



THE DEPARTMENT OF  
**COMPUTER SCIENCE  
& ENGINEERING**  
計算機科學及工程學系



# **(LZ2) Pairs Trading with Machine Learning**

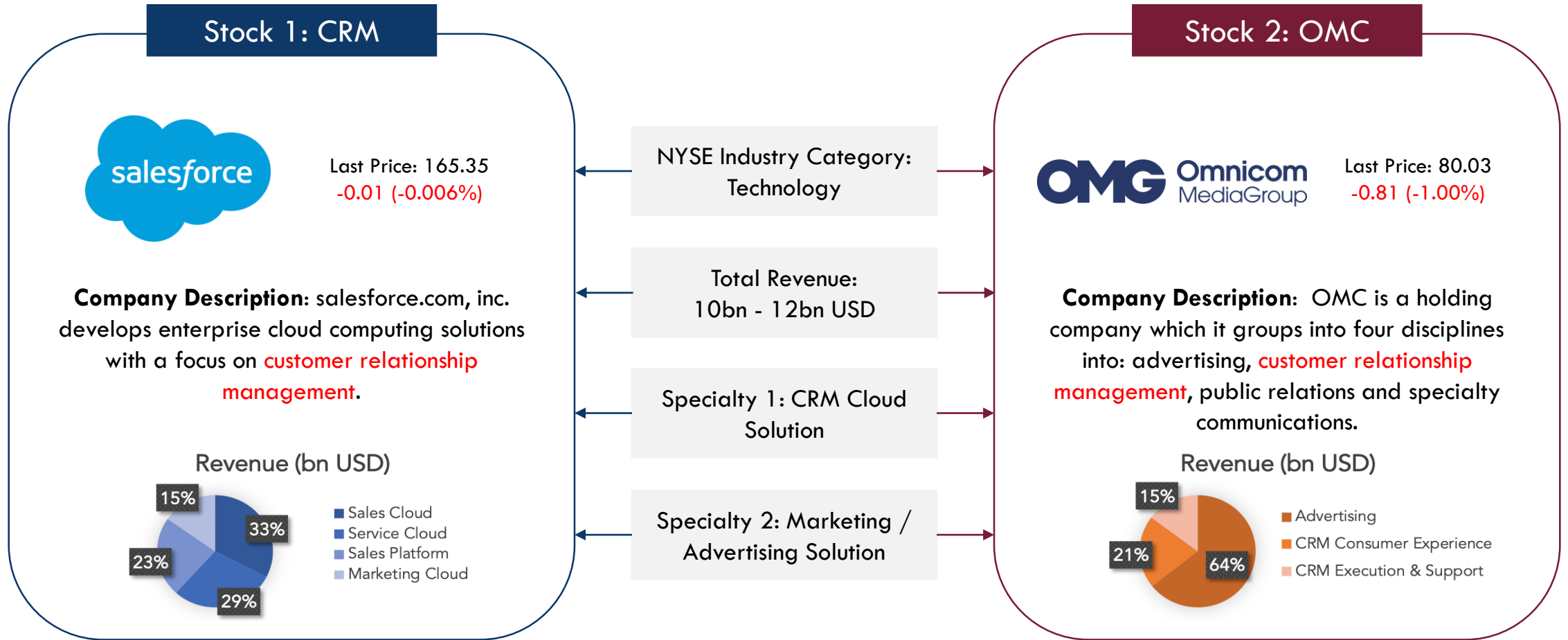
---

A novel approach via Reinforcement Learning



# Introduction to Pairs Trading

## Concept of Relative Pricing



**Key Concept:** Similar stocks (assets) with **similar risks** should be **priced similarly** / have similar price movements

# Introduction to Pairs Trading

## Idiosyncratic Risks – The Source of Trade Opportunities

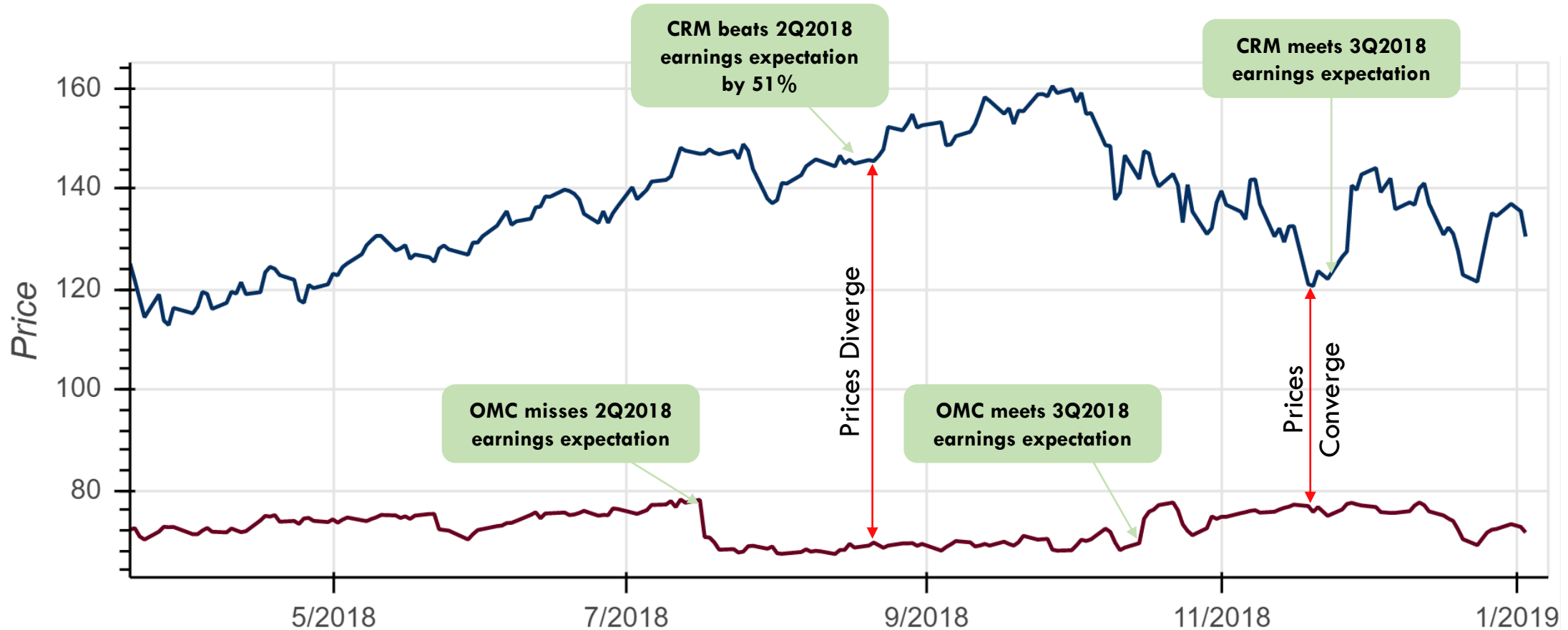


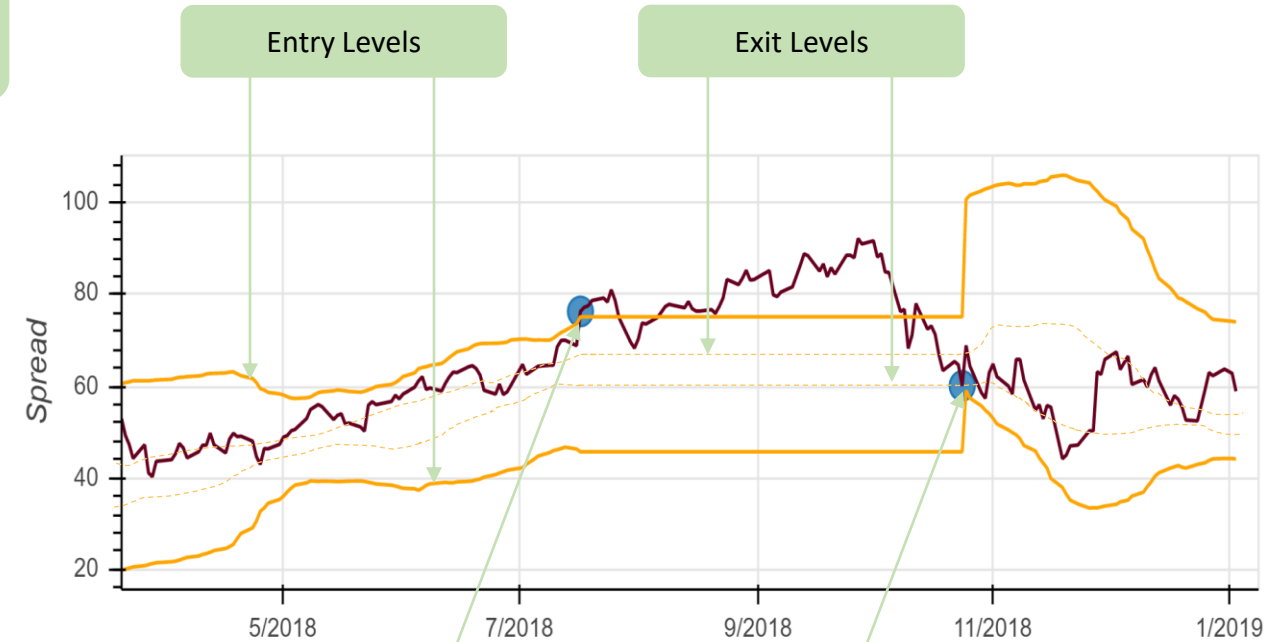
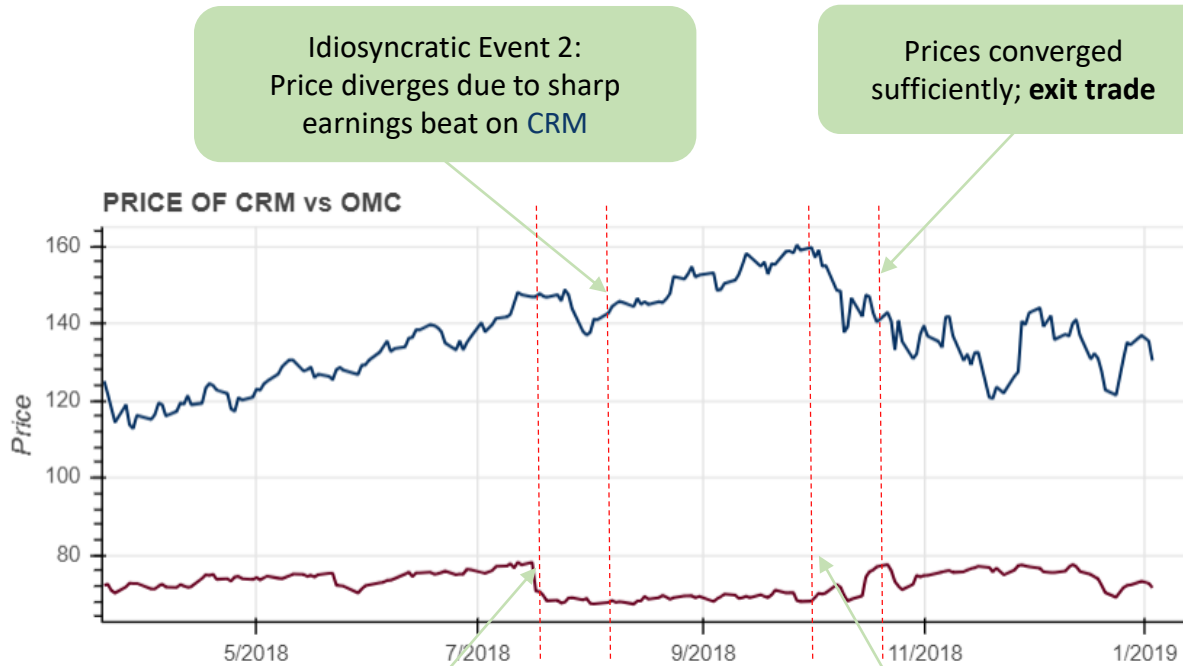
Figure 2: Prices of CRM (blue) and OMC (brown).

