

# **Statistical Decision, Reinforcement Learning and Finance**

Version 2.0

---

Dr. Ye Luo  
HKU Business School  
Jan 2025

## Statistical Decision Process

- State Variable  $S_t$ , decision  $a_t$ , and decision policy  $a_t = P(S_t)$ .
- A reward (utility)  $R(S_t, a_t)$  is collected at time  $t$ .
- $P(\cdot)$  is a function that is parametrized by  $P_\theta(S)$ ,  $\theta$  is a parameter that needs to be learned by solving:

$$\theta^* = \operatorname{argmax}_{\theta \in \Theta} \sum_{i=1}^T R(S_t, P_\theta(S_t)),$$

- The state variable  $S_t$  can include but not limited to:
  1. Market data: Past information on prices and trading volumes.
  2. News and Events: past news and events being triggered.
  3. Macro economic state variables, such as interest rate, unemployment, GDP growth rate, exchange rates, etc.
- The variable  $S_t$  must be obtained before  $t$ .
- Action  $a_t$ : the portfolio weight to hold at time  $t$ .
- Reward: the portfolio return  $R_t$  at time  $t$ . May adjust to cost.

## Mean-Variance Policy

- Investor has mean-variance utility (reward)

$$R(w) = w^T \mu - \frac{\gamma}{2} w^T \Sigma w.$$

- $w$  is  $N \times 1$  vector of asset allocation weights.  $\gamma$  is risk aversion parameter.  
Aggregate  $\gamma \approx 2$ .
- Optimal decision rule is  $w^* = \pi(\Sigma, \mu) = \frac{1}{\gamma} \Sigma^{-1} \mu$ .
- This is a special case of statistical decision, with state variable  $= (\Sigma, \mu)$ .

## Different Approaches to Learn a Policy Function

- Directly solving the policy optimization problem can be difficult.
- Could lead to overfitting if the policy function is too complex.
- One solution is the Prediction - decision process. The other one is to perform decision directly.

## The Prediction - Decision Process

- Construct a prediction model of asset returns  $R_{it}$  based on predictors  $W_{it}$ .  
For example, linear regression:

$$\hat{\beta}_i = \operatorname{argmin} \sum_{t=1}^T (R_{it} - W_{it}^\top \beta_i)^2.$$

- Often leads to overfit as time series  $T$  is short.  $T \sim 100$  to 1000.
- Parameter Sharing:  $\beta_i = \beta$  - this puts a more robust estimation of  $\beta_i$ .  
Ideally we could involve many predictors. In this case, LASSO and  
Boosting can be used.
- Namely, we solve:

$$\hat{\beta} = \operatorname{argmin} \sum_{i=1}^N \sum_{t=1}^T (R_{it} - W_{it}^\top \beta)^2.$$

## The Decision Process

- Suppose your prediction model  $\hat{R}_{i,T+1} := W_{i,T+1}\hat{\beta}_i$ , where  $W_{i,T+1}$  is the state variable that you can obtain before time  $T + 1$ .
- Let

$$a_{T+1} = (a_{1,T+1}, \dots, a_{N,T+1})$$

be the vector of assets to long from time  $T$  to  $T + 1$ .

- Simple cut-off rule for a long only strategy:

$$a_{i,T+1} = \frac{1}{q_T} \mathbf{1}(W_{i,T+1}\hat{\beta} > c),$$

where  $q_T$  is a re-scaling factor, for some number  $c > 0$ . It means that you will long the stocks with predicted return  $> c$ .

- For equally weighted portfolio (the naive  $1/N$  strategy), we can adopt

$$q_T = \sum_{i=1}^N \mathbf{1}(W_{i,T+1}\hat{\beta} > c).$$

## What's in the Prediction-Decision Process

- A prediction model is constructed first. The better quality the prediction is, the better the model is. In our case, we would like to learn a robust model that delivers good out-of-sample MSE.
- A decision is made based on the “cut-off” rule of the prediction model. In many cases, the “cut-off” rule is widely used.
- The parameter to choose is the cut-off rule  $c$ , which can be learned and optimized using past data as well.
- Essentially, the cumulative return of the strategy

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

is dependent on the cut-off rule  $c$ . We can choose  $c$  to maximize in sample cumulative return.

## Backtesting: out-of-sample test

- One way to conduct a reasonable test is to learn the policy using the training data set:  $t = 1, 2, \dots, T_{train}$ , and then test the policy using the testing data set:  $t = T_{train} + 1, \dots, T$ .
- We focus on the out-of-sample evaluation of policy in the testing data, using finance metrics such as annualized return, Sharpe ratio, maximum drawdown, etc.
- This approach have the danger of overfitting when the hyper parameters are tuned in order to have better out-of-sample performance. Also, it may simply hit the chance that the pattern learned only works for the past training data rather than more recent testing dataset, or vice-versa, leading to unexpectedly good result.

## Robustness: The Rolling-Window Testing Approach

- One way to improve the robustness is to perform this out-of-sample test for many times.
- For example, we can define a rolling window with length  $K$  as:

$$t + 1, \dots, t + K.$$

Within this rolling window, the first  $\tilde{K}$  data, e.g.,  $\tilde{K} = 0.8K$  is used as training sample, and the rest  $K - \tilde{K}$  data are used as testing sample.

- Then, the out-of-sample performance in the testing sample

$$t + \tilde{K} + 1, \dots, t + K$$

is evaluated.

## Robustness: The Rolling-Window Testing Approach

- The entire testing process is divided into a bunch of rolling window out-of-sample tests.
- The window moves forward at a step size of  $Q$ . So, the next test uses data

$$t + Q + 1, \dots, t + Q + K.$$

- This time, we produce predictions and evaluations on the sample:

$$t + Q + \tilde{K} + 1, \dots, t + Q + K.$$

- The test in the previous window takes data point from  $t + \tilde{K} + 1, \dots, t + K$ . So, it is important to set  $t + Q + \tilde{K} + 1 = t + K + 1$  to allow the test results to exist consecutively.

## Robustness: The Rolling-Window Testing Approach

- For example, suppose in total we have  $T = 5000$ . Each rolling window last for  $K = 500$ . Rolling window moves forward at a step of  $Q = 100$ .
- $\tilde{K}$  is set as 400 - it means that in each window with  $K = 500$  samples, the first 400 are the training sample, and the rest 100 data points are the testing sample.
- The window moves forward at a step size of 100. The first window ranges from sample point 1 to 500, with testing sample from 401 – 500.
- In total there will be 46 testing windows, and the rolling-window out-of-sample test starts from the 401<sup>th</sup> sample to the 5000<sup>th</sup> sample. The first 400 samples do not have out-of-sample testing results as they are used as the initial training sample.

