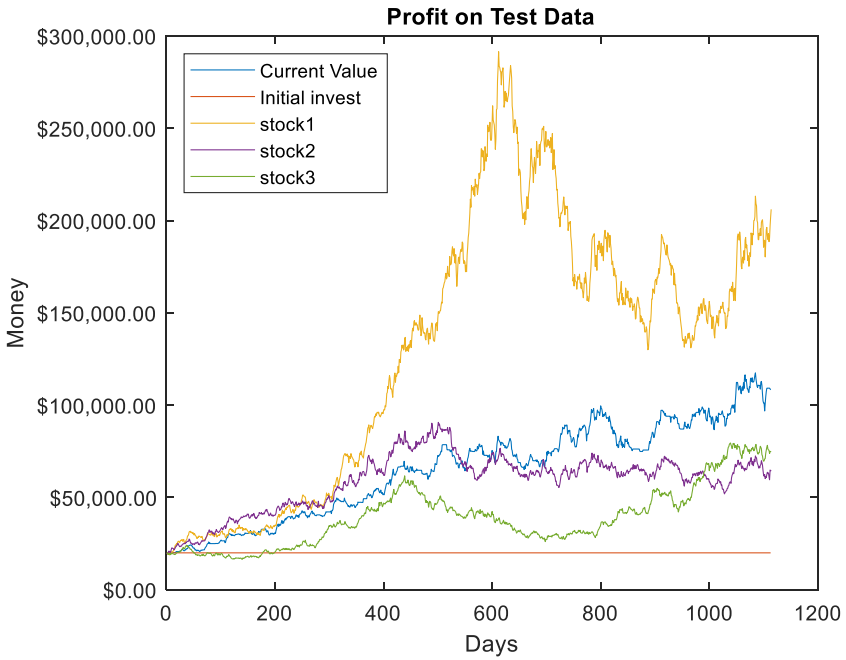
Reinforcement Learning for Financial Trading

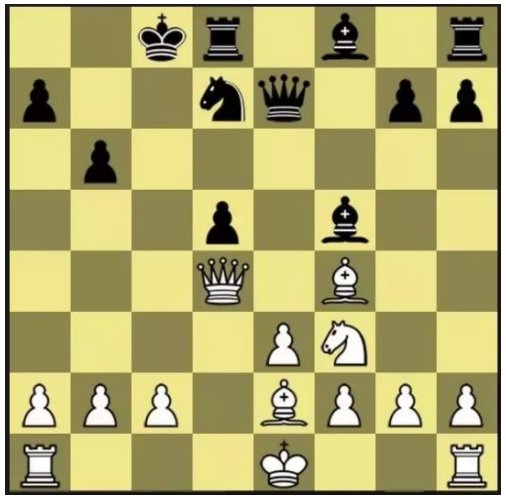
By David Willingham

Reinforcement Learning enables the use of Deep Learning for Controls and Decision Making Applications