### Source:

- CRM 2Q2018: <https://www.nasdaq.com/article/salesforcecom-crm-q2-earnings-and-revenues-surpass-estimates-cm1015210>
- CRM 3Q2018: <https://www.nasdaq.com/article/salesforce-crm-solid-cloud-offerings-to-aid-q3-earnings-cm1060129>
- OMC 2Q2018: <https://www.nasdaq.com/article/omnicom-omc-q2-earnings-surpass-estimates-revenues-lag-cm992377>
- OMC 3Q2018: <https://www.nasdaq.com/asp/call-transcript.aspx?StoryId=4211957&Title=omnicom-group-inc-omc-ceo-john-wren-on-q3-2018-results-earnings-call-transcript>

# Introduction to Pairs Trading

How to obtain the 'spread' from two prices



# Key Questions For The Project

---

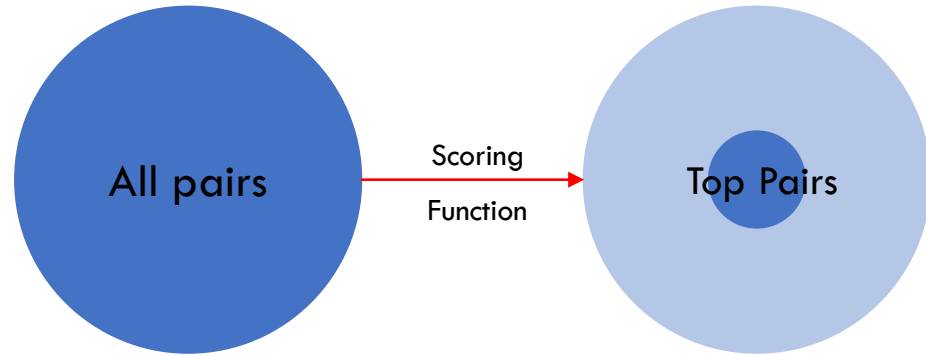
*Takeaways from the CRM-OMC pairs trade framework*

- 1. How do we determine what are good pairs?*
- 2. How do we determine the optimal time to enter and exit trades?*
- 3. What are the existing pairs trading approaches?*
- 4. How do we model the problem in machine learning setting?*

# The Pairs Trading Framework

Pair Selection + Pair Trading Logic = **Pairs Trading Strategy**

## Pairs Selection



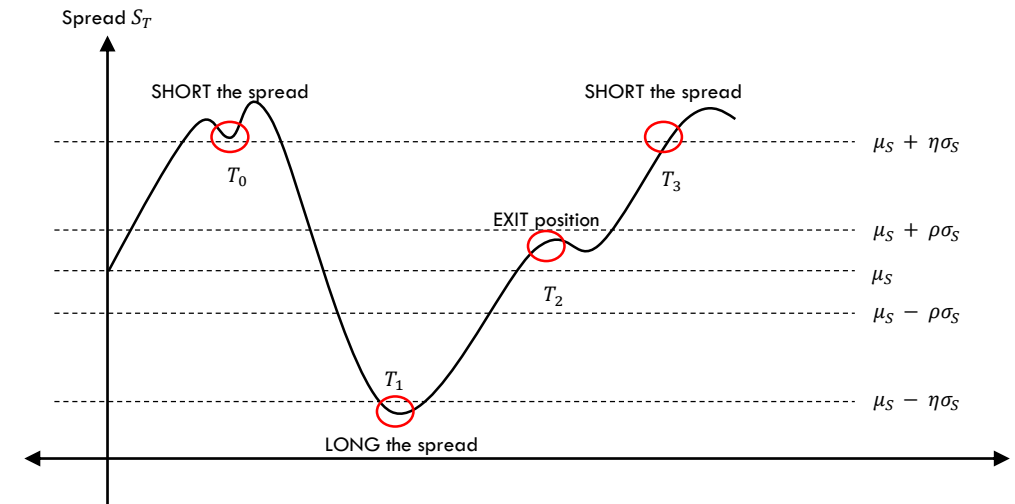
Given  $N$  stocks, there are  $\binom{N}{2}$  possible pairs to consider.

The top  $k$  % of pairs with highest scores are selected.

### Essential Criteria

The scoring function should determine if the spread is **mean-reverting**

## Pairs Trade Logic


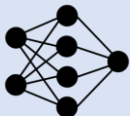



### Essential Criteria

The spread definition is **case-by-case** basis: depends on the approach used

# Project Scope & Objectives

Applying reinforcement learning to pairs trade logic

Scope	Pairs Trading Framework in Statistical Arbitrage Trading		
Areas of Focus	Backtesting	Trading Agent	User Interface
Objectives	<div><div>1</div><div><p>Implement existing strategies as <b>baselines</b> for comparison (Distance, Cointegration)</p></div></div>	<div><div>2</div><div><p>Develop profitable pairs trading agent using <b>Reinforcement Learning (RL)</b></p></div></div>	<div><div>3</div><div><p>Implement a system with <b>User Interface (UI)</b> for visualizing backtest performance</p></div></div>
Outcome	<div><div>i)</div><div>Create a universal trading agent that outperforms both baseline strategies and market indices.</div></div> <div><div>ii)</div><div>Create a trading agent that can generalize to trade other sectors</div></div>		





THE DEPARTMENT OF  
**COMPUTER SCIENCE  
& ENGINEERING**  
計算機科學及工程學系



# Section 1: Design of Baseline Strategies

---

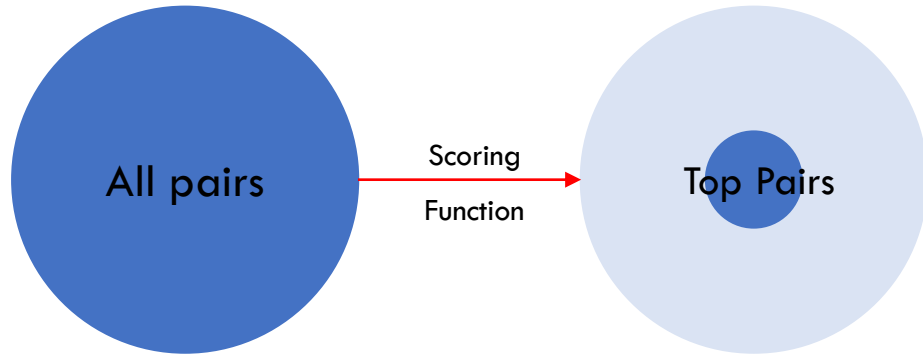




# Baseline Strategy – Distance Method

## Scoring function and spread definition

### Pairs Selection



Given  $N$  stocks, there are  $\binom{N}{2}$  possible pairs to consider.

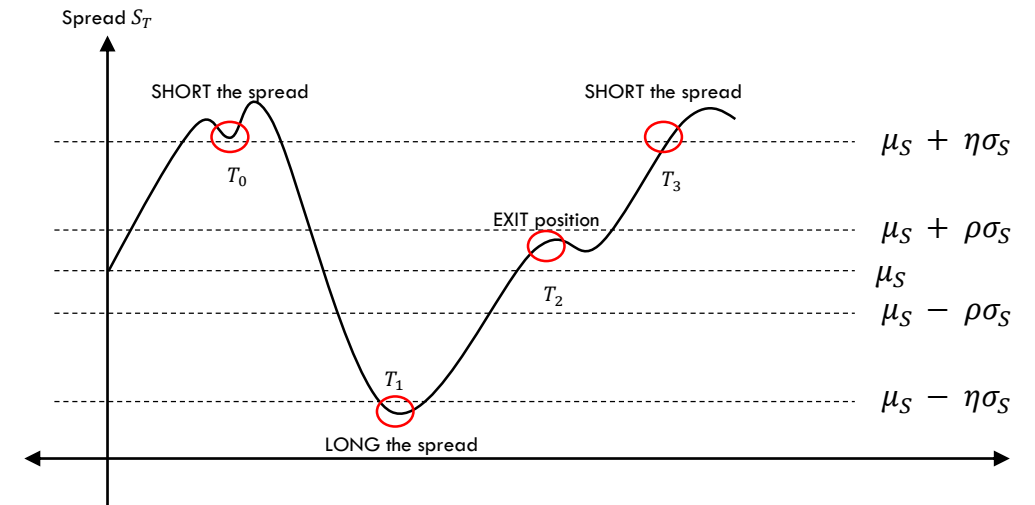
The top  $k$  % of pairs are traded after ranking a by scoring function

