Investment Association Presentation

Machine Learning in Algorithm Trading:
An Example of Recurrent Reinforcement Learning

(Partial Achievement of CS534 Artificial Intelligence Project)

Ethan Song

April 11, 2016

Concepts

- Machine Learning (ML):
  - Develop the computer's ability of learning without explicitly programmed
  - E.g. Facebook face recognition

- Algorithm Trading (AT):
  - The trading strategies are programmed and implemented by computers
  - Previous work: Emerging from 1980s, predictive math models, statistical arbitrage, event-driven system have been using for decades
  - New method: application of machine learning

Concepts

- Reinforcement Learning (RL):
  - Example: Chess game learning
  - Model Components:

I. Environment states $\{S_i\}_n$

II. Actions $\{A_i\}_n$

III. Transition Model: How does the state change from step i to step i+1? e.g. weather change model, transitional probability matrix

IV. Reward $\{R_i\}_n$: immediate step reward

V. Utilities: A goal function regarding total rewards

RRL: Initialization

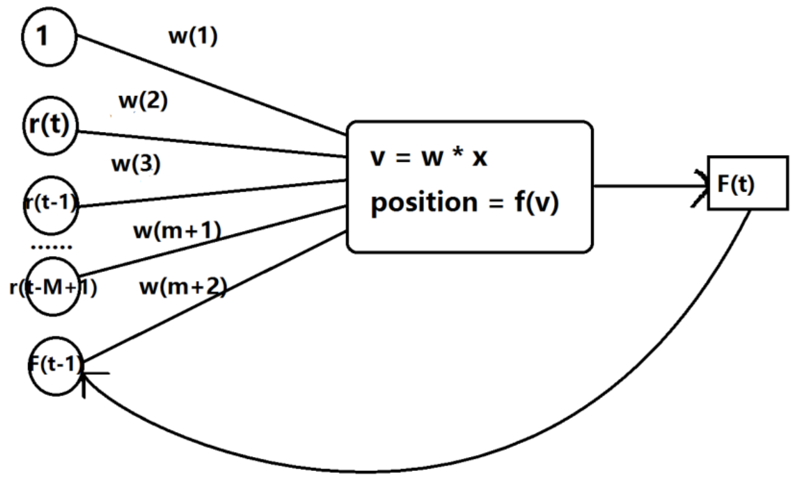
- Sample Data:
  - Simplified Data: Time and close price
  - Assumptions & Denotations:
    - Only trade one share per transaction
    - $P_{t:}$ price at date t
    - $r_{t:}$ price increment at date t, $r_t = P_t - P_{t-1}$
    - N: dimension of observations (number of the observed previous dates)
    - w: weights, $w \epsilon R^{N+2}$ (explained later)
    - u: commission fees per trade
    - T: length of time period

2014/9/1	10740
2014/9/2	10735
2014/9/3	10725
2014/9/4	10730
2014/9/5	10735

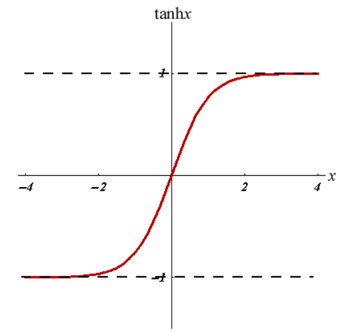
- Model Components:
  - States x_t, defined as the vector:
    - $[1, r_t, r_{t-1}, r_{t-2}, \ldots, r_{t-N+1}, F_{t-1}]$
    - e.g. For 2014/9/5 , if N = 4, $x_t = [1, 5, 5, -10, -5, 1]$ as showed in the sample data with a previous long position F_{t-1} denoted as $F_{t-1} = 1$ on 2014/9/4.
  - Actions/Positions F_t: The position on date t. F_t can be 1 as a long position, 0 as no position, -1 as a short position.
  - Immediate Rewards: $R_t = F_{t-1} * r_t - u * abs(F_t - F_{t-1})$
  - Utility: $S_T = E[R_t] / Std(R_t) = E[R_t] / sqrt(E[R_t\text{^}2] - (E[R_t])\text{^}2)$ "profit per unit risk"
    - Denote $A = E[R_t]$, $B = E[R_t\text{^}2]$, then $S_T = A / sqrt(B - A\text{^}2)$ (Similar to Sharp Ratio)