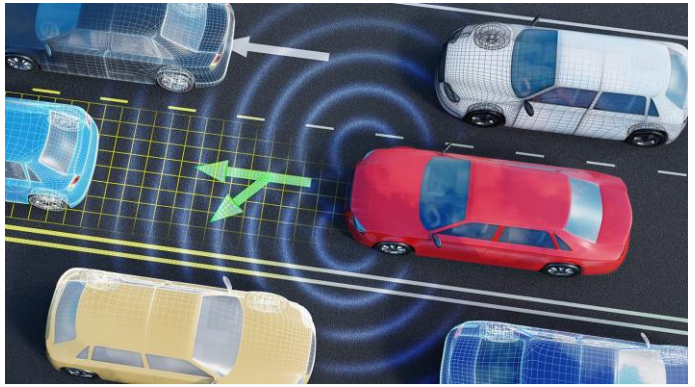


Finance

Robotics

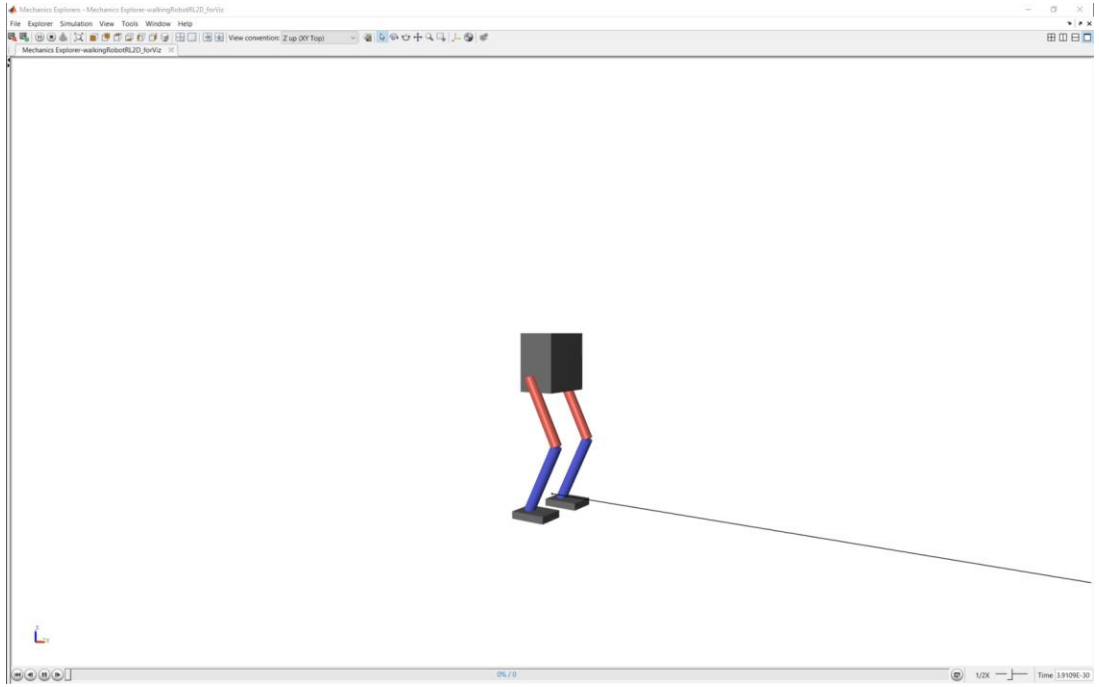


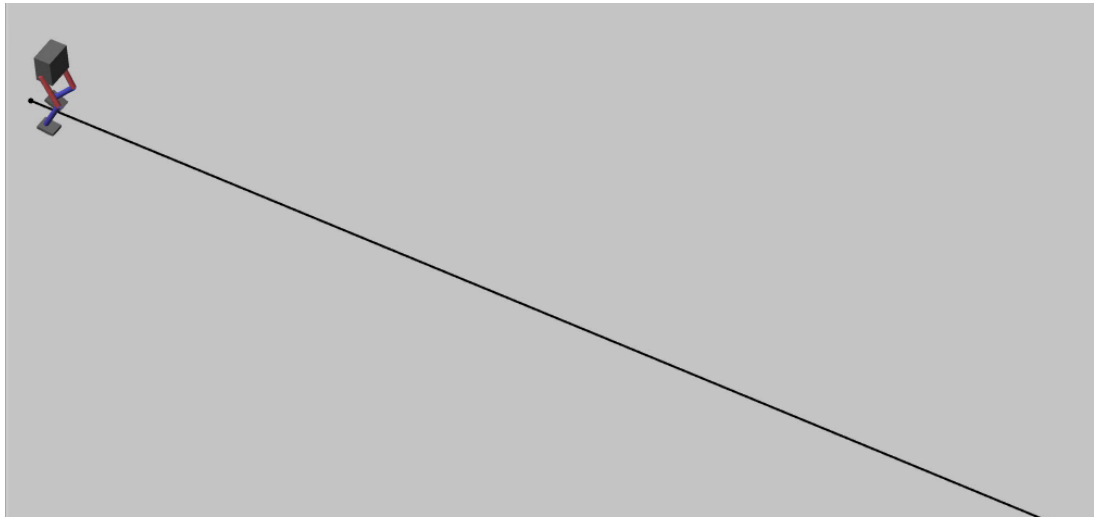
A.I. Gameplay



Autonomous driving

Example: Robot walking with deep reinforcement learning

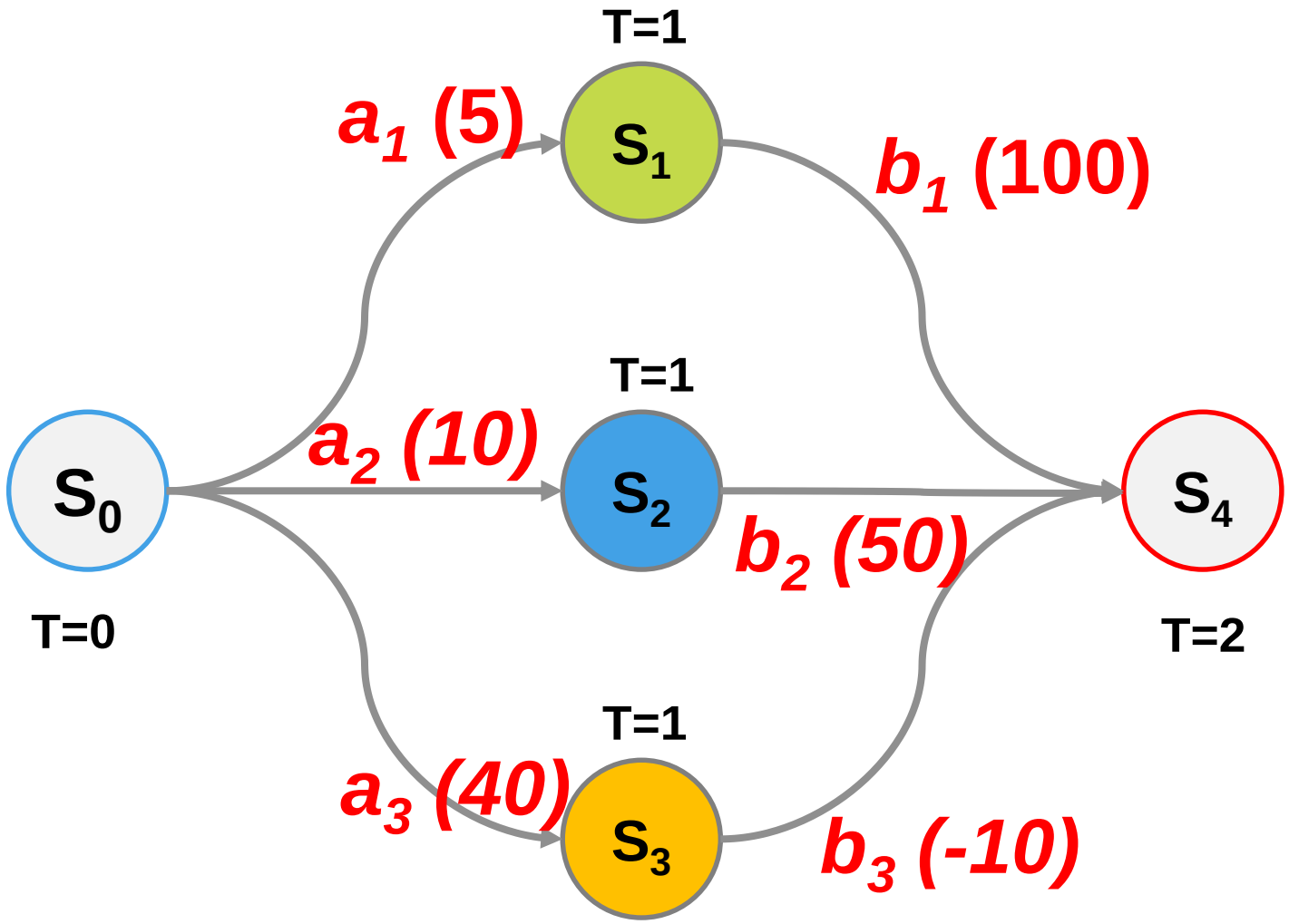




Simulation During Training

Trained Agent

Pop Quiz: What Would You Do?



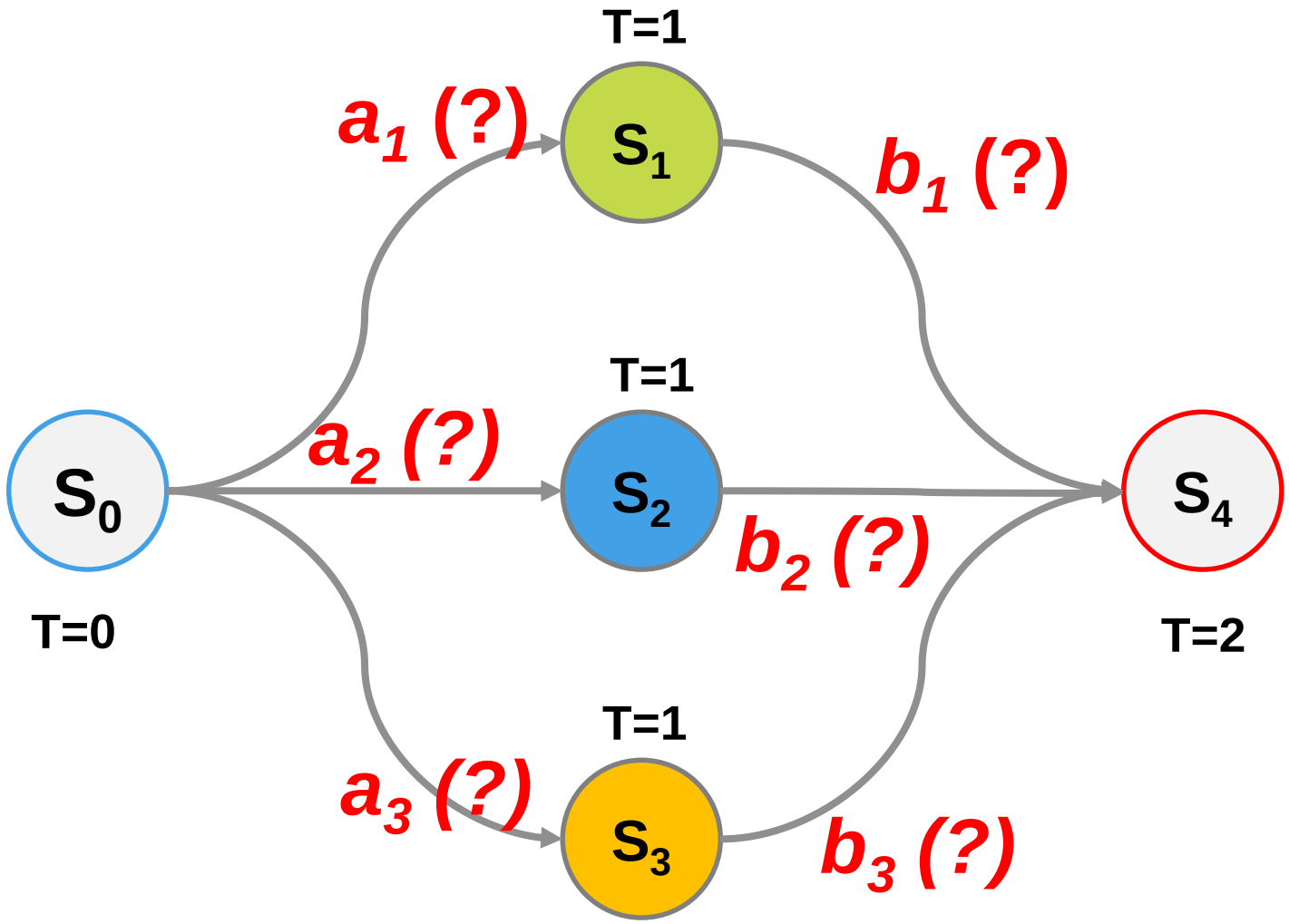
Action Value Function (Q)

- $Q(S_0,a_1)$ = **+105 (optimal)**
- $Q(S_0,a_2)$ = **+60**
- $Q(S_0,a_3)$ = **+30**

State Value Function (V)

- $V(S_0)$ = **+65 (avg)**
- $V(S_0)$ = **+105 (optimal)**

Reinforcement Learning uses Simulations to find the Optimal Action given a Certain State



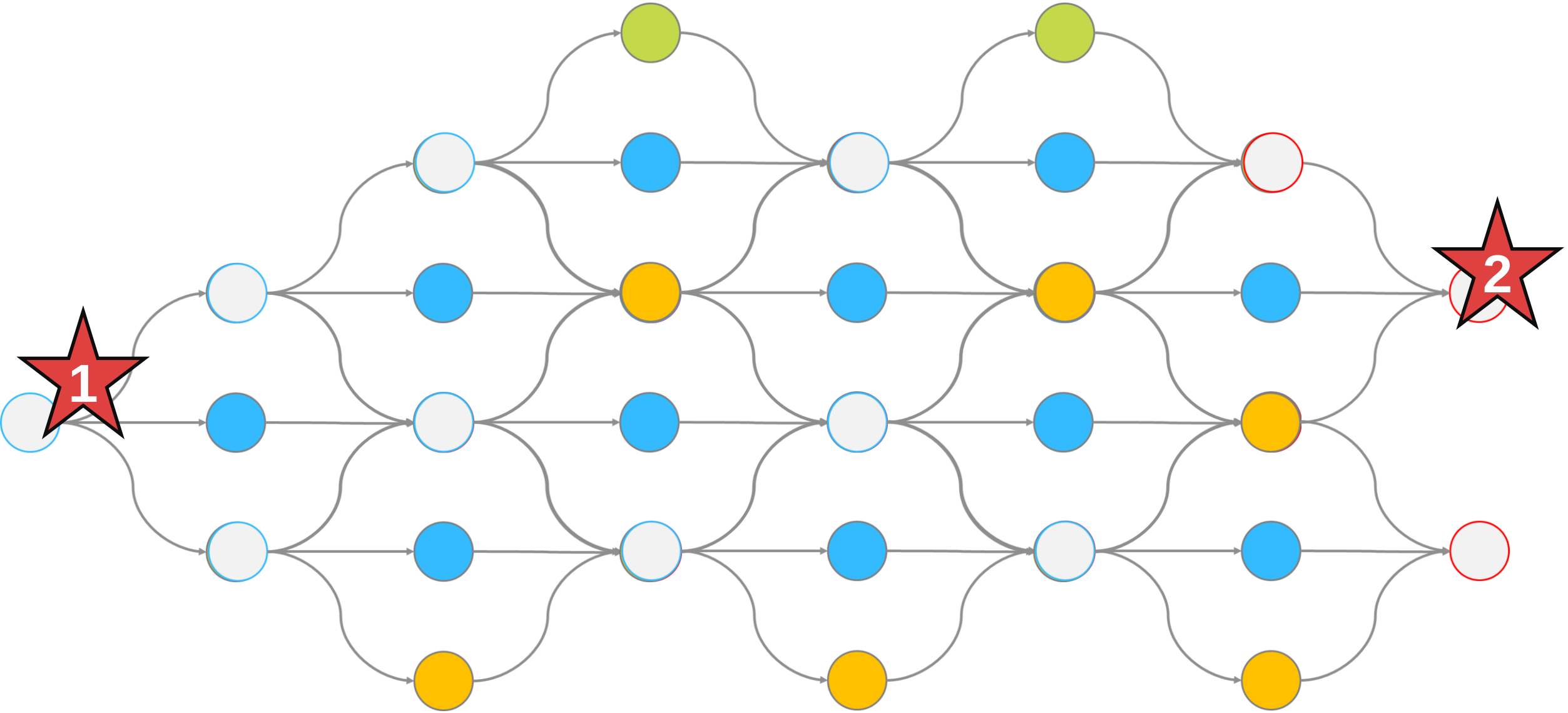
Action Value Function (Q)