#### Scoring Function

$$Score[X, Y] = - \sum_{t=0}^T (Y_t - X_t)^2$$

*\*\* This is the sum of squared-error of prices*

### Pairs Trade Logic



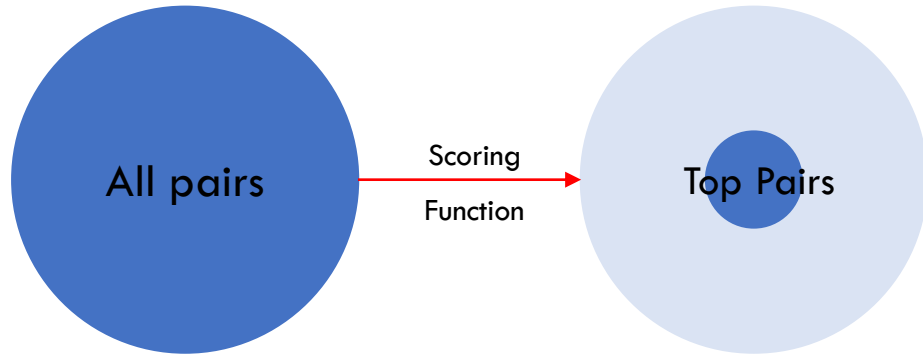
#### Spread Definition

$$S_t = Y_t - X_t$$

# Baseline Strategy – Cointegration method

## Scoring function and spread definition

### Pairs Selection



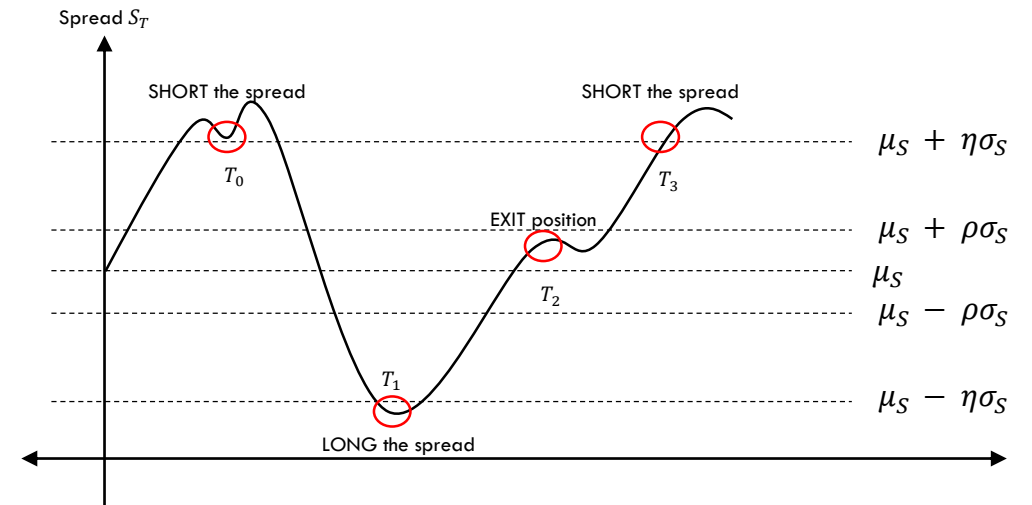
Given  $N$  stocks, there are  $\binom{N}{2}$  possible pairs to consider.

The top  $k$  % of pairs are traded after ranking a by scoring function

#### Scoring Function

$$\text{Score}[X, Y] = - \left( \begin{array}{l} p\_value \text{ of } ADF \\ stationarity \text{ test on} \\ spread \text{ of } X \text{ and } Y \end{array} \right)$$

### Pairs Trade Logic



#### Spread Definition

$$S_t = \log Y_t - \alpha \log X_t - \beta$$

# More on Cointegration Method ...

---

*Approaches to estimation of parameters  $\alpha$  and  $\beta$*

Recall that the spread definition in Cointegration method is

$$S_t = \log Y_t - \alpha \log X_t - \beta$$

How do we estimate  $\alpha$  and  $\beta$  during trading time? (since they might change over time)

## Rolling OLS (ordinary least squares)

At every time step, use OLS to fit the linear model

$$\log Y_t = \alpha \log X_t + \beta + \varepsilon_t$$

using the K most recent data points of X and Y.

## Kalman Filter

*Assume that  $\alpha$  and  $\beta$  follow random walk processes:*

$$\alpha_{t+1} = \alpha_t + \epsilon_{\alpha,t} \qquad \beta_{t+1} = \beta_t + \epsilon_{\beta,t}$$

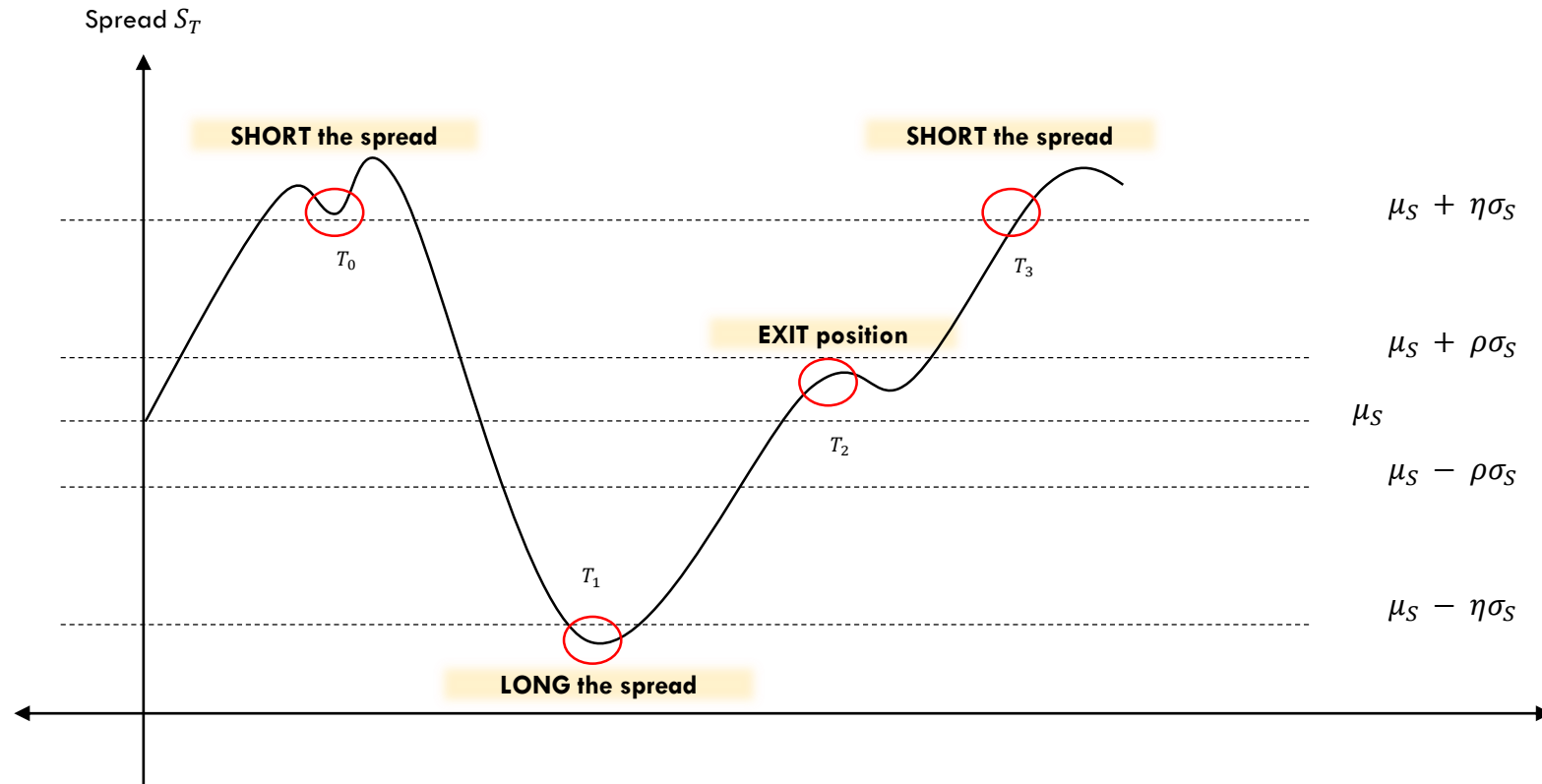
*Estimate them Dynamically using the Kalman Filter framework:*

$$\text{State transition equation: } \begin{bmatrix} \beta_{t+1} \\ \alpha_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \beta_t \\ \alpha_t \end{bmatrix} + \begin{bmatrix} \epsilon_{\beta,t} \\ \epsilon_{\alpha,t} \end{bmatrix}$$

$$\text{Observation equation: } \log Y_t = \begin{bmatrix} 1 & \log X_t \end{bmatrix} \begin{bmatrix} \beta_t \\ \alpha_t \end{bmatrix} + \epsilon_t$$

## Closer look: Pair Trade Logic (Used In Baselines)

We will be proposing a new Pair Trade Logic later. Let's review a look at the existing one



### Key idea:

- When spread diverges  $\pm 2$  SD away from mean, SHORT/LONG the spread.
- When spread converges to within  $\pm 0.5$  SD from mean, EXIT spread to take profit.



THE DEPARTMENT OF  
**COMPUTER SCIENCE  
& ENGINEERING**  
計算機科學及工程學系



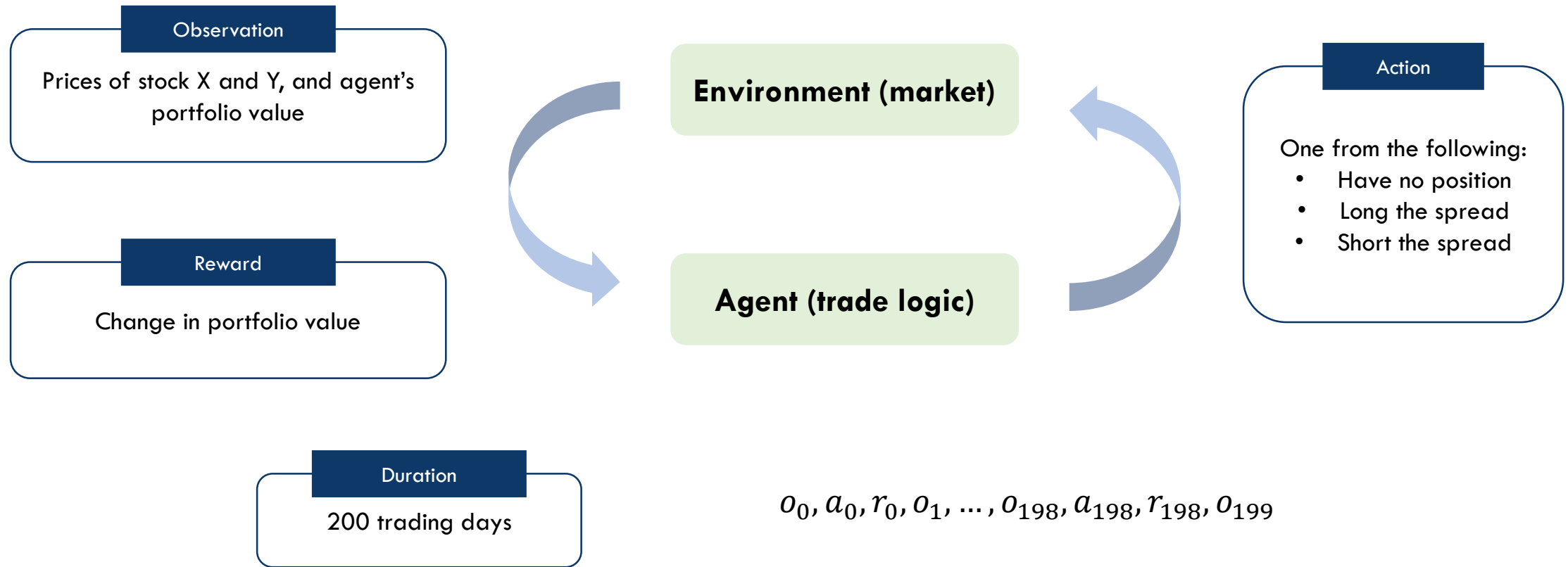
## Section 2: Design of RL Trading Agent

---



# Design of RL Trading Agent

*The pair trading problem (one pair at a time) can be modelled as follows:*





# Design of RL Trading Agent

---

*The imaginary environment state*

State

The relative pricing level between stock X and Y, and agent's portfolio value

In this problem, observation  $\neq$  state (Partially observable):

Burger 1: \$40

Burger 2: \$40

! ! ! ! ! ! ! !

