UNIVERSITY OF ALBERTA

The Intelligent Robot Learning Laboratory

Matthew E. Taylor (Matt)

Reinforcement Learning:
From Video Games to Stock Trading

air
AI Redefined

Matt’s background

PhD in AI in 2008

Since then:

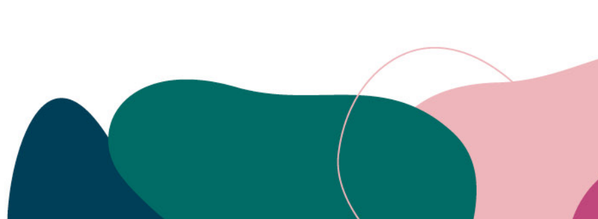
- Postdoc
- Prof. at liberal arts college
- Prof. at research university
- 2.5 years with Borealis AI – applied + fundamental research

Now: UofA CS + Amii + AI-Redefined

Reinforcement Learning

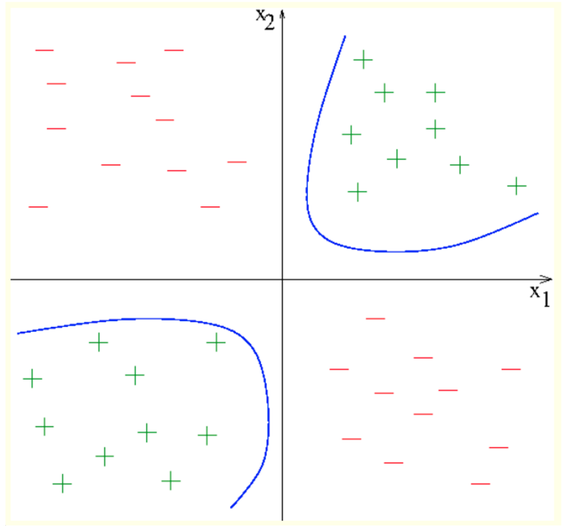
Reinforcement Learning @ RBC

Identifying good problems

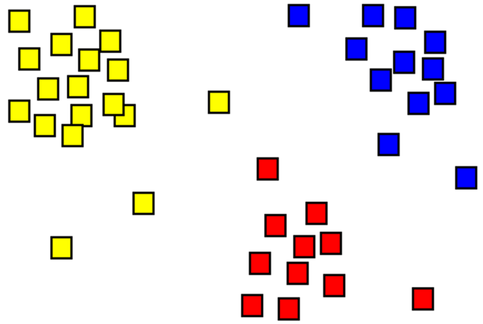


Machine Learning

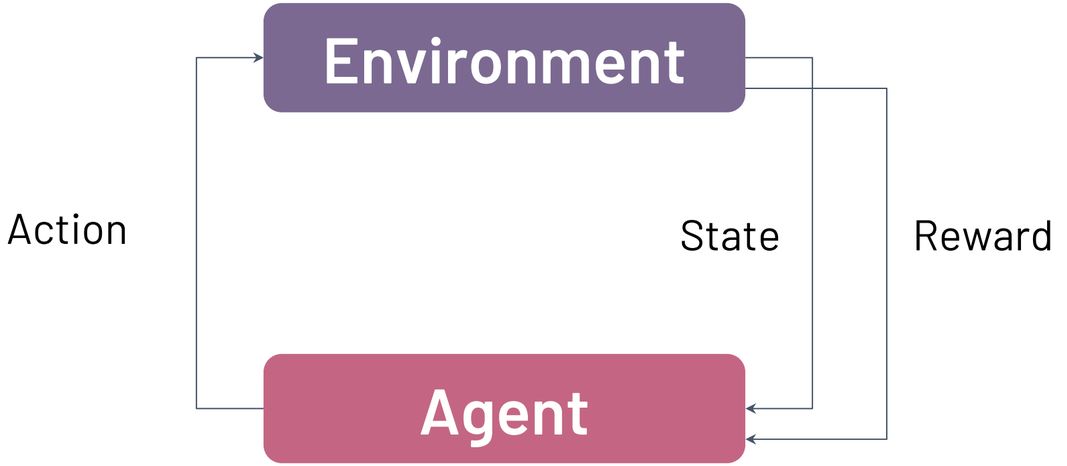