RRL: Model Desciption

- Neuro Structure:



- Input: $v = w^T * x = w_1+w_2r_t+w_3r_{t-1}+\ldots+w_{N+1}r_{t-M+1}+w_{N+2}F_{t-1}$
- $f(v) = \tanh(v) = e^x-e^{-x} / e^x+e^{-x}$ (Hyperbolic Tangent)



r

RRL: Mathematical Principle

- $w^{i+1} = w^i + alpha * (dS_T / dw^i)$ (online learning, gradient ascent)
- How to calculate (dS_T / dw^i)?

Recall $\frac{dS_T}{dw} = \frac{d}{w}\left\{\frac{A}{\sqrt{B-A^2}}\right\}$ where $A = E(R_t) = \frac{1}{T}\sum_{t=1}^{T} R_t$ $B = E(R_t{}^2) = \frac{1}{T}\sum_{t=1}^{T} R_t{}^2$

$$\frac{dS_T}{dw} = \frac{d}{w}\left\{\frac{A}{\sqrt{B-A^2}}\right\} = \frac{dS_T}{dA}\frac{dA}{dw} + \frac{dS_T}{dB}\frac{dB}{dw} = \sum_{t=1}^{T}\left(\frac{dS_T}{dA}\frac{dA}{dR_t} + \frac{dS_T}{dB}\frac{dB}{dR_t}\right)\frac{dR_t}{dw}$$

$$= \sum_{t=1}^{T}\left(\frac{dS_T}{dA}\frac{dA}{dR_t} + \frac{dS_T}{dB}\frac{dB}{dR_t}\right)\left(\frac{dR_t}{dF_t}\frac{dF_t}{dw} + \frac{dR_t}{dF_{t-1}}\frac{dF_{t-1}}{dw}\right)$$ (back propagation)

Where $\frac{dS_T}{dA}, \frac{dA}{dR_t}, \frac{dS_T}{dB}, \frac{dB}{dR_t}$ can be obtained at each step, and

$$\frac{dR_t}{dF_t} = -u \cdot \mathrm{sign}(F_t - F_{t-1}) \qquad \frac{dR_t}{dF_{t-1}} = r_t + u \cdot \mathrm{sign}(F_t - F_{t-1})$$

$$\frac{dF_t}{dw} = \frac{d}{w}\{\tanh(w^T x_t)\} = (1 - \tanh^2(w^T x_t)) \cdot \left(x_t + \frac{dF_{t-1}}{dw} \cdot w_{N+2}\right)$$

$\frac{dF_t}{dw}$ is recurrent and depend on the previous $\frac{dF_{t-1}}{dw}$, which can be iteratively programmed

RRL: Implementation

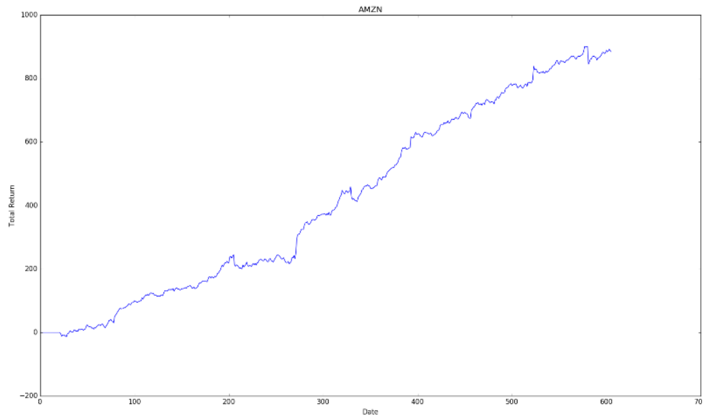
- Implementation:
  - Split historical data into training and test set
  - Using training data to determine the optimal weight
  - Apply the weight on test set and observe the performance
  - Assume trade only one share per transaction