Burger 1: \$10, \$10, \$11, \$12, \$40

Burger 2: \$36, \$37, \$38, \$39, \$40

# Design of RL Trading Agent

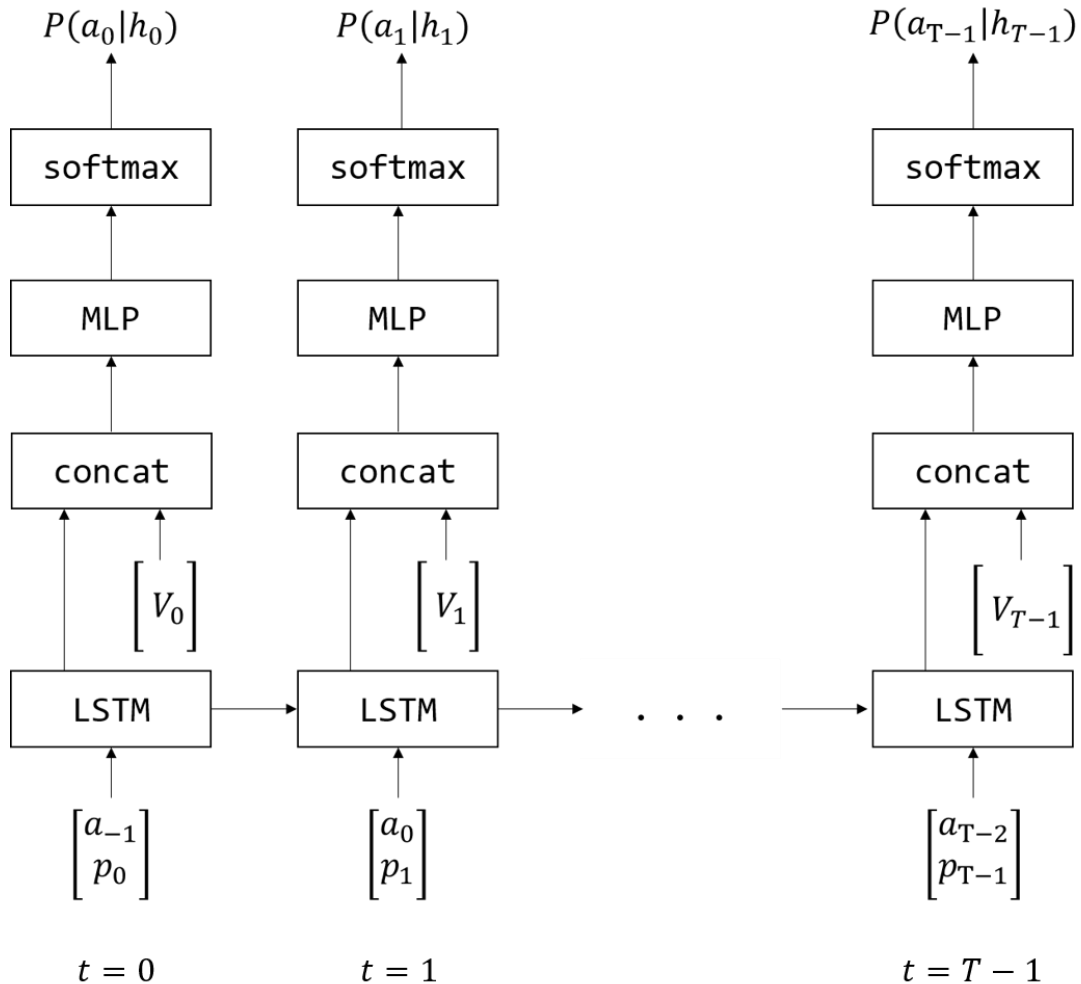
---

*How to solve the problem by reinforcement learning?*

- Policy Gradient Method
- We need a policy that:
  - accepts the historical observations and actions as input due to the partial observability issue mentioned earlier, and
  - outputs a probability distribution over the action space (have no position, long the spread, short the spread).
- Since the sequence of historical observations and actions is in variable length, we can use an Recurrent Neural Network (RNN) as part of the model architecture of our policy network.

# Design of RL Trading Agent

A policy network  $\pi(h_t) = P(a_t|h_t)$ . The architecture is defined as follows:



- $a_t$  is a one-hot encoding vector that indicates the action taken at time  $t$ .
- $p_t = (p_Y^t \ p_X^t)^T$ , where  $p_Y^t$  and  $p_X^t$  are the normalized log price of stock  $Y$  and  $X$  respectively at time  $t$ .
- $V_t$  is the (normalized) portfolio value.
- $h_t$  is the sequence  $a_{-1}, p_0, a_0, p_1, \dots, a_{t-1}, p_t$  together with  $V_t$ .
- At time  $t$ , the next action to take is sampled from  $\pi(h_t) = P(a_t|h_t)$ .
- $T$  is the terminal time step (which is equal to 199).
- Since we think that the portfolio value will not contribute to the state dynamics, we concatenate the  $V_t$  to the output from the LSTM.  
After the concatenation, the output is used to approximate the environment state at the given time  $t$ .

# Design of RL Trading Agent

---

## *Training process*

1. Initialize a random policy network.
2. Sample a subset of all possible pairs.
3. For each sampled pair  $i$ , run trading simulation (given the policy network) and generate trajectory  $\tau_i$  that is a sequence of observations, actions taken, and rewards received.
4. Compute the **gradient** of the **objective function** with respect to the policy network parameters.
5. Update the policy network by the **gradient ascent update rule**.
6. Go to step 2 until some iteration  $K$ .

Highlighted words will be mentioned in next slide.

# Design of RL Trading Agent

---

*How does the agent learn?*

- The objective function that we want to maximize:

$$J(\theta) = \mathbb{E}_{\tau \sim p(\tau; \theta)}[r(\tau)] = \mathbb{E}_{\tau \sim p(\tau; \theta)} \left[ \sum_{t=0}^{T-1} r_t \right],$$

where  $\theta$  is the policy network parameters,  $\tau$  is a trajectory generated in the trading simulation given the policy network.

- By <http://people.idsia.ch/~alexander/2007/2/icann2007.pdf>, the gradient of the objective function with respect to  $\theta$  is:

$$\nabla_{\theta} J \approx \frac{1}{N} \sum_{n=1}^N \sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_t | h_t) R_t^n,$$

where  $N$  is the number of sampling of trajectories and  $R_t^n = \sum_{k=t}^{T-1} r_k$ .

- Together with the update rule, we can enable the trading agent to learn from experience:

$$\theta := \theta + \eta \nabla_{\theta} J(\theta),$$

where  $\eta$  is the learning rate.



