda e(s, a)$
*s = s' ; a = a';*
*until s is terminal*

**Figure 6:** Sarsa(λ) with replacing traces

**Parameter values.** The agent used Sarsa(λ) shown in figure 6 with replacing traces[6]. Replacing traces are used because due the fluctuating nature of the stock prices several states are visited very frequently before the end of the episode even though revisiting the states was not necessary for episode to end. Replacing traces ensured that repeated visits to the state don't increase their eligibility above 1. After exploring different parameter values the following values $\gamma = .9$, $\lambda = .9$, $\alpha = .8$ and $\varepsilon = .01$ were found to give best results for the agent.

### 3.2 Learning Curve

Figure 7 shows a typical learning curve for Sarsa-trader on a steady fluctuating price market condition when $1000 was set as unit of for each trade action and was trained for different durations. The trader started off with random actions and as it learned the optimal policy it improved the profit per trade very fast but after 200K trades the improvement in profit per trade became very slow and finally turned steady over 400K trades.

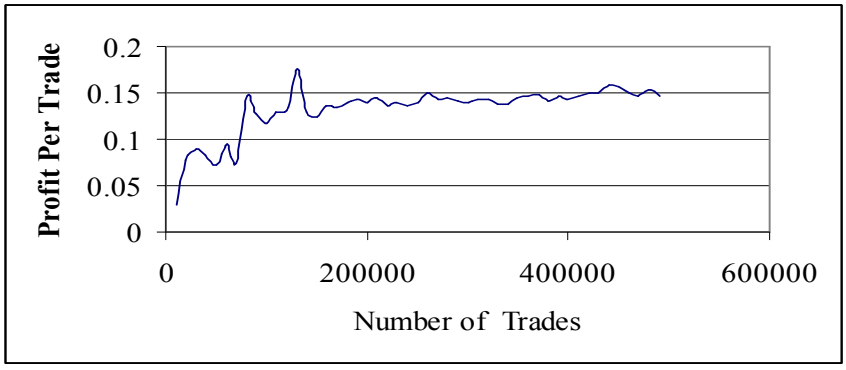


**Figure 7:** Learning curve of Sarsa-trader

### 3.3 Learned Policy

Figure 8 shows a typical policy learned by Sarsa-trader after training it over 200K trades. The values 0, 1 and 2 stands for no trade, buy and sell actions. The state numbers 0-99 represent a current price lower than estimated price, 100 represent a state in which the current price is equal to the estimated price and 101-200 represent states where current price is higher than the estimated price. The agent correctly learns to buy at lower price and sell at higher prices.

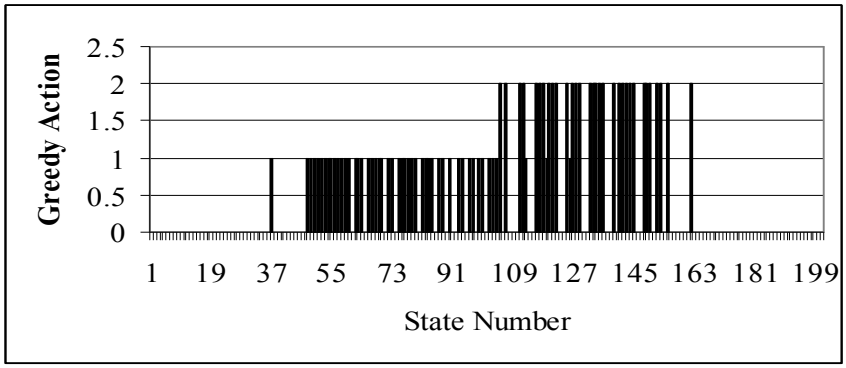


**Figure 8:** A typical policy learned by Sarsa-trader after 200K trades.

Careful inspection of the graph shows that there are lots of white spaces around state hundred and many thick black lines farther away from state 100 indicating that the agent learned that its safer and more profitable to buy at states further away from estimated price and to avoid trades at states close to the state 100 which represent the equilibrium

price. By choosing which states to trade the stock at, the agent can balance its risk and profitability. The states 0-50 and states 160-200 have a "no trade" action value because these states are very far from the equilibrium price of the stock and are rarely visited during the training period. This shows that the agent is flexible enough to learn the current fluctuations of the market and start trading in those states which are visited during its operations.

## 4 The Reverse Strategy

Two common hand coded trading strategies are Trend Following strategy and the Reverse Strategy. In the Trend Following strategy the trader buys stocks when the price is rising and sells when it is falling. The Reverse strategy does exactly the opposite: it sells when the price is rising and buys when it is falling. Although this strategy appears counter-intuitive, it works by exploiting the price micro-movements (small price spikes in both direction), which make up the evolution of the price of a stock during each trading day.

Initial testing of the Trend Following strategy revealed that it lost more money than it made and therefore it was abandoned for the counter intuitive Reverse strategy which consistently made profits in all markets conditions used for the experiments. The Reverse strategy was implemented as follows. The trend of the price movement was monitored using the slope of the regression line of the last 100 trade prices and when the slope was positive the agent sold the fixed amount of shares and when the slope became negative the agent bought the same fixed amount of shares.