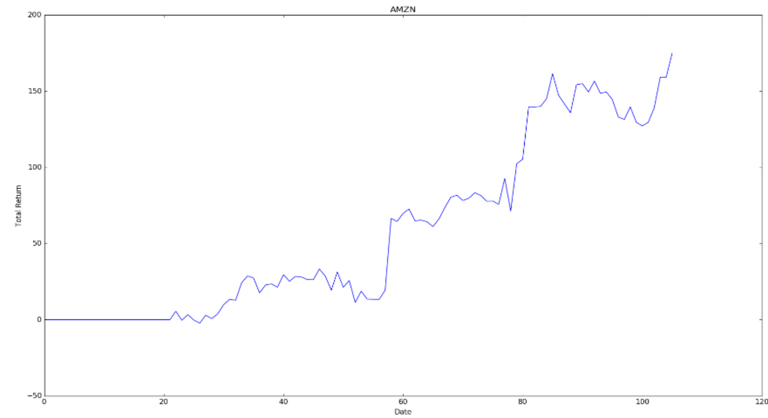
  - As the learning steps increases, the “Sharp Ratio” increases

```
D:\python34\python.exe C:/Users/acer/Desktop/Presentation/rrl_stock.py
Please enter the initial weight coefficient (0.01-1.00): 0.1
Please enter the learning Speed (0.1-5.0): 2
Reinforcement Learning 1 th step, Sharp Ratio: -0.015538848940336492
Reinforcement Learning 2 th step, Sharp Ratio: 0.06929876556275599
Reinforcement Learning 3 th step, Sharp Ratio: 0.1287925043522688
Reinforcement Learning 4 th step, Sharp Ratio: 0.12907035149680088
Reinforcement Learning 5 th step, Sharp Ratio: 0.13037694126475244
Reinforcement Learning 6 th step, Sharp Ratio: 0.14789008367970302
Reinforcement Learning 7 th step, Sharp Ratio: 0.15682476764671893
Reinforcement Learning 8 th step, Sharp Ratio: 0.1506964686043493
Reinforcement Learning 9 th step, Sharp Ratio: 0.14426021998371022
Reinforcement Learning 10 th step, Sharp Ratio: 0.15453325780698657
Reinforcement Learning 11 th step, Sharp Ratio: 0.1680635929509651
Reinforcement Learning 12 th step, Sharp Ratio: 0.16540300311604716
Reinforcement Learning 13 th step, Sharp Ratio: 0.17892174648168402
Reinforcement Learning 14 th step, Sharp Ratio: 0.17956181701849072
Reinforcement Learning 15 th step, Sharp Ratio: 0.18498154040414663
```

RRL: Implementation on Stocks

- AMZN, Day-Granularity, **no loss limit control**, using 21 days price:
  - Training Data: 2013/1/2 – 2015/5/29, 606 days
  - Test Data: 2015/6/1 – 2015/9/30, 86 days
  - Training natural net return 171.08, test natural net return (nnt) 80.97
  - Initial weight 0.02, learning speed 0.3, commission 0.05, steps 1000



