Reinforcement Learning in Trading: Project Scoping

Tanay Trivedi and Padraic McAtee

Machine Learning for Trading: Ritter

- Uses Q-Learning algorithm
- Simulates the dataset of financial asset using stochastic process estimated from real time series
  - Assumes that the dataset has some mean reversion dynamic
  - Suppose there is a tradeable security with positive price process $p_t>0$
  - Further suppose the security has some equilibrium price p_e such that

$$x_t = log\left(\frac{p_t}{p_e}\right)$$
$$dx_t = -\lambda x_t + \sigma\zeta$$

Where $\zeta_t \sim N(0,1)$ and each ζ_t is independent

- Means that p_t will revert to p_e with rate ƛ

Ritter (cont) Action Space

- Equities are usually traded in integer multiples of “round lots”, Ritter takes his to be 100 shares, with maximum position in asset to be *M*.
- Allows only *K* lots to be taken in each trade
- Therefore, possible trades and actions space is:

$$\mathcal{A} = LotSize \cdot \{-K, -K+1, \ldots, K\}$$

- Has cardinality $|A| = 2K + 1$

Ritter (cont) State Space

- Price in markets are naturally discretized to ticks, meaning 0.01 precision
- Figure out what the maximum movement of time series is, and setup simulation parameters as such $\mathcal{P} = TickSize \cdot \{1, 2, \ldots, 1000\}$
  - For his application, there was vanishing probability that the price exits [0.01,100]
- Now let *H* denote the possible ways of holding up to the maximum position size:

  $$\mathcal{H} = \{-M, -M+1, \ldots, M\}$$

  H has cardinality: $|\mathcal{H}| = 2M + 1$

- State definition for time *t* is $s_t = (p_t, n_{t-1})$
  - Price at time *t*, position size in shares at time *t-1*

State in other words is Cartesian Product between *H* and *P*: $\mathcal{S} = \mathcal{H} \times \mathcal{P}$

Ritter (cont) Reward

- Differential value of portfolio $\delta v_t = h_{t-1} \cdot r_t - c_t$
  - *v*-value
  - *h*-holdings/position
  - *r*-return from price change, *t-1* to *t*
  - *c*-cost from financing and slippage at *t*
- The following maximization works for wealth as well as utility of wealth, given a risk neutral investor:

$$\max_{\pi} \sum_{t} (\mathbb{E}[\delta w_t] - \frac{\kappa}{2} \mathbb{V}[\delta w_t])$$

w-wealth

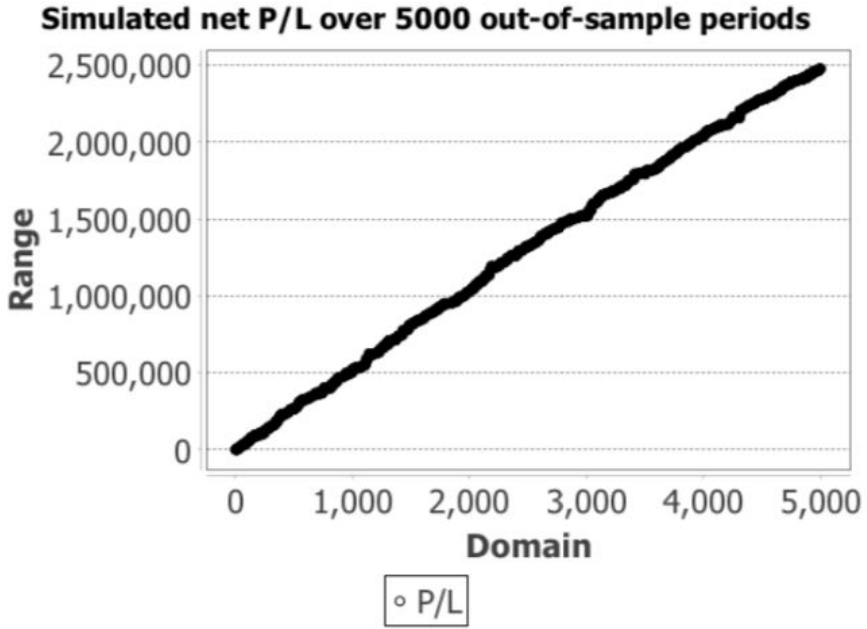
k-constant defined for mean variance equivalence