# A Rolling Window Test Result Using A-share Market Assets



Figure 1: A Trained Strategy After Transaction Cost

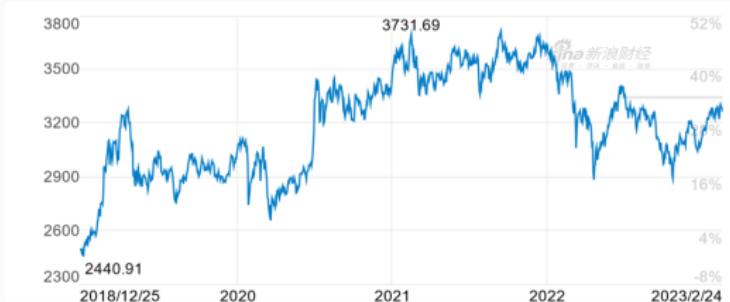
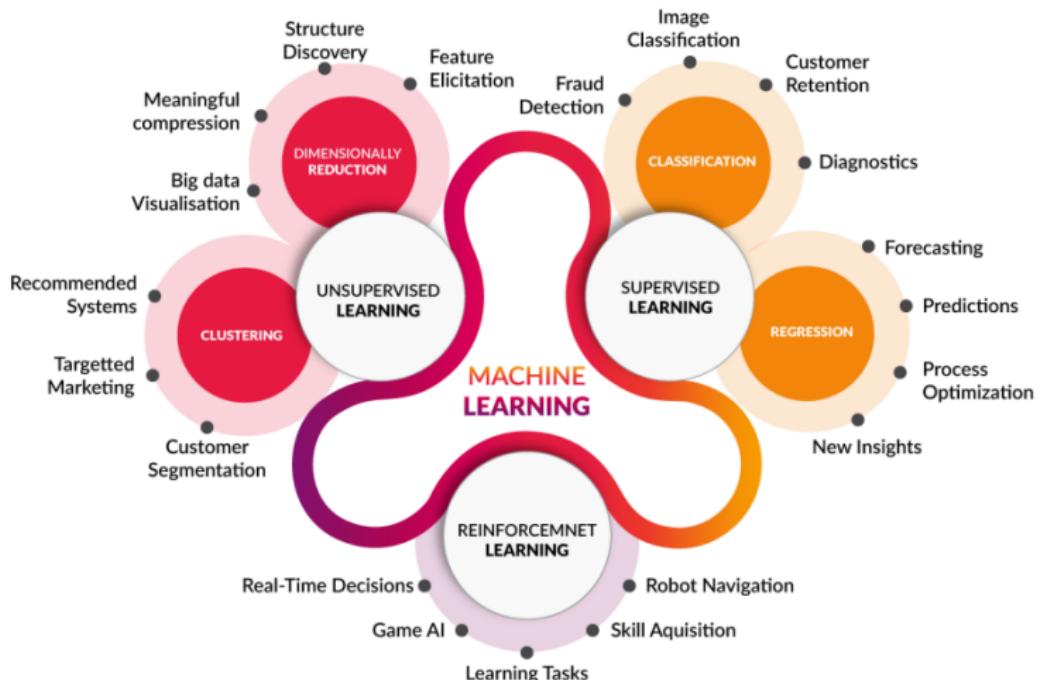


Figure 2: SH-A index

# Class of MLs



## Direct Approach

- $R(s, a)$  represents the reward function at state  $s$ , with action  $a$ . In finance,  $R(s, a)$  represents the return of assets at state  $s$ , with decision  $a$ .
- A trading strategy is a policy  $\pi : S \mapsto \mathcal{A}$ .
- We attempt to learn  $\pi$  from past data in order to maximize Sharpe ratio or cumulative reward.

## A Markov Decision Process (MDP)

- At time  $t$ , a state variable  $S_t$  is observed, then, an action  $a_t \in \mathcal{A}$  is taken by the decision maker.
- The decision maker chooses the policy  $a_t$  by a policy function  $\pi_\theta(s_t)$ , where  $\theta$  is a parameter that needs to be optimized. For example:

1. Linear policy:

$$a_t = s_t^\top \theta.$$

2. Binary policy:

$$a_t = 1(s_t^\top \theta > c).$$

- A reward function  $R(s_t, a_t)$  is revealed by the nature, and collected by the decision maker.

- State Transition:  $S_t, A_t \mapsto S_{t+1}$ . Transition probability:

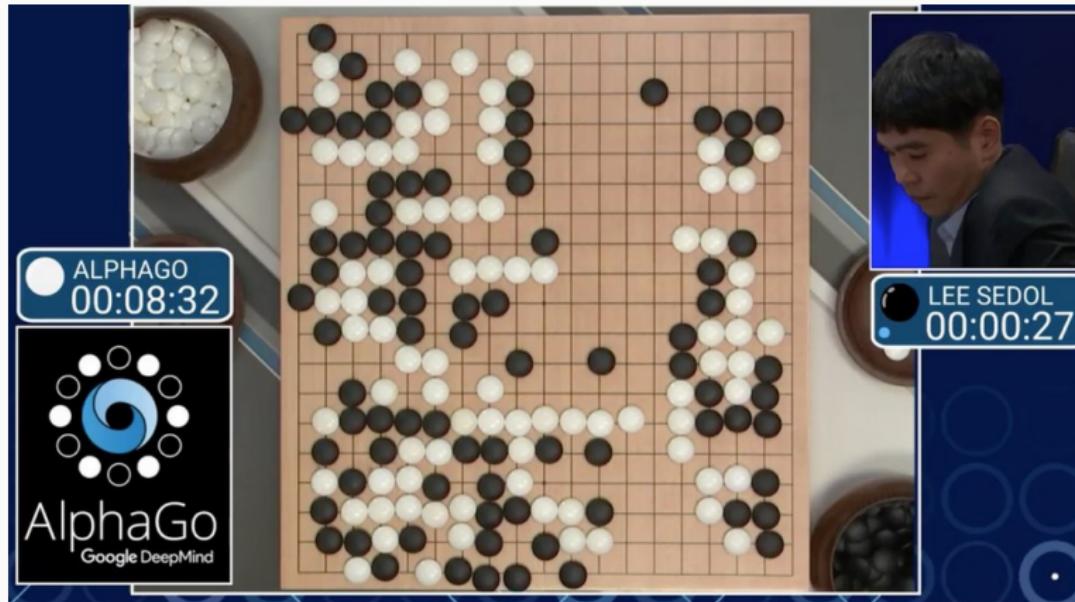
$$P(S_{t+1}|S_t, A_t).$$

- The next state may be affected by your action. For example, your order may affect market structure, and leads to different prices afterwards. Or your current portfolio allocation may affect your future transaction cost.
- Therefore, there exists a strategic effect of the current action on the future states.
- Agent cares about cumulative reward

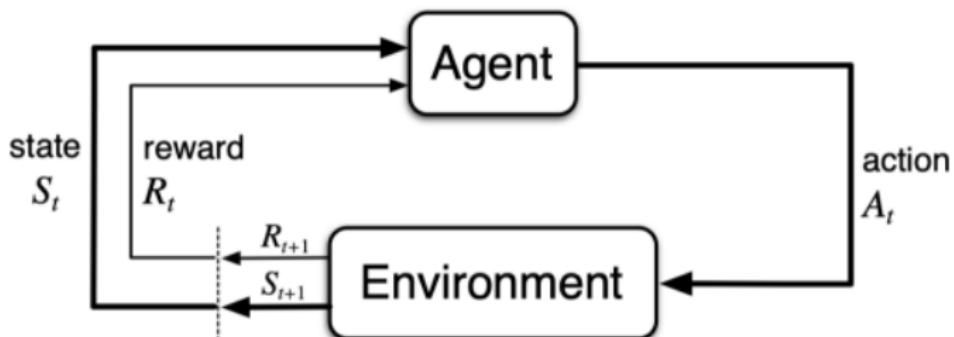
$$\sum_{t=0}^T \gamma^t R(S_t, A_t).$$

- $\gamma$  here represents the discount rate.