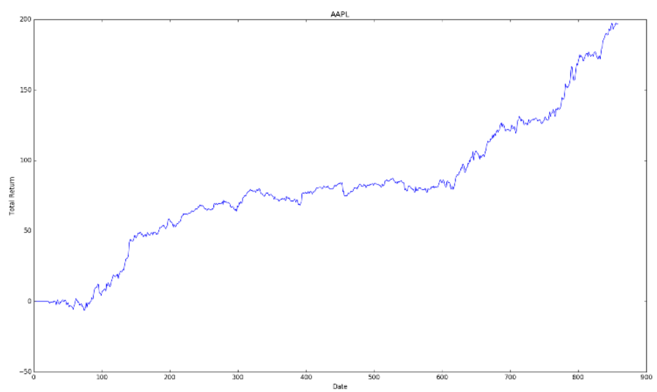


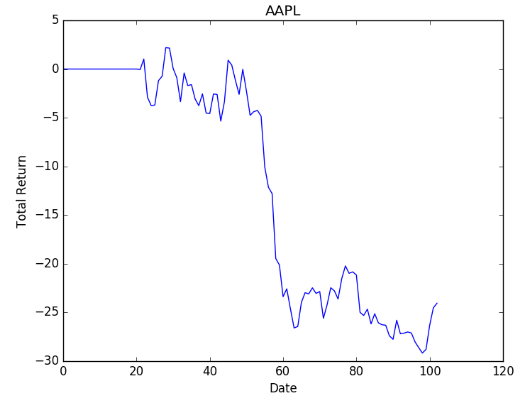
  - Training: net return around 850, 500% nnt
  - Test: net return around 220, 270% nnt
  - If only hold long position net return 140, 170% nnt

r

RRL: Implementation on Stocks

- AAPL, Day-Granularity, **no loss limit control**, using 21 days price:
  - Training Data: 2012/7/2 – 2015/11/30, 859 days
  - Test Data: 2015/12/1 – 2016/3/31, 83 days
  - Training natural net return 33.65, test natural net return -12.24
  - Initial weight 0.1, learning speed 1.0, commission 0.05, steps 1000





  - Training: net return around 197.0, 600% nnt
  - Test: net return around -25.0, 200% nnt (loss)
  - Biggest drawback happened at the second half of Jan, 2016, when the stock price was bumpy