Ritter (cont) Reward

- To satisfy $\mathbb{E}[R_t] = \mathbb{E}[\delta w_t] - \frac{\kappa}{2}\mathbb{V}[\delta w_t])$, $R_t := \delta w_t - \frac{\kappa}{2}(\delta w_t - \hat{\mu})^2$
  - $\hat{\mu}$ is an estimate of the mean wealth increment over one period
  - Circular, because it is dependent on optimal policy. To solve, define it to be zero first, which will overestimate variance
  - Overestimation will not be bad, because as long as Risk Adjusted Return is not extremely high: $\mathbb{E}[(\delta w_t - \hat{\mu})^2] \approx \mathbb{E}[(\delta w_t)^2]$, meaning that $R_t \approx \delta w_t - \frac{\kappa}{2}(\delta w_t)^2$
  - Once policy has converged initially, use proper mean estimate
- Incremental wealth is approximately incremental value, so: $\delta w_t \approx \delta v_t = h_{t-1} \cdot r_t - c_t$
- This makes the incremented reward: $R_{t+1} \approx \delta v_{t+1} - \frac{\kappa}{2}(\delta v_{t+1})^2$

Ritter (cont) Results

Under the following parameters for Q-Learning: $\kappa = 10^{-4}\ \gamma = 0.999 \alpha = 0.001\ \epsilon = 0.1$



An Investigation into the Use of Reinforcement Learning Techniques within the Algorithmic Trading Domain- James Cumming

- Cummings uses Least Squares Temporal Difference Learning (LSTD)
  - Requires episodic and Markovian definitions
  - Defined days as episodes, Markovian because any indicators used are calculated in that episode making the state definition internal to the episode, end of day is terminal state

Approximate the State Value function using matrix inversion :

$$V^{\pi}(s) = \beta^{T}\phi(s)$$

V-State Value Function

Pi- Policy to be evaluated

s- state

Beta- Coefficient Matrix

phi- Feature Vector representing state

Cummings (cont)- Why LSTD

- LSTD has complexity $O(NK^3)$, N is the number of episodes used and K is the size of the feature vector
- Advantages include:
  - No Step Size Parameter, avoids overfitting and parameter hacking
  - No initial estimation
  - Only requires λ initial parameter, does not affect convergence speed much
- Chosen to minimize initial parameter inputs, because the space is complex and it is hard for humans to guess right
  - Results in really slow convergence, but very good results with less trial and error

Cummings (cont)

State Variables:

- Current Position (Long, Short, Neutral) quantified through PnL
- Price Price History over N intervals

Smaller size means more price history

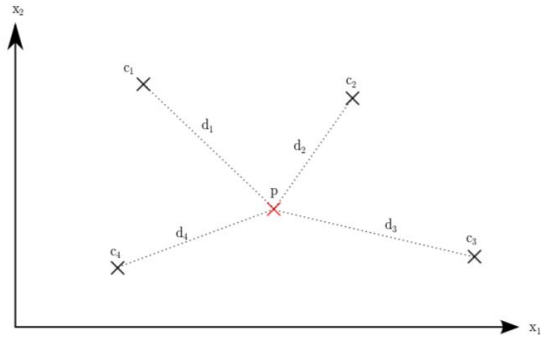
$$\mathbf{s_h} = \begin{pmatrix} \mathbf{d_1} \\ \mathbf{d_2} \\ \vdots \\ \mathbf{d_n} \end{pmatrix} = \begin{pmatrix} open_1 \\ high_1 \\ low_1 \\ close_1 \\ \vdots \\ open_n \\ high_n \\ low_n \\ close_n \end{pmatrix}$$

$$\mathbf{s_h} = \begin{pmatrix} \mathbf{d_1} \\ \frac{\mathbf{d_2}+\mathbf{d_3}}{2} \\ \frac{\mathbf{d_4}+\mathbf{d_5}+\mathbf{d_6}}{3} \\ \frac{\mathbf{d_7}+\mathbf{d_8}+\mathbf{d_9}+\mathbf{d_{10}}}{4} \\ \frac{\mathbf{d_{11}}+\mathbf{d_{12}}+\mathbf{d_{13}}+\mathbf{d_{14}}+\mathbf{d_{15}}}{5} \end{pmatrix}$$

