Risk, Reward & Reinforcement

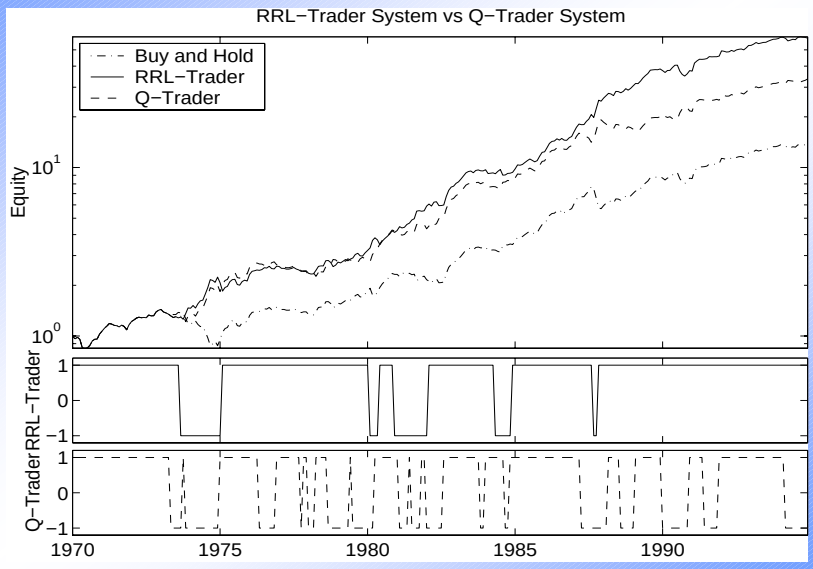
John Moody
Department of Computer Science
OGI School of Science & Engineering
Oregon Health & Science University

Machine Learning, Statistics & Discovery
AMS Workshop, Snowbird Utah, June 25, 2003

Goals of This Talk

- Introduce Reinforcement Learning
- Present Direct Reinforcement
  - Contrast w/ Value Function RL Methods
  - Causal, Non-Markovian, Partially-Observed
- Describe Risk-Averse Reinforcement
- Demonstrate application to
  - A Competitive Game
  - Trading & Asset Allocation

Preview: S&P-500 / T-Bill Asset Allocation



What is Reinforcement Learning?

RL Considers:

- A Goal-Directed Agent
- interacting with an Uncertain Environment
- that attempts to maximize Reward / Utility

RL is a Dynamic Learning Paradigm:

- Trial & Error Discovery of Strategy
- Actions result in Reinforcement

Time Plays a Critical Role:

- Rewards depend on sequences of actions
- Rewards may be delayed or received over time

Reinforcement vs. Supervised Learning

Sound Bites:
- "Learning from Examples" (SL)
- "Learning by Trial and Error" (RL)

Distinctions:
- Static (SL) vs. Dynamic (RL)
- Feedback: "Instructive" (SL) vs. "Evaluative" (RL)
- SL usually ignores larger problem (goals, utility)
- RL agents take action, may influence environment

Characteristics of RL Applications:
- Dynamical model of world is not known
- Labeled examples expensive or unavailable
- Temporal credit assignment problem

Origins of Reinforcement Learning

- Psychology and Animal Behavior
  - Thorndike's "Law of Effect", *Animal Intelligence* (1911)
  - Skinner's "Operant Conditioning", *The Behavior of Organisms* (1938)
  - "Trial and Error" Learning & "Reinforcement" Theories
- Computational Intelligence
  - Turing, "Computing Machinery and Intelligence" (1950)
  - Minsky, "Neural-Analog Reinforcement" (1954)
  - * Farley & Clark's Policy Gradient Learner (1954)
  - Samuel's Checkers Program (1959)
- Operations Research & Control Engineering
  - Bellman, *Dynamic Programming* (1957)

Modern RL

- Value Function Methods
  - Sutton's "Temporal Difference $TD(\lambda)$ " (1988)
  - Watkins' "Q-Learning" (1989)
  - Tesauro's "TD-Gammon" (1994)
- Actor-Critic Methods
  - Barto, Sutton & Anderson (1983)
  - Werbos' Taxonomy (1992)
  - Konda & Tsitsiklis, NIPS*1999 (2000)
- Direct Reinforcement: Policy Gradient & Policy Search
  - Williams' "REINFORCE" (1988, 1992)
  - Moody, et al: "RRL" and Finance (1996 -- present)
  - Baxter & Bartlett: "Direct Gradient-Based RL" (1999)
  - Ng & Jordan: "Pegasus: Policy Search" (2000)
  - NIPS*2000 Workshop
    "Learn the Policy or Learn the Value-Function?"

Is a paradigm shift occurring?

Dynamic Programming

Discrete Time Stochastic Control

Markov Decision Process (MDP): Agent

operating with discrete states x

taking actions a

receiving rewards R

MDP System Model:

Transition probability $P_{xy}(a)$ for actions $a_t : x_t \rightarrow y_{t+1}$

Distribution $P_R(x,a)$ of rewards $R(x,a)$

Value Function and Policy:

$$V^{\pi}(x) = E\left[\sum_{t=0}^{\infty} \gamma^t R[x_t, \pi(x_t)] \mid x_0 = x\right]$$

$$\pi = \text{"policy"} = \{a_t = \pi(x_t)\ ;\ t = 0, \ldots, \infty\}\ ,\ P_{\pi}(x,a)$$

Goal: find optimal V and hence π

Dynamic Programming, cont

Bellman's Recursion Equation:

$$V^{\pi}(x) = \sum_{a} P_{\pi}(x,a) \sum_{y} P_{xy}(a) \left\{ E(R(x,a)) + \gamma V^{\pi}(y) \right\}$$

The optimal policy satisfies:

$$V^{*}(x) = \max_{a} \left[E(R(x,a)) + \gamma \sum_{y} P_{xy}(a) V^{*}(y) \right]$$

Optimal Policy π^{*} is defined implicitly:

$$V^{*}(x) = \max_{\pi} V^{\pi}(x) \quad \text{and} \quad \pi^{*}(x) = \arg\max_{\pi} V^{\pi}(x)$$

Finding π^{*} requires also determining V^{*}

knowing the System Model:

$$P_{xy}(a) \ , \ P_{R}(x,a) \ .$$

and computing expectations!

