Deep Q-Learning for Stock Trading

Yang Wang

CSLT / RIIT

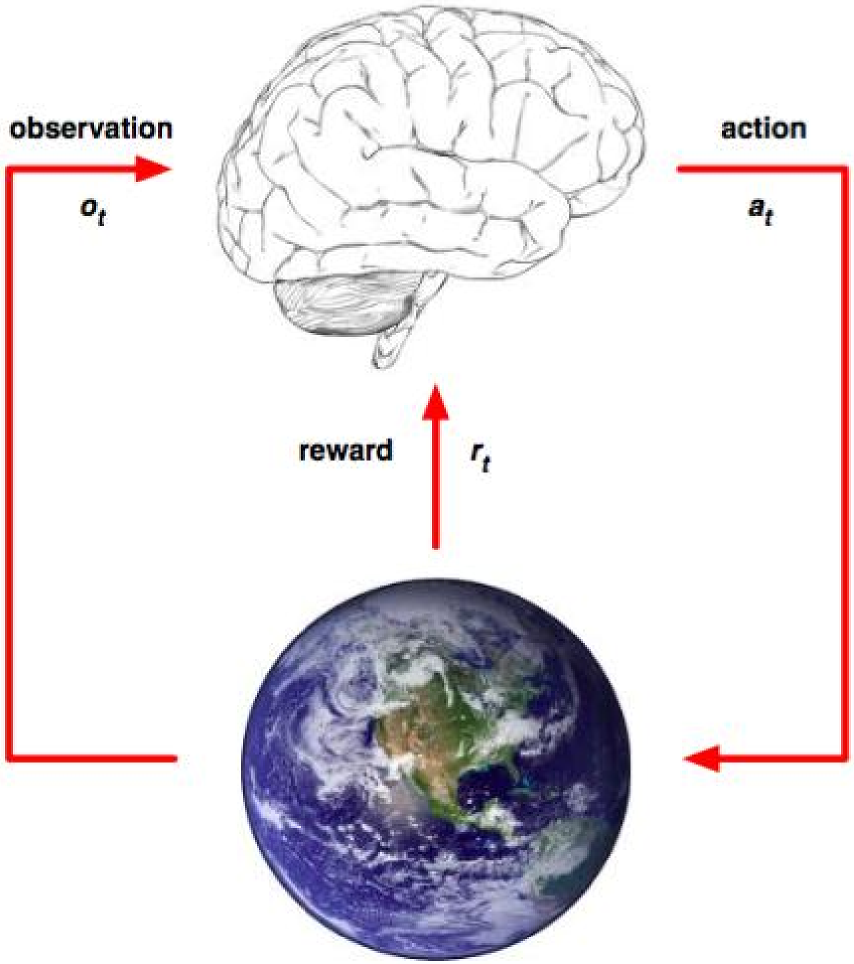
Tsinghua University

wangyang@cslt.riit.tsinghua.edu.cn

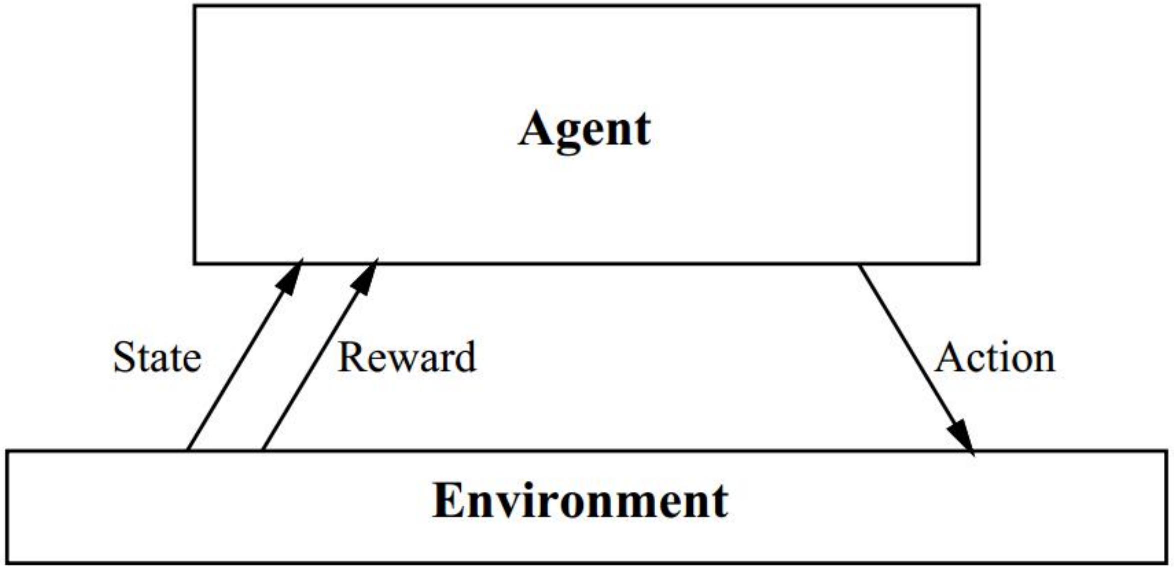
What is reinforcement learning?

