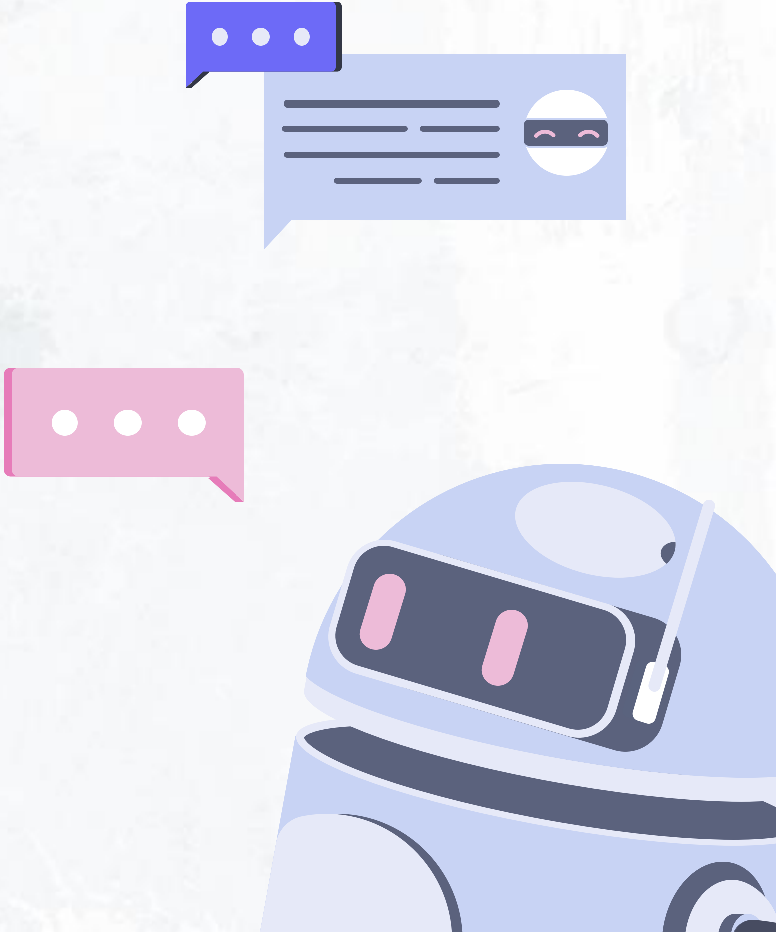
Reinforcement Learning



(AI)

Table of contents

01 → Prerequisites and revisiting Neural Networks

02 → A basic investment strategy

03 → Reinforcement Learning: Core concepts

04 → Reinforcement Learning: Q-Learning

01 →

Prerequisites and revisiting Neural Networks

(AI)

Requirements for this seminar

Knowledge	Origin	Relevance
Deep Neural Networks	Deep Learning	High
Gradient Descent	Deep Learning	Medium
Cost Function	Data Analysis	Medium
Keras/Tensorflow	Deep Learning	High
R Programming	Data Analysis	High
Math Skills	School/Uni	Low
Knowledge Transfer	Uni	High