Reinforcement Learning: Beyond Dynamic Programming

RL Algorithms offer approximate solutions to:

- Dynamic Programming Problems
- Stochastic Control Problems

RL Algorithms:

- Do not require a model of the system
  - Learn via simulation or live trial & error experience
- Avoid Bellman's "curse of dimensionality"
  - By using "function approximation" to smooth state space
- Learn on-line:
  - Stochastic optimization
  - "Exploration" vs. "Exploitation"

Direct Reinforcement

Risk & Reward

Q-Learning: Adaptive DP

Q-Function: state $\otimes$ action $\rightarrow$ value

$$V^*(x) = \max_a \left[Q^*(x,a) \right]$$

$$Q^*(x,a) = E(R(x,a)) + \gamma \sum_y P_{xy}(a) \max_b \left[Q^*(y,b) \right]$$

Q-Learning (Watkins 1989): estimates $\hat{Q}^*$ iteratively via *simulation* without a system model $P_{xy}(a),\ P_R(x,a)$:

$$\Delta \hat{Q}(x,a) = \eta \left[R(x,a) + \gamma \max_b \left[\hat{Q}(y,b) \right] - \hat{Q}(x,a) \right]$$

Optimal Policy $\hat{\pi}^*$ is defined <u>implicitly!</u>

$$\hat{a}^*(x) = \arg\max_b \left[\hat{Q}^*(x,b) \right]$$

<u>Problems</u>: representations & robustness

Direct Reinforcement

Represent / learn the policy: observation $\rightarrow$ action directly <u>without</u> learning a value function!

Motivation:

- Simpler, *more natural* problem representations
- Only the local gradient of the value function matters
  - Local estimates of performance often available
- Solve non-Markovian problems
- Find solutions with problems with only "partial observability"
- Seek a "good" policy, not an "optimal" policy

RRL (Recurrent Reinforcement Learning):
a "policy gradient" algorithm

Learning via Direct Reinforcement

Direct Reinforcement

Risk & Reward

DR Agent:

- "Partially Observes": information I_t, not full state S_t
- "Non-Markovian": Recurrent policy $F_t = F(\theta_t; F_{t-1}, I_t)$
- Takes action, Receives reward $R_t\left(F_t, F_{t-1}; S_t\right)$
- Causal performance function (Generally path-dependent) $U(R_t, R_{t-1}, ..., R_1)$
- Learn policy $F(\theta_t; F_{t-1}, I_t)$ by varying θ_t

GOAL: Maximize performance U_T

or marginal performance $D_t \equiv \Delta U_t = U_t - U_{t-1}$

Recurrent Reinforcement Learning (RRL)

Direct Reinforcement

Risk & Reward

Deterministic gradient (batch):

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

with recursion: $\frac{dF_t}{d\theta} = \frac{\partial F_t}{\partial \theta} + \frac{dF_t}{dF_{t-1}}\frac{dF_{t-1}}{d\theta}$

Stochastic gradient (on-line):

$$\frac{dU_t(\theta)}{d\theta_t} \approx \frac{dU_t}{dR_t}\left\{\frac{dR_t}{dF_t}\frac{dF_t}{d\theta_t} + \frac{dR_t}{dF_{t-1}}\frac{dF_{t-1}}{d\theta_{t-1}}\right\}$$

stochastic recursion: $\frac{dF_t}{d\theta_t} \approx \frac{\partial F_t}{\partial \theta_t} + \frac{dF_t}{dF_{t-1}}\frac{dF_{t-1}}{d\theta_{t-1}}$

Stochastic parameter update (on-line):

$$\Delta\theta_t = \rho \frac{dU_t(\theta_t)}{d\theta_t}$$

Constant ρ : adaptive learning. Declining ρ : stochastic approx.

RL Algorithms Compared

Direct Reinforcement

Risk & Reward

Q-Learning

- Learn Q-Function
- Value=Q(Action)
- Q: state $\otimes$ action $\rightarrow$ value
- Action $F = \underset{b}{\operatorname{argmax}}\, N(x, b, \theta)$

Direct Reinforcement

- Learn Policy F
- Use local performance est.
- F: observations $\rightarrow$ action
- Action $F = N(x, \theta)$

Properties

- Bellman's Equation: A-Causal
- MDP Assumption
- Complex representations
- Curse of Dimensionality
- Computations expensive
- Policies often unstable

Properties

- Causal: Forward in Time
- Recurrent, Partially Observable
- Enables simpler reps
- Reduces Curse of Dimensionality
- More efficient in practice
- Yields more robust policies

The Oracle Problem

(How to Turn an Easy Problem into a Harder Problem)

Direct Reinforcement

Risk & Reward

Problem Description

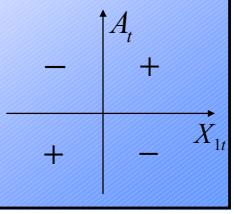
- Binary actions: $A_t \in \{-1, +1\}$
- Rewards for {incorrect, correct} : $R_t \in \{-1, +1\}$
- Vector of inputs: $\vec{X}_t$
  - Oracle input: X_1 (tells the agent the correct action A_t)
  - Noise inputs: $X_2, X_3, \ldots X_N$ (boolean random variables)

Complexity of DR Agent

- Perceptron Policy Learner: $A_t = \operatorname{sign}(\vec{W} \cdot \vec{X}_t)$
  Optimal policy w/ single threshold: $A_t = \operatorname{sign}(X_{1t})$

Complexity of Q Agent

- Optimal policy $Q(A_t, X_{1t})$ is XOR function:
  Representation requires at least two thresholds.