## 5 Experimental Evaluation

The Sarsa-trader and the hand coded agent with Reverse strategy in a series of experiments under several different market conditions. In each experiment the Sarsa-trader was first trained for 200K trading rounds and the performance of the trained agent was then compared against the hand coded agent during 20K trading rounds. Each result presented is an average over 10 runs of the same experiment. Each trader was given and cash asset of 10K at the beginning of the experiment and the graphs in figures 9, 10, 11 and 12 shows the growth in total asset during the experiments.

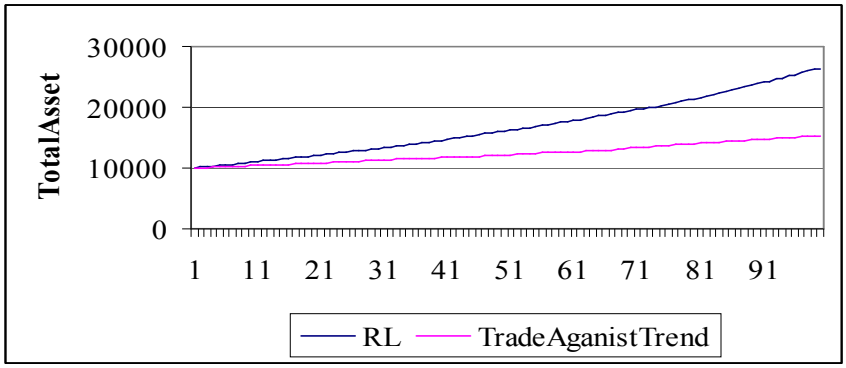


**Figure 9:** Performance of the agents in a steady price market condition

Figures 9, 10, 11 and 12 compares the performance of Sarsa-trader with the hand coded agent during steady price, rising price, falling price and mixed price market conditions. The graphs show that Sarsa-agent outperforms the hand coded agent in each of the market conditions with a significant margin. The graphs also indicate that profits earned by both the agents are steadily rising over the trading period without big fluctuations. This is partly because the price movements used the market stimulator are minor and doesn't include any drastic rise or fall in prices.