Organization

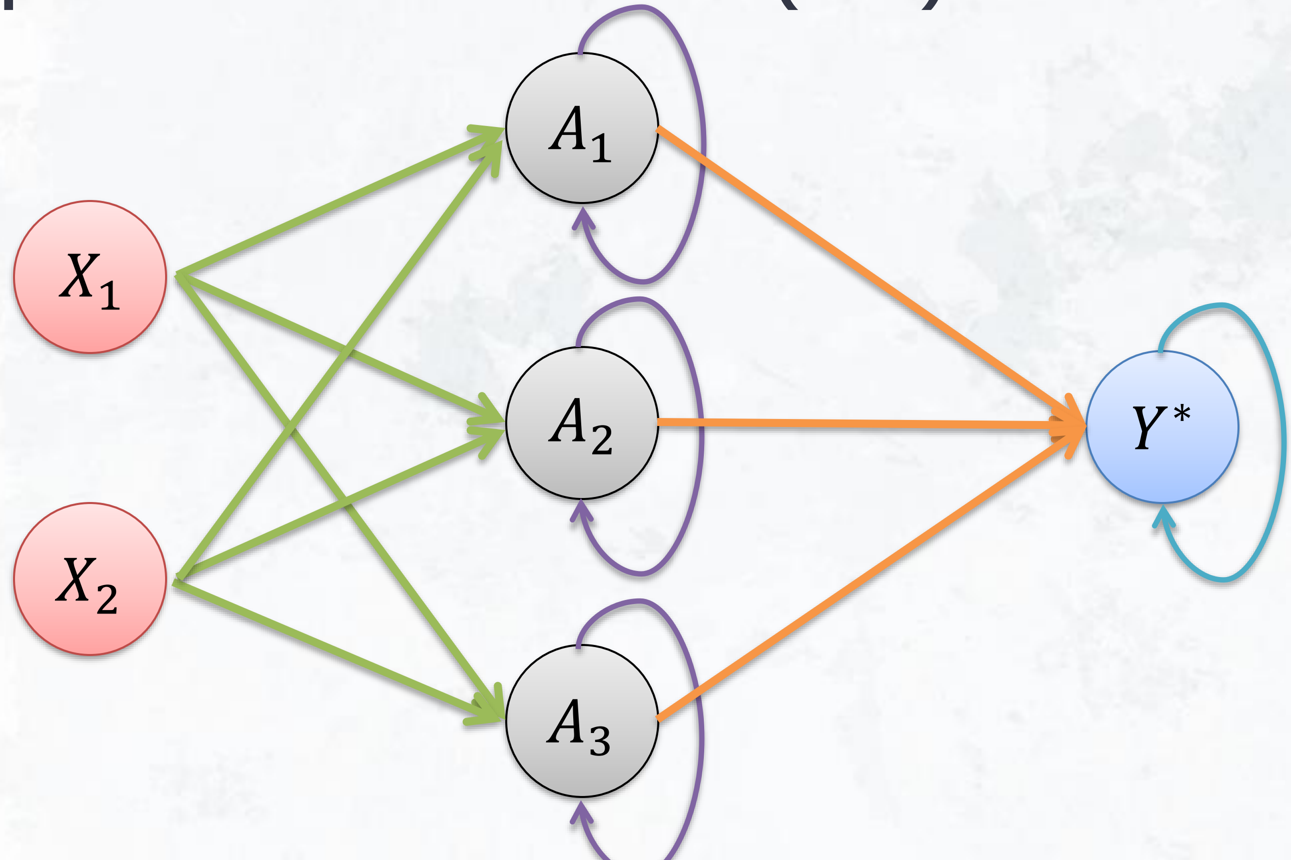
In week one you will receive a **brief introduction** to the current topic. Afterwards, you must make a **binding registration** for the seminar until Sunday, the 08.06.2025 23:59, via mail to steinert@europa-uni.de. This registration always leads to a grade.

In weeks 3, 4 and 5 there will be three preparation tasks you need to solve leading to **mandatory presentations** on the group level including graded questions by me. Weeks six to eight will be independent group work with individual **meetings with me on demand**. The submission deadline for the fourth and **final task** is Sunday, 27.07.2025, at 23:59.

Your seminar project

- Create a standard NN which aims to learn the optimal investment strategy
- Build the environment and replay buffer for reinforcement learning
- Build and train a Q-learning NN architecture
- Design and train your own Q-Learning NN architecture

Simple Neural Network (NN)