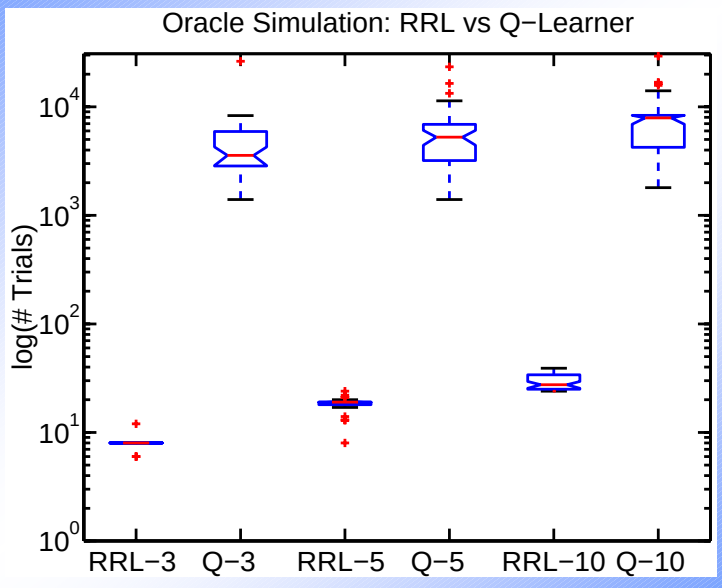


The Oracle Simulation

- Measure how many trials required to learn representation
- Convergence criteria:
  - RRL: Correct policy for all possible input vectors
  - Q-Learner: MSE < 0.01 for all possible input vectors
  - Maximum of 30,000 trials per run
- Repeated runs for multiple learning rates
  - Choose learning rate with quickest convergence on average
- 50 random initializations; N = 1,2,3,4,5,10 inputs

	RRL	Q-Learner
- Min # trials:	1	1150
- Max #trials:	39	29350
- # Runs Non-Converged:	0	15

The Oracle: Simulation Results



Direct Reinforcement
Risk & Reward

RoShamBo (RPS)

Rules : Rock beats Scissors.

Paper beats Rock.

Scissors beats Paper.

" The rules are simple, but the game itself is as complex as the mind of your opponent."

(www.worldrps.com)

Character of human throws:

- Rock: commonly perceived as the most aggressive throw
- Paper: considered the most subtle throw
- Scissors: often perceived as clever or crafty

Competitions: World Championship, Computer Olympiad

RPS Player Representation

Softmax representation for probability(action):

$$F^a() = \frac{\exp[f^a()]}{\sum_{b=1}^{m} \exp[f^a()]} \text{ for } a = 1,....,m$$

Inputs & Weights:

Opponent's two previous moves: $\vec{W} \bullet \vec{B}$

Player's two previous moves (recurrent): $\vec{V} \bullet \vec{A}$

Learning Algorithm:

Stochastic Direct Reinforcement (SDR)

A generalization of RRL

Simulated Competition

Direct Reinforcement

Risk & Reward

' Human Champions' Dataset:

→ The "Great Eight" Gambits are the eight most widely used by recent champions.

1. Avalanche (RRR) 2. Bureaucrat (PPP) 3. Crescendo (PSR) 4. Dénouement (RSP)
5. Fistfull o' Dollars (RPP) 6. Paper Dolls (PSS) 7. Scissor Sandwich (PSP) 8. Toolbox (SSS)

→ "Great Eight Plus Four" yields equal probability of R,P,S.

Training a 'Dynamic Opponent' using SDR

→ Learn by playing against "Great Eight Plus Four"

Learn to beat the opponent using Recurrent SDR

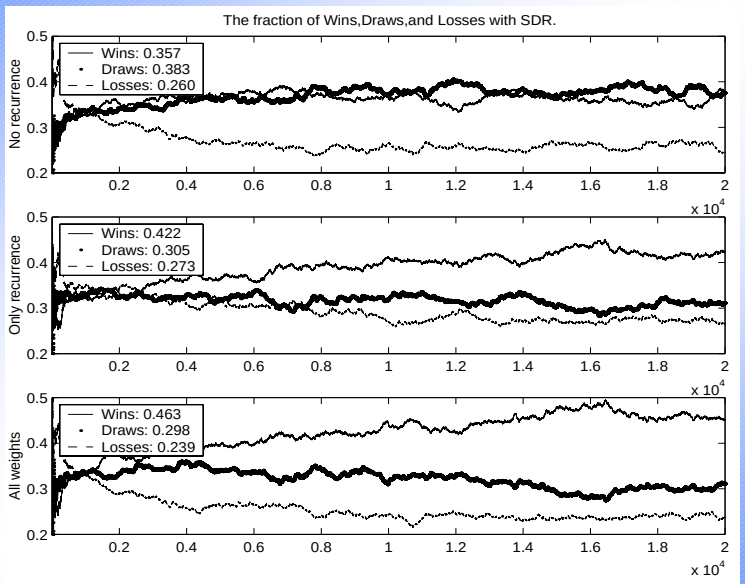
→ Learn by playing against the Dynamic Opponent