- $Q(S_0, a_1)$ = **?**
- $Q(S_0, a_2)$ = **?**
- $Q(S_0, a_3)$ = **?**

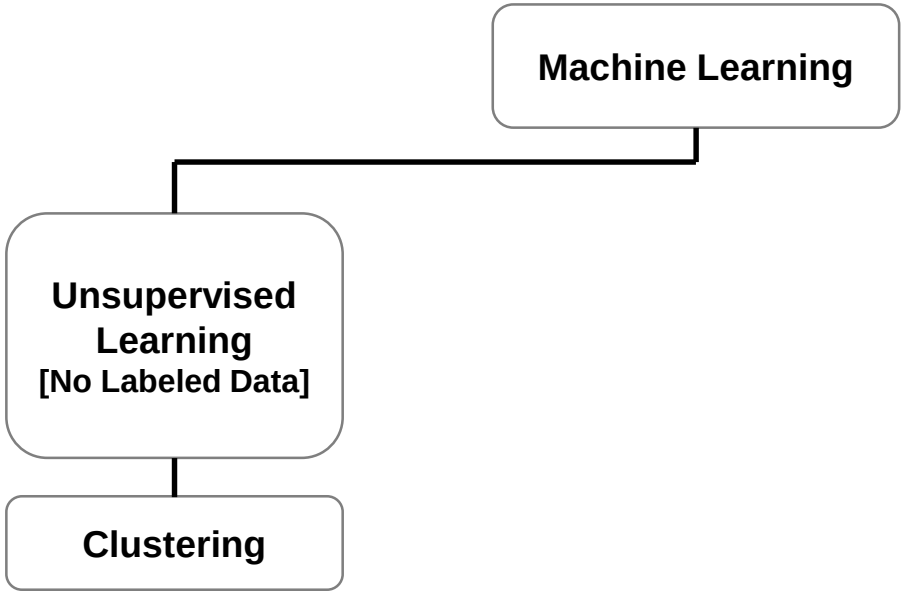
State Value Function (V)

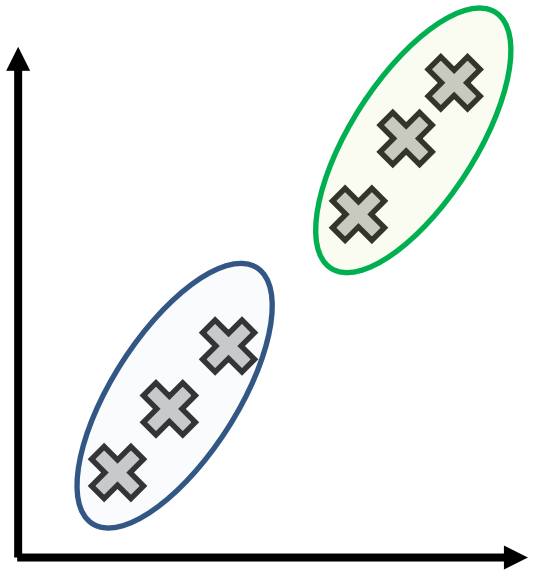
- $V(S_0)$ = **?**
- $V(S_1)$ = **?**

It can get very complicated very quickly

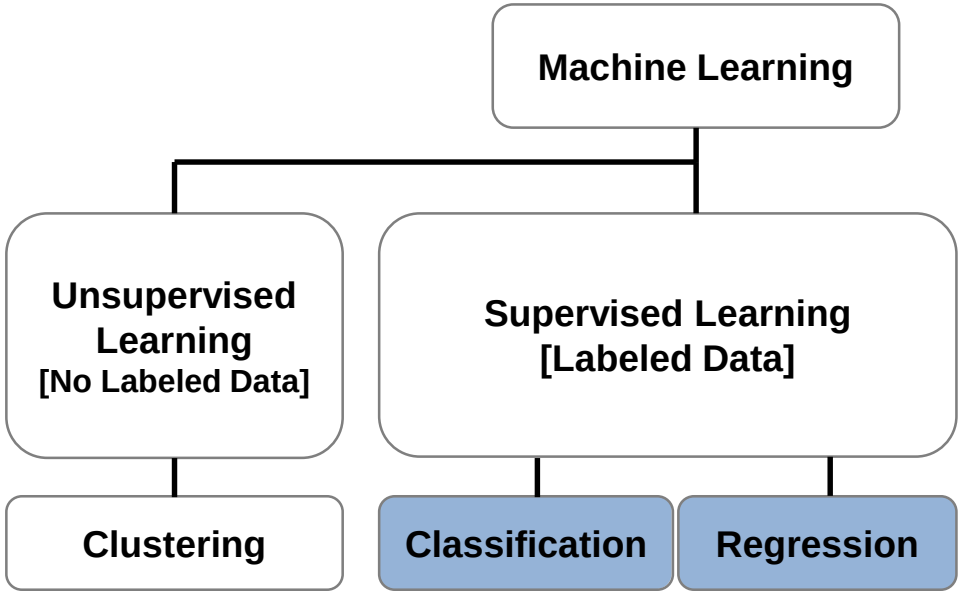


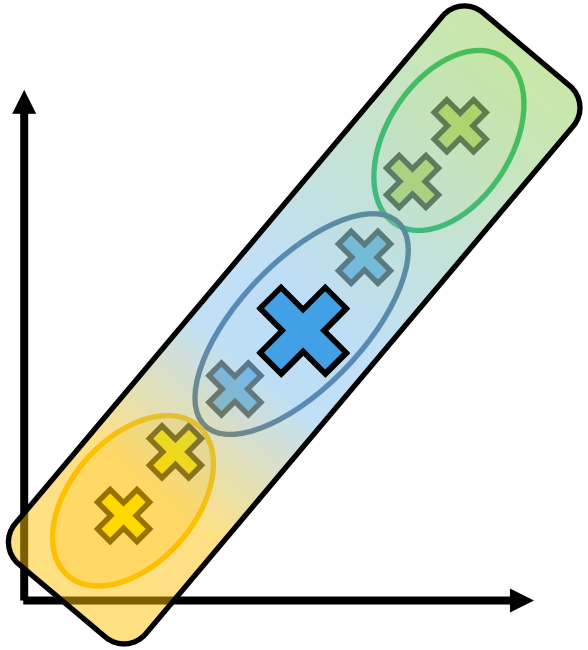
Machine Learning, Deep Learning, and Reinforcement Learning