THE DEPARTMENT OF  
**COMPUTER SCIENCE  
& ENGINEERING**  
計算機科學及工程學系



## Section 3: Experiments and Results

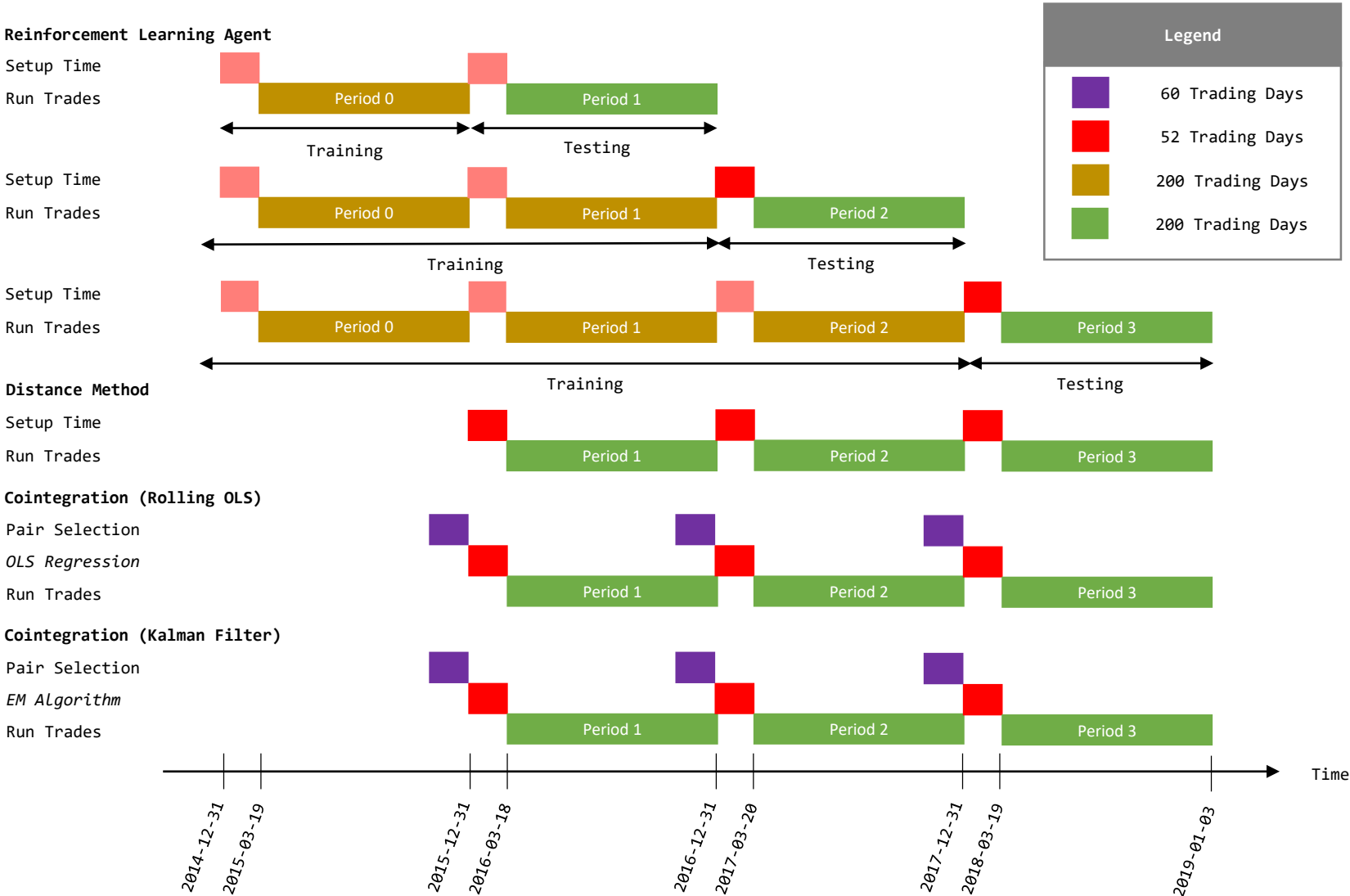
---





# Experimental Setup

## Standardization Procedures and Asset Pool



### Stock Universe / Assets

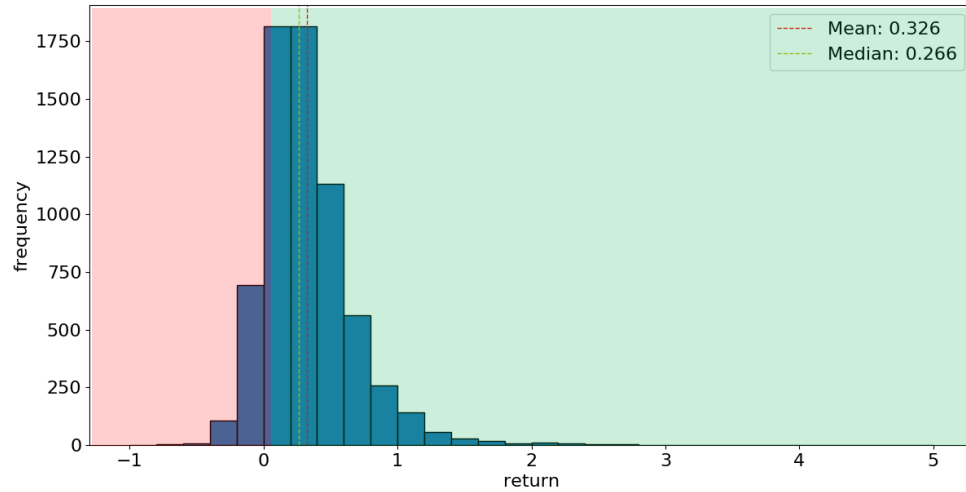
- All possible pairs from 116 NYSE listed Technology sector stocks
- All possible pairs from 155 NYSE listed Energy sector stocks

### Training vs Testing

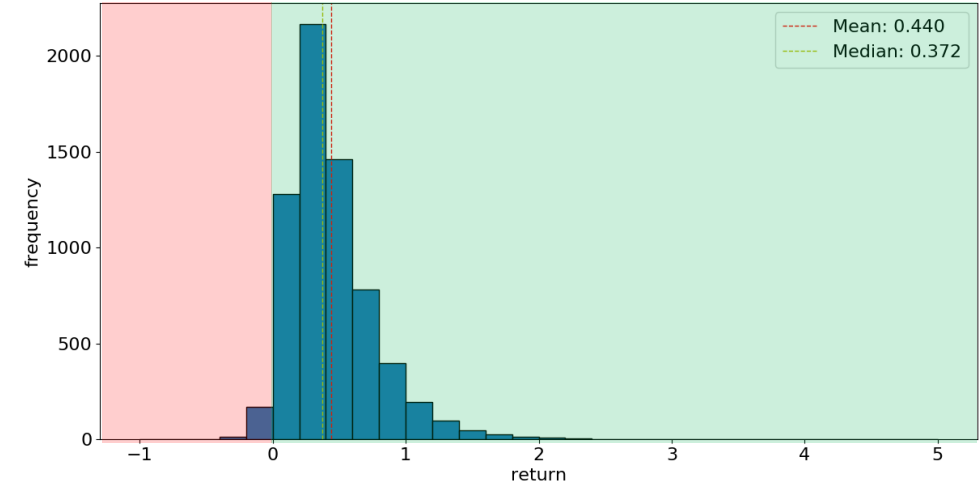
- Train the agent using Technology sector stocks
- Test the agent using both Technology and Energy sector stocks

# Testing Results – Technology Sector Stocks

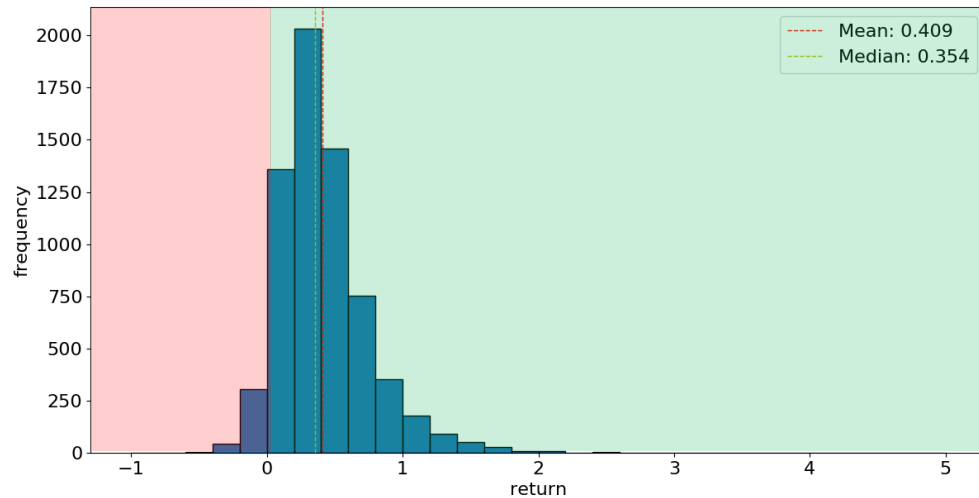
## Returns Distribution of Trading Agent Consistently Positive



Testing on Period 1



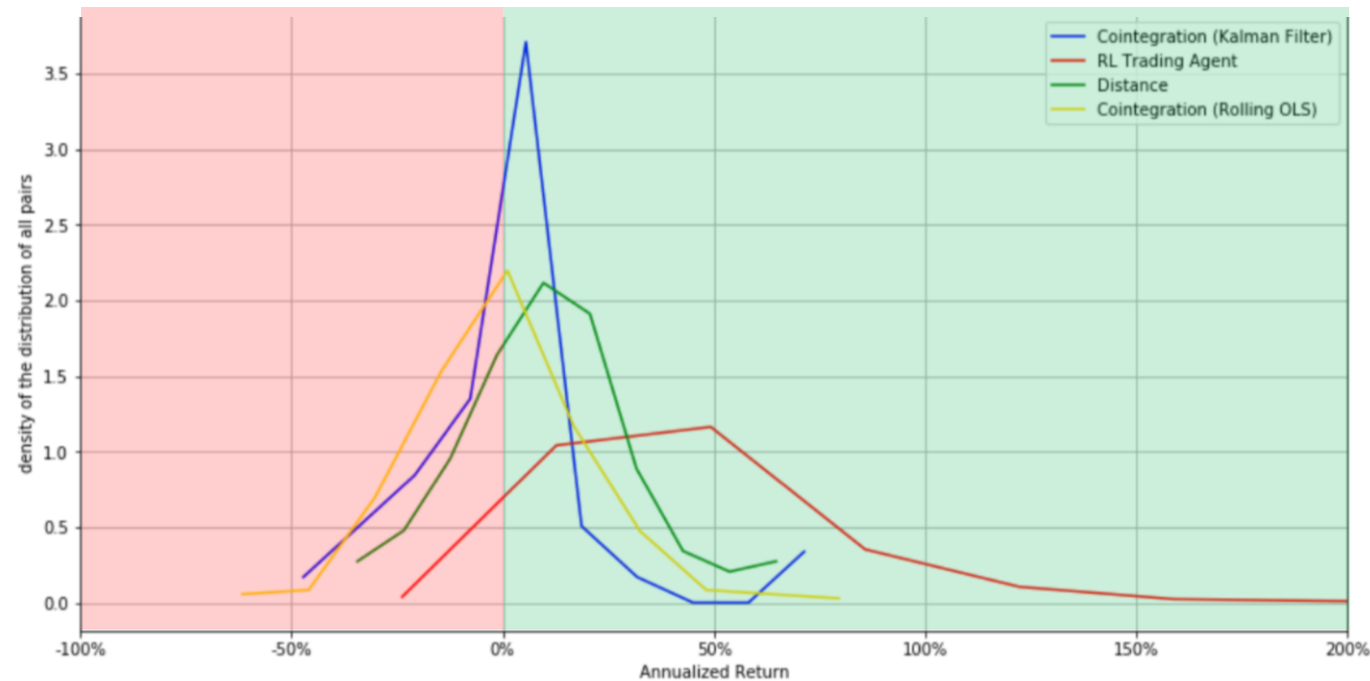
Testing on Period 3



Testing on Period 2

# Testing Results – Comparison of Trading Agent vs Baselines

RL Trading Agent Has Positively Skewed Returns Distribution vs Baseline Strategies



Comparison of returns distribution for RL trading agent vs baseline strategies (in period 3)

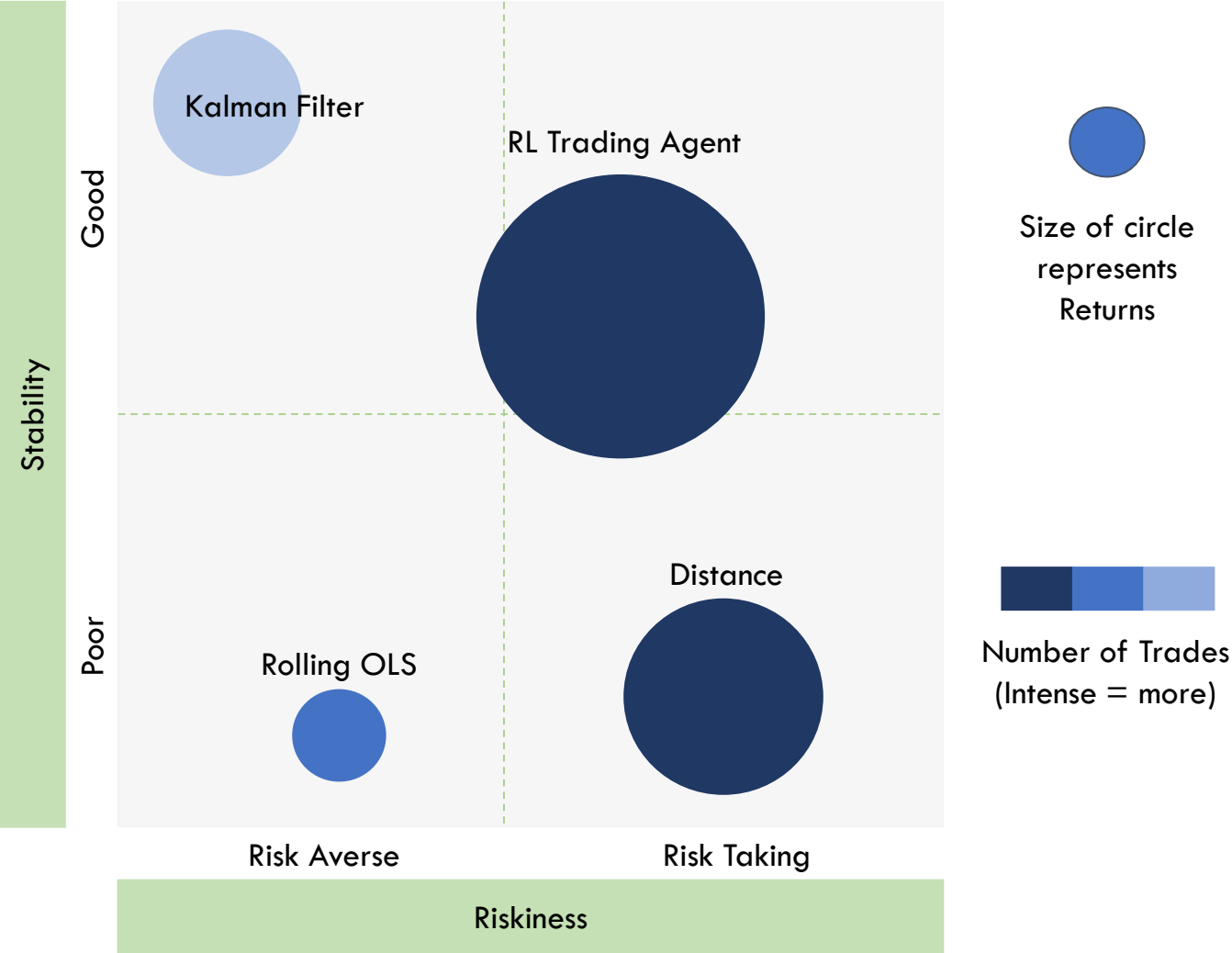
Strategy/Portfolio	Period 1	Period 2	Period 3
Distance	+2.86%	-1.79%	+10.48%
Cointegration (Kalman Filter)	+4.28%	+0.46%	+0.26%
Cointegration (Rolling OLS)	+3.21%	-3.34%	-0.59%
Reinforcement Learning Agent	+32.69%	+40.83%	+43.78%
S&P 500 Index	+9.23%	+12.65%	-7.60%
PSE Index	+14.80%	+18.74%	-7.76%