Opponent tries to predict Player's move, and vice versa

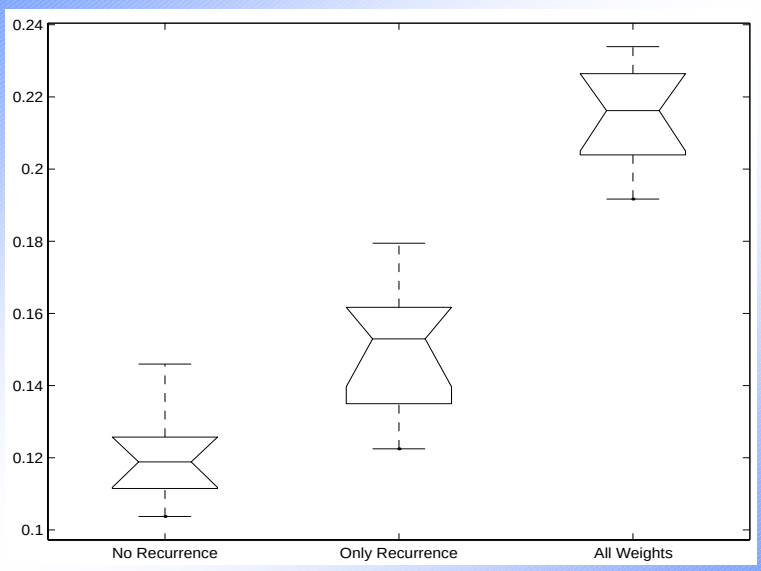
Three Model Players: Learning Curves

Direct Reinforcement

Risk & Reward



[Wins – Losses] for Three Model Players

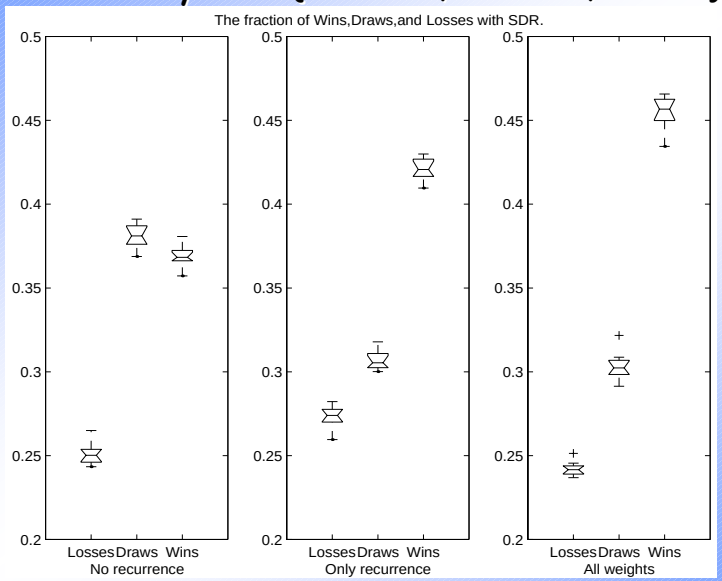


Summary Stats: 10 Simulations

Direct Reinforcement

Risk & Reward

Three Players: {Losses, Draws, Wins}



Summary Stats: 10 Simulations

Direct Reinforcement

Risk & Reward

Risk Averse Reinforcement

Sources of Uncertainty in RL:

- Stochastic Rewards
- Stochastic Environment
- Stochastic Policies

Standard RL framework takes expectations of above!

- "Optimal Policies" are only "optimal" in expectation

Partial Observability: Another source of uncertainty

Risk: Distributions of trajectories, rewards, outcomes

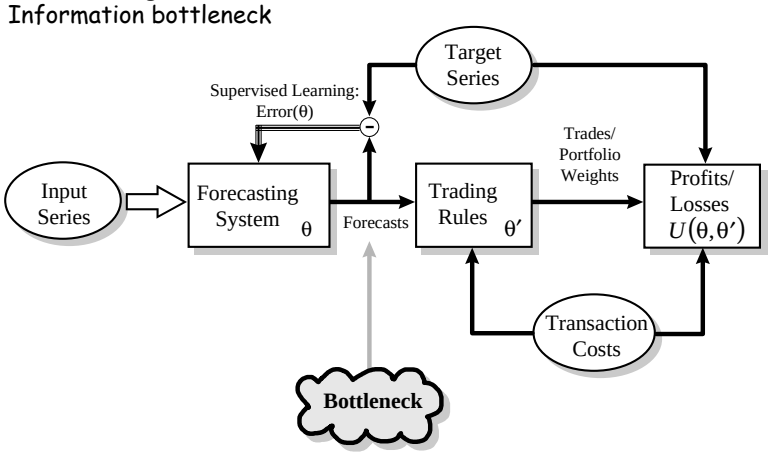
- Probability of Poor Performance
- Risk of Ruin

Asset Management as a Microcosm
for Reinforcement Learning Research

- Simulations
  - Bang-Bang Control Problem
    Simple Buy / Sell Decisions ("Long" / "Short" Positions)
  - Transaction Costs → Recurrent, Non-Markovian Rep.
- Uncertain Environment:
  - Details of markets / economy are Unobservable
  - Prices / fundamentals / economic data are Very Noisy
  - Markets react quickly to Unpredictable News / Events
- Challenging Problem:
  - Efficient Markets Theory: You Can't Beat the Market.
  - Competitive Game: Trading opportunities will be discovered, exploited and eliminated by others.
  - Prediction Accuracy Limited: "$1/2 + \varepsilon$"