02 →

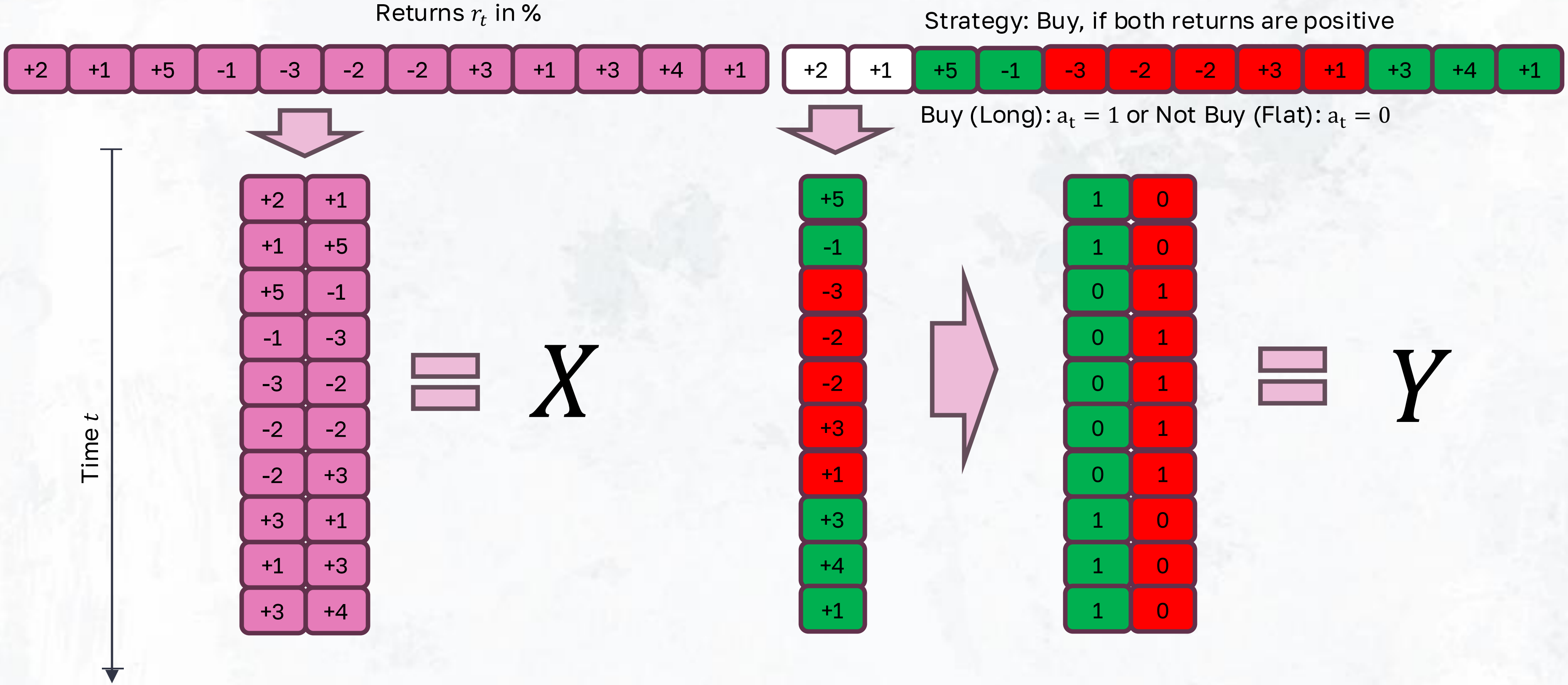
A basic investment strategy

(AI)

To buy or not to buy – that‘s the question!

Time t	$t = 0$											$t = T$
Returns r_t in %	+2	+1	+5	-1	-3	-2	-2	+3	+1	+3	+4	+1
Situation: Two past returns observable	+2	+1	?	?	?	?	?	?	?	?	?	?
Possible trades	Buy (Long): $a_t = 1$ or Not Buy (Flat): $a_t = 0$											
Strategy: Buy, if both returns are positive	+2	+1	🟩 +5	🟩 -1	🟥 -3	🟥 -2	🟥 -2	🟥 +3	🟥 +1	🟩 +3	🟩 +4	🟩 +1