Summary of mean returns (Tech)

**Key takeaway:** Returns of RL Trading Agent outperforms *both* baseline strategies and market indices

# Analysis of Strategy Behavior

## Ranking and Use Case Conditions



## Behaviour of Strategies

Strategy/Portfolio	Period 1	Period 2	Period 3
Distance	+2.86%	-1.79%	+10.48%
Cointegration (Kalman Filter)	+4.28%	+0.46%	+0.26%
Cointegration (Rolling OLS)	+3.21%	-3.34%	-0.59%
Reinforcement Learning Agent	+32.69%	+40.83%	+43.78%

**RL Trading Agent:** Overall, our proposed RL Trading Agent performs best under general conditions

**Cointegration (Kalman Filter):** Among all the baseline strategies, Kalman can be considered the **most risk averse strategy**, because the returns distribution has very positive kurtosis, making it a low risk low reward strategy.

**Distance:** In contrast to Kalman variant of cointegration, the **returns are highly unstable**. Analysis of our results suggests that the returns are somewhat related to general market volatility.

**Cointegration (Rolling OLS):** This approach has noisy estimation of essential parameters ( $\alpha$  and  $\beta$ ).

# Review of Key Issues of Machine Learning

---

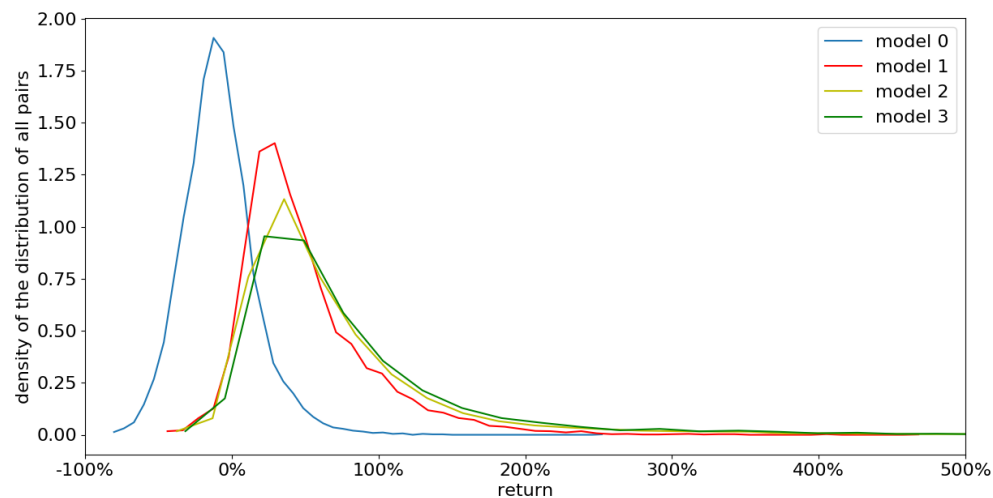
*Learning & Generalization Capabilities of RL Trading Agent*

1. *What has the agent learned?*
2. *Can it generalize to unseen data?*
3. *Can it generalize to other sectors?*

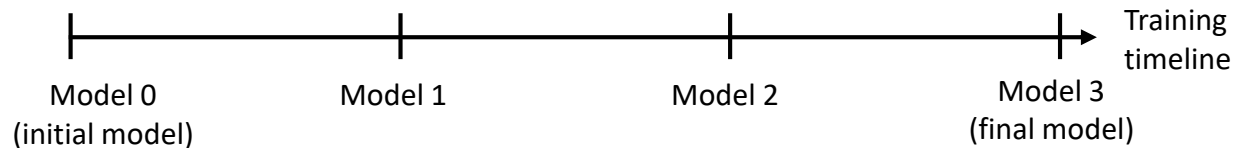
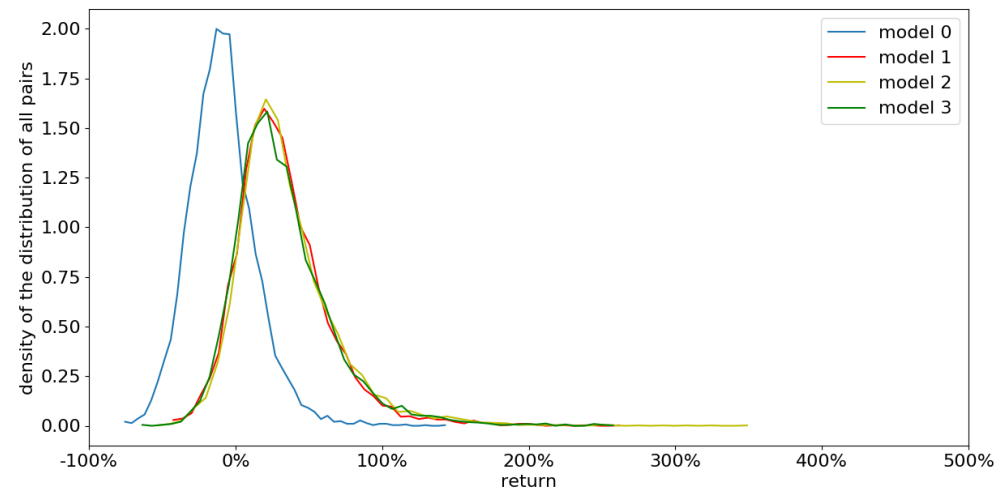
# Review of Key Issues of Machine Learning

## Learning & Generalization Capabilities of RL Trading Agent in Experiment 1 (Technology)

### Learning (training in period 0)



### Generalization (testing in period 1)





# Review of Key Issues of Machine Learning

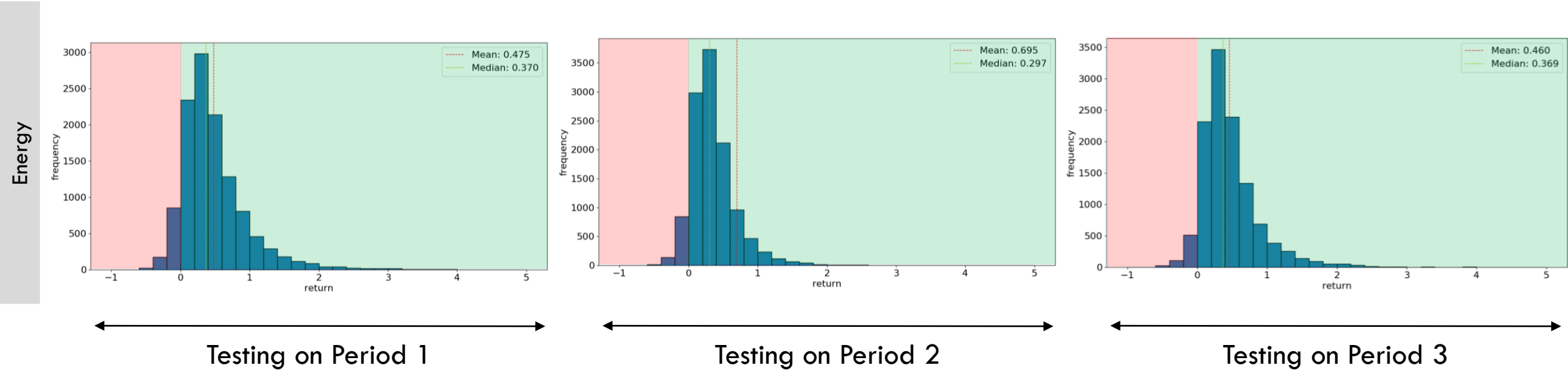
---

*Learning & Generalization Capabilities of RL Trading Agent*

1. *What has the agent learned?*
2. *Can it generalize to unseen data?*
3. *Can it generalize to other sectors?*

# Review of Key Issues of Machine Learning

## Learning & Generalization Capabilities of RL Trading Agent in Energy Sector



Strategy/Portfolio	Period 1	Period 2	Period 3
Reinforcement Learning Agent	+53.80%	+69.60%	+46.02%
S&P 500 Index	+9.23%	+12.65%	-7.60%
NYSE Index	+2.69%	-5.15%	-2.58%

Summary of mean returns (Energy)

# Review of Model Assumptions

*Are the Profits Scalable to Multi-Billion Dollar Funds?*

## Pre-Trade

### Latency of Building Position



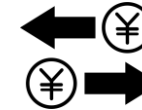
To enter a position of very large size may take long time

### Trading Costs



Realistically, one should buy at ask price and sell at bid; short selling-incurs borrowing cost

### Liquidity



There must be adequate volume traded to transact quickly and capture price

## Post-Trade

### Dividends



Long positions are entitled to dividends; short sells require payment to stock lender

### Borrowing Stability



Legally, a borrower may terminate the stock loan and recall at any time


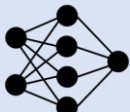

### Portfolio Construction



Netting Effect may occur; Pairs may be correlated to each other

# Conclusion

We made money! (hypothetically)