Supervised



Unsupervised



Reinforcement Learning



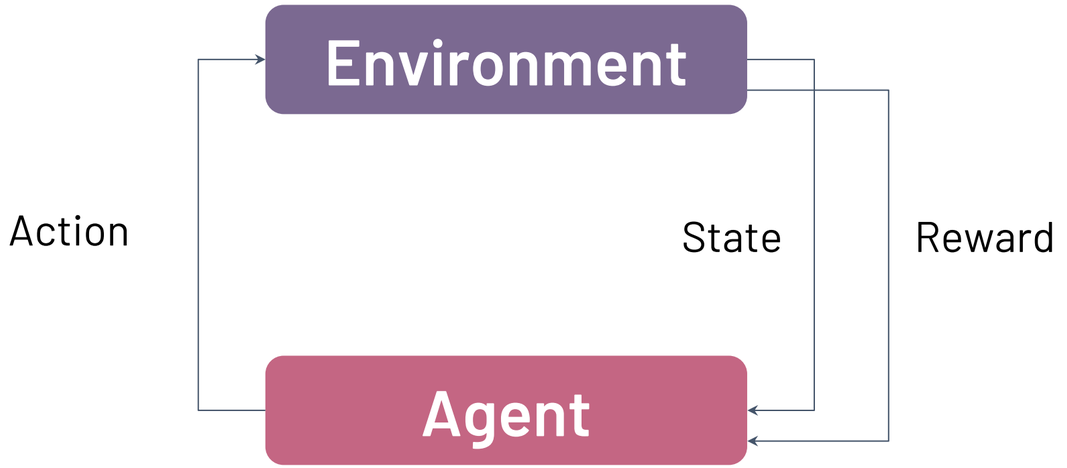
Reinforcement Learning (RL)

No labels: agent never told right or wrong

Agent interacts with environment (simulator or real world)

Typically can gather data, possibly at cost, by interacting with environment

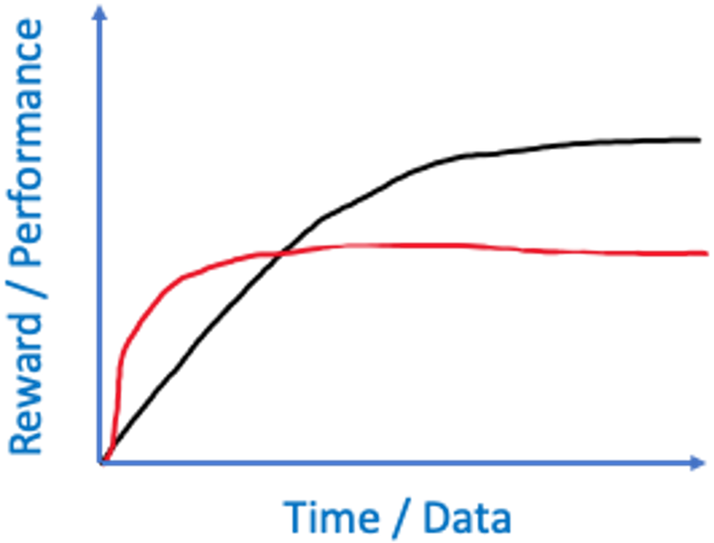
Learns via exploring vs. exploiting



RL Goals

Learn to maximize real-valued reward signal

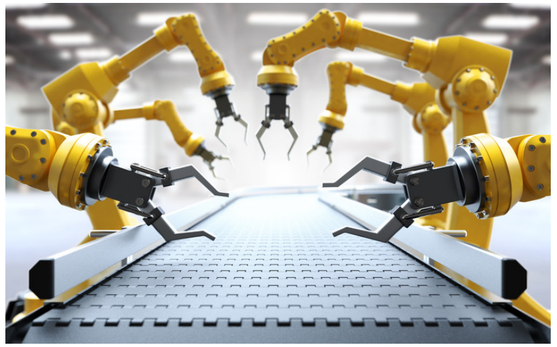
- With maximal final performance
- With little data
- Reducing human effort
- Discovering novel solutions
- Handling non-stationary environments