$$\mathbf{s_h} = \begin{pmatrix} \mathbf{d_1} \\ \frac{\mathbf{d_2}+\mathbf{d_3}}{2} \\ \frac{\mathbf{d_4}+\mathbf{d_5}+\mathbf{d_6}+\mathbf{d_7}}{4} \\ \frac{\mathbf{d_8}+\mathbf{d_9}+\mathbf{d_{10}}+\mathbf{d_{11}}+\mathbf{d_{12}}+\mathbf{d_{13}}+\mathbf{d_{14}}+\mathbf{d_{15}}}{8} \end{pmatrix}$$

More compression means less precision

Cummings (cont) Vectorization of State



$$\mathbf{p_{pos}} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{pmatrix}$$

$$\phi_{gauss}(\mathbf{x}, \mathbf{c}) = e^{-(\epsilon \|\mathbf{x}-\mathbf{c}\|^2)}$$

$$\boldsymbol{\phi}_{pos}(s) = \begin{pmatrix} \phi_{gauss}(\mathbf{s_h}, \mathbf{c_1}) \\ \phi_{gauss}(\mathbf{s_h}, \mathbf{c_2}) \\ \vdots \\ \phi_{gauss}(\mathbf{s_h}, \mathbf{c_m}) \end{pmatrix}$$

Cummings (cont) State Feature Vector Representation

$$\boldsymbol{\phi}(s) = \begin{cases} \boldsymbol{\phi}_S(s) & \text{if } s \text{ is a short state} \\ \boldsymbol{\phi}_I(s) & \text{if } s \text{ is an idle state} \\ \boldsymbol{\phi}_L(s) & \text{if } s \text{ is a long state} \end{cases}$$

First Feature Vector Representation

$$\boldsymbol{\phi}_S^1(s) = \begin{pmatrix} \phi_{gauss}(\mathbf{s_h}, \mathbf{c_1}) \\ \phi_{gauss}(\mathbf{s_h}, \mathbf{c_2}) \\ \vdots \\ \phi_{gauss}(\mathbf{s_h}, \mathbf{c_m}) \\ S_{pnl} \\ \mathbf{0}^{m+1} \\ \mathbf{0}^{m+1} \end{pmatrix} \quad \boldsymbol{\phi}_I^1(s) = \begin{pmatrix} \mathbf{0}^{m+1} \\ \phi_{gauss}(\mathbf{s_h}, \mathbf{c_1}) \\ \phi_{gauss}(\mathbf{s_h}, \mathbf{c_2}) \\ \vdots \\ \phi_{gauss}(\mathbf{s_h}, \mathbf{c_m}) \\ I_{pnl} \\ \mathbf{0}^{m+1} \end{pmatrix} \quad \boldsymbol{\phi}_L^1(s) = \begin{pmatrix} \mathbf{0}^{m+1} \\ \mathbf{0}^{m+1} \\ \phi_{gauss}(\mathbf{s_h}, \mathbf{c_1}) \\ \phi_{gauss}(\mathbf{s_h}, \mathbf{c_2}) \\ \vdots \\ \phi_{gauss}(\mathbf{s_h}, \mathbf{c_m}) \\ L_{pnl} \end{pmatrix}$$

Cummings (cont) Actions and Transition Prob.

$$\mathcal{A}(\text{long}) = \{\text{close}, \text{stay}\}$$
$$\mathcal{A}(\text{short}) = \{\text{close}, \text{stay}\}$$
$$\mathcal{A}(\text{idle}) = \{\text{open}_{\text{long}}, \text{open}_{\text{short}}, \text{stay}\}$$