RL is a general-purpose framework for decision-making

- RL is for an agent with the capacity to act
- Each action influences the agent's future state
- Success is measured by a scalar reward signal
- Goal: select actions to maximise future reward



- At each step t the agent:
  - Executes action a_t
  - Receives observation o_t
  - Receives scalar reward r_t
- The environment:
  - Receives action a_t
  - Emits observation o_{t+1}
  - Emits scalar reward r_{t+1}



$$s_0 \xrightarrow[r_0]{a_0} s_1 \xrightarrow[r_1]{a_1} s_2 \xrightarrow[r_2]{a_2} \dots$$

Goal: Learn to choose actions that maximize

$$r_0 + \gamma r_1 + \gamma^2 r_2 + \dots \text{, where } 0 \leq \gamma < 1$$

Agent's learning task

Execute actions in environment, observe results, and

- learn action policy $\pi : S \rightarrow A$ that maximizes

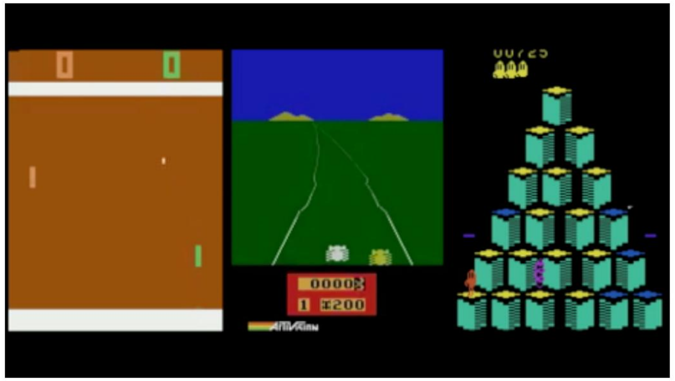
$$E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots]$$