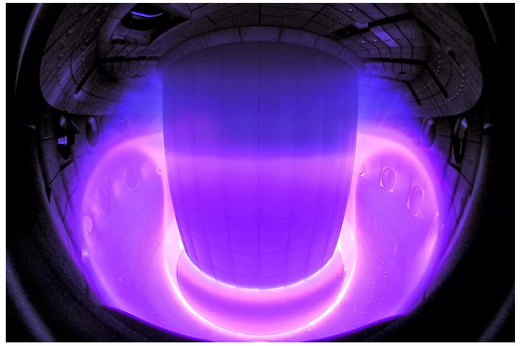
RL Applications

(Un)Supervised learning performs well for many real-world applications

Dota



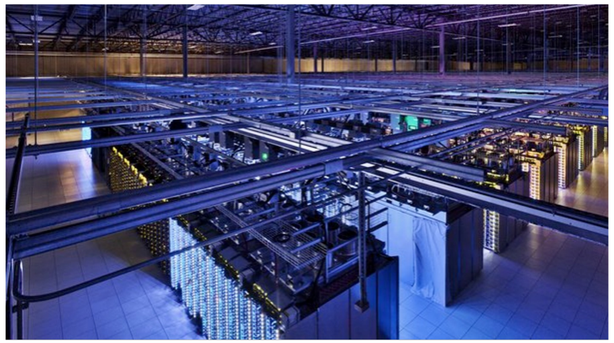
Robotics



Fusion: Tokamak Plasmas

Stock Trading

AlphaGO



Data Center Cooling

1. RL is mature
2. You should know if/where it applies

https://www.youtube.com/watch?v=0Jw4HTWvGdY

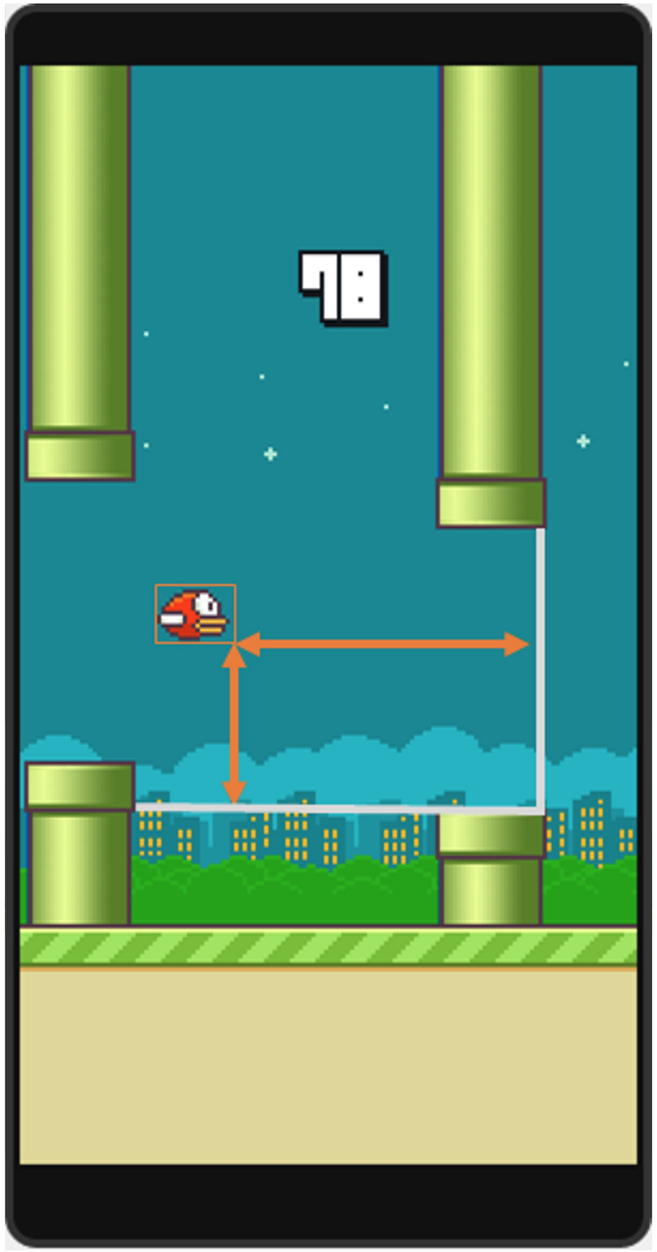
Example 1: Flappy Bird

Transition function: Controlled by game

Action?