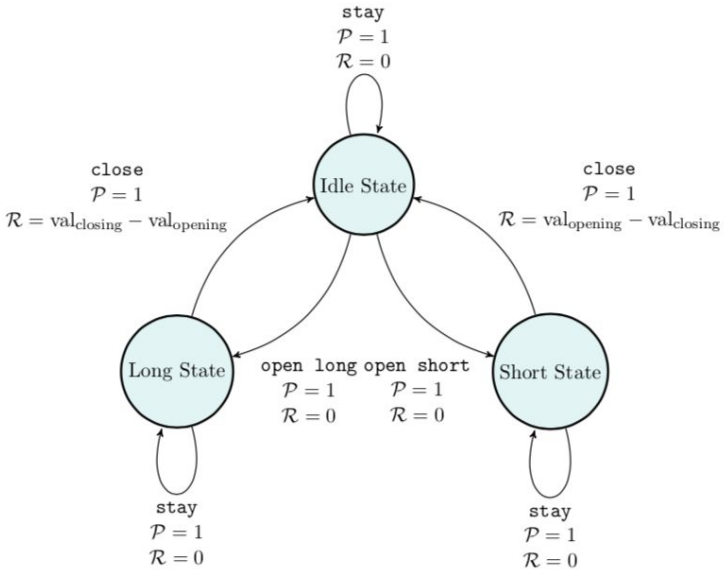
- Probabilities are 1 that every action indicated is executed
- Actual trading involves missed orders, partial fills, ECN rejection etc etc, but it's hard to estimate and simulate those parameters
  - Communicating with sandbox does some simulated interaction

Cummings (cont) Reward and MDP

- Learner is meant to maximize profitability
- Rewards are as follows:
  - Learner places new order or stays idle=0
  - Learner closes position=Realized PnL from position

$s = s_t$	$s' = s_{t+1}$	$a = a_t$	$\mathcal{P}^a_{ss'}$	$\mathcal{R}^a_{ss'}$
I_t	I_{t+1}	`stay`	1	0
I_t	L_{t+1}	`open long`	1	0
I_t	S_{t+1}	`open short`	1	0
L_t	L_{t+1}	`stay`	1	0
L_t	I_{t+1}	`close`	1	$\text{value}_{\text{closing}} - \text{value}_{\text{opening}}$
S_t	S_{t+1}	`stay`	1	0
S_t	I_{t+1}	`close`	1	$\text{value}_{\text{opening}} - \text{value}_{\text{closing}}$

Cummings (cont) Transition Diagram and Parameters



Parameter	Values
Number of Gaussian RBF centres	$m \in \mathbb{N}$
Width of Gaussian RBFs	$\epsilon \in \mathbb{R}^+$
Reward discount factor	$\gamma \in (0, 1)$
History structure	List of number of days to average over in history in the form $[l_n, l_{n-1}, \ldots, l_1]$
Lambda	$\lambda \in [0, 1]$
Exploration epsilon	$\epsilon \in (0, 1)$

Cummings (cont) Parameters and Results

Parameter	Values
Number of Gaussian RBF centres	8
Width of Gaussian RBFs	5
Reward discount factor	1
History structure	[3,2,1,1,1]
Lambda	0
Exploration epsilon	0.05

4.2.1 LSTD(λ) Results

Currency Pair	Annualised Return Rate (%)
Euro - Pound	0.00250
US Dollar - Canadian Dollar	0.000715
US Dollar - Swiss Franc	0.00296
US Dollar - Japanese Yen	0.00335
Euro - Canadian Dollar	0.0281
Euro - US Dollar	0.0164
Euro - Australian Dollar	0.0228
Pound - US Dollar	0.00363
Euro - Japanese Yen	0.00229
Australian Dollar - US Dollar	0.0118
Pound - Japanese Yen	−0.00230
Total	0.00839

Cummings (cont) Possible Improvements

- Having the learner attain a grasp of risk aversion in the process
  - Reward on Risk Adjusted Return rather than PnL
  - Have functionality in the state machine to check the Expected Return from an action versus the current risk and make a decision
- Use PILCO or other methods to better use data and reduce oscillation in convergence

Other Papers (Using RRL)

- Moody- Recurrent Reinforcement Learning for Trading (1998)
- Dempster- Computational Learning Techniques for Intraday FX Trading Using Popular Technical Indicators (2001)(Improvement on Moody)
- LvDuZhai- Algorithm Trading using Q-Learning and Recurrent Reinforcement Learning (2009)

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

- Viewed as a recurrent single layer neural network, optimizing live instantaneous reward metric

$$D_t = \frac{d\widehat{S}(t)}{d\eta}|_{\eta=0}$$

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