## A real example



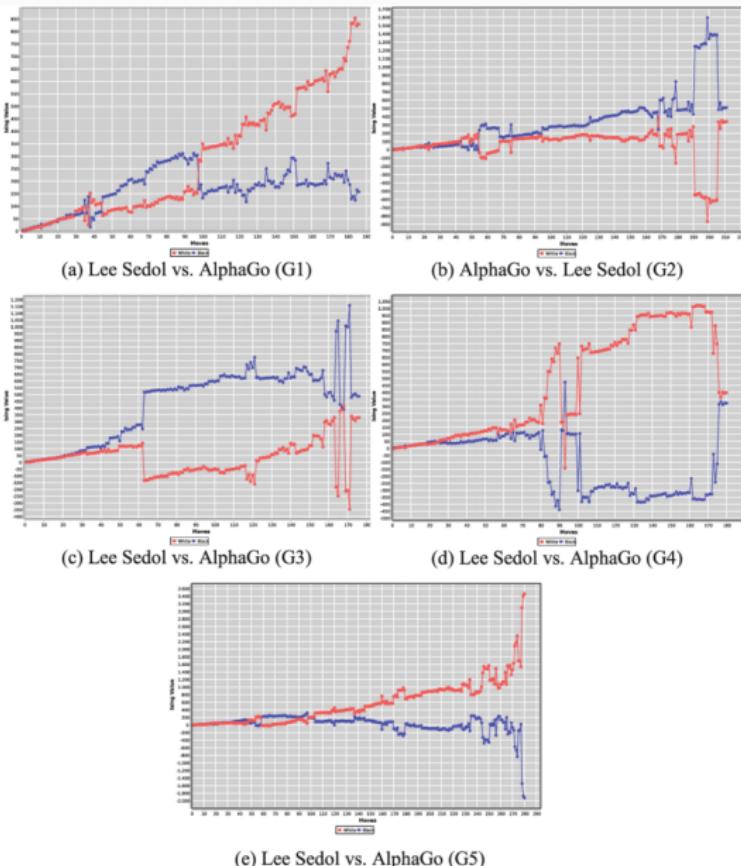
## A MDP environment



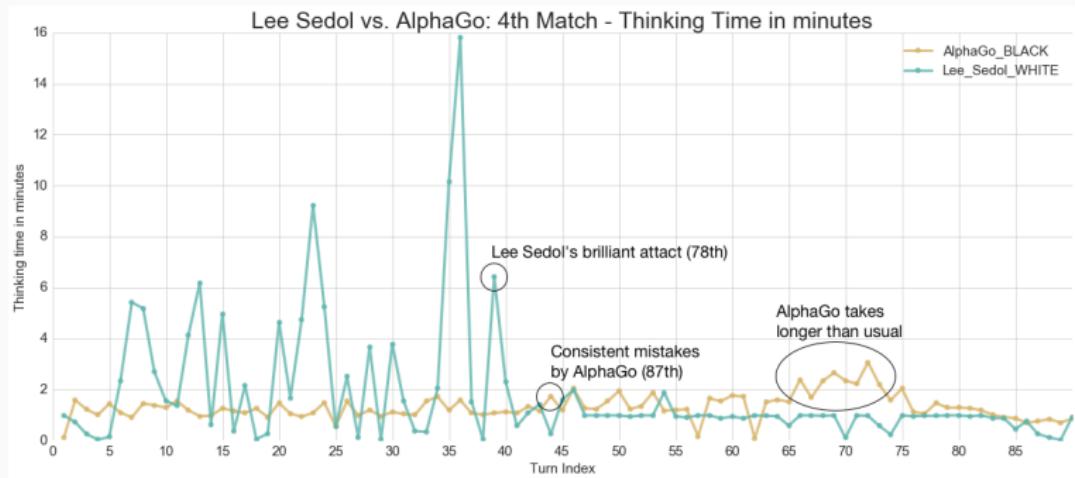
## State, Actions, Transition and Reward in AlphaGo

- State is the allocations of stones on the board.
- Action is the next play.
- Transition is defined by the rules of the game of Go.
- Reward is the expected probability of winning in the end. The cumulative reward is captured by “Value”.

# Value Comparison



# Thinking Time



## MDP as a strategic feedback system

- Often times the problem is to search for where the reward locates.
- The reward function and the future reward all together defines the quality of policy.
- This is captured by the “Q-value” function.

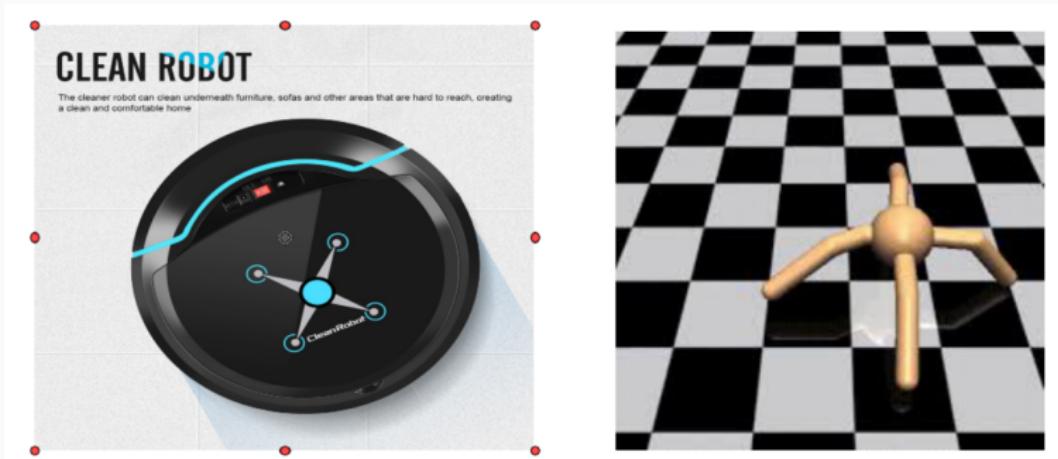
$$Q^*(s, a) = \mathbb{E}[R(s, a) + \gamma \max_{a', s' | a, s} Q^*(s', a')].$$

- These conditions are recursive, and they are referred as the “Bellman” equation.
- $R(s, a)$  represents the immediate payoff.  $\max_{a', s' | a, s} Q^*(s', a')$  represents the future payoff.
- $Q^*(s, a)$  captures the value of action  $a$  under state  $s$ .

## Examples



## Examples



## Other concepts

- $V^*(s) := \max_{a \in \mathcal{A}} Q^*(s, a)$  represents the value of state  $s$ , named as “V-value” function.
- $\pi^*(s) = \operatorname{argmax}_{a \in \mathcal{A}} Q^*(s, a)$  represents the optimal policy. The optimal policy  $\pi^*(\cdot)$  chooses the best action  $a$  to maximize  $Q^*(s, a)$  under state  $s$ .
- $V^*, Q^*$  are value functions at optimal policy. Learning  $\pi^*$  is essentially the same as learning  $V^*, Q^*$ .

## Policy Evaluation

- Policy evaluation: for a given policy  $\pi$ , we can define the policy-related value function as:

$$Q^\pi(s, a) = \mathbb{E}_{a' \sim \pi(s'), s' | s, a} [R(s, a) + \gamma Q^\pi(s', a')]$$

$$V^\pi(s) = \mathbb{E}_{a \sim \pi(s)} Q^\pi(s, a).$$

- This function evaluates the quality of the policy  $\pi$ , and we have that:

$$Q^*(s, a) = \max_{\pi} Q^\pi(s, a), V^*(s) = \max_{\pi} V^\pi(s).$$

- Expanding the recursive condition, we have that:

$$Q^\pi(s_0, a_0) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_{a_t \sim \pi(s_t)} [R(s_t, a_t)],$$

which is the cumulative reward achieved by  $\pi$ .

## The Core of Modern AI - Reinforcement Learning

- The MDP problem reduces to learn/estimate the value function and policy function,  $Q$  and  $\pi$ . For policy optimization, we attempt to learn  $Q^*$  and  $\pi^*$ .
- Reinforcement Learning is a class of Algorithms that learns  $Q$  and  $\pi$ .
- This kind of algorithms provide the base foundations of the modern Artificial Intelligence.
- It's also known as statistical control (decision) theory.

## Basic Algorithms: Temporal Difference

- The temporal difference algorithm is aimed at evaluating  $Q^\pi(s, a)$ .
- Behavior policy is fixed at  $\pi$ .
- The Bellman's equation sets:

$$\mathbb{E}[R(s_t, a_t) + \gamma Q^\pi(s_{t+1}, a_{t+1}) - Q^\pi(s_t, a_t)] = 0.$$

- Temporal Difference Algorithm (TD(0)) on  $Q$ , updating equation:

$Q^\pi(s_t, a_t)$  is updated as:  $Q^\pi(s_t, a_t) + \alpha_t(R(s_t, a_t) + \gamma Q^\pi(s_{t+1}, a_{t+1}) - Q^\pi(s_t, a_t))$ ,

where  $\alpha_t$  is the learning rate.

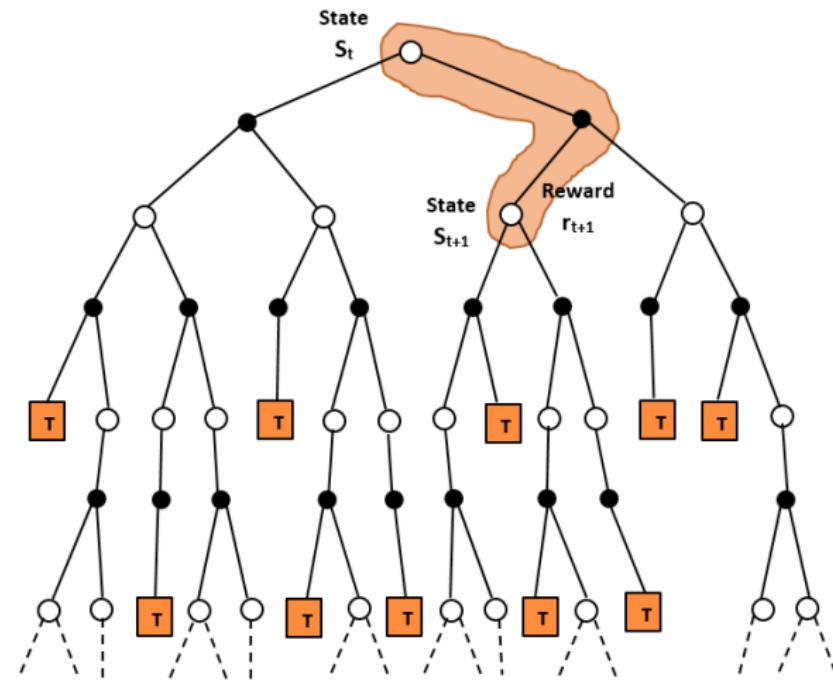
- TD(0) on  $V$ :

$V^\pi(s_t)$  is updated as:  $V^\pi(s_t) + \alpha_t(R(s_t, a_t) + \gamma V^\pi(s_{t+1}) - V^\pi(s_t))$ .