Reward?

State representation?



http://sarvagyavaish.github.io/FlappyBirdRL/

Example 2: Water Treatment

Developed @ UofA

ISL Adapt, UofA, and Amii

No ground truth

Raw water from North Saskatchewan River

State: Sensors added to filtration plant

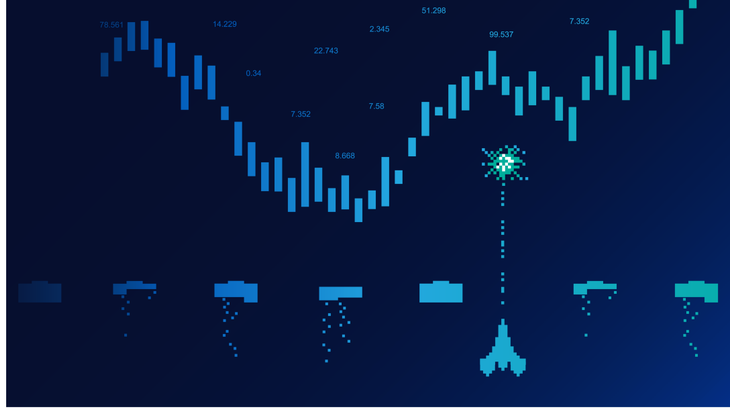
Actions: Changes like chemicals, backwash cleaning, etc.

Reward: Environmental and fiscal benefits

Drayton Valley

bit.ly/3ouscL0

Example 3: Aiden



https://www.borealisai.com/en/applying-ai/aiden/

Optimal Order Execution

State: Info about stock & market

Actions: Do nothing, buy/sell a little, buy/sell a lot

Rewards: Based on VWAP (Volume-weighted average price)

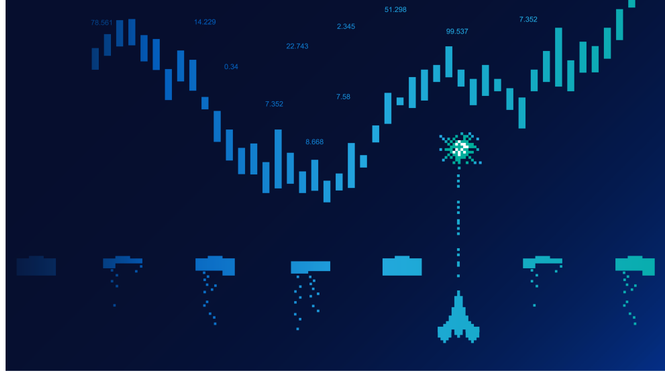
Transition function: Stock market (real or simulated data)

Reinforcement Learning

Reinforcement Learning @ RBC

Identifying good problems

Rewards



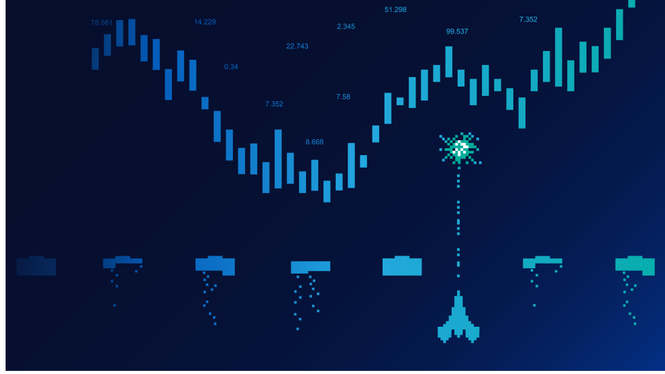
1) VWAP: Volume-weighted average price

$$\frac{\sum Price \times Volume}{\sum Volume}$$

Better than just the average price
Other benchmarks possible (e.g., arrival price)