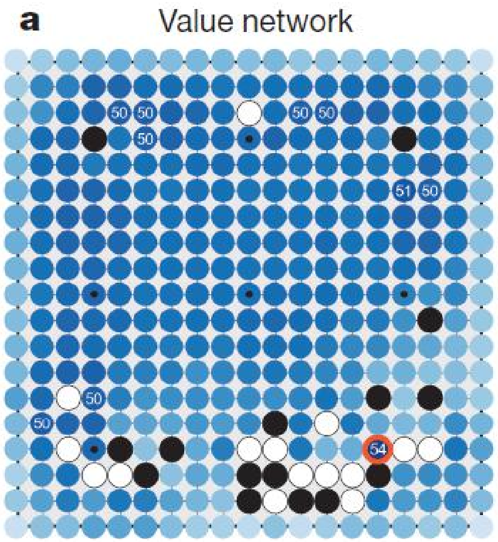
  from any starting state in S

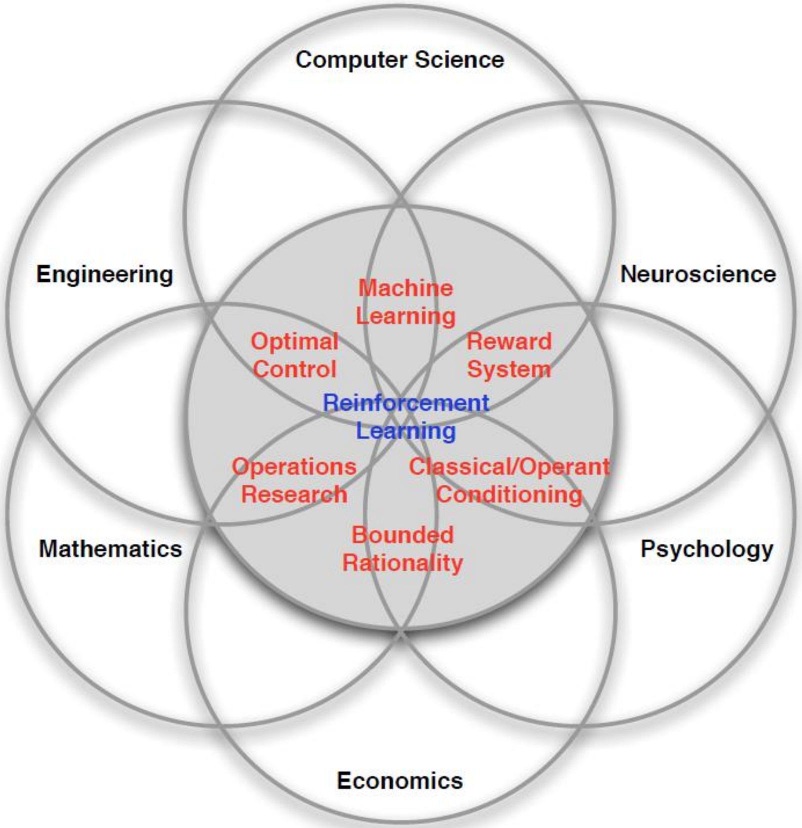
- here $0 \leq \gamma < 1$ is the discount factor for future rewards

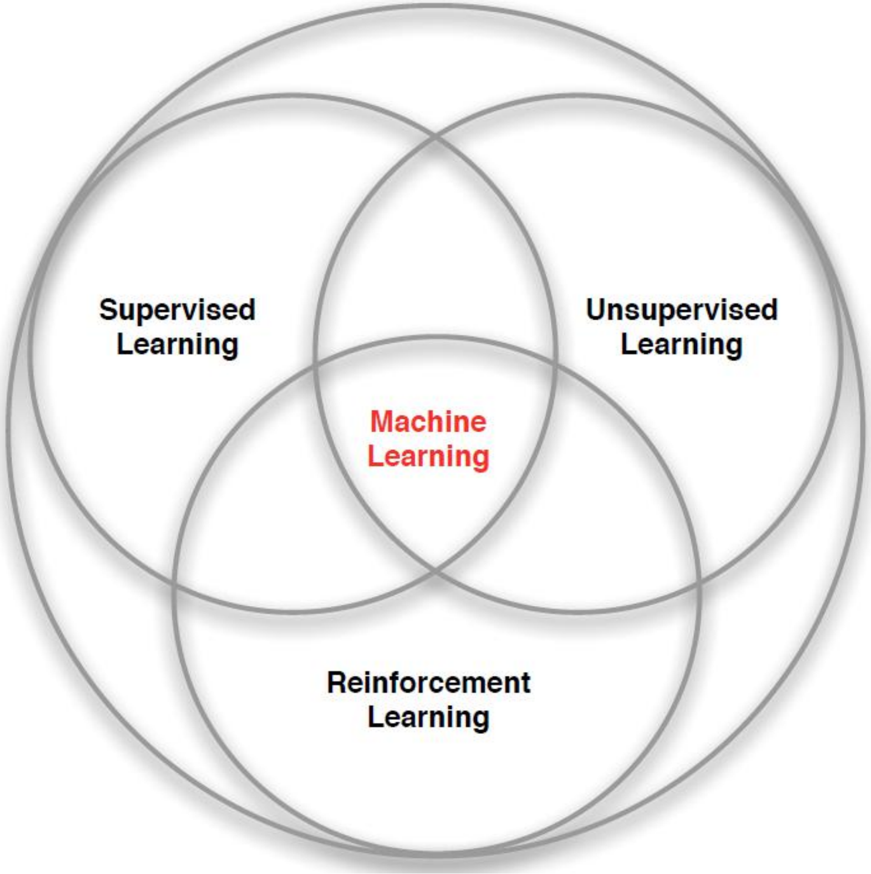
Some applications

- Play many Atari games better than humans.
- Defeat human experts at Go game.
- Solve some simulated physics tasks.
- Finance: Investment decision, portfolio design.