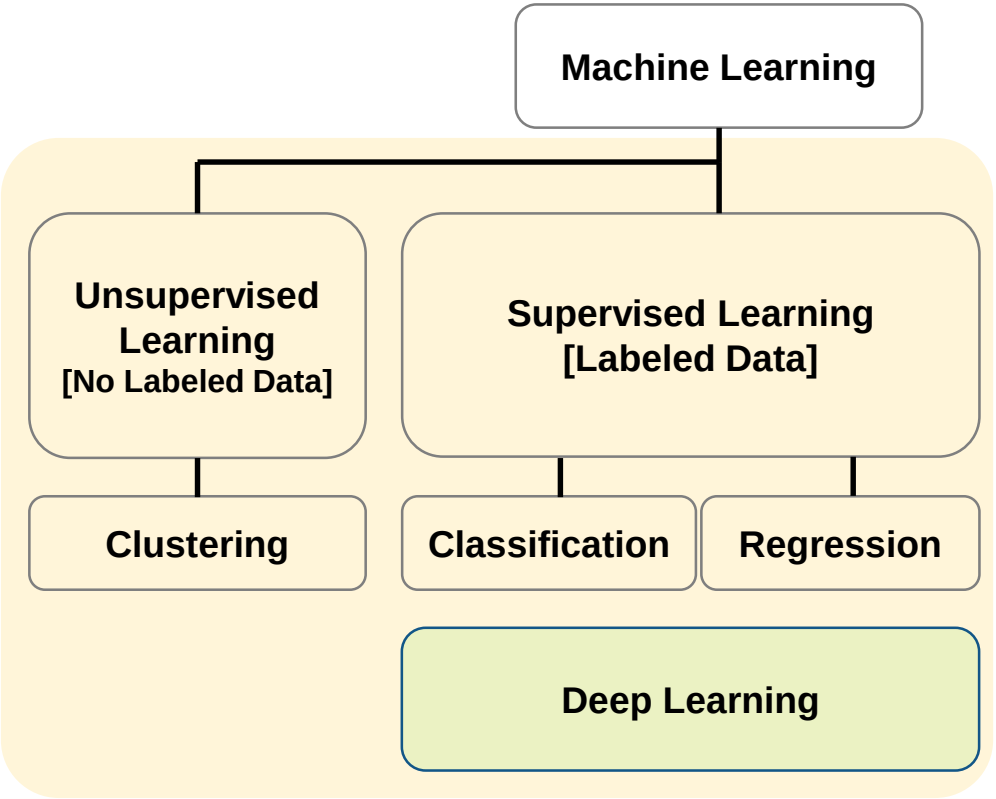


Machine Learning, Deep Learning, and Reinforcement Learning





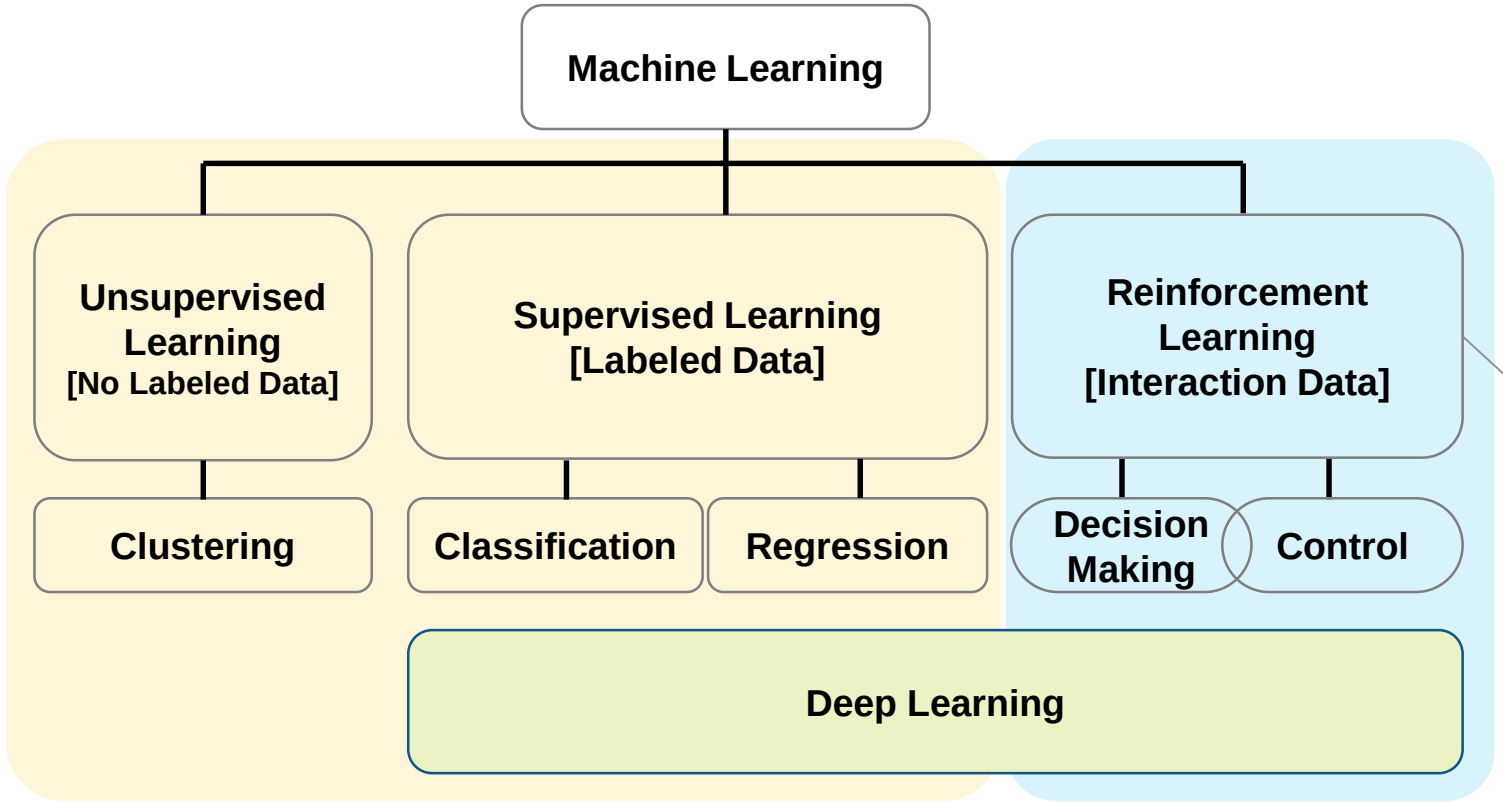
Machine Learning, Deep Learning, and Reinforcement Learning



Supervised learning typically involves feature extraction

Deep learning typically does not involve feature extraction

Machine Learning, Deep Learning, and Reinforcement Learning



Reinforcement learning:

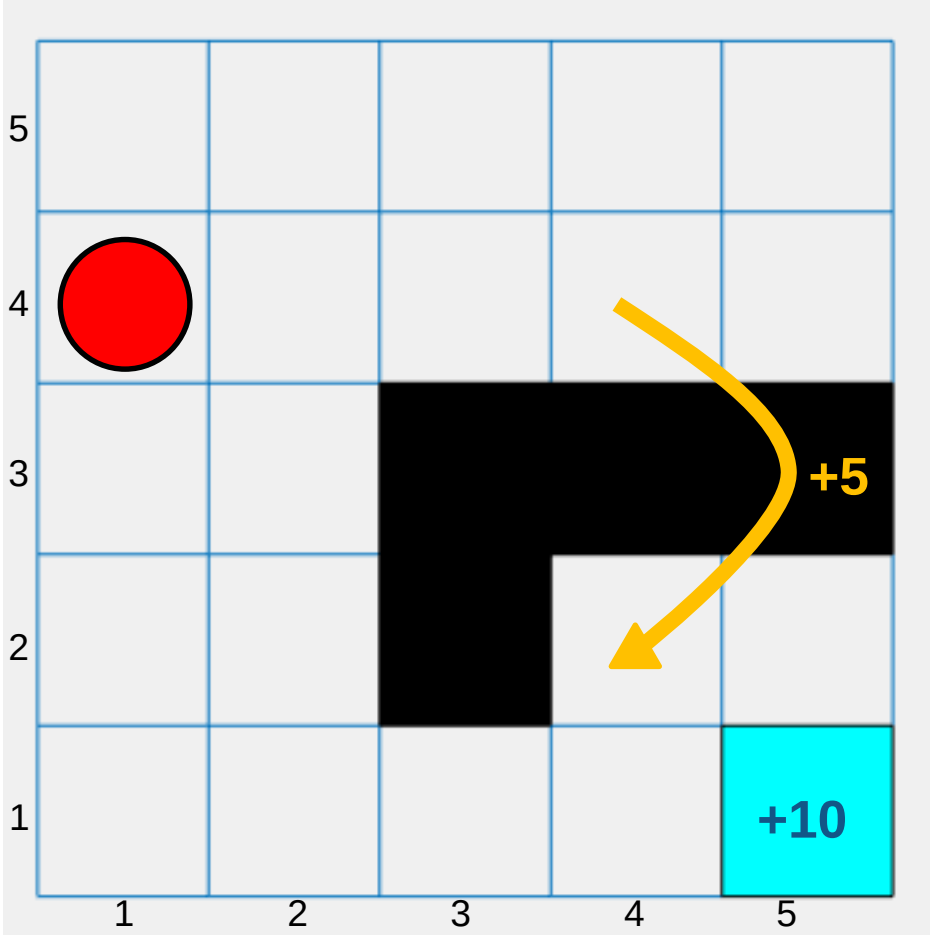
- Learning through trial & error [*interaction*]
- Complex problems typically need deep learning [*Deep Reinforcement Learning*]
- It's about learning a **behavior** or accomplishing a **task**

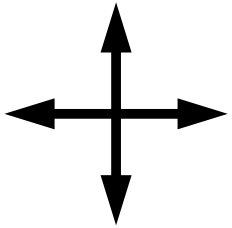
Reinforcement Learning Workflow

Environment **Reward** **Policy** **Agent** **Training** **Deploy**

Glossary of Common Terms in Reinforcement Learning

- **Agent**: **Red Circle** that learns how to navigate the grid to reach the **blue square** by trial and error
- **Environment**: 5x5 grid that is being navigated
- **State**: The current square the **red circle** is in
- **Action**: One of the 4 possible actions the red circle can take at each time step
- **Reward**: Points the **red circle** gets for taking an action
  - -1 for any move except
  - **+5** when you land on teleportation square [4,4]
  - **+10** when you land on [5,1]



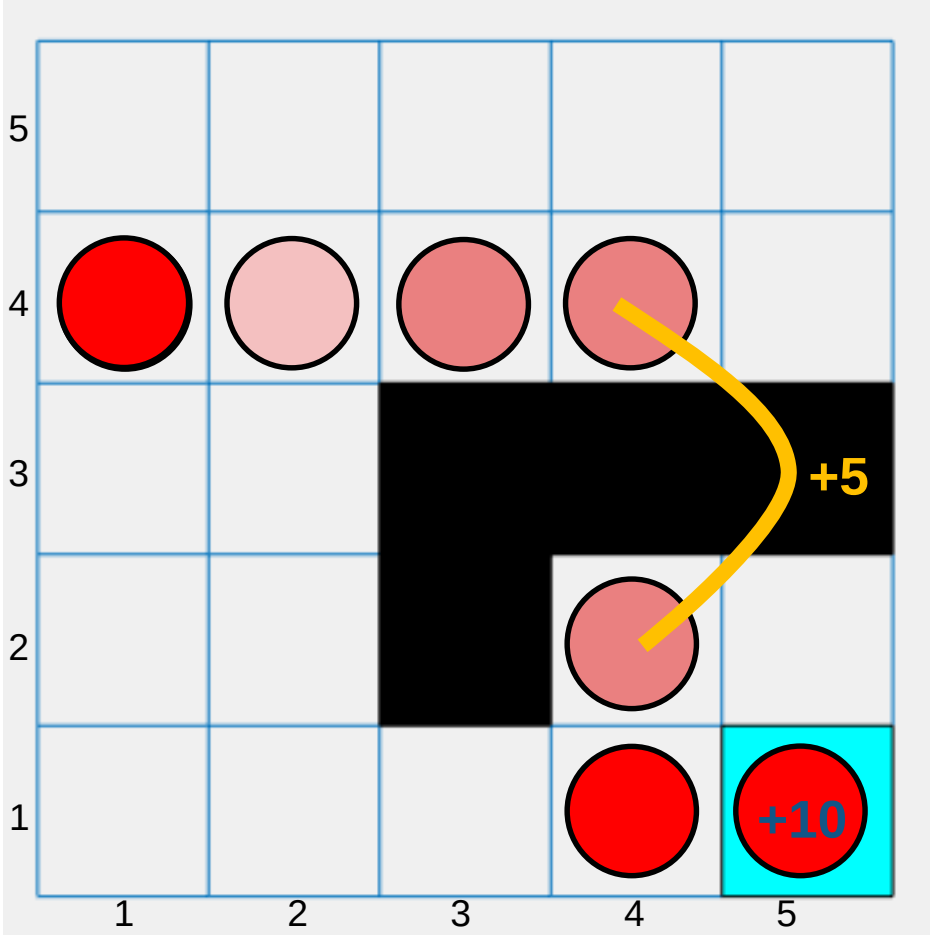


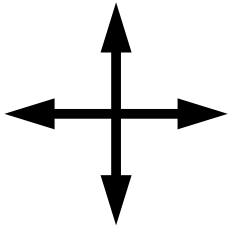
4 Possible Actions

Red circle does not know what possible reward values are

Glossary of Common Terms in Reinforcement Learning

- Trained Agent: **Red Circle** that **has learned** how to navigate the grid by taking the **best possible** actions
- Final Reward: +11 points
- **Policy**: The **logic** that is learned by **red circle** to implement the best possible actions. E.g.
  - If **red circle** is in [1,4], move **right**
  - If **red circle** is in [2,4], move **down**
- Reinforcement Learning Algorithm: The **trial-and-error algorithm** that developed this policy