2) Complete desired trade

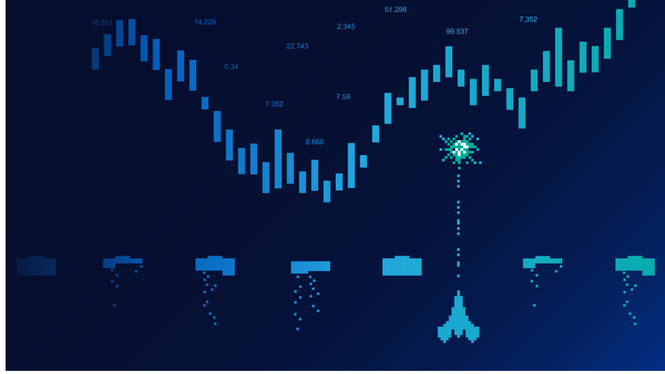
Actions



Know what direction you're going
Do nothing, buy/sell a little, buy/sell a lot

Discrete vs. continuous?
How much freedom?

State

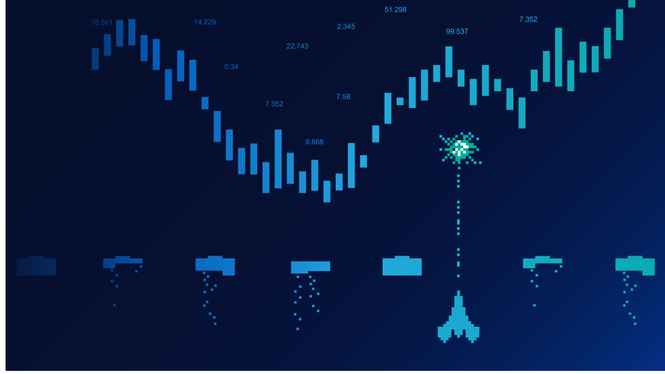


Core trade secrets
Not included in patent filings
Researchers only knew some of them

Necessary & Sufficient?

Memory?