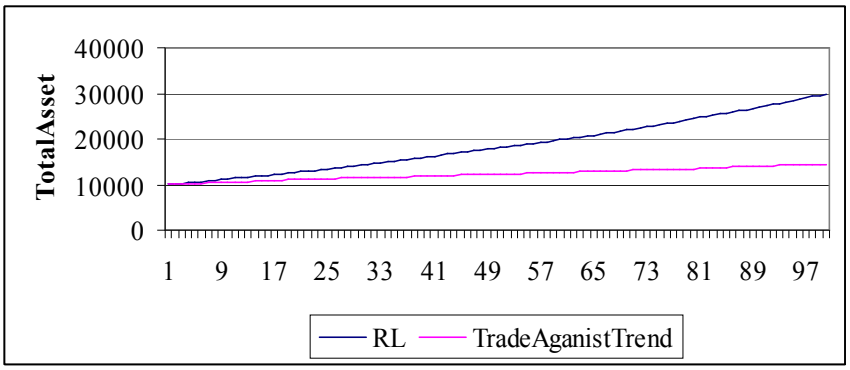


**Figure 10:** Performance of the agents in a rising price market condition

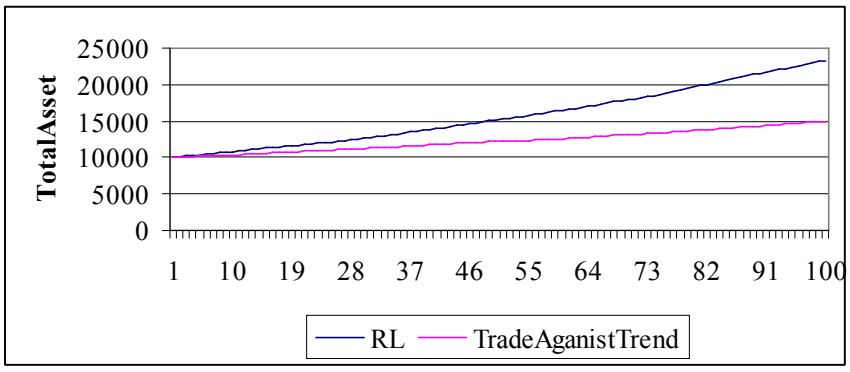


**Figure 11:** Performance of the agents in a falling price market condition

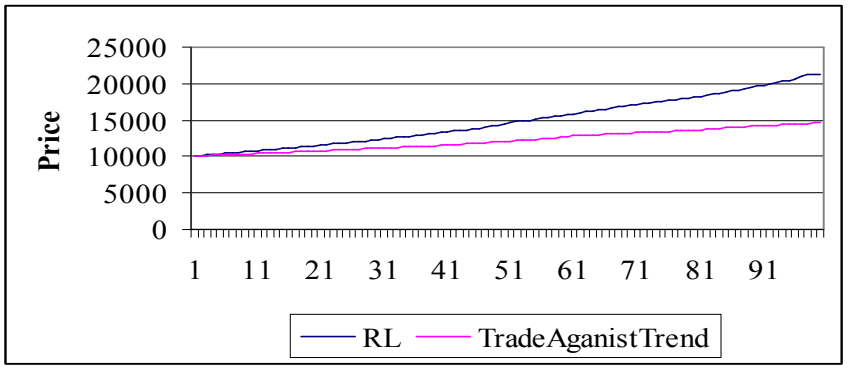


**Figure 12:** Performance of the agents in a mixed price market condition

Figure 13 compares the performance of the Sarsa-trader in different market conditions. The Sarsa-trader shows best performance in a rising market condition as price rise enables it to make profits even when some of the buy options are executed at less than optimal prices. Similarly a falling market condition makes its difficult to make profits as purchases made at lower prices should be sold immediately at higher prices before the price goes down. I posit that the Sarsa-trader performs worst in a mixed market because the linear regression based estimation used for predicting the prices has a maximum likelihood of failing a mixed market condition where the price rises for certain period and then falls for the remaining part of the day.

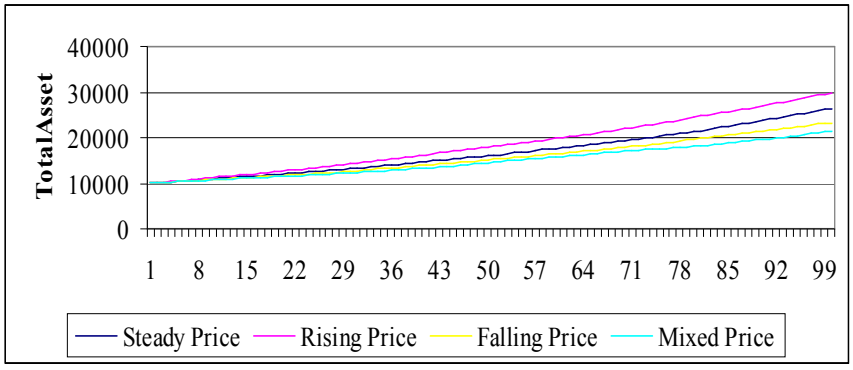


**Figure 13:** Performance of the RL agent in various market conditions

The Sara-trader and Reverse strategy agents did not perform very well in few experimental market conditions with steep changes in the stock prices. This is one of the limitations of the linear regression price estimation tool used the by the agent which fails to anticipate any such changes. Unless such steep changes recur in a certain pattern it would be hard for any method of price prediction to accurately predict such a change.

The agent also does not use any order book information to make price estimations. If order book information is available to the agent then it would be able to anticipate the price changes before it becomes visible in the market but once again the complex prediction problem would be predicting sharp changes in the order book entry due to influence from the outside economy. Predicting such sharp changes in the order book entries would then be a bottle neck for performance of the order book based agent.

## 6 Conclusion and Future Work

This paper describes a Sarsa λ based autonomous stock-trading agent which outperforms hand coded agent under several simulated market conditions. Autonomous trading agents is an active research area with huge economic significance and also very well suited for applying reinforcement learning algorithms which enables the agents to learn polices for a stochastic and dynamic environment. The Sarsa-trader presented in this paper has a limited action space and further research needs to be done to incorporate more actions into the agent and at the same time converge to successful policies quickly. Another area of improvement is to incorporate the order book information and live news releases for price prediction which would bridge the gap between the information available to human traders and the electronic counterparts.

## 7 Acknowledgements

I thank Professor Peter Stone for his guidance during the project and also actively sharing his expertise in reinforcement learning theory through his lectures, assignments, reviews and discussions.

## References

[1] R.S. Sutton and A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 1998.

[2] Sherstov, A. and Stone, P. 2004. Three Automated Stock-Trading Agents: A Comparative Study. In AAMAS 2004 Workshop on Agent Mediated Electronic Commerce VI.

[3] Feng, Y., Yu, R. and Stone, P. 2004. Two Stock-Trading Agents: Market Making and Technical Analysis. In Agent Mediated Electronic Commerce V, Lecture Notes in Artificial Intelligence, pp. 18-36, Springer Verlag.

[4] Moody, J. and Saffell, M. 2001. Learning to trade via direct reinforcement. IEEE Trans. Neural Networks 12(4):875-889.

[5] Dempster, M.A.H. and Jones, C.M. 2001. A real-time adaptive trading system using genetic programming. Quantitative Finance 1:397-413.

[6] Singh, S.P., Sutton, R.S. (1996). Reinforcement learning with replacing eligibility traces. Machine Learning 22(1/2/3):123-158.

[7] Watkins, C. J. C. H. and Dayan, P. 1992. Q-learning. Machine Learning 8(3):279–292.