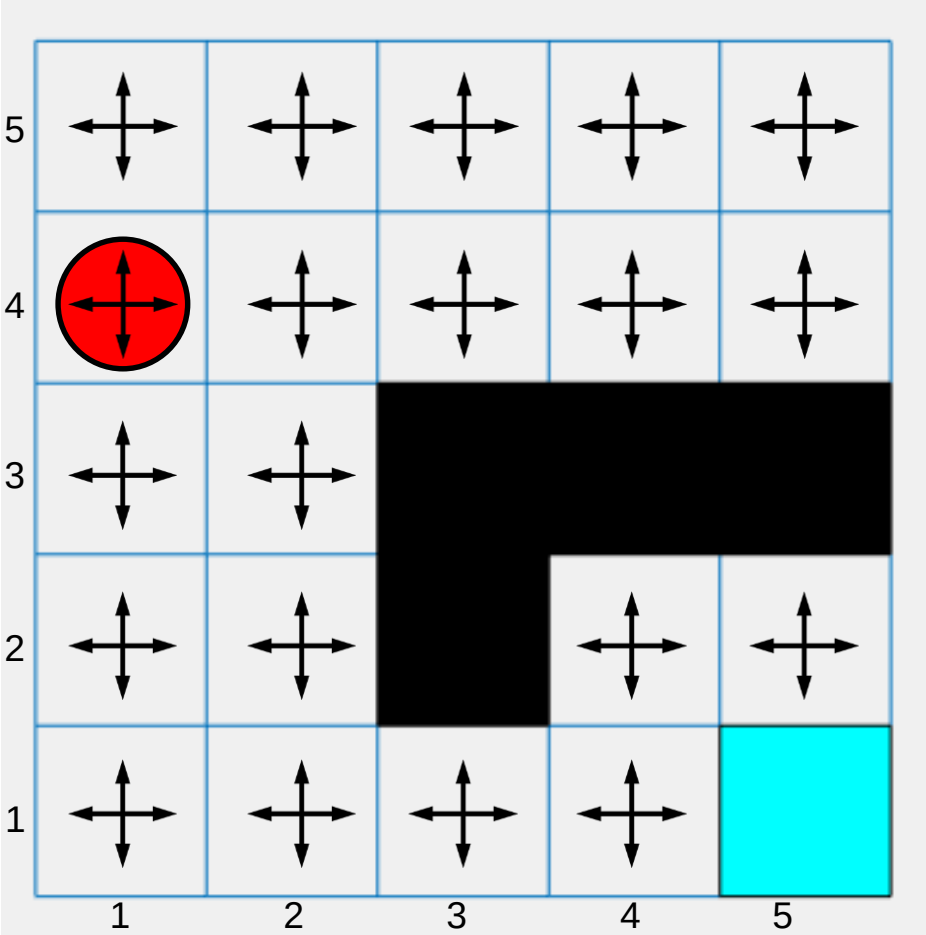




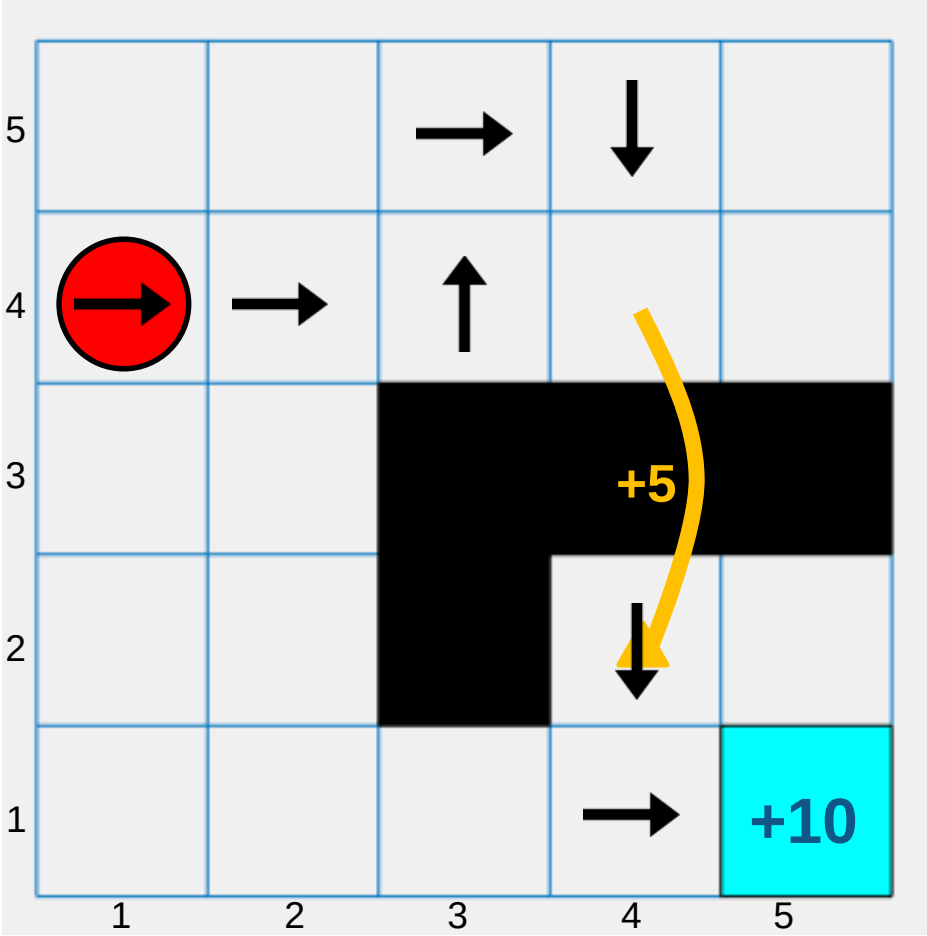
4 Possible Actions

The best action to take depends on the state

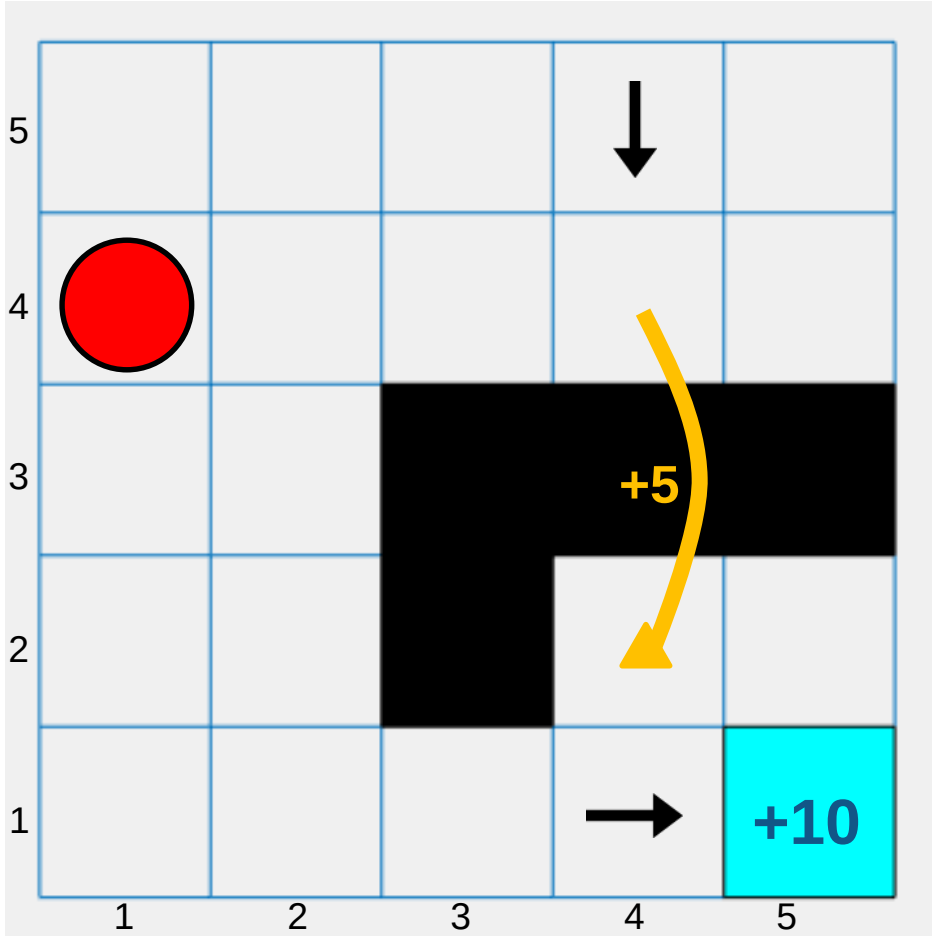
This Simple Example Can Illustrate Why



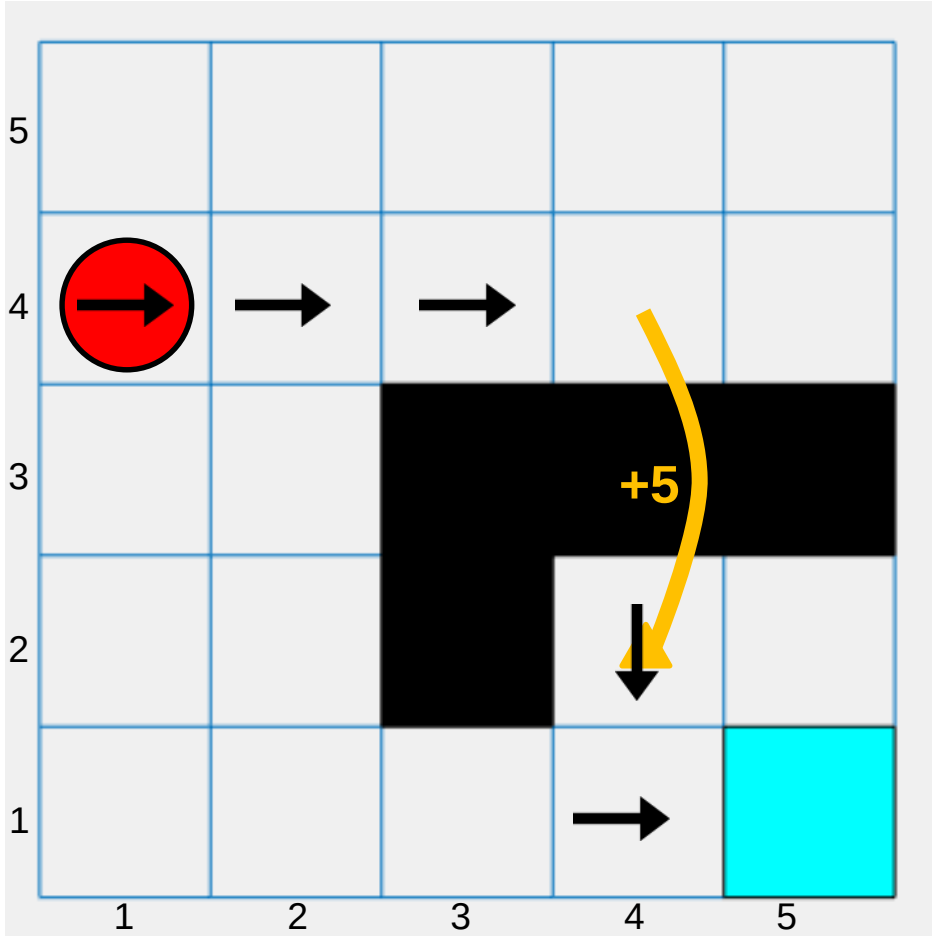
In This Sample Trajectory, We Luckily Receive Two Rewards



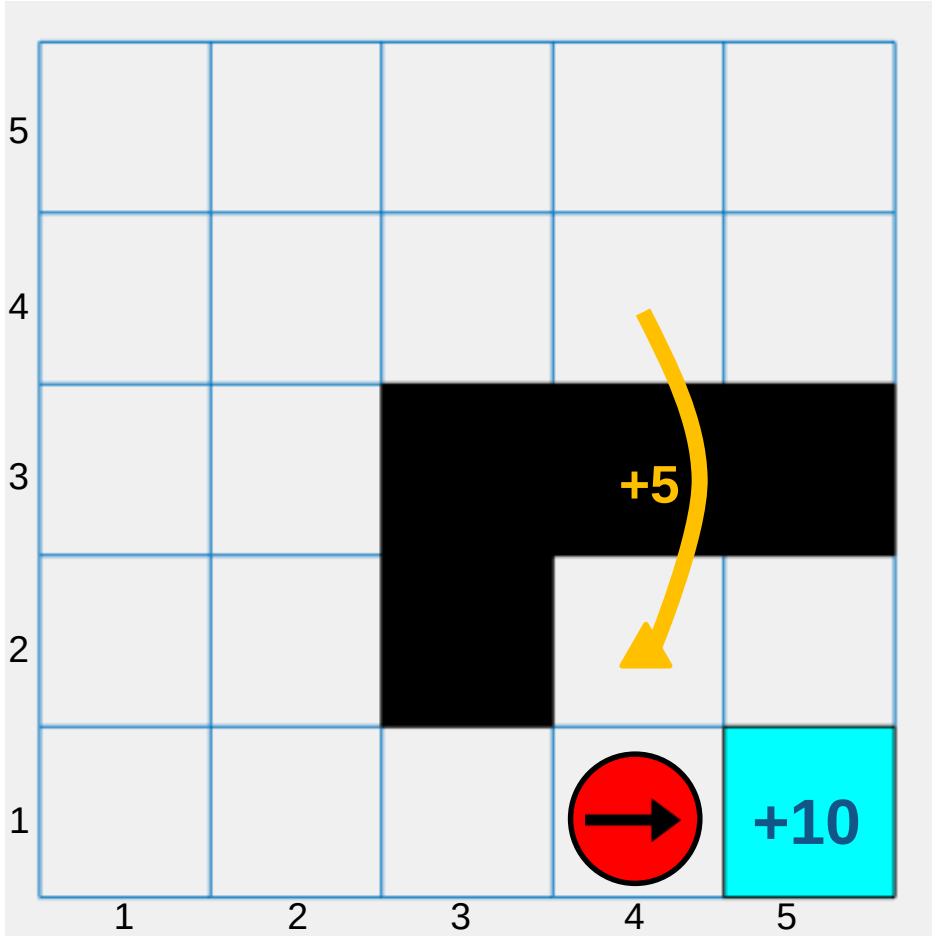
And Now, the Agent Remembers Which Two Actions Led to the Reward