Learning

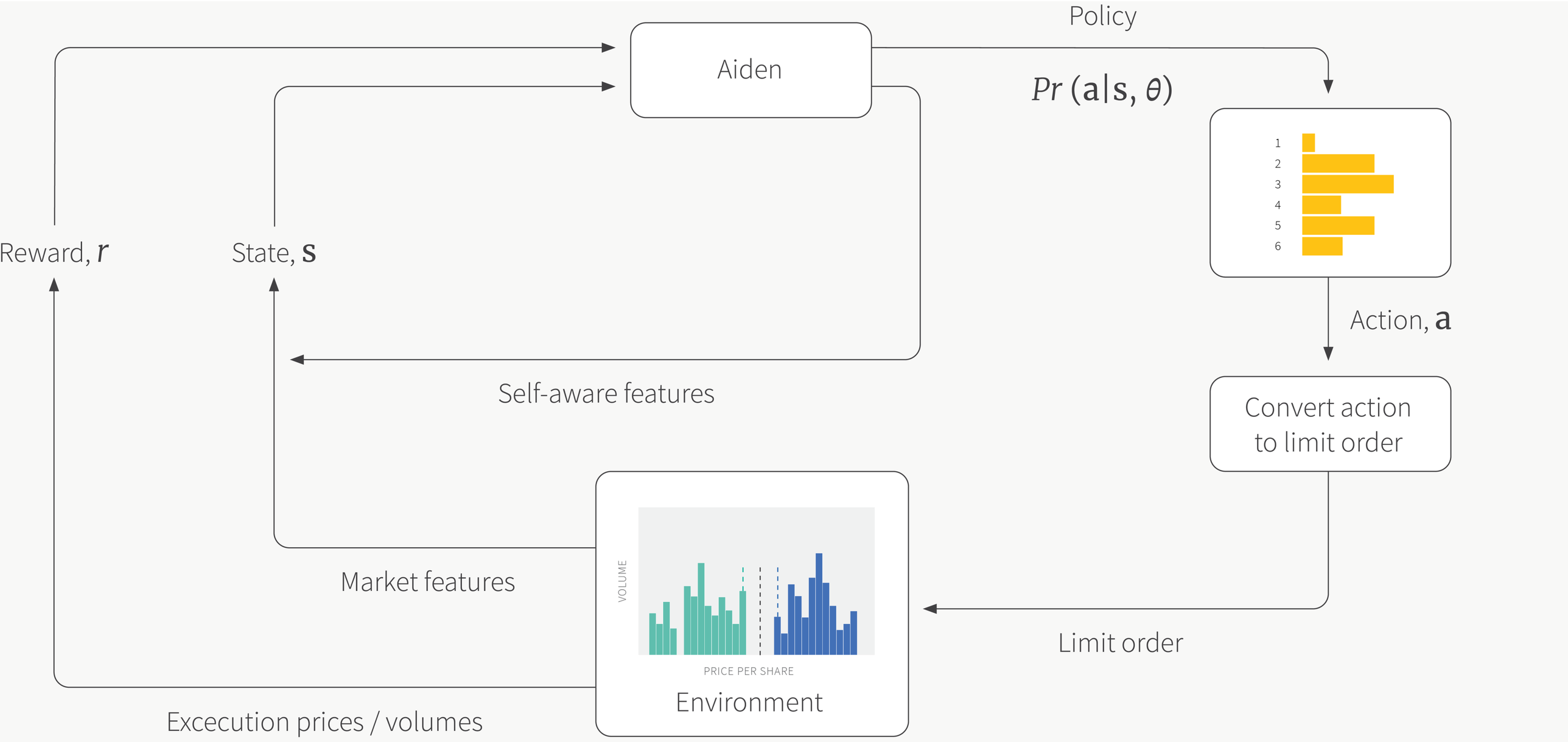


Policy gradient method: PPO

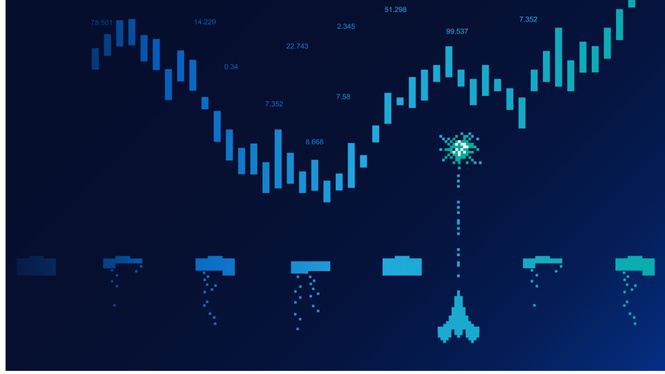
Trust region with clipping and/or KL-divergence penalty

Simulator first

Worked surprisingly well

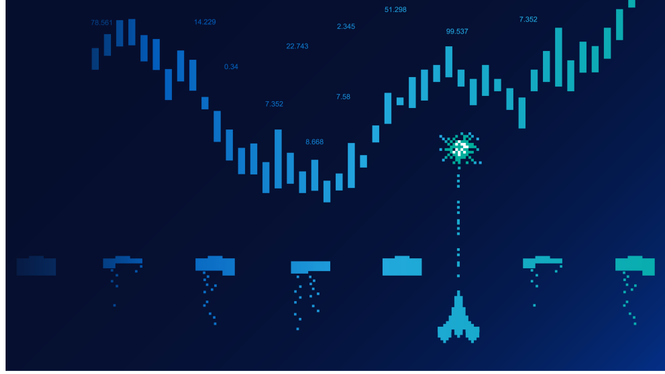

Aiden
Policy
$Pr(\mathbf{a}|\mathbf{s},\theta)$
1
2
3
4
5
6
Action, $\mathbf{a}$
Convert action to limit order
Limit order
Reward, r
State, $\mathbf{s}$
Self-aware features
Market features
VOLUME
PRICE PER SHARE
Environment
Excecution prices / volumes