From trades to supervised learning



Returns r_t in %

+2	+1	+5	-1	-3	-2	-2	+3	+1	+3	+4	+1

Time t

+2	+1
+1	+5
+5	-1
-1	-3
-3	-2
-2	-2
-2	+3
+3	+1
+1	+3
+3	+4

$= X$

Strategy: Buy, if both returns are positive

+2	+1	+5	-1	-3	-2	-2	+3	+1	+3	+4	+1

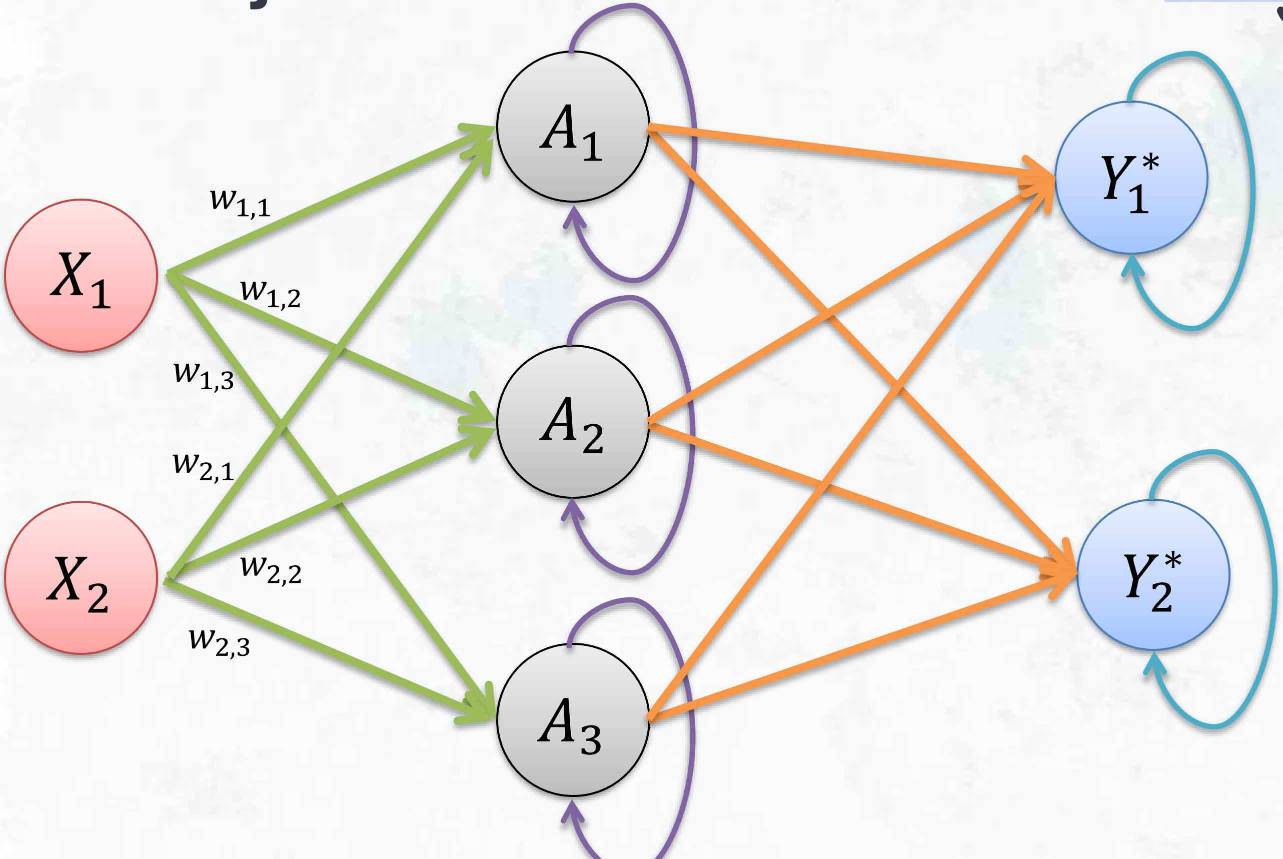
Buy (Long): $a_t = 1$ or Not Buy (Flat): $a_t = 0$

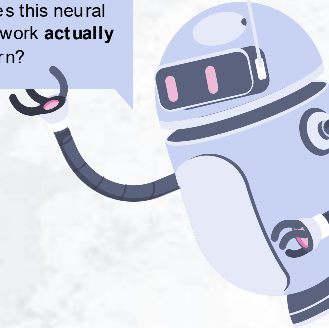
+5	1	0
-1	1	0
-3	0	1
-2	0	1
-2	0	1
+3	0	1
+1	0	1
+3	1	0
+4	1	0
+1	1	0

$= Y$

Idea: Now just model it with NN!