Trading based on Forecasts

Four limitations:
- Two sets of parameters
- Forecast error is not Utility
- Forecaster ignores transaction costs
- Information bottleneck

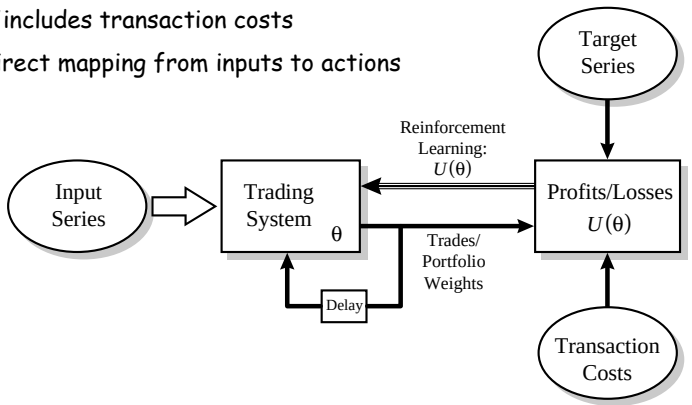


Direct Reinforcement

Risk & Reward

Learning to Trade via Direct Reinforcement

Four advantages:
- One set of parameters
- A single utility function
- U includes transaction costs
- Direct mapping from inputs to actions



Direct Reinforcement

Risk & Reward

Structure of Traders

- Single Asset
  - Price series z_t
  - Return series $r_t = z_t - z_{t-1}$ simple returns
    or $r_t = \frac{z_t}{z_{t-1}} - 1$ rates of return
- Traders
  - Discrete position size $F_t \in \{-1, 0, 1\}$
  - Recurrent policy $F_t = F(\theta_t; F_{t-1}, I_t)$
- Information Set: $I_t = \{z_t, z_{t-1}, z_{t-2}, \ldots; y_t, y_{t-1}, y_{t-2}, \ldots\}$
  - Full system state is not known

Returns, Profit & Wealth for Traders

- Simple Trading Returns and Profit:

$$R_t = F_{t-1} r_t - \delta |F_t - F_{t-1}|$$

$$P_t = \mu \sum_{t=1}^{T} R_t$$

- Compounded Trading Returns and Wealth:

$$R_t = (1 + F_{t-1} r_t) \cdot (1 - \delta |F_t - F_{t-1}|) - 1$$

$$W_T = W_0 \prod_{t=1}^{T} \{1 + R_t\}$$

- Transaction Costs: represented by δ .
- Risk Free Rate:
  suppressed for simplicity ($r_t^f = 0$).
- Note: Market impact: δ = function(trade size)

Balancing Reward with Risk:
Financial Performance Measures

Performance Functions:

- Path independent: $U_t = U(W_t)$
  (Standard Utility Functions)
- Path dependent: $U_t = U(W_t, W_{t-1}, ..., W_1, W_0)$
- In general: $U_t = U(R_t, R_{t-1}, ..., W_0)$

Performance Ratios:

- Sharpe Ratio: $\dfrac{\text{Average}(R_t)}{\text{Standard Deviation}(R_t)}$
- Downside Deviation Ratio: $\dfrac{\text{Average}(R_t)}{\text{Downside Deviation}(R_t)}$
- Sterling Ratio: $\dfrac{\text{Average}(R_t)}{\text{Draw-Down}(R_t)}$

For Learning:

- Per-Period Returns: R_t
- Marginal Performance: $D_t \equiv \Delta U_t = U_t - U_{t-1}$

Maximizing the Sharpe Ratio

Sharpe Ratio:

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

Exponential Moving Average **Sharpe Ratio**:

$$S_\eta(t) = \frac{A_t}{K(B_t - A_t^2)^{1/2}}$$

with time scale η^{-1} and

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

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

$$K = \left(\frac{1-\eta/2}{1-\eta}\right)^{1/2}$$

Motivation: EMA Sharpe ratio

- emphasizes recent patterns;
- is causal & can be updated incrementally.

Differential Sharpe Ratio for Adaptive Optimization

Expand $S_\eta(t)$ to first order in :

$$S_\eta(t) \approx S_\eta(t-1) + \eta \frac{dS_\eta(t)}{d\eta}|_{\eta=0} + O(\eta^2).$$

Define Differential Sharpe Ratio as:

$$D_\eta(t) \equiv \frac{dS_\eta(t)}{d\eta} = \frac{B_{t-1}\Delta A_t - \frac{1}{2}A_{t-1}\Delta B_t}{(B_{t-1} - A_{t-1}^2)^{3/2}}$$

where

$$\Delta A_t = R_t - A_{t-1}$$

$$\Delta B_t = R_t^2 - B_{t-1}$$

Motivation for DSR:

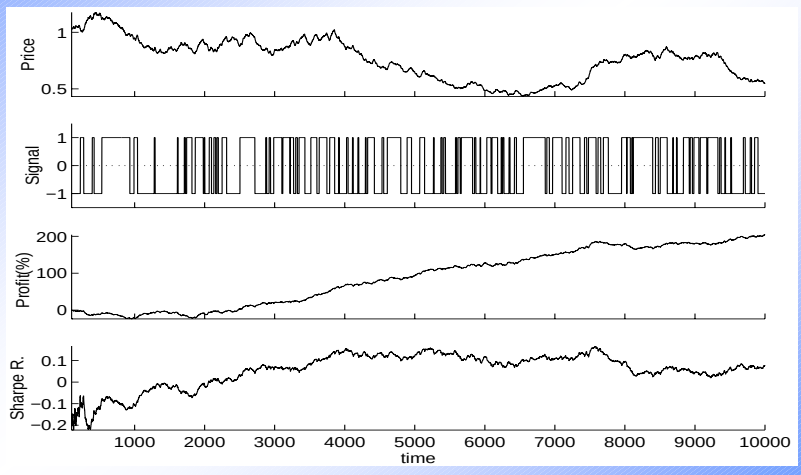
- isolates contribution of R_t to U_t ("marginal utility");
- provides interpretability;
- adapts to changing market conditions;
- facilitates efficient on-line learning (stochastic optimization).