r

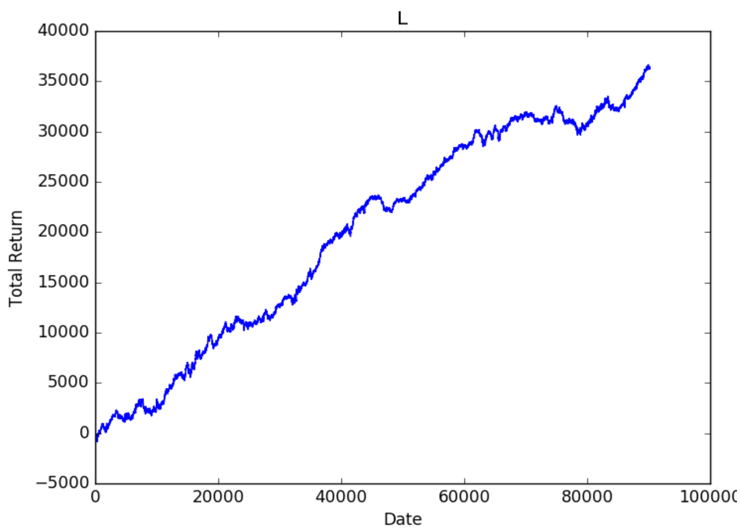
RRL: Implementation on Futures

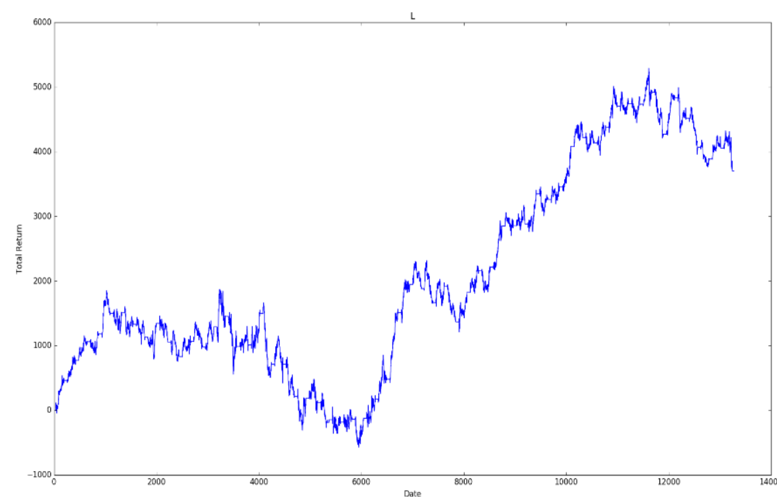
- China PE(L) Futures, Minute-Granularity
  - Point value 5, Commission fee 3 per share
  - Assume no slippage (but can be included by raising commission fee)
  - Simplified data, only use close price
  - Training Data: 2013/1/4 – 2014/8/29, 90224 tuples
  - Test Data: 2014/9/1 – 2014/11/28, 13275 tuples
- More control: decrease risk exposure by lowering trading times and forbid overnight positions, cut loss by setting absolute loss limit
  - Trading Period: 9:30 A.M. – 2:30 P.M. Overnight not allowed
  - Loss limit control: e.g. cut when sum loss greater than 200 in 3 bars
  - Lower trading frequency: Hold same position for at least 5 bars

r

RRL: Implementation on Futures

- Performance
  - Initial weight 0.03, learning speed 0.5, steps 200, #bars 21, loss limit 200
  - Training: Net profit 36409.0, 820% the original margin; Trading times 9114, around 24 transactions per day (still too high)
  - Test: Net profit 3702.0, 86% the original margin; Trading times 1357, around 20 transactions per day

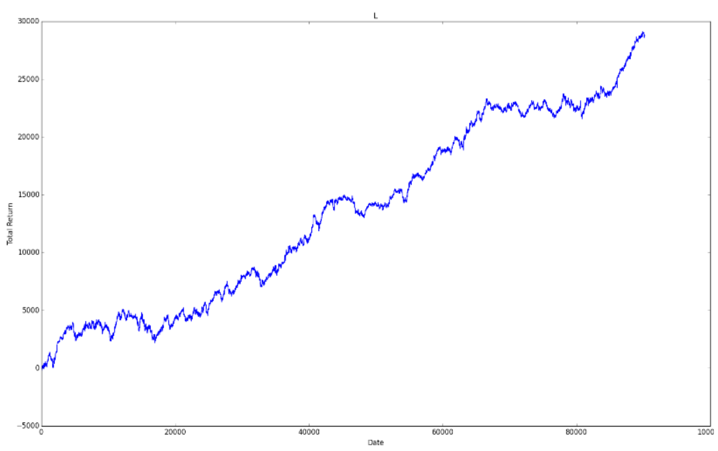


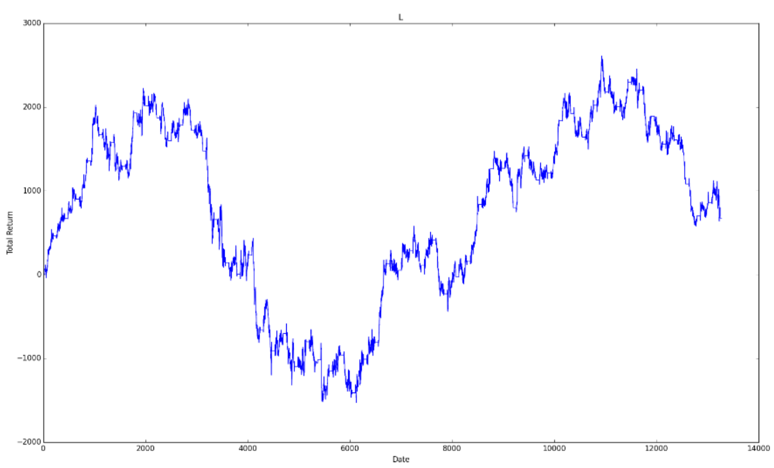


r

RRL: Implementation on Futures

- Performance
  - Initial weight 0.02, learning speed 0.8, steps 300, #bars 30, loss limit 175
  - Training: Net profit 28783.0, 650% the original margin; Trading times 9427, around 25 transactions per day (still too high)
  - Test: Net profit 673.0, 15% the original margin; Trading times 1392, around 20 transactions per day, significant drawback