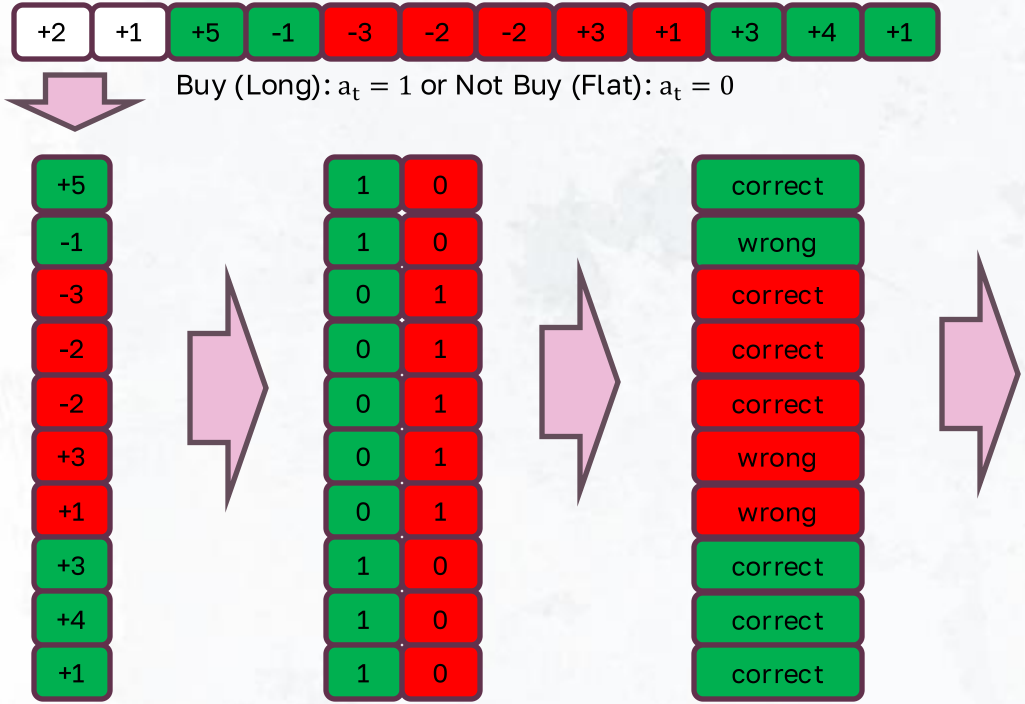
Let us focus, what does this neural network **actually** learn?





How good was this strategy?