# Basic Algorithms: Temporal Difference

## Temporal Difference Learning

$$V(S_t) = V(S_t) + \alpha (R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$



## Basic Algorithms: Q-Learning

- The Q-Learning aims to learn  $Q^*(s, a)$  function rather than  $Q^\pi$ .
- $Q^*(s_t, a_t)$  is updated as:

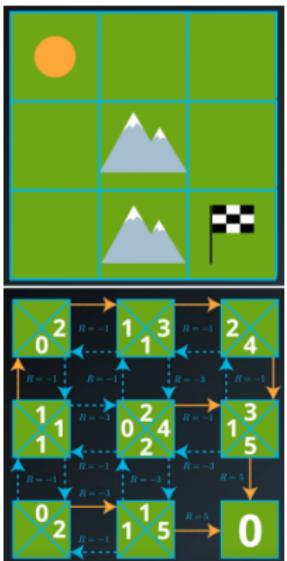
$$Q^*(s_t, a_t) \leftarrow Q^*(s_t, a_t) + \alpha_t (R(s_t, a_t) + \gamma \max_{a_{t+1} \in \mathcal{A}} Q^*(s_{t+1}, a_{t+1}) - Q^*(s_t, a_t)).$$

- The Q-learning gives estimates of the  $Q$  function directly, and therefore, leads to optimal policy that:

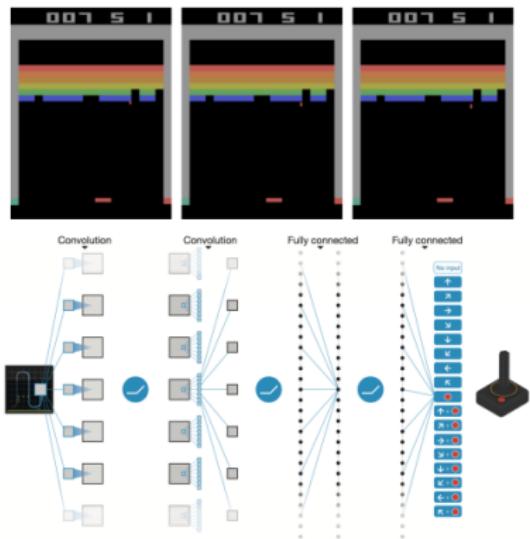
$$\pi^*(s) := \operatorname{argmin}_a Q^*(s, a).$$

# Q-Learning

Tabular



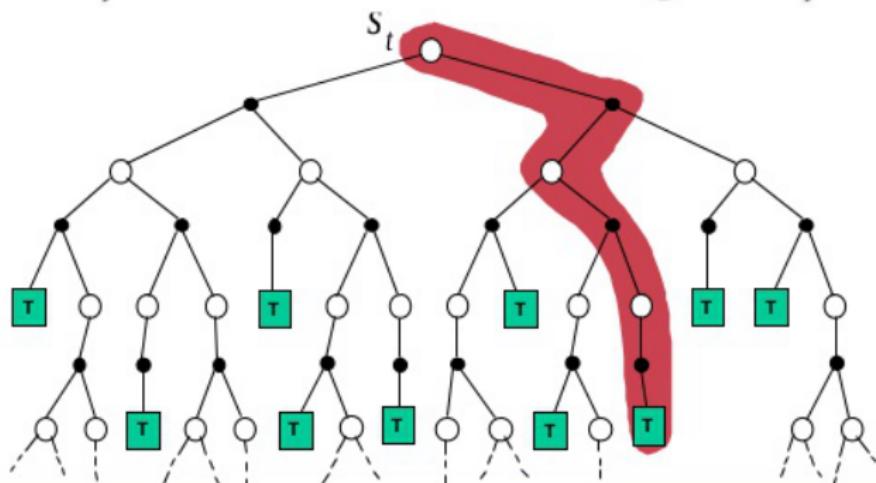
Deep neural networks



## Simple Monte Carlo

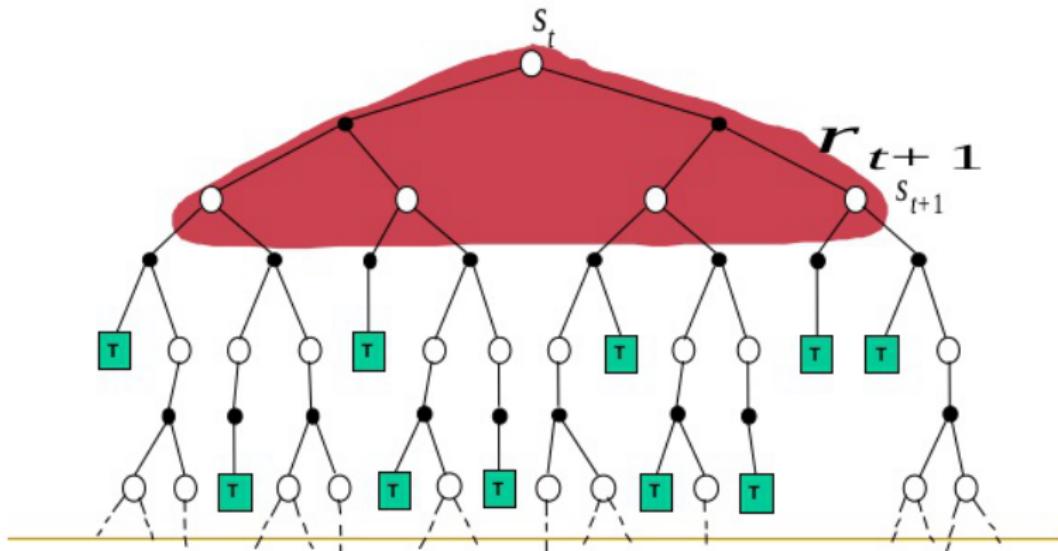
$$V(s_t) \leftarrow V(s_t) + \alpha [R_t - V(s_t)]$$

where  $R_t$  is the actual return following state  $s_t$ .



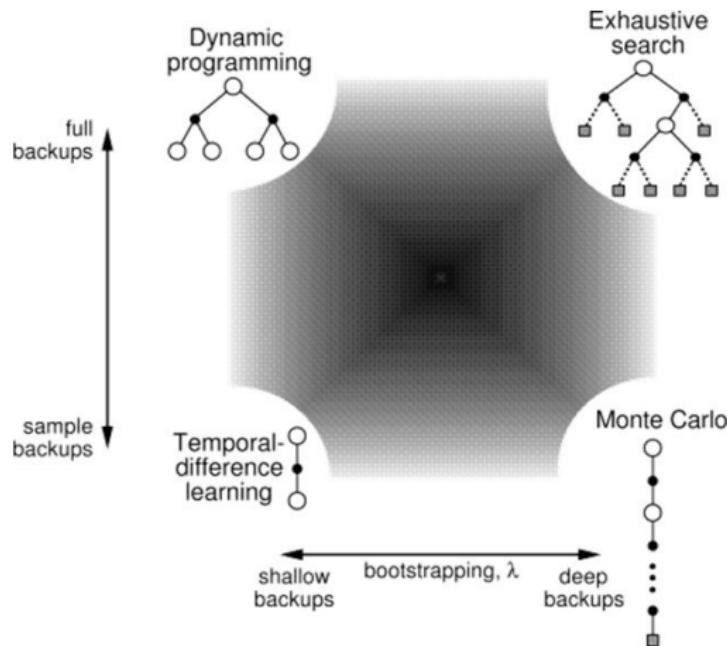
## Dynamic Programming

$$V(s_t) \leftarrow E_{\pi}\{r_{t+1} + \gamma V(s_t)\}$$



# Different Scopes of Reinforcement Learning

## Dimensions of Reinforcement Learning



# Complex Control Problems Solved by RL



# Winning Probability Table

Table 1: Evaluation Results.