Scope	Pairs Trading Framework in Statistical Arbitrage Trading		
Areas of Focus	Backtesting	Trading Agent	User Interface
Objectives	<div>1</div> <div></div> <div>Implemented existing strategies as <b>baselines</b> for comparison (Distance, Cointegration)</div>	<div>2</div> <div></div> <div>Developed profitable pairs trading agent using <b>Reinforcement Learning (RL)</b></div>	<div>3</div> <div></div> <div>Implemented a system with <b>User Interface (UI)</b> for visualizing backtest performance</div>
Outcome	<div>i) Created a universal trading agent that outperforms both baseline strategies and market indices.</div> <div>ii) Created a trading agent that can generalize to trade other sectors</div>		

# Appendix: Actual implementation of Pairs Trade Logic in pseudo code

---

*A lot more complex!*

---

**Algorithm 1** Trade logic at time  $t$

---

```
 $S_t \leftarrow$  spread at time  $t$ 
 $\mu_S \leftarrow$  mean of spread, estimated by rolling window
 $\sigma_S \leftarrow$  standard deviation of spread, estimated by rolling window
 $\eta \leftarrow$  a positive number for setting entry threshold level
 $\rho \leftarrow$  a positive number for setting exit threshold level, should be less than  $\eta$ 
 $L \leftarrow$  loss limit, a value for determining when we should give up on a losing trade

if current position is “no position” then
  if  $S_t > \mu_S + \eta\sigma_S$  then
    SHORT the spread
  else if  $S_t < \mu_S - \eta\sigma_S$  then
    LONG the spread
  end if

else if current position is “long spread” then
  if  $S_t > \mu_S + \eta\sigma_S$  then
    EXIT the spread, and then SHORT the spread
  else if  $S_t > \mu_S - \rho\sigma_S$  then
    EXIT the spread
  else if return of trade  $< -L\%$  then
    EXIT the spread
  end if

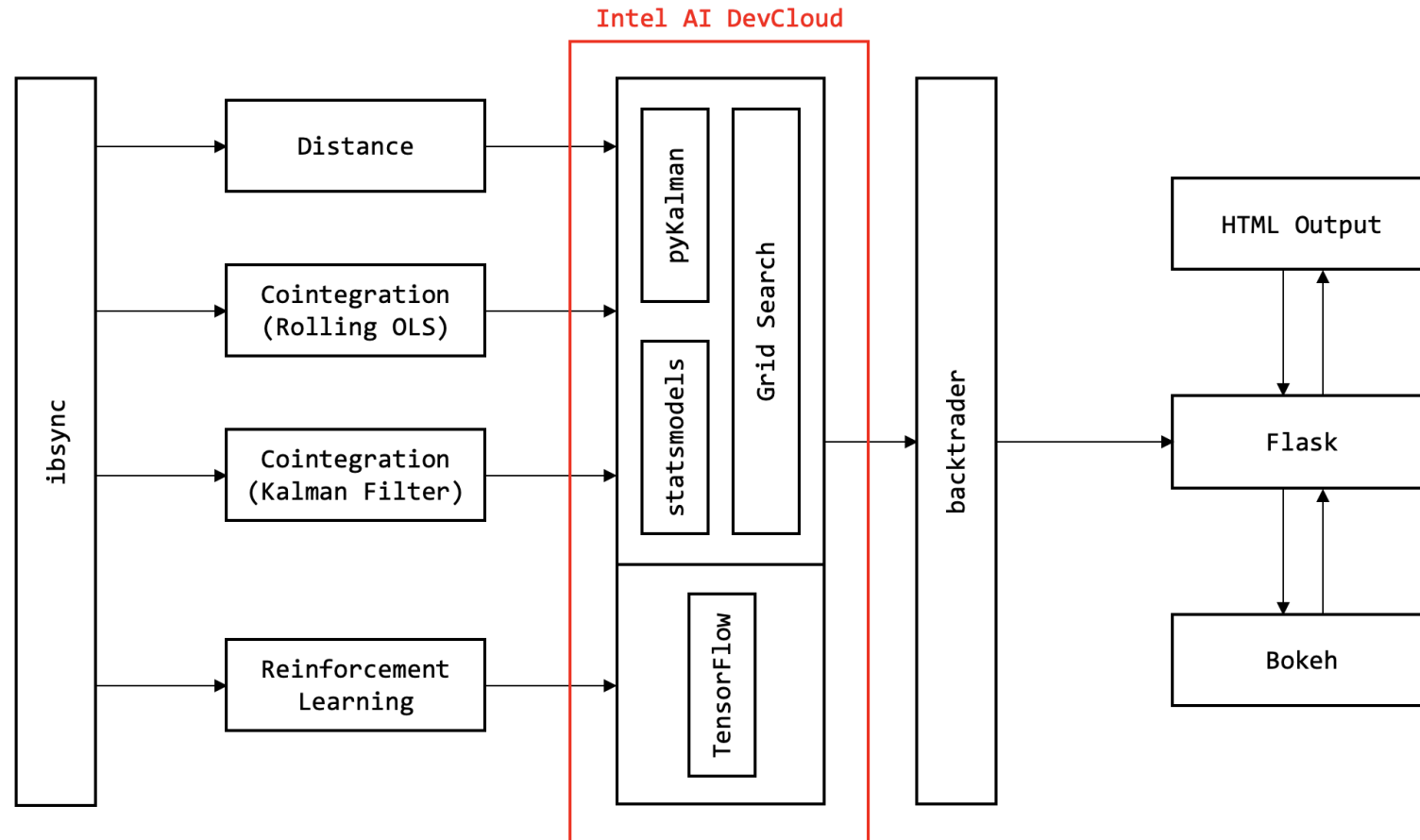
else if current position is “short spread” then
  if  $S_t < \mu_S - \eta\sigma_S$  then
    EXIT the spread, and then LONG the spread
  else if  $S_t < \mu_S + \rho\sigma_S$  then
    EXIT the spread
  else if return of trade  $< -L\%$  then
    EXIT the spread
  end if

end if
```

---

# System Overview

*Unified view of our project*







THE DEPARTMENT OF  
**COMPUTER SCIENCE  
& ENGINEERING**  
計算機科學及工程學系



# Motivation for Reinforcement Learning

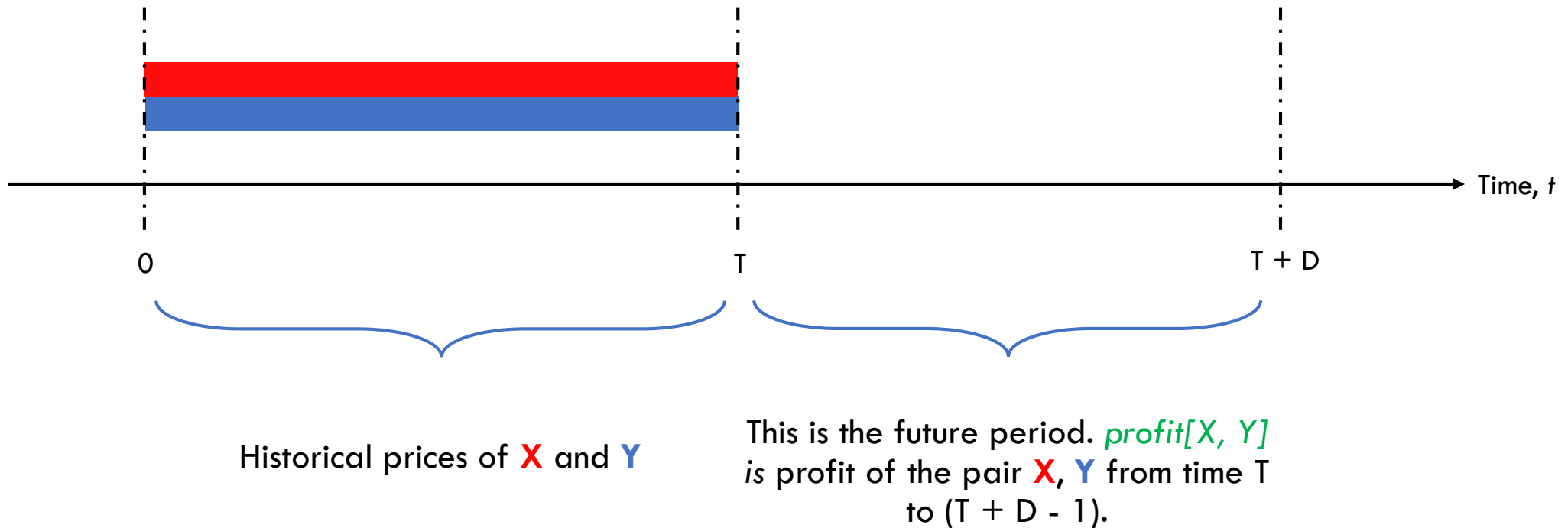
---



# How To Develop A **Novel** Pairs Selection Algorithm ... Using Machine Learning?

Recall that Pairs Trade Strategy = Pairs Selection + Pairs Trade logic

**Initial idea: Train a regressor (supervised learning) to predict the profitability of a pair**  
**Input: historical prices of  $X$  and  $Y$ , Output:  $\text{profit}[X, Y]$**



**Regression model = scoring function for pairs.**

**The higher the predicted future profit (score), the better the pair !**

# How to prepare the training data?

---

The training data should be of the form  $\langle \text{historical prices of stock } X, \text{ historical prices of stock } Y, \text{ profit}[X, Y] \rangle$ .

## Features

$\langle \text{historical prices of stock } X, \text{ historical prices of stock } Y \rangle$  is just a segment of the historical prices:

$\text{historical prices of stock } X = \langle X[0], X[1], X[2], \dots, X[T-1] \rangle$

$\text{historical prices of stock } Y = \langle Y[0], Y[1], Y[2], \dots, Y[T-1] \rangle$

where  $T$  is a predetermined value (eg. 200).

## Labels

$\text{profit}[X, Y]$  can be the annualized return earned by trading the pair  $X, Y$  when **backtested** over a future period:

- from time  $T$  to  $(T + D - 1)$

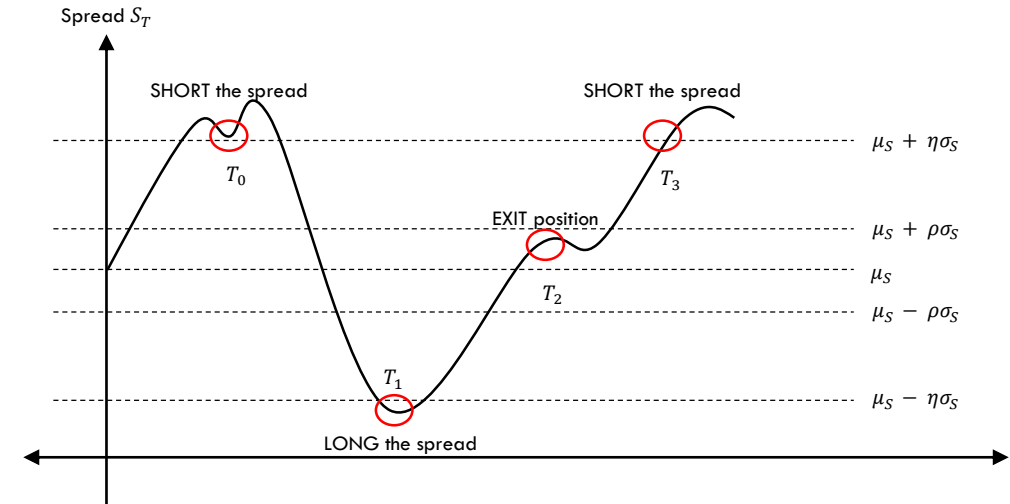
where  $D$  is a predetermined value (eg. 200).