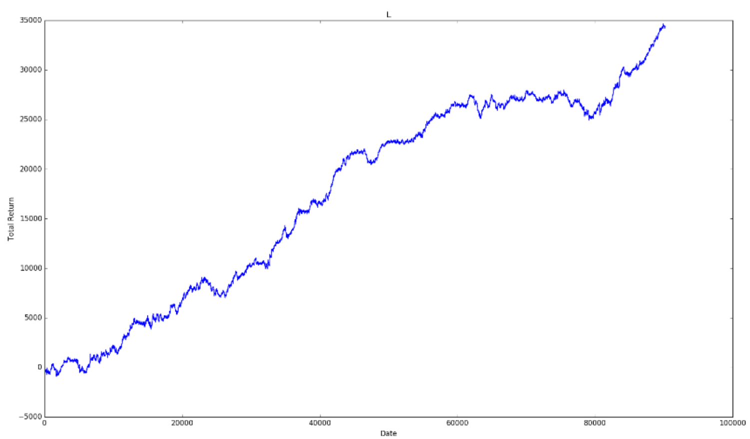


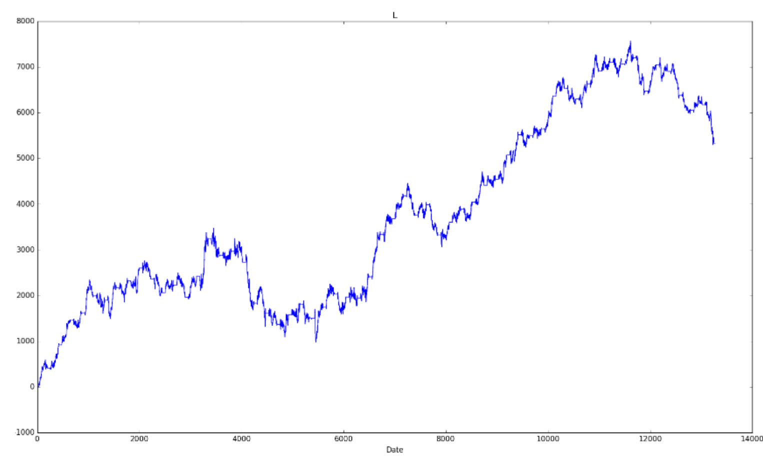


r

RRL: Implementation on Futures

- Performance
  - Initial weight 0.1, learning speed 1.0, steps 300, #bars 14, loss limit 150
  - Training: Net profit 34424.0, 650% the original margin; Trading times 9096, around 24 transactions per day (still too high)
  - Test: Net profit 5331.0, 124% the original margin; Trading times 1349, around 20 transactions per day





r

RRL: Conclusion

- Conclusion
  - Training by RRL, the utility function (S_T) can be considerably optimized
  - The raw algorithm shows promising result on test data, generally it will perform better than the natural return
  - The strategy developed by RRL is likely to be a trend following strategy, for it performs worse when it is bumpy, and much better when there is a good trend
  - The style of the strategy on higher frequency data is like “hit and run”, very swift and sensitive. Under minute-granularity, we have to filter the trading signals to lower the trading frequency, thus avoiding slippage.
  - The strategy with such trading style may perform better on even higher frequency data, e.g. High frequency trading on Forex under tick-level.

r

RRL: Future Work

- Improvement
  - Improve loss control mechanism (e.g. trailing stop loss limit)
  - Use cross validation to split, train, and test.
  - Optimize the parameters other than weight. e.g. #bars
  - Revise the benchmark metrics, include in more factors. e.g. Trading times
  - Test the feasibility on tick-level forex trading

r