Opponent's Type	Non-cheating (Training)							Cheating (No-training)		
	1	2	3	4	5	6	7	8	9	10
Difficulty Level	1	0.99	0.94	0.99	0.95	0.88	0.78	0.70	0.73	0.60
Combat Rule	1	0.99	0.45	0.47	0.39	0.73	0.66	0.56	0.52	0.41
Combat Network	0.98	0.99	0.45	0.47	0.39	0.73	0.66	0.56	0.52	0.41
Mixture Model	1	1	0.99	0.97	1	0.90	0.93	0.74	0.71	0.43

## Case Study: LOXM

- LOXM uses a trained **neural network** to optimize between liquidity demand and passive trading, based on the prevailing market state, adapting as market conditions change.
- Using **reinforcement learning** techniques, LOXM can analyze millions of simulated transactions in light of historical data to zero in on hard-to-spot optimal behaviors.
- LOXM delivered significant savings and far outperformed both manual and automated existing trading methods in trials <sup>1</sup>.

## Case Study: LOXM

### Quote

"In traditional quant finance, we try to be as general as possible. We are trying to develop theories that resemble physics in some respect. This works great to an extent, but we lose the idea that markets are very specific, they are very competitive and they are Darwinian. With AI, we can do better, we can create things that handle very specific situations very well."

*Vaslav Glukhov, Head of EMEA e-Trading Quantitative Research at JPMorgan*

## Case Study: LOXM

- Limit order placement
- Actions: how much to place, what price, and for how long
- Algorithm: standard reinforcement learning <sup>3</sup>
- Some concerns
  - Explainability: important to regulators and customers
  - Performance under increasing competitions: Passive strategies are now 35% of the volume traded in the US market. That figure was 5% 10 years ago.

## Case Study: RL on Quantitative Trading

- Looking for patterns from historical market data, such as price, trading volume, high, low, etc.
- Try to associate these patterns with trading decisions.

Deng, Y., Bao, F., Kong, Y., Ren, Z., & Dai, Q. (2017). Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems*, 28(3), 653-664.

Team: Department of Automation, Tsinghua University

*Cited by 72 (up to Mar 5, 2019, according to Google Scholar)*

- Raw input: price sequences  $p_1, p_2, \dots, p_t, \dots$
- Reward  $z_t = p_t - p_{t-1}$
- Action (discrete)  $\delta_t \in \{\text{long, neutral, short}\} = \{1, 0, -1\}$
- Profit  $R_t = \delta_{t-1} z_t - c |\delta_t - \delta_{t-1}|$ , where  $c$  is the transition cost
- Return  $U_t = \sum_{t=1}^T R_t$  or  $U_t = \frac{\text{mean}(R_t)}{\text{std}(R_t)}$

## Policy Function as Deep Neural Network

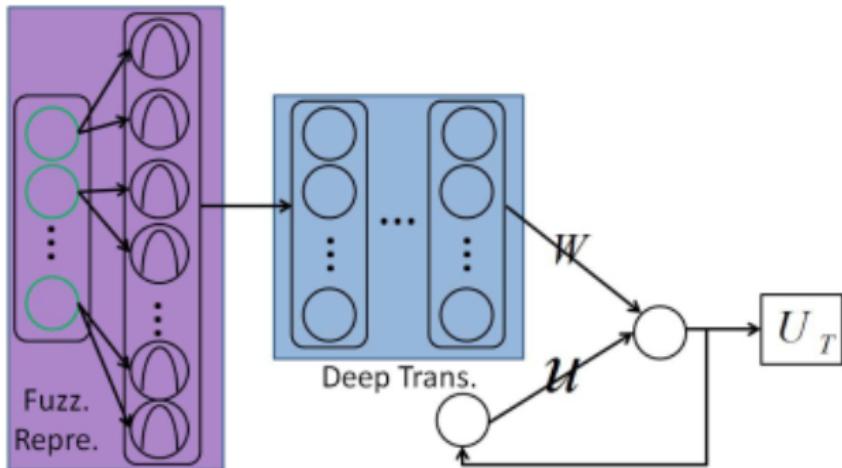


Fig. 2. Overview of fuzzy DRNNs for robust feature learning and self-taught trading.

# Training/Testing Setup

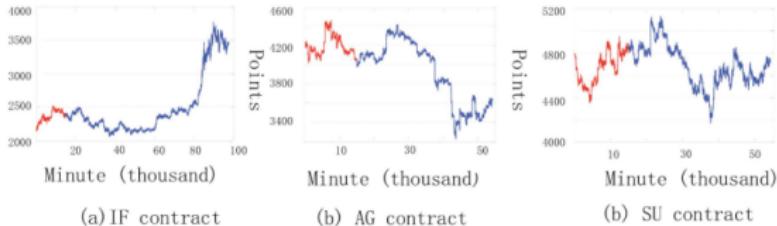
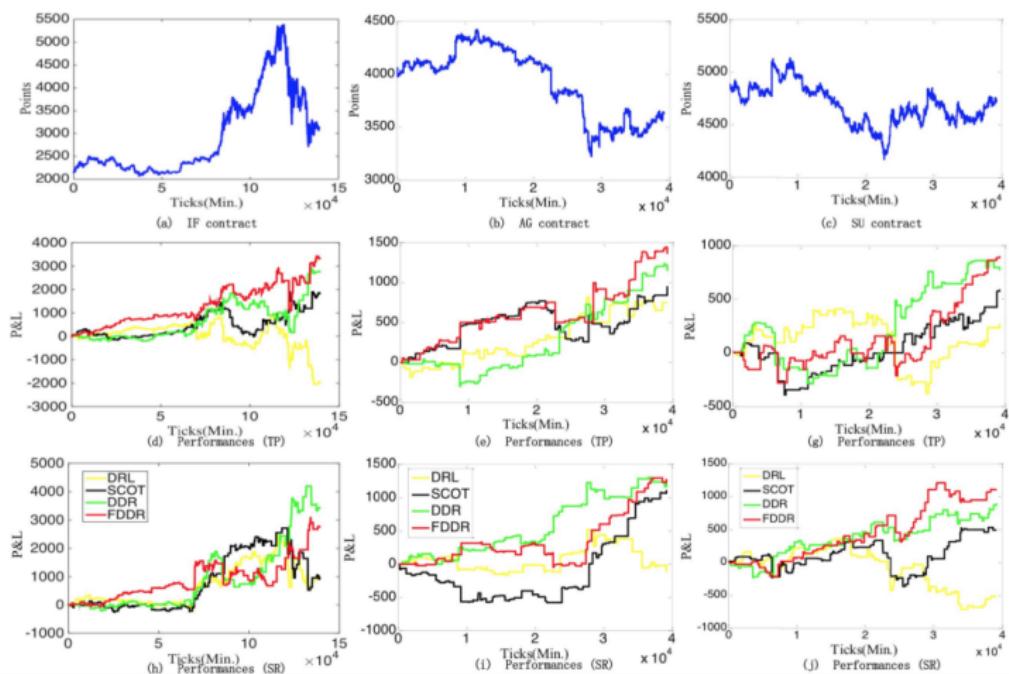


Figure: Prices (based on minute resolutions) of the three tested futurecontracts. Red parts: RDNN initializations. Blue parts: out-of-sample tests

TABLE I  
SUMMARY OF SOME PRACTICAL PROPERTIES  
OF THE TRADED CONTRACTS

Contract	Periods	CNY/pnt	TC	$c$
IF	1/14–9/15	300	30 (0.1pnt)	1pnt
AG	1/14–1/15	15	5 (0.4pnt)	2pnt
SU	1/14–1/15	5	3 (0.3pnt)	1.5pnt