Eventually, We Find the Best Path Possible Based On Our Initial State

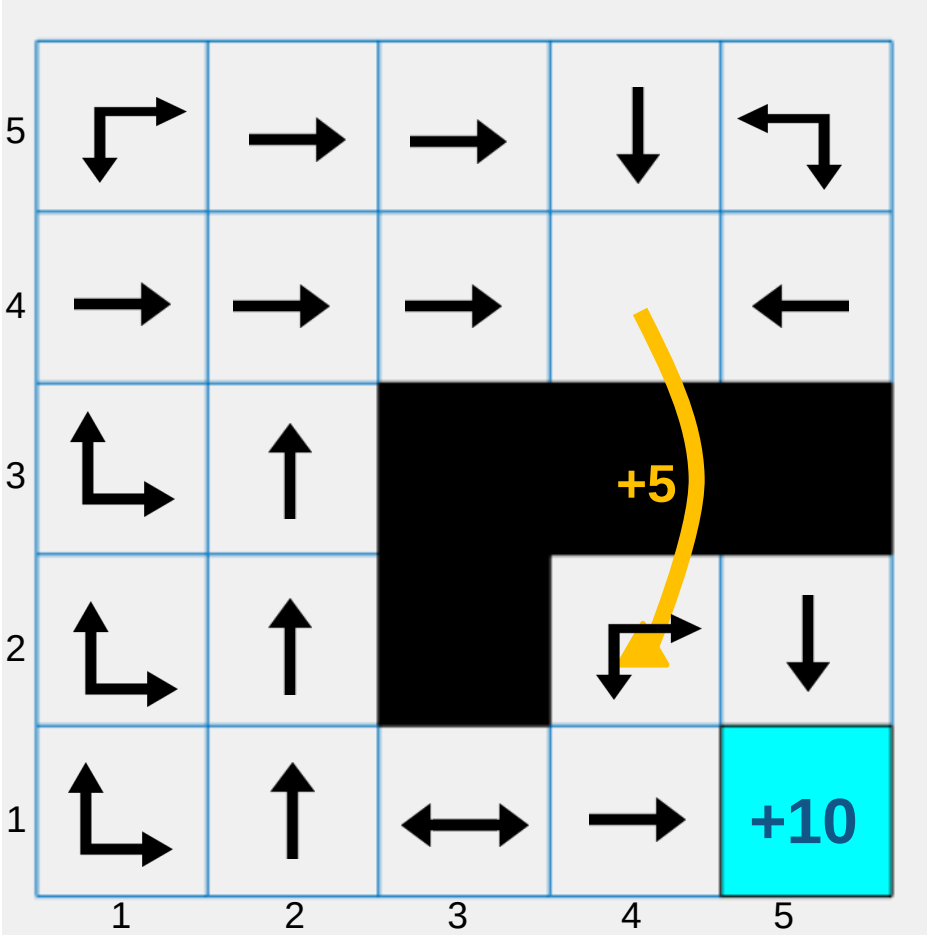


But What If We Had a Different Initial State? Would the Same Path Be the Best Choice?



Clearly, the Best Action to Take Depends on the State We Are In

In this case, we only have 21 possible states

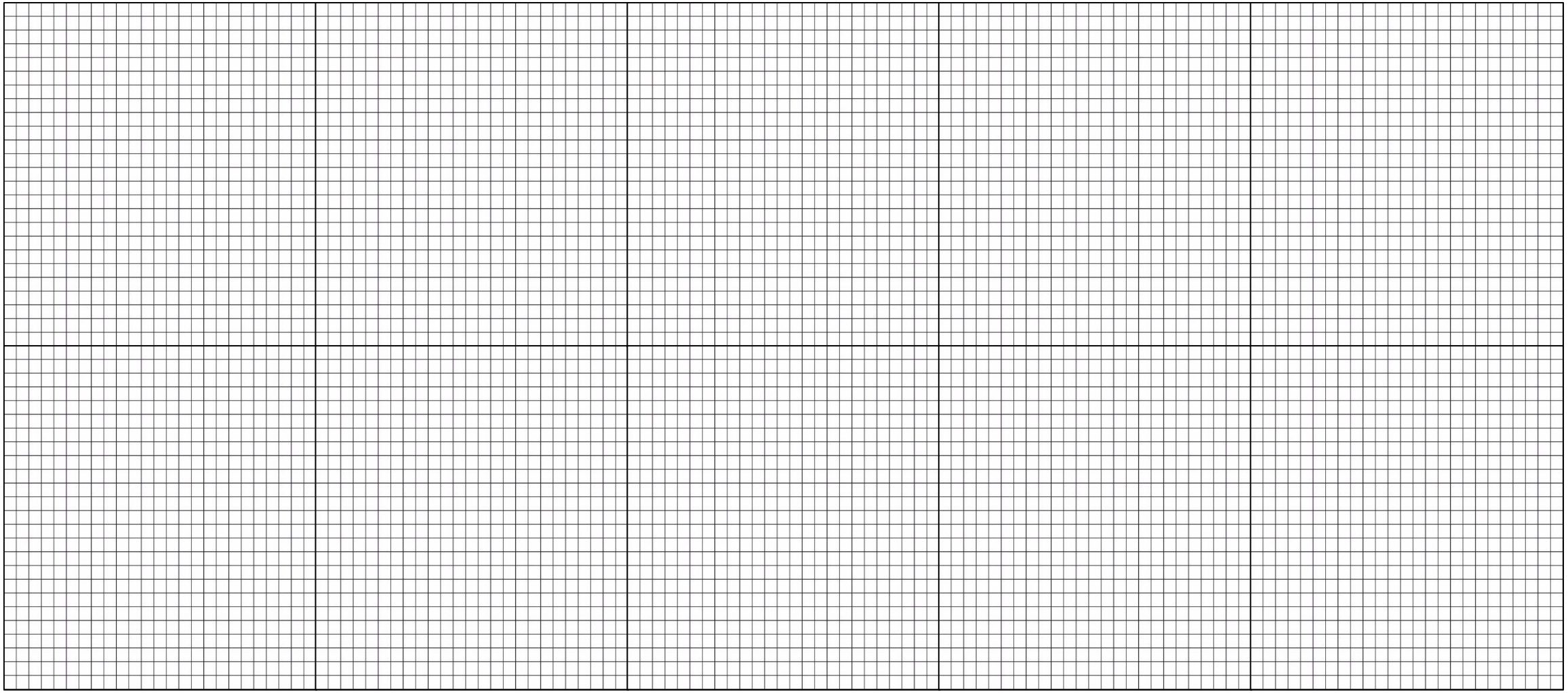


In this case, we can run a small and finite number of simulations to find the best possible path irrespective of our initial state