Computer Science
Engineering
Neuroscience
Machine Learning
Optimal Control
Reward System
Reinforcement Learning
Operations Research
Classical/Operant Conditioning
Bounded Rationality
Mathematics
Psychology
Economics


Supervised Learning
Unsupervised Learning
Machine Learning
Reinforcement Learning

MDP (Markov Decision Process)

Assume

- finite set of states S
- set of actions A
- at each discrete time agent observes state $s_t \in S$ and chooses action $a_t \in A$
- then receives immediate reward r_t
- and state changes to s_{t+1}
- Markov assumption: $s_{t+1} = \delta(s_t, a_t)$ and $r_t = r(s_t, a_t)$
  - i.e., r_t and s_{t+1} depend only on *current* state and action
  - functions δ and r may be nondeterministic
  - functions δ and r not necessarily known to agent

Value function

To begin with, consider deterministic worlds...

A value function is a prediction of future reward

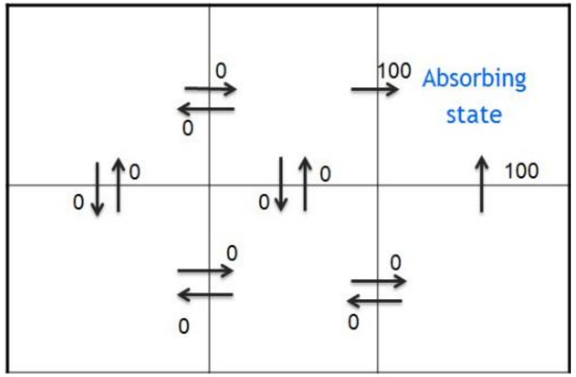
$$V^{\pi}(s) \equiv r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots$$
$$\equiv \sum_{i=0}^{\infty} \gamma^i r_{t+i}$$

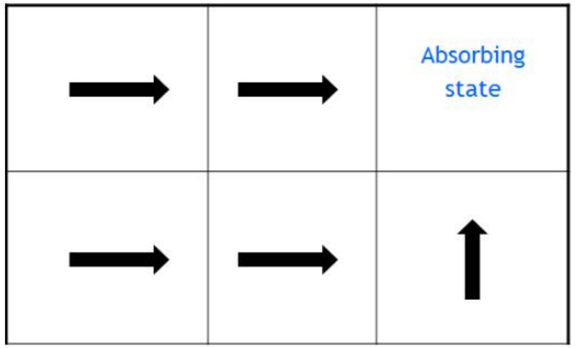
$$\begin{aligned} v(s) &= \mathbb{E}\left[G_t \mid S_t = s\right] \\ &= \mathbb{E}\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \mid S_t = s\right] \\ &= \mathbb{E}\left[R_{t+1} + \gamma \left(R_{t+2} + \gamma R_{t+3} + \ldots\right) \mid S_t = s\right] \\ &= \mathbb{E}\left[R_{t+1} + \gamma G_{t+1} \mid S_t = s\right] \\ &= \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid S_t = s\right] \end{aligned}$$

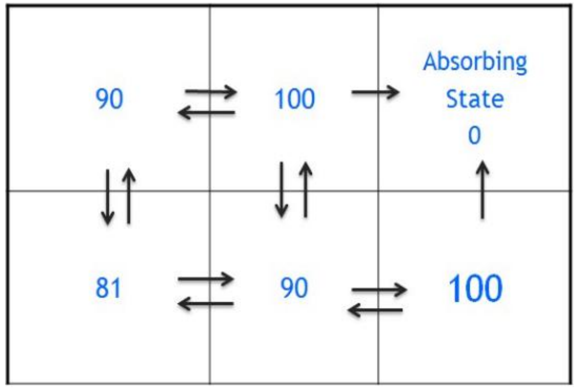
Restarted, the task is to learn the optimal policy

$$\pi^* \equiv \underset{\pi}{\operatorname{argmax}}\, V^{\pi}(s), (\forall s)$$

One example







$V^{\pi}(s_{12}) = r(s_{12},a_r) = r(s_{13}|s_{12},a_r) = 100$

$V^{\pi}(s_{11}) = r(s_{11},a_r)+\gamma*V^{\pi}(s_{12}) = 0+0.9*100 = 90$

$V^{\pi}(s_{23}) = r(s_{23},a_u) = 100$

$V^{\pi}(s_{22}) = r(s_{22},a_r)+\gamma*V^{\pi}(s_{23}) = 90$

$V^{\pi}(s_{21}) = r(s_{21},a_r)+\gamma*V^{\pi}(s_{22}) = 81$

What to learn

Learn the evaluation function V^{π^*} (which we write as V^*)

Do a lookahead search to choose best action from any state s beacause

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}[r(s,a) + \gamma V^*(\delta(s,a))]$$

So easy !!! ???