Direct Reinforcement

Risk & Reward

Long / Short Trader Simulation

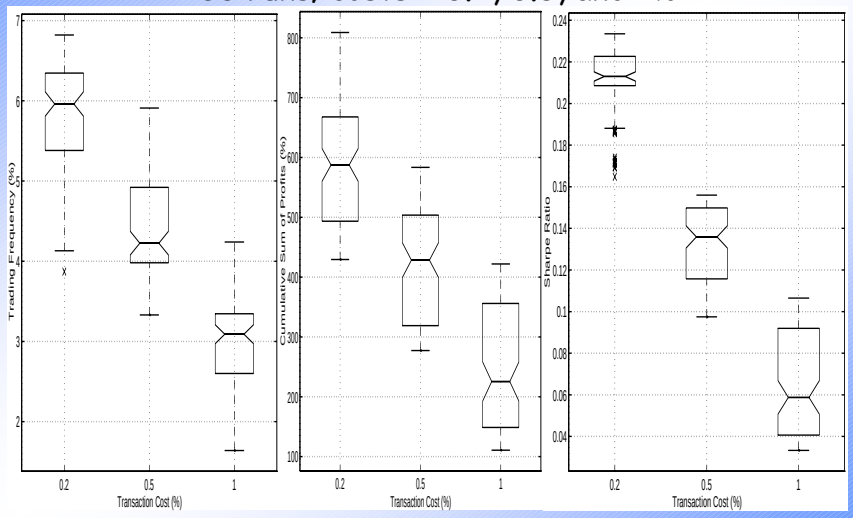
- Learns from scratch and on-line
- Moving average Sharpe Ratio with η = 0.01



Trader Simulation (summary stats)

Effects of transaction costs on performance

100 runs; costs = 0.2, 0.5, and 1%

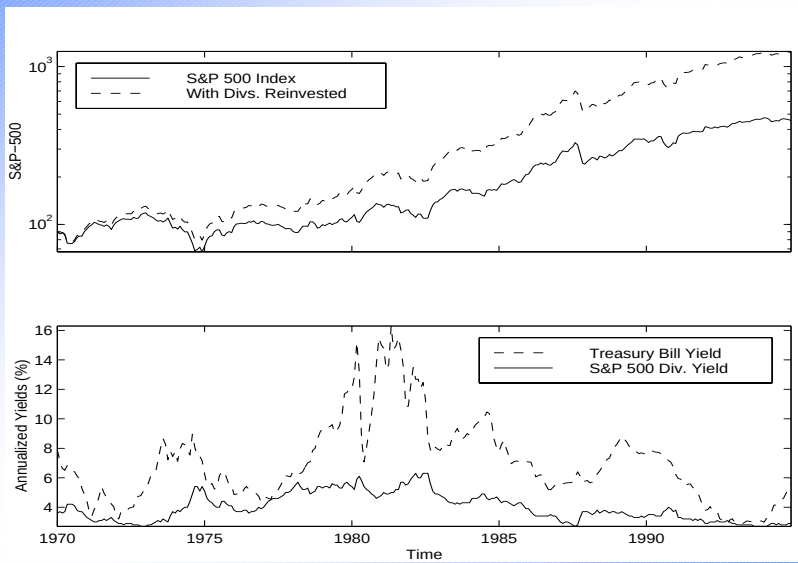


Direct Reinforcement

Risk & Reward

Asset Allocation Example

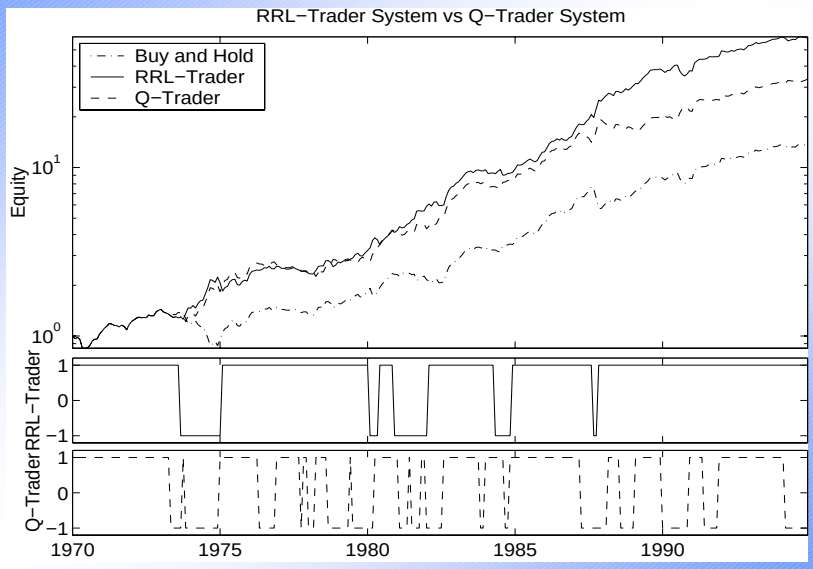
S&P-500 Index and 3-Month T-Bill



Direct Reinforcement

Risk & Reward

Maximizing the Differential Sharpe Ratio: S&P-500 / T-Bill Asset Allocation



Direct Reinforcement

Risk & Reward

Gaining Economic Insights by Opening Up the "Black Box"

Which of the 85 economic / financial input series for the S&P-500 / T-Bill trader are most important?