Strategy: Buy, if both returns are positive



+2	+1	+5	-1	-3	-2	-2	+3	+1	+3	+4	+1

Buy (Long): $a_t = 1$ or Not Buy (Flat): $a_t = 0$

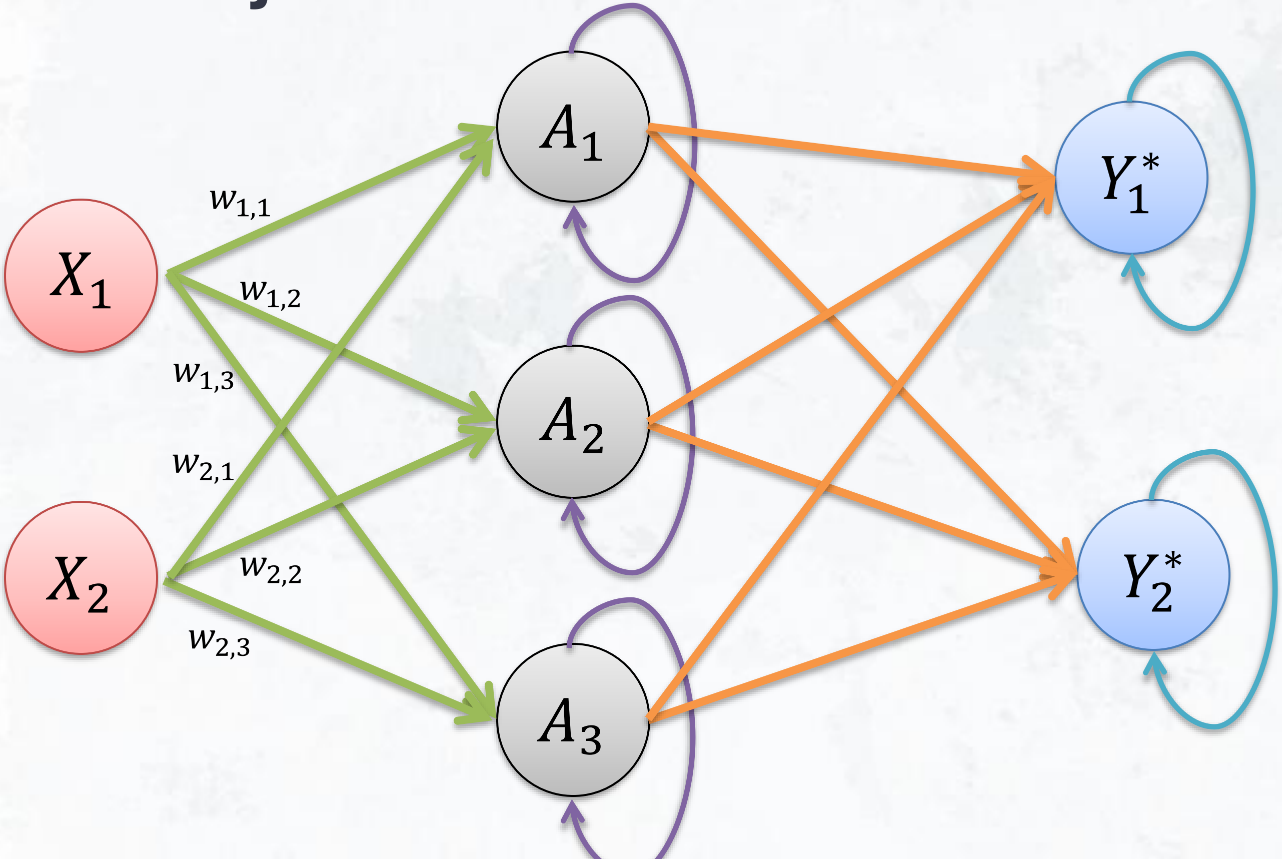
+5	1	0	correct
-1	1	0	wrong
-3	0	1	correct
-2	0	1	correct
-2	0	1	correct
+3	0	1	wrong
+1	0	1	wrong
+3	1	0	correct
+4	1	0	correct
+1	1	0	correct

The strategy was imperfect, so we need to learn from other (better) traders!

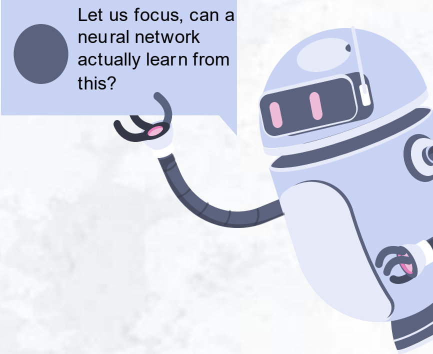
Idea: Just buy, when it goes up!