# Results from RL



# Results from RL - Commodities

TABLE II  
PERFORMANCES OF DIFFERENT RL SYSTEMS ON DIFFERENT MARKETS

	IF				AG				SU			
	TP	SR(%)	TP	SR(%)	TP	SR(%)	TP	SR(%)	TP	SR(%)	TP	SR(%)
BH	739	-	-	-	-415	-	-	-	-121	-	-	-
DRL	-1998.2	-5.4	1054.4	3.1	-113	-2.4	745	12.2	271	4.3	-514	-7.4
SCOT	1873.2	5.6	1056.2	3.5	559	6.5	953	16.2	590	11.2	492	7.8
DDR	2785.4	9.3	3424.4	12.9	985	16.4	1147	19.4	787	13.0	898	15.8
FDDR	3256.6	11.2	2760.2	10.0	1567	21.2	1335	20.0	901	13.6	1118	19.3

## Results from RL - SP500

Data: SP500m daily resolution, Jan 1990 –Sep 2015

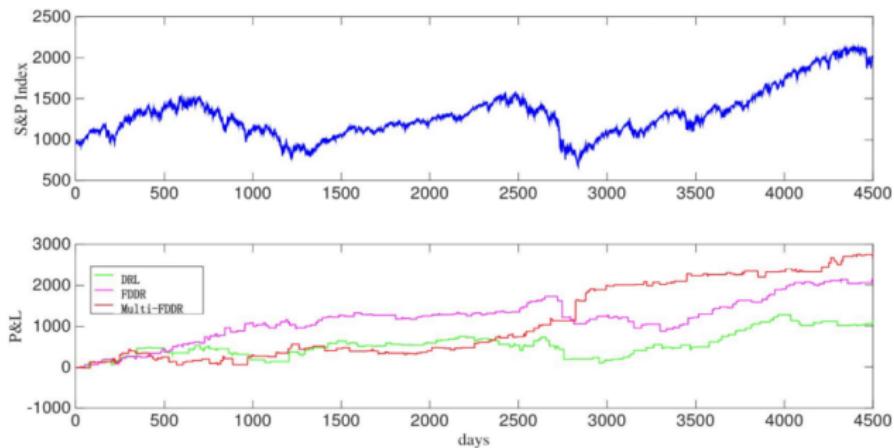
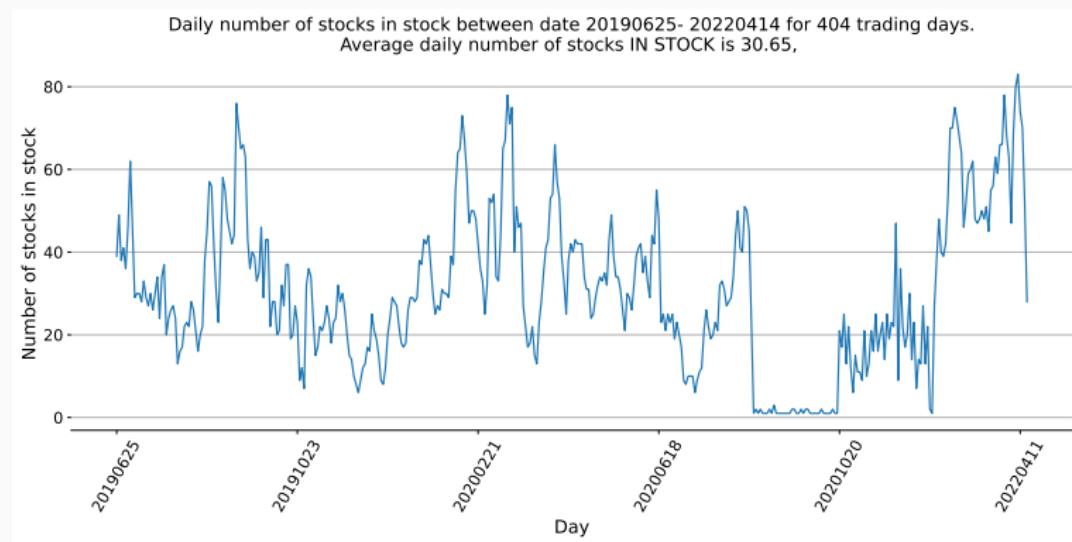


Figure: Testing S&P data and the P&L curves of different trading systems

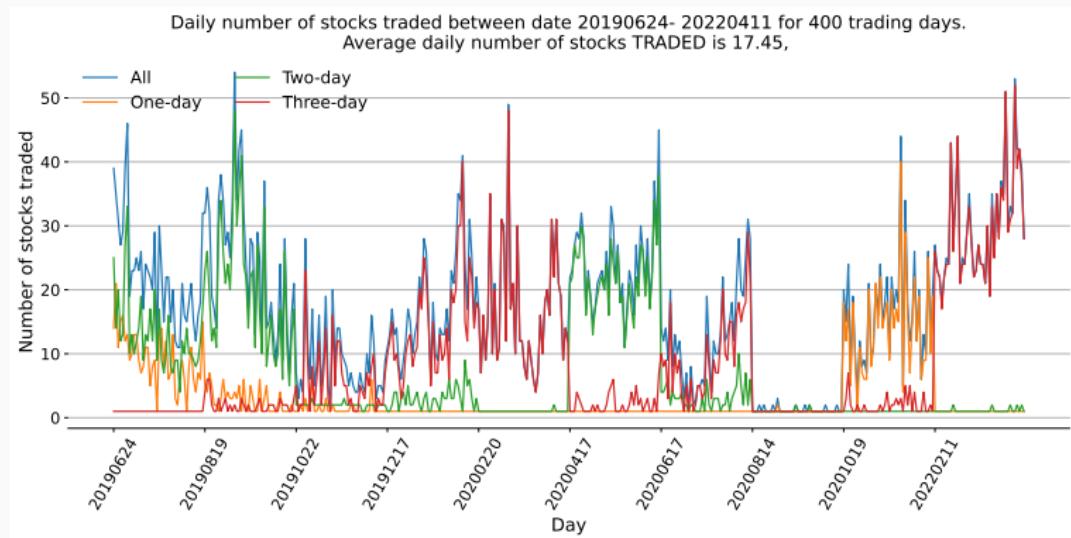
## Case Study: Chinese A-Share Market

- State Variables: past prices, trading volumes, and fundamental information.
- Decision: Choosing which stock to hold, and for 1, 2, or 3 days.
- Reward: cumulative return.

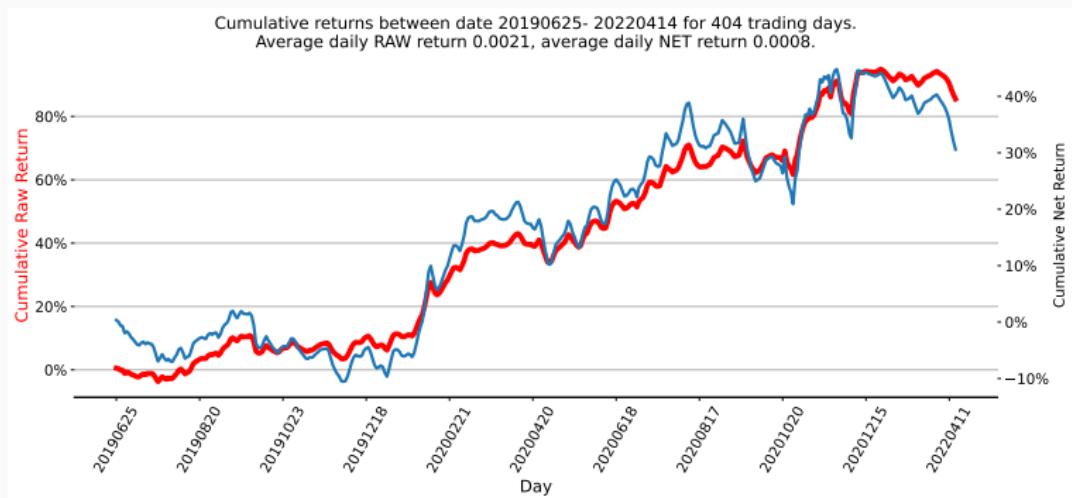
# Number of Stock Selected Each day



# Day distributions



# Cumulative Return Curve



- A rising trend is AI investment.
- Related Papers:
  - Alejandro Lopez-Lira and Yuehua Tang, "Can ChatGPT Forecast Stock Price Movements? Return Predictability and Large Language Models", 2024
  - J. Jiang, B. Kelly and D. Xiu, "Expected Returns and Large Language Models", 2024
- AI investing: Large Language Models + RL.
  - LLM for extracting information/factors.
  - RL for optimizing trading strategy.