- This works well if agent knows $\delta : S \times A \rightarrow S$ and $r : S \times A \rightarrow \Re$
- But when it doesn't, it can't choose actions this way

Q function

Define new function very similar to V*

$$Q(s,a) \equiv r(s,a) + \gamma V^*(\delta(s,a))$$

If agent learns Q, it can choose optimal action even without knowing δ

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}[r(s,a) + \gamma V^*(\delta(s,a))]$$

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q(s,a)$$

Q is the evaluation function the agent will learn

Training rule to learn Q

Note Q and V* closely related:

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

Which allows us to write Q recursively as

$$\begin{aligned} Q(s_t, a_t) &= r(s_t, a_t) + \gamma V^*(\delta(s_t, a_t))) \\ &= r(s_t, a_t) + \gamma \max_{a'} Q(s_{t+1}, a') \end{aligned}$$

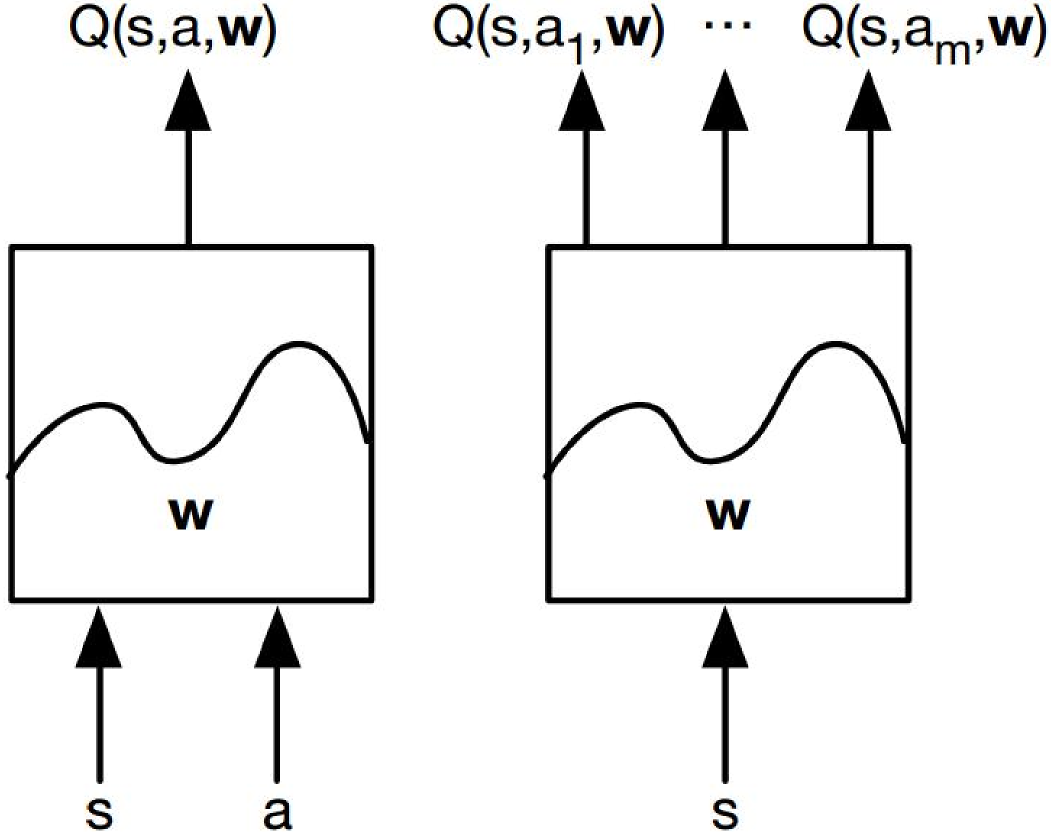
Nice! Let $\hat{Q}$ denote learner's current approximation to Q. Consider training rule