Deployment and Acceptance



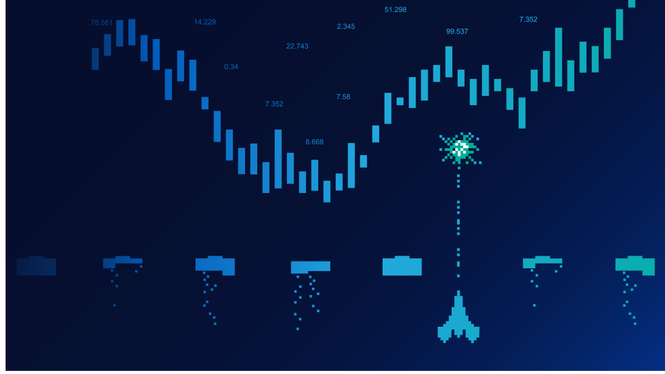
Small project, early Borealis AI employee
Fully backed by Foteini Agrafioti
Got Capital Markets on board
Some politics, of course
Deployment (i.e., 1st rodeo)

Marketing



Add-on benefits
You could imagine that many clients were very excited

Explainability



Trust: Customer / User / Manager
What went wrong?
Debugging
Marketing / Sales

Can you improve performance using explainability?
Active area of research

Other (Known) RL Applications

Portfolio optimization
Pricing options & financial derivatives

I believe there's many more opportunities

Reinforcement Learning

Reinforcement Learning @ RBC

Identifying good problems

Sample Cost

Fully Virtual vs. Approximate Simulator available vs. No Simulator

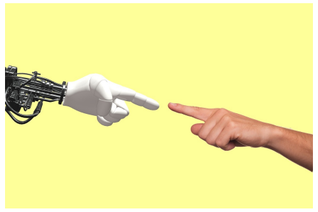
Balancing exploration vs. exploitation continuously, with real world costs
Training could be too slow to be useful (non-stationarity)
If can parallelize, can be OK, but need simulator
Trade off hyperparameter tuning, neural architecture search
How many independent training runs?

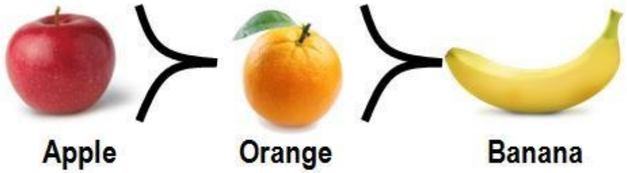
Initial Performance

How bad are random actions?

What is the cost of exploring?

Can you use an existing agent or a human?





Safety Constraints

Maximizes expected return

- E.g., number of points in an Atari game
- Does not enforce “what not to do” (critical states)
- Could lead to unintended behaviors
- May not correctly factor in risk