https://www.mathworks.com/help/reinforcement-learning/ug/train-q-learning-agent-to-solve-basic-grid-world.html

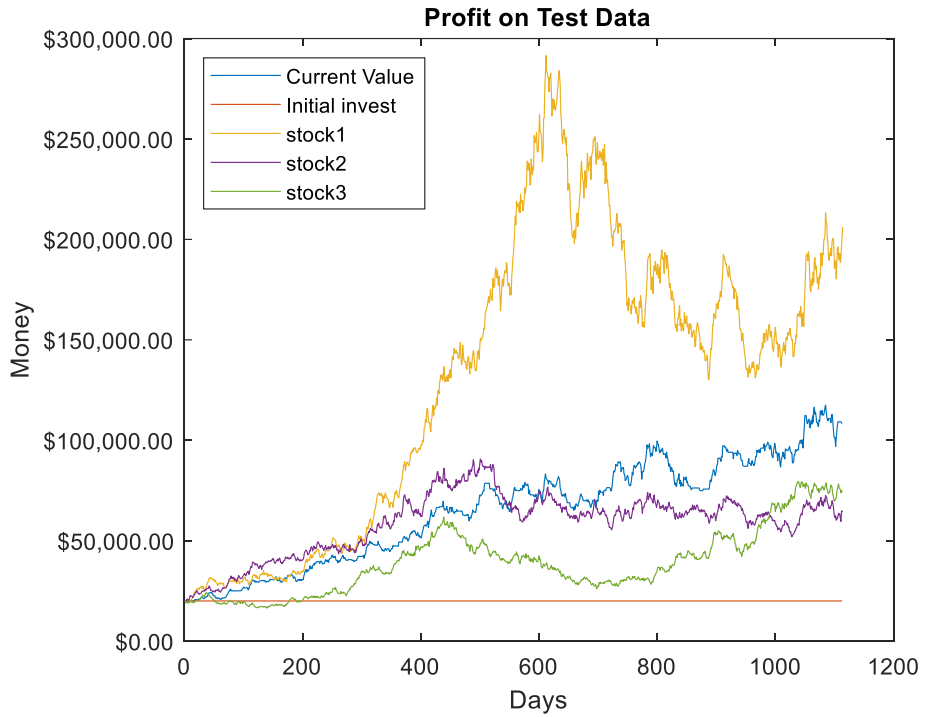
>> openExample('rl/BasicGridWorldExample')

But What If We Have Many States?

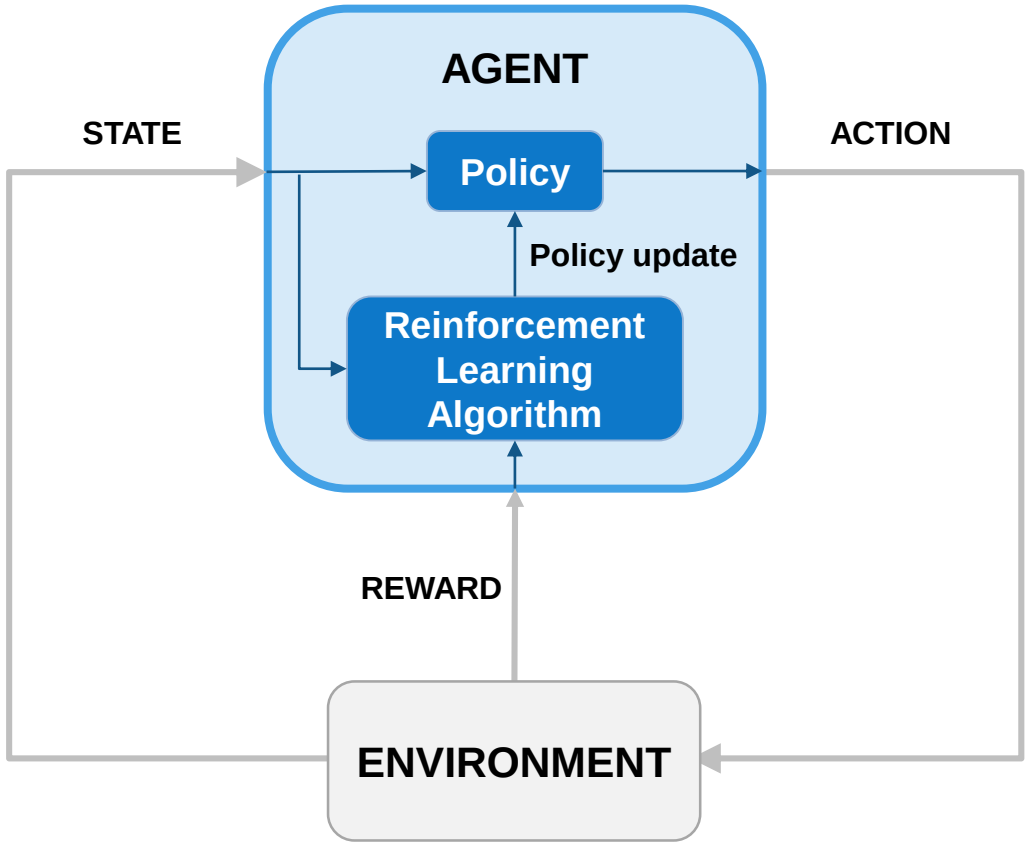


Financial Trading Example

- The goal:
  - Learn how to trade the financial markets without ever losing money.



A Practical Example of Reinforcement Learning
Financial Trading Example



- A computer learns how to trade… (**agent**)
- using financial market indicators (daily/weekly rets etc),… (**state**)
- that represent a financial market,… (**environment**)
- by generating buy, hold, sell signals… (**actions**)
- based on an internal state-to-action mapping… (**policy**)
- that tries to optimize good trading behavior (not profit)… (**reward**).

- The policy is updated through repeated trial-and-error by a **reinforcement learning algorithm**

Financial Trading Example

- Our environment consists of 3 stocks, cash & historical data:
  - Stocks are Simulated via Geometric Brownian Motion
  - $20,000, 15 years of Closing Prices
- Actions (buy, sell ,hold) for 3 stocks = 27 total actions
- The States being observed are:
  - Stocks Owned
  - % Price Different when Bought
  - Cash In Hand
  - % Price change from yesterday
  - % Price change from 2 days ago
  - % Price change from 7 days ago
  - % Price change from average price of 7 days ago

Strategy

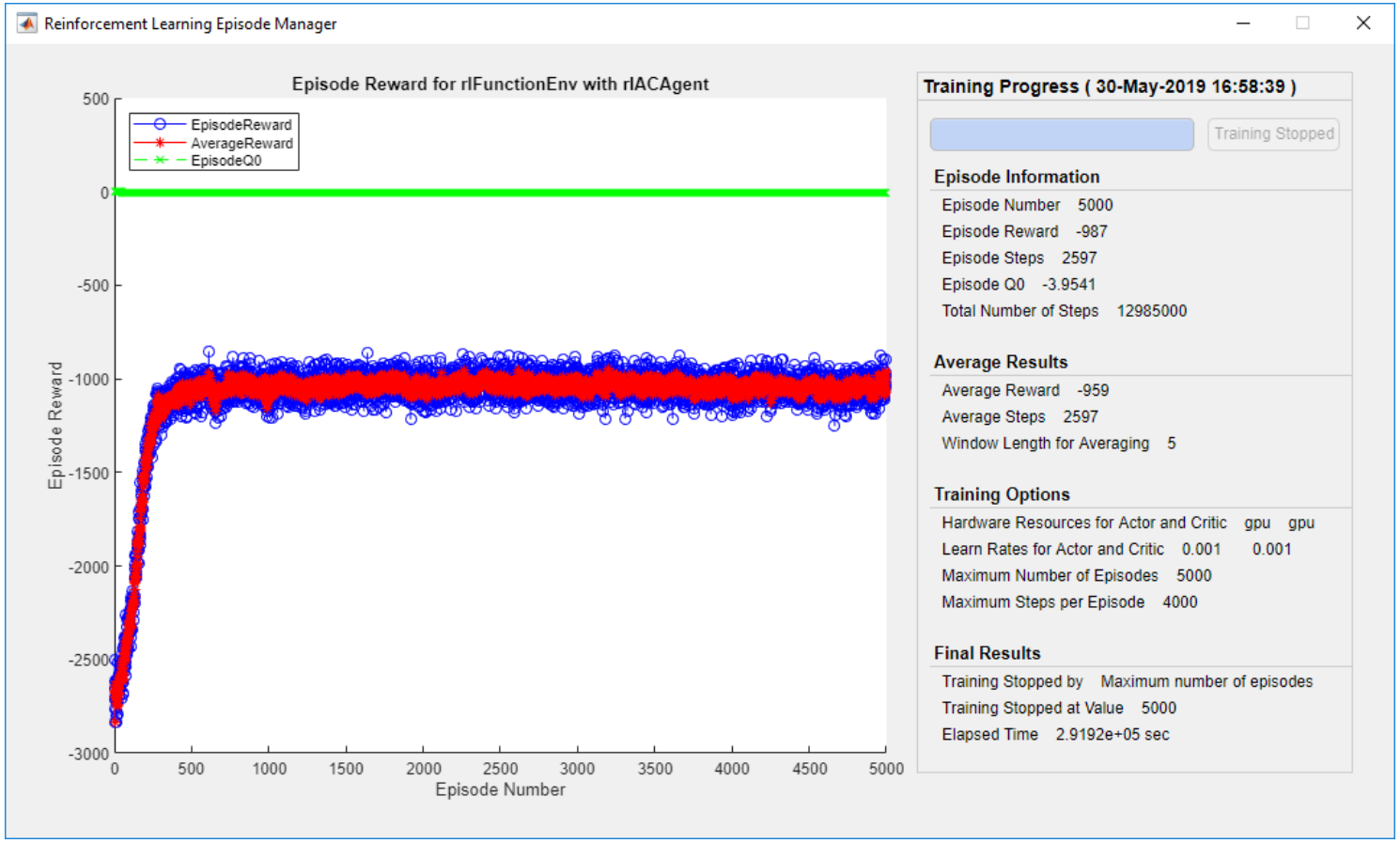
- Given 3 stocks
- Try to find the best time to buy, sell, or hold each stock
- If selling a stock, sell all of it.
- If buying a stock, buy the maximum allowed given cash in hand.