$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

Where s' is the state resulting from applying action a in state s

Q-networks

Represent value function by Q-networks with weight **w**



$$Q(s, a, \mathbf{w}) \approx Q^*(s, a)$$

Q-learning

Optimal Q-values should obey Bellman equation

$$Q^*(s,a) = \mathbb{E}_{s'}\left[r + \gamma \max_{a'} Q(s',a')^* \mid s,a \right]$$

Treat right-hand side $r + \gamma \max_{a'} Q(s',a',\mathbf{w})$ as a target

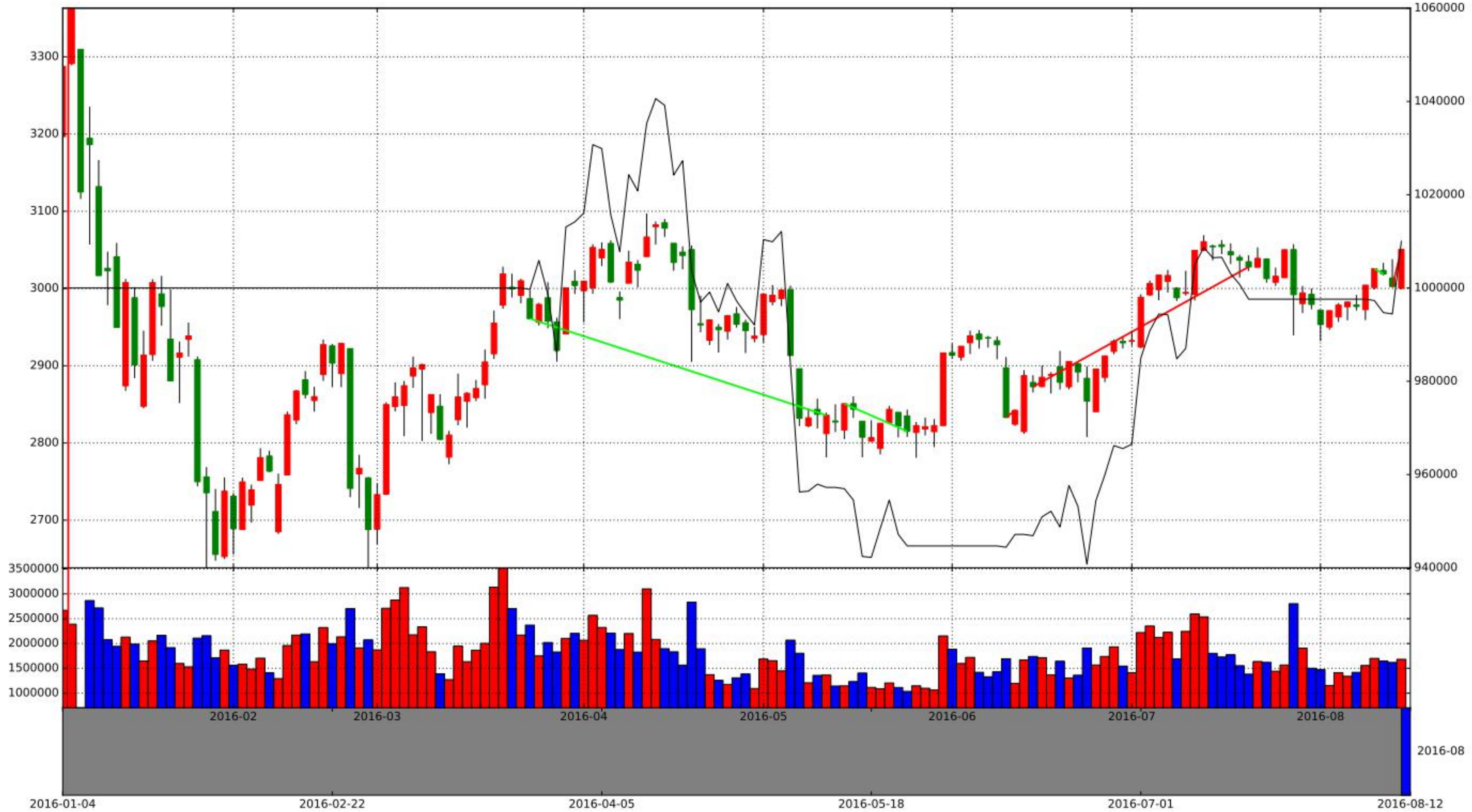
Minimise mean square erroe (MSE) loss by stochastic gradient descent

$$I = \left(r + \gamma \max_{a} Q(s',a',\mathbf{w}) - Q(s,a,\mathbf{w}) \right)^2$$