# Predicting Future Stock Returns with Large Language Models

- Based on news, headlines, Ravenpack database, and stock return data.
- Prompt: Forget all your previous instructions. Pretend you are a financial expert. You are a financial expert with stock recommendation experience. Answer "YES" if good news, "NO" if bad news, or "UNKNOWN" if uncertain in the first line. Then elaborate with one short and concise sentence on the next line. Is this headline good or bad for the stock price of the company name in the \_term?

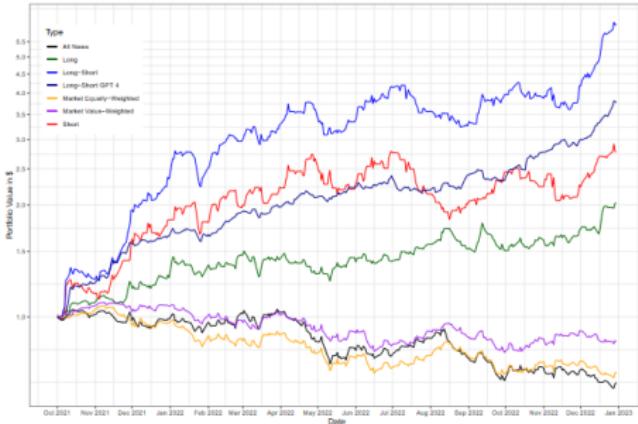
Headline: headline

## Predicting Future Stock Returns with Large Language Models

- Based on the current news data, we can obtain buy, sell, or no signal indications.
- Based on these signals, we construct a portfolio.
- This portfolio achieves significant excess returns, with an annualized return rate exceeding 200

# Excess Return for GPT Factors

Figure 1: Cumulative Returns of Investing \$1 (Without Transaction Costs)



This figure presents the results of different trading strategies without considering transaction costs. We assume that if a piece of news is revealed before the market close, we buy (or short-sell) a position at the market close price. If a piece of news is announced after the market closes, we assume we buy (or short-sell) a position at the next opening price. All the strategies are rebalanced daily. The “All-news” black line corresponds to an equal-weight portfolio in all companies with news the day before. The green line corresponds to an equal-weighted portfolio that buys companies with good news, according to ChatGPT 3.5. The red line corresponds to an equal-weighted portfolio that short-sells companies with bad news, according to ChatGPT 3.5. The light blue line corresponds to an equal-weighted zero-cost portfolio that buys companies with good news and short-sells companies with bad news, according to ChatGPT 3.5. The dark blue line corresponds to an equal-weighted zero-cost portfolio that buys companies with good news and short-sells companies with bad news, according to ChatGPT 4. The yellow line corresponds to an equally weighted market portfolio. The purple line corresponds to a value-weighted market portfolio.

- By combining generative AI, financial domain knowledge, and financial solvers, we can establish a natural language-based deep industrial-grade automated decision-making platform for the finance domain, tentatively named GPT-For-Finance.
- Through automated data processing and meta-machine learning optimization methods, this platform can provide fast and effective assistance or solutions to a wide range of industrial problems in the finance field, such as banking, insurance product design, wealth management services, risk control, financial asset pricing and quantitative trading, industrial finance, enterprise merger estimation and decision-making advice, financial education, data management and operations, report analysis, report generation, and more.
- These services are entirely achieved through natural language interaction between human managers and GPT-For-Finance. Compared to existing financial institutions, this approach can reduce costs, improve efficiency, significantly increase product development speed and work accuracy, and enhance product quality and management level.
- Fine-tuning using financial domain data enables localization and on-premises deployment.

- Perform natural language understanding around core tasks in the finance domain.
- Learn how to connect with the backend optimizer, which is the core knowledge carrier in the finance domain and not inherently known by GPT. GPT treats this optimizer as a black box.
- Provide appropriate dialogue and feedback to users.

## The role of GPT



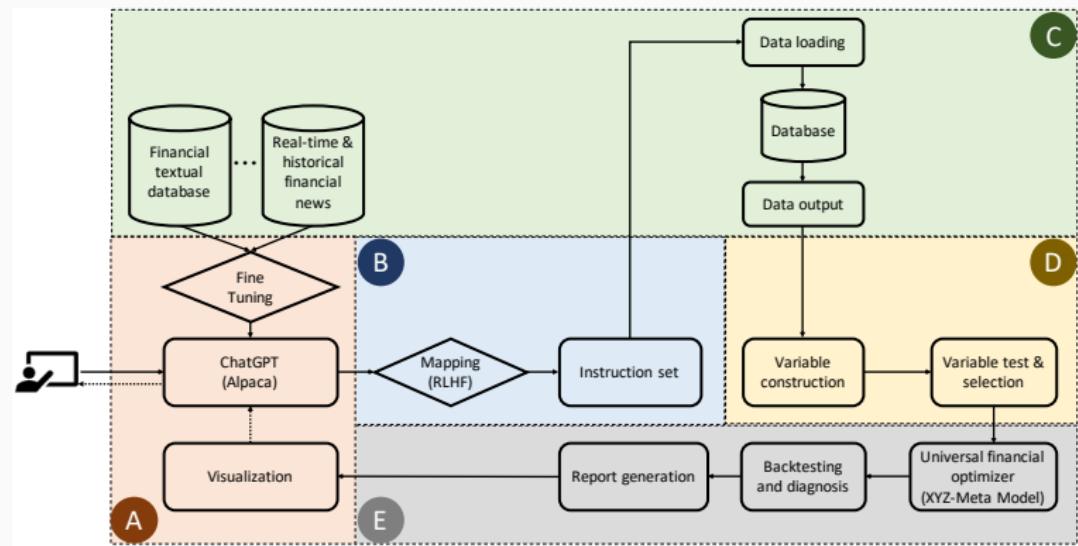
## GPT and Human-Assisted Reinforcement Learning

- A mapping between natural language and the instruction set needs to be established between the user command and the black box.
- Design a human feedback system for users to adjust instructions.
- Multiple rounds of interaction between natural language commands, feedback, and users.
- Establish an instruction decision function through human-assisted reinforcement learning (RLHF) to correctly call the backend module.

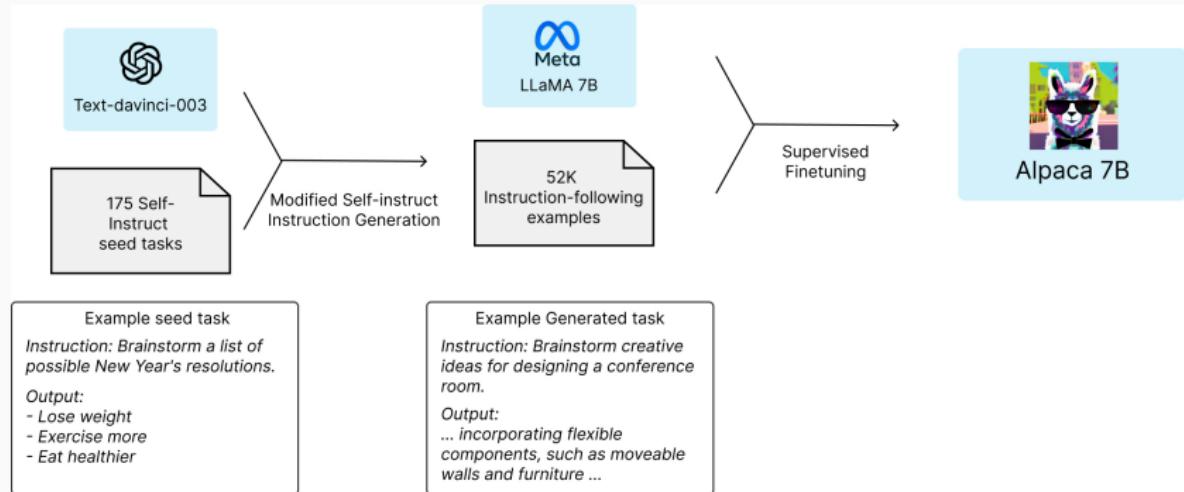
## Integration of GPT and the Financial Industry

- Enable industrial-grade production and decision-making in the financial industry through conversational interactions.
- Rely on the combination of GPT and specific task optimizers or models, such as risk models, asset pricing models, merger models, and more.
- Zero code, natural language-driven. Moving from industry-specific platforms with low-code capabilities, such as Joint Quant, World Quant, to a platform with natural language capabilities.
- One-click optimization, fully automated generation, and natural language production and customization of analysis results.