Reward

- A good reward is given when
  - A selling trade results in profit
  - A selling trade occurs with momentum
    - I.e. selling when prices start to fall
  - Holding a stock results in the portfolio value increasing
- A bad reward is given when
  - Selling results in a negative profit
  - A selling trade occurs against momentum
    - I.e. selling when prices are increasing, buying when prices are falling
  - Holding a stock results in the portfolio value decreasing
  - A good reward is overwritten to be bad if ***any*** illogical trade occurs
    - I.e. sell a stock you don't have or buy when you don't had cash

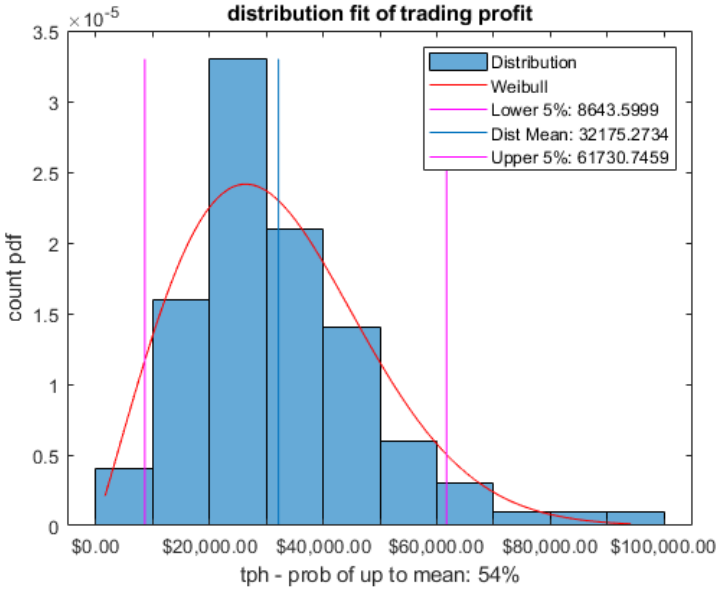
Training Progress

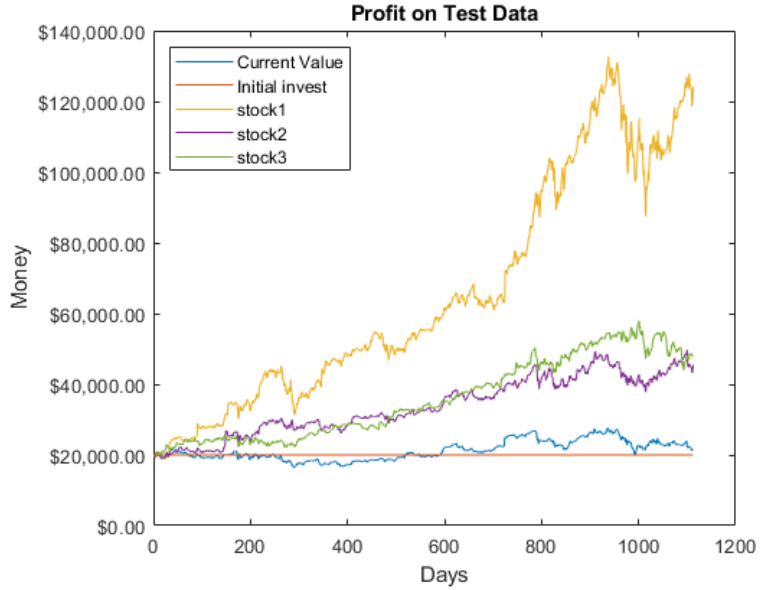
- Based 12years of data
- 3000 episodes
- ~80hrs to train



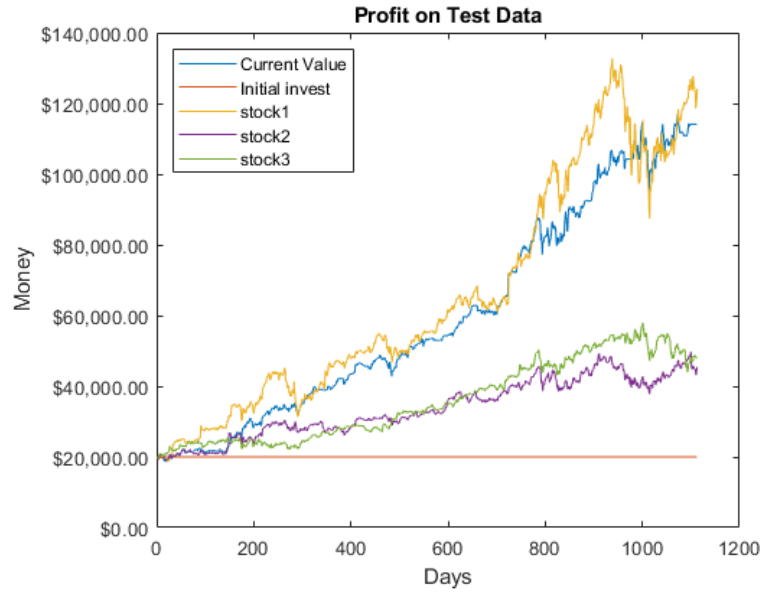
Results

- 100 Simulations of 3 years:
  - Average profit - ~$30k
  - 0 simulations returned negative
  - But most did not outperform individual stocks





Worst Sim



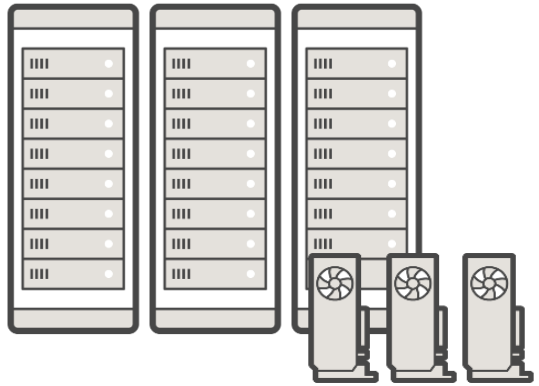
Best Sim

Further Improvements

- Transaction costs
- Cover the Hi/Lo spread
- Refined reward system
- Try different agents

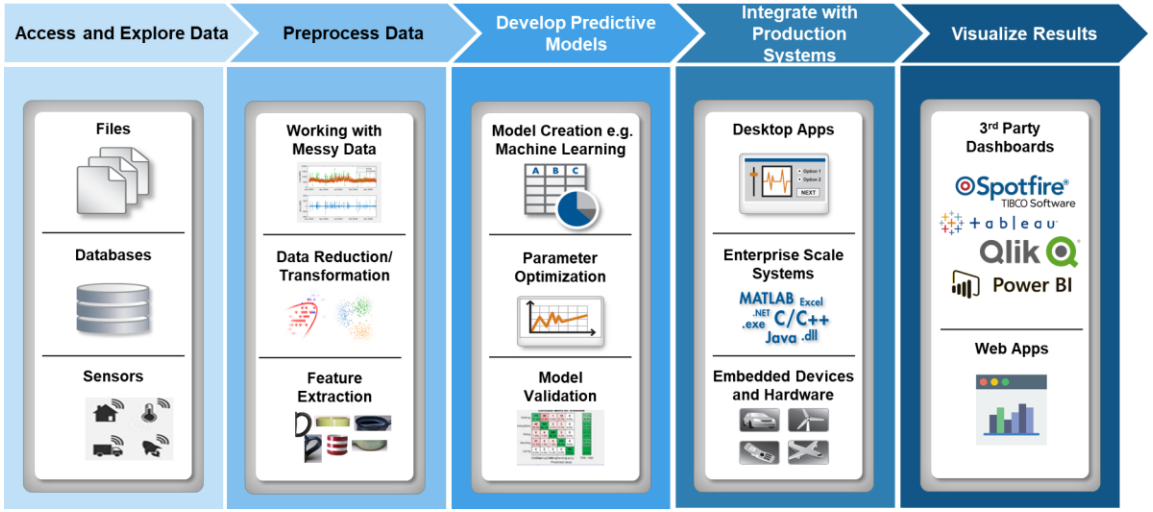
Why MATLAB?

- Flexible programming environment
  - The case study took ~2 weeks to develop



- Easy to scale computations
  - Cloud, Parallel & GPU computing are just “options”

- Supports the entire workflow
  - From data to model to deployment



Further Resources

- Reinforcement Learning – [YouTube Series]