Experiments

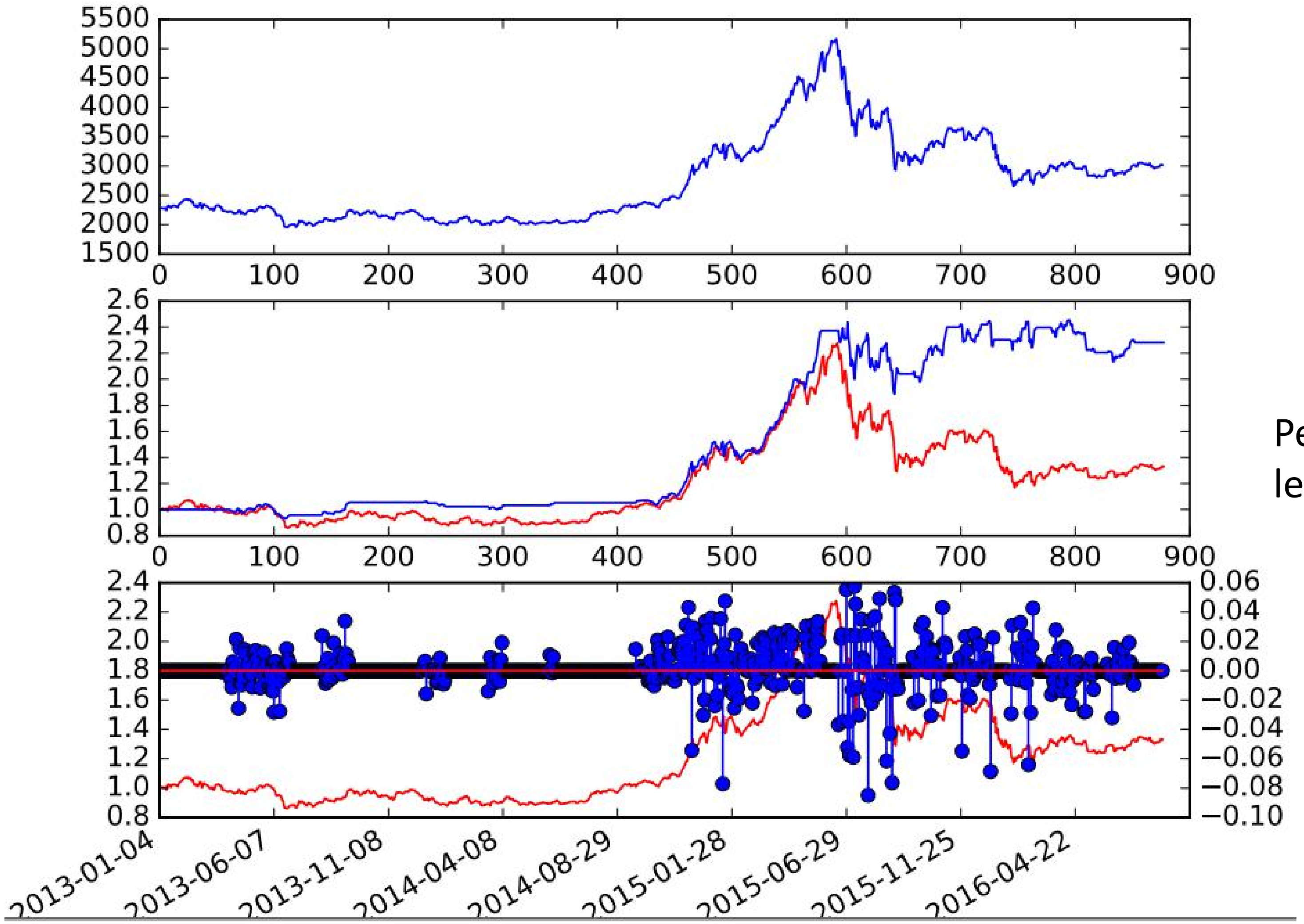
Performance on training dataset

Experiments



Performance with pretrained model on test dataset

Experiments



Performance with on-line learning on test dataset

Thank you for your attention！

Stay Hungry, Stay Foolish.

wangyang@cslt.riit.tsinghua.edu.cn