# Problem: Which Pairs Trade Logic do we use for the backtesting?

Remember this Pairs Trade Logic? The one we used for baseline Pairs Trading Strategies:

*“If spread is far away from its mean (eg. outside  $\pm 2.0$  standard deviations), this means that spread has diverged, then we enter position by betting that the spread will converge in the future.”*

*“Keep holding and wait until spread converges back to its mean (eg. within  $\pm 0.5$  standard deviations), then unwind positions to take profit.”*

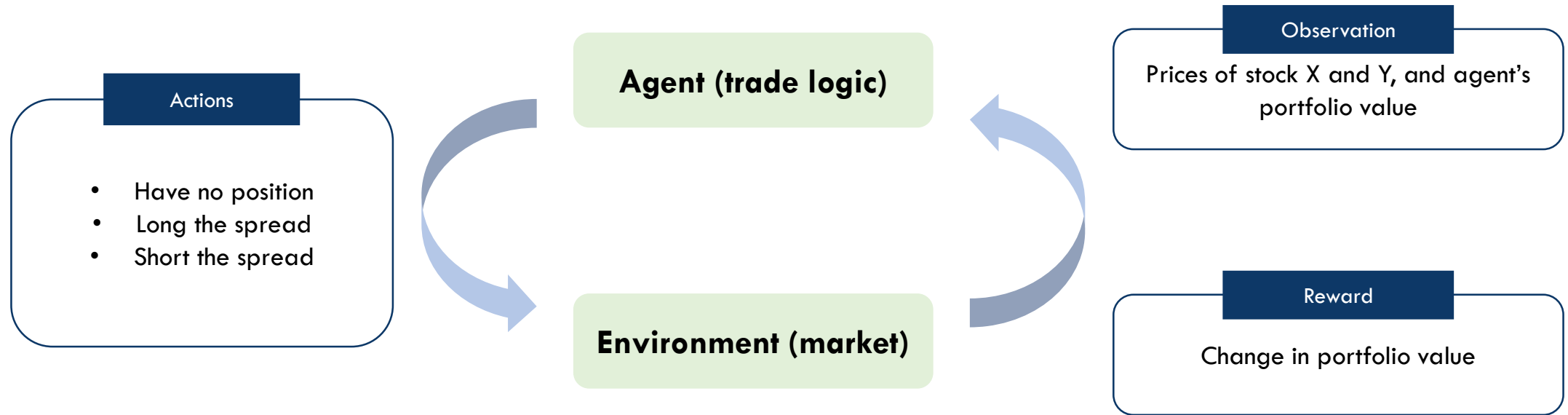


The problem with this trading logic is that, **does not maximize the potential profit for any arbitrary pair.**  
**(if we label data wrongly, even the best classifier might not give the true best pair)**

- We need a Pairs Trade Logic which **makes the optimal decision for any arbitrary pair!**
- In other words, the desired Pairs Trade Logic should maximize the expected profit of any arbitrary pair.

# New Idea: Train a Universal Trading Agent!

*The RL Trading agent will then act as our “desired” Pairs Trade Logic.*



**The “core” part of our FYP is how to design and train such a RL trading agent.**

*Actual setup of RL trading agent in next few slides.*

# Reinforcement Learning Trading Agent – Setup

*Mapping from textbook example (Atari game) to pairs trading*



Atari Breakout



Pairs Trading Problem (one pair of stocks at a time)

# Reinforcement Learning Trading Agent – Setup

*How does the simplest pairs trading problem looks like?*

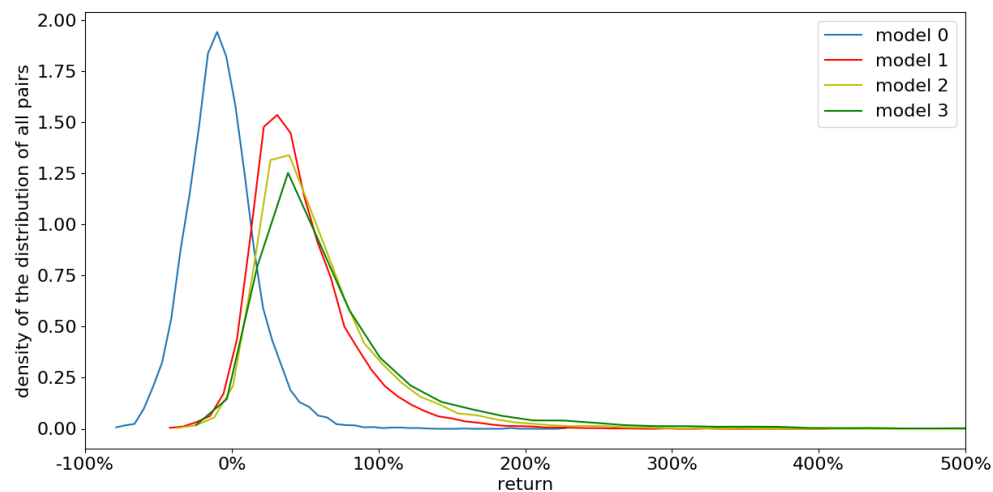
	Atari Breakout	Pairs Trading Problem (One Pair)
Duration of a game?	When the ball fell below the bottom line, or all bricks were broken	Fixed (200 trading days)
What the agent can observe?	One frame at a time	Prices of the two stocks and the current portfolio value in a trading day.
What action can the agent take?	Don't move, left, or right	No position, buy the spread, or sell the spread
Environment transition?	Frame transition with fixed frequency	Trading day transition
What is the reward after an action was taken?	The change in score	The change in portfolio value
Fully observable? (Does the agent observe enough information at any given time to make an optimal decision?)	No. If the agent only observes one frame at a time <u>without memory</u> , the agent cannot know the full dynamics of the ball (velocity and acceleration).	No. The agent only observes the price of two stocks but have no idea whether the spread is overpriced, underpriced, or just in the normal level.

To solve the problem of partial observability, the agent may need to consider past observations and actions.  
A suitable model architecture for the agent would be an Recurrent Neural Network (RNN) which can summarize a variable length sequence of inputs into a fixed dimension vector.

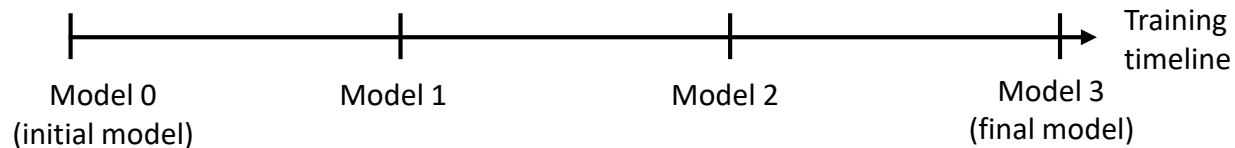
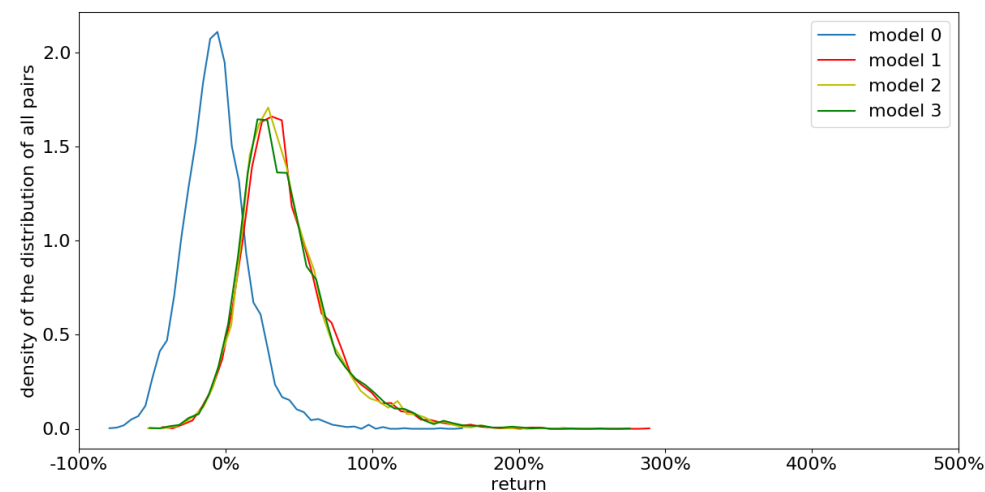
# Review of Key Issues of Machine Learning

## Learning & Generalization Capabilities of RL Trading Agent in Experiment 2 (Technology)

### Learning (training in period 0, 1)



### Generalization (testing in period 2)

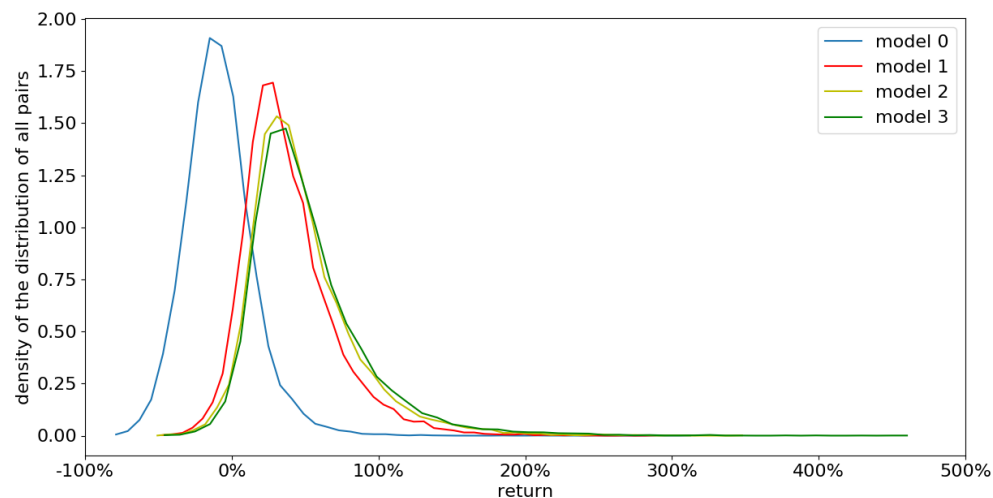




# Review of Key Issues of Machine Learning

## Learning & Generalization Capabilities of RL Trading Agent in Experiment 3 (Technology)

### Learning (training in period 0, 1, 2)



### Generalization (testing in period 3)