Time t	$t = 0$											$t = T$
Returns r_t in %	+2	+1	+5	-1	-3	-2	-2	+3	+1	+3	+4	+1
Situation: Two past returns observable	+2	+1	?	?	?	?	?	?	?	?	?	?
Possible trades	Buy (Long): $a_t = 1$ or Not Buy (Flat): $a_t = 0$											
Strategy: Buy, if future returns are positive	+2	+1	🟩 +5	🟥 -1	🟥 -3	🟥 -2	🟥 -2	🟩 +3	🟩 +1	🟩 +3	🟩 +4	🟩 +1

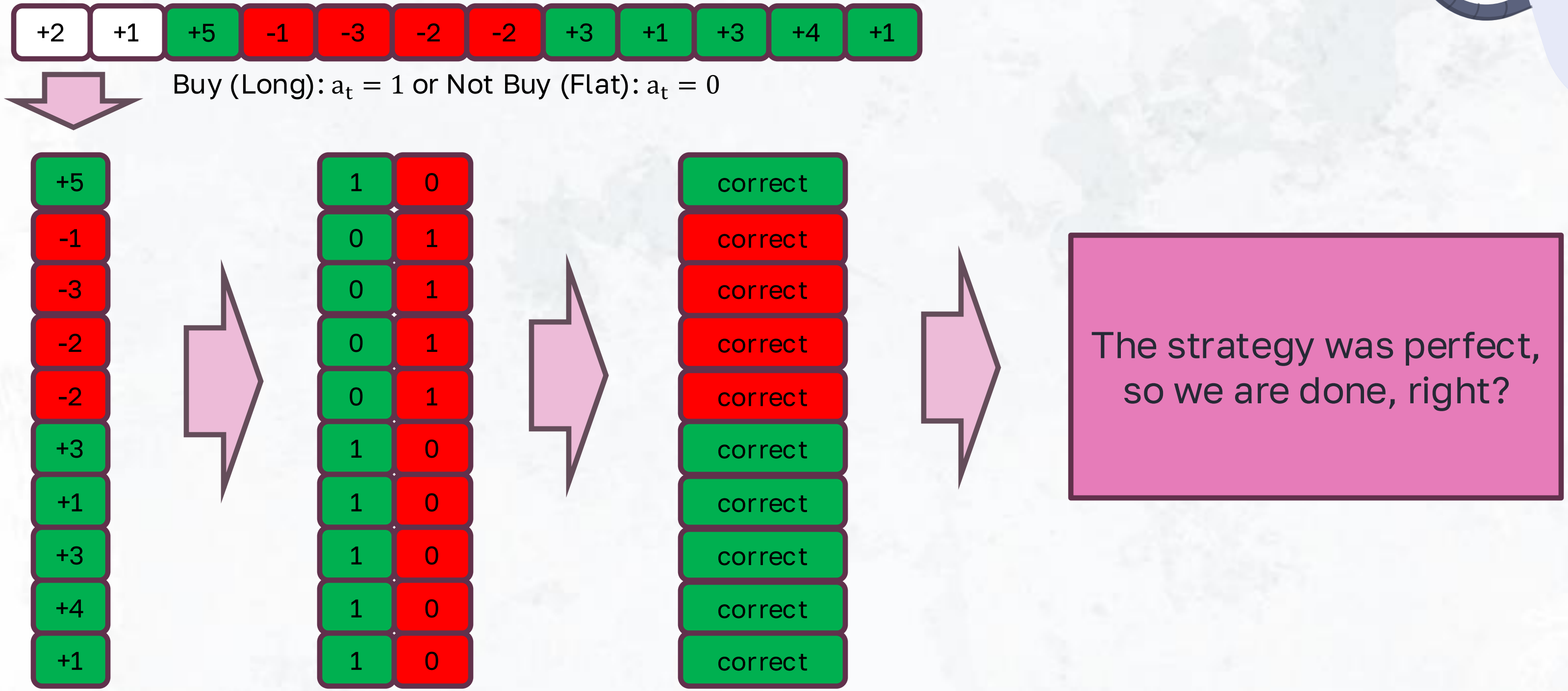
Idea: Now just model it with NN!



How good was this strategy?


Let us focus, can a neural network actually learn from this?

Strategy: Buy, if future returns are positive



+2	+1	+5	-1	-3	-2	-2	+3	+1	+3	+4	+1

Buy (Long): $a_t = 1$ or Not Buy (Flat): $a_t = 0$

+5	1	0	correct
-1	0	1	correct
-3	0	1	correct
-2	0	1	correct
-2	0	1	correct
+3	1	0	correct
+1	1	0	correct
+3	1	0	correct
+4	1	0	correct
+1	1	0	correct

The strategy was perfect, so we are done, right?