# GPT+Decision Pipeline



# LLM Fine Tuning



# LLM Comparison

▲ Vicuna Evaluation Examples Demo Blog Github

## Battles between State-of-the-Art Chatbots

Category Question

Finance Q1: PE和PB分别做什么?

8/10  ✎  9/10

Question: PE和PB分别做什么?

Assistant #1 ChatGPT-3.5

PE指市盈率 (Price-to-Earnings Ratio)，是指公司股价与每股收益的比率，用来衡量公司的股票价格是否合理。PB指市净率 (Price-to-Book Ratio)，是指公司股价与每股净资产的比率，用来衡量公司的股价是否高估或低估。

Assistant #2 (Vicuna-13b, our model)

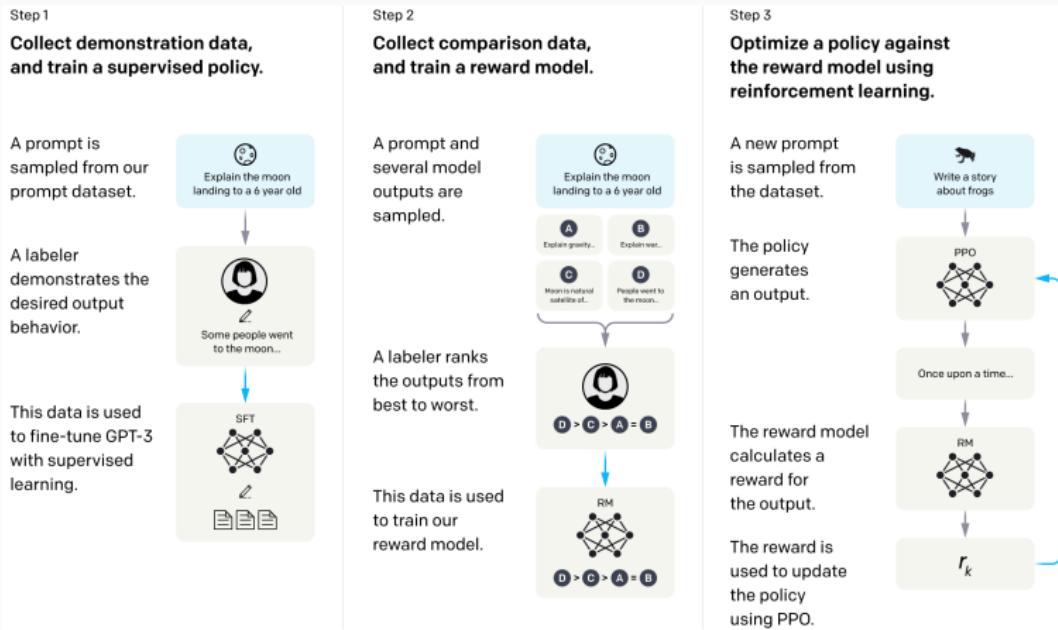
PE和PB是股票评估中常用的两个指标。具体含义如下：

1. PE (市盈率)：PE指的是每股的市值与净利润之比，通常用来衡量股市对公司未来盈利能力的预期。 $PE = \text{市值} / \text{净利润}$
2. PB (市净率)：PB指的是每股的市值与净资产之比，通常用来衡量股市对公司的基本价值。 $PB = \text{市值} / \text{净资产}$

Show more

GPT-4 Evaluation (Score: ChatGPT-3.5 8/10, Vicuna-13b 9/10)

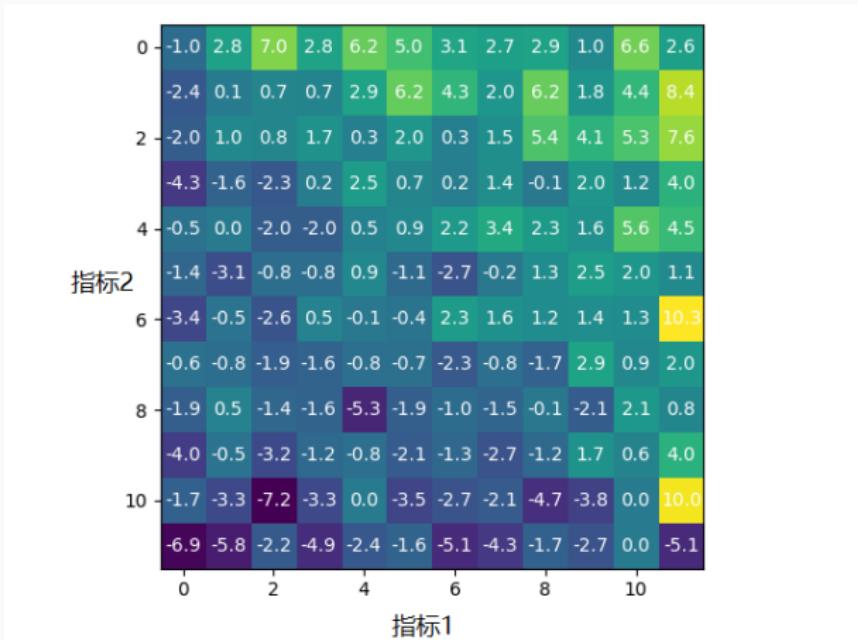
我认为两个助手都给出了正确、相关、清晰且详细的答案。他们准确解释了PE和PB的含义和用途，并提醒了投资者不应该只依靠这两个指标来做出决策。助手2的回答更加详细，还解释了净利润和净资产的含义，因此我给助手2的表现打了稍微高一些的分数。



## General Optimizer in the Finance Field: The XYZ-Meta Model

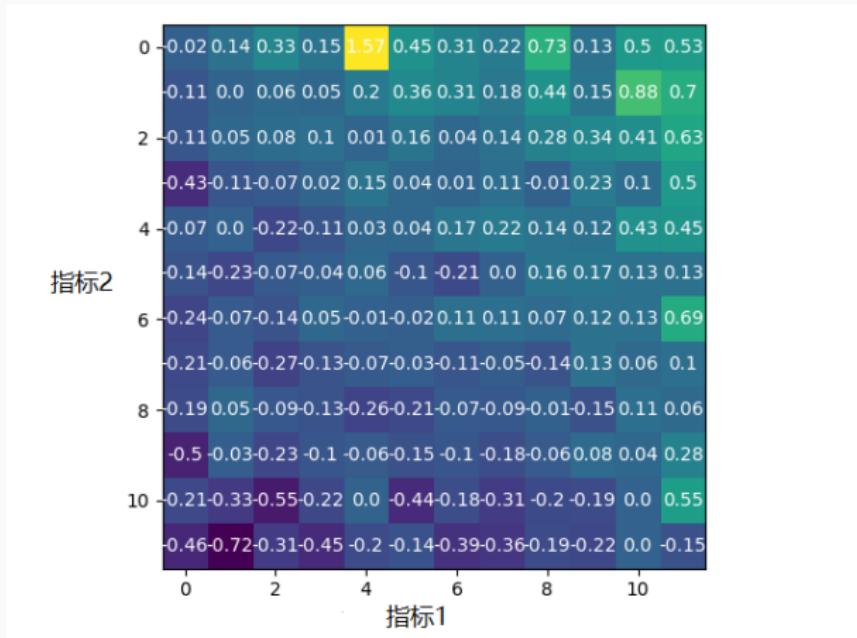
- Based on Meta-ML algorithms and financial domain knowledge, the XYZ-Meta Model is a general optimizer for financial market asset pricing models.
- Define three trading elements: the target asset, the hedging tool, and the buy/sell state variables.
- User-defined preferences, such as risk rate, return rate, return rate per transaction, Sharpe ratio, and more.
- Automatic optimization of models and trading algorithms, with automatic robustness testing.
- Automatic optimization of timing algorithms, auto fine-tuning.

# Data Visualization



**Figure 3:** The distribution of total returns of a certain stock index futures during a certain period, in units of 1%.

# Data Visualization



**Figure 4:** The distribution of average returns per transaction of a certain stock index futures during a certain period, in units of 1%.

# Back Test Result of the Optimizer