Relative sensitivity of input i:

$$S_i = \frac{\left|\frac{dF}{dx_i}\right|}{\max_j \left|\frac{dF}{dx_j}\right|}$$

Each year, average the sensitivity for each input

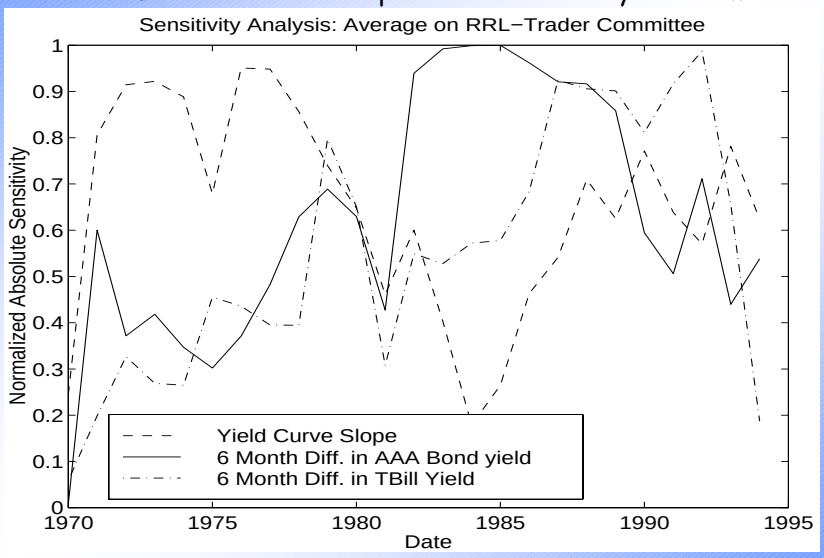
Note:
Sensitivity Analysis is straightforward for Direct RL, but not for Q-Learning

Direct Reinforcement

Risk & Reward

S&P-500: Three Most Important Variables

85 series: Learned relationships are nonstationary over time



Minimizing Downside Risk

Downside Deviation:

$$DD_T = \left[\frac{1}{T} \sum_{t=1}^{T} \min\{R_t - \theta_t, 0\}^2 \right]^{1/2}$$

N^{th}- Degree Downside Deviation:

$$DD_T^{(n)} = \left[LPM_T(n) \right]^{1/n}$$

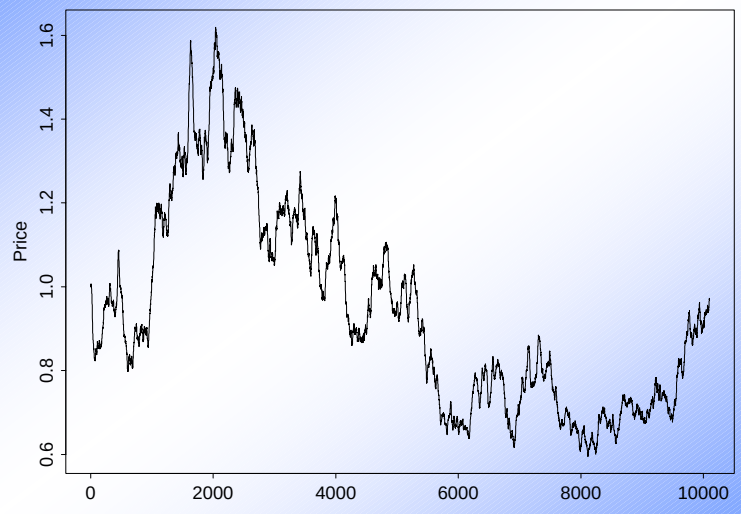
N^{th} Lower Partial Moment:

$$LPM_T(n) = \frac{1}{T} \sum_{t=1}^{T} \max\{\theta_t - R_t, 0\}^n$$

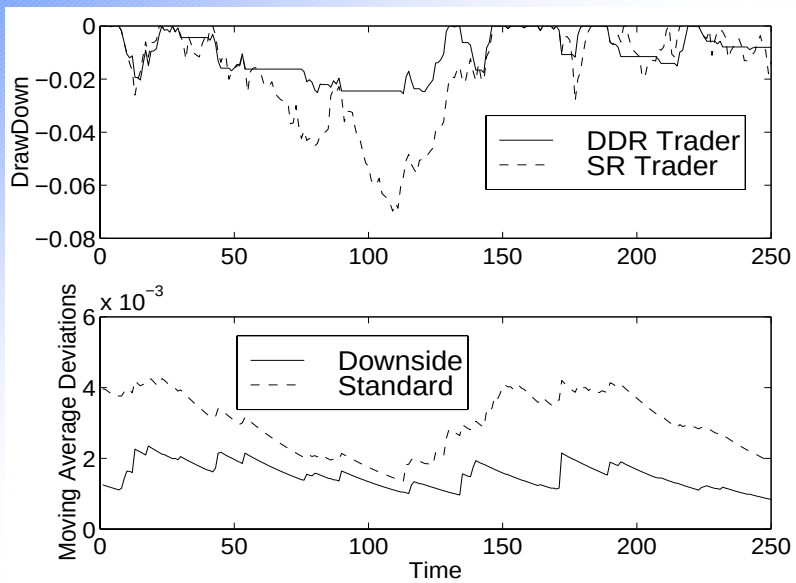
Downside Deviation Ratio:

$$DDR_T = \frac{\text{Average}(R_t)}{\text{Downside Deviation}(R_t)}$$

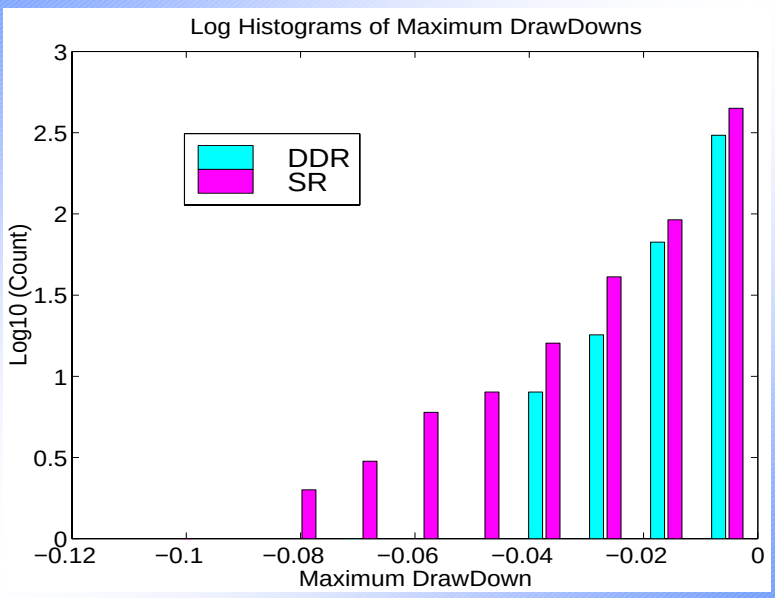
Artificial Price Series



Performance Results (cont'd)

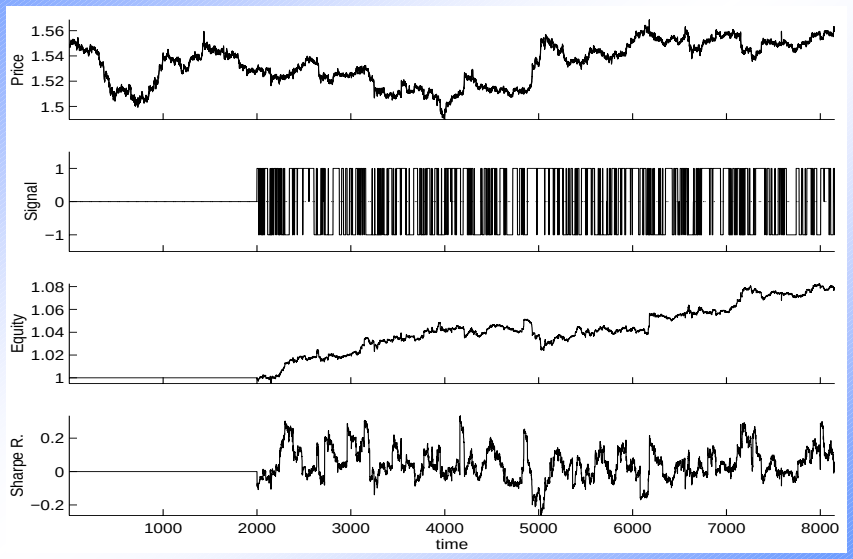


Draw-Down Comparison



Position Comparison

British Pound:
$\text{Return}_A = 15\%$, $\text{SR}_A = 2.3$, $\text{DDR}_A = 3.3$



Comments on the British Pound Results

The Simulations are Suggestive, not Conclusive:

- BP performance better than Deutschmark or Yen
- Price data are Reuters "indicative" quotes, not transactions
- Market microstructure effects could influence profitability.
- We spent little time designing the trader.
- From a real-world standpoint, the work is preliminary.

Efficacy of RRL for FX confirmed by Carl Gold (Caltech):

- More extensive simulations w/ Olsen 30 minute FX quotes
- IEEE CIFER *2003 Proceedings

Looking Ahead:

- Further analysis of microstructure / transaction costs is needed. FX broker transaction prices would help.
- A true test requires live trading.

Closing Remarks

- Direct Reinforcement
  Advantages over Value Function RL:
  - Natural representations
  - Efficiency & robustness
  - Causal algorithms and nonstationary problems
- Recurrence
  - Naturally occurs in real-world problems
  - Must abandon MPD framework
- Risk Averse Reinforcement
  - Conventional RL considers only expected rewards
  - Risk: The distribution of rewards matters
  - Robust policies for the real world must be low risk!
- Interesting research opportunities for Statisticians, Computer Scientists

URL: www.cse.ogi.edu/~moody

Some References:

URL: www.cse.ogi.edu/~moody

Papers:

John Moody and Matthew Saffell, 'Learning to Trade via Direct Reinforcement', Special Issue on Financial Engineering, IEEE Transactions on Neural Networks, 12(4):875-889, July 2001.

John Moody and Matthew Saffell, 'Minimizing Downside Risk via Stochastic Dynamic Programming', in "Computational Finance 1999", Y. S. Abu-Mostafa, B. LeBaron, A. W. Lo, and A. S. Weigend, editors, MIT Press, Cambridge, MA, pp. 403-416, 2000.

John Moody and Matthew Saffell, `Reinforcement Learning for Trading', in Advances in Neural Information Processing Systems, S. Solla, M. Kearns and David Cohn, eds., v. 11, pp 917-923, MIT Press, 1999.

Moody, J., Wu, L., Liao, Y. & Saffell, M., `Performance Functions and Reinforcement Learning for Trading Systems and Portfolios', Journal of Forecasting v. 17, pp. 441-470, 1998.