Could use action filter

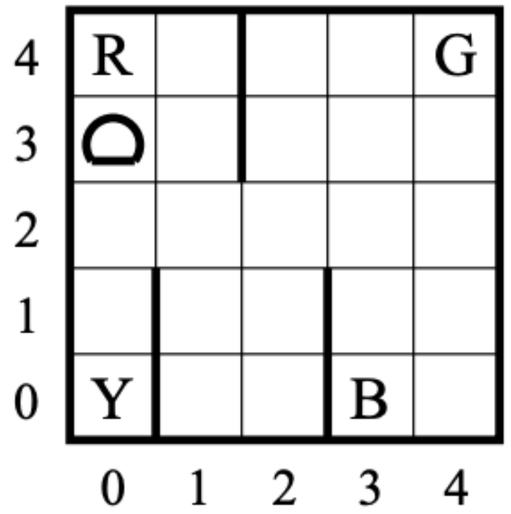
Might have access to safe policy

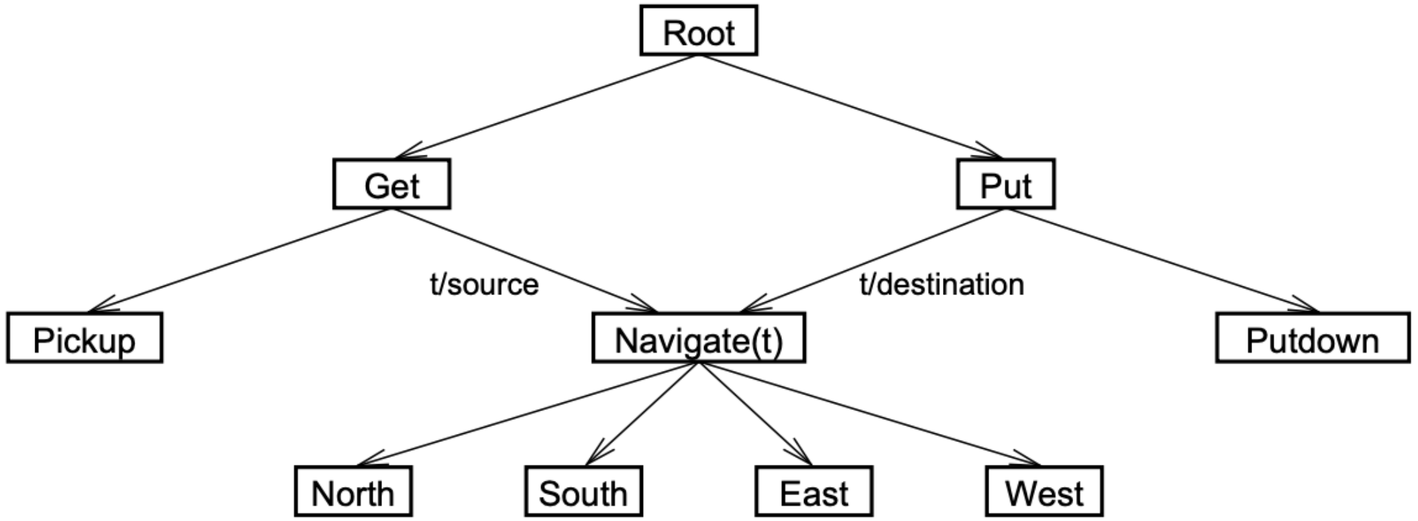
Add specific constraints

Large Action Spaces

Larger action space require more exploration

- Continuous action spaces
- Movie recommender agent: action space of size = number of movies
- Subsets of actions (e.g., movie genre) could help: hierarchical actions





Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition: Dietterich, 1999

Reward Function

Can be difficult to define

- Additional rewards can help for faster learning
- But… might discover high reward policy without doing the intended task!
- High-scoring, counter-intuitive behaviors

MDP reward

- Expert elicitation?
- User preferences?
- Shaping reward from expert?

Partial Observability, Non-Stationarity

Partially Observable MDP (POMDP) introduces additional complexity

- Agent gets observations, not true state
- DQN uses frame-stacking to handle POMDPs
- DQN + RNN obtains similar performance

Changes in the environment (state and/or reward distribution) over time

→ Sudden jumps vs. concept drift

Explainability / Interpretability

Explainability for failure cases
→ Helpful for debugging

Helps to build trust with users / encourage adoption
Might learn new strategies from the agent
→ AlphaGo learned unseen strategies in Go

RL Strengths

Agent can autonomously learn to maximize rewards

Programmer just specifies goals

Often much less work than directly programming

Can achieve superhuman performance

Can handle unanticipated changes in the environment

RL Weaknesses

Agent maximizes reward whether it's what you actually wanted or not!

Can require lots of computation and/or interaction with the real world
→ Interacting with real world can have cost: time, money, wear, etc.

Solutions are often black box
→ Explainability is not well understood (yet)

Initial performance could be very poor

OpenAI Five uses **180 years** of gameplay data **each day** across tens of thousands of simultaneous games, consuming **128,000 CPU** cores and **256 GPUs**

Summary: Targeting a Good RL Task

How expensive are failures (e.g., from exploration or early performance)?

What improvement can we expect over state of the art?

Are there second-order effects (e.g., marketing)?

How difficult will it be to set up an environment where an agent can get the data it needs and execute actions?

Can I first tackle an abstract version of the task to understand how difficult it is?

Conclusion: Lots of Opportunities!

RL is awesome

Lot's of (free) resources to learn more

Identifying good problems can be non-trivial

The Intelligent Robot Learning Laboratory

http://irll.ca