Current Issues

We only modeled one trader

- In order to model the whole universe of strategies, we would need to observe all possible trades. This would amount to 2^T strategies!

We tried to model the perfect scenario

- We used the future return to retrieve the best outcome, but this is not observable and thus not learnable in real life.

We looked at results in the next time period

- Buying a stock gives the next days' return, but also comes with consequences later on, e.g. the need to sell, which involves transaction costs.

03 →

Reinforcement Learning: Core concepts

(AI)

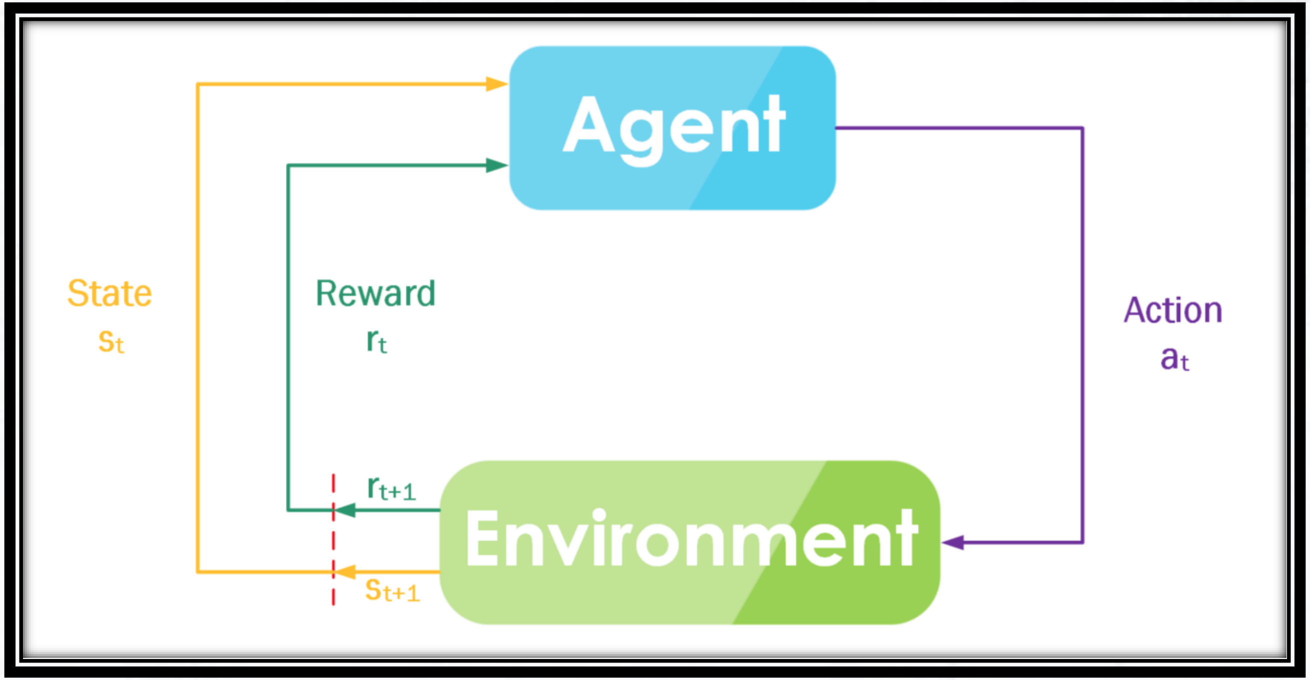
Key idea of Reinforcement Learning

The main idea of Reinforcement Learning (RL) is to teach the AI to **interact** with the surrounding **environment**. By doing so, it **receives rewards**, which teaches it to improve.

In particular, it balances between trying out (**exploration**) and earning from what it has learned (**exploitation**). Just likes humans, the AI needs to learn from the mistakes it makes as much as from the successes.

RL is applied in various fields, from robotics over trading to self-driving and gaming you will find a lot of implementations.

The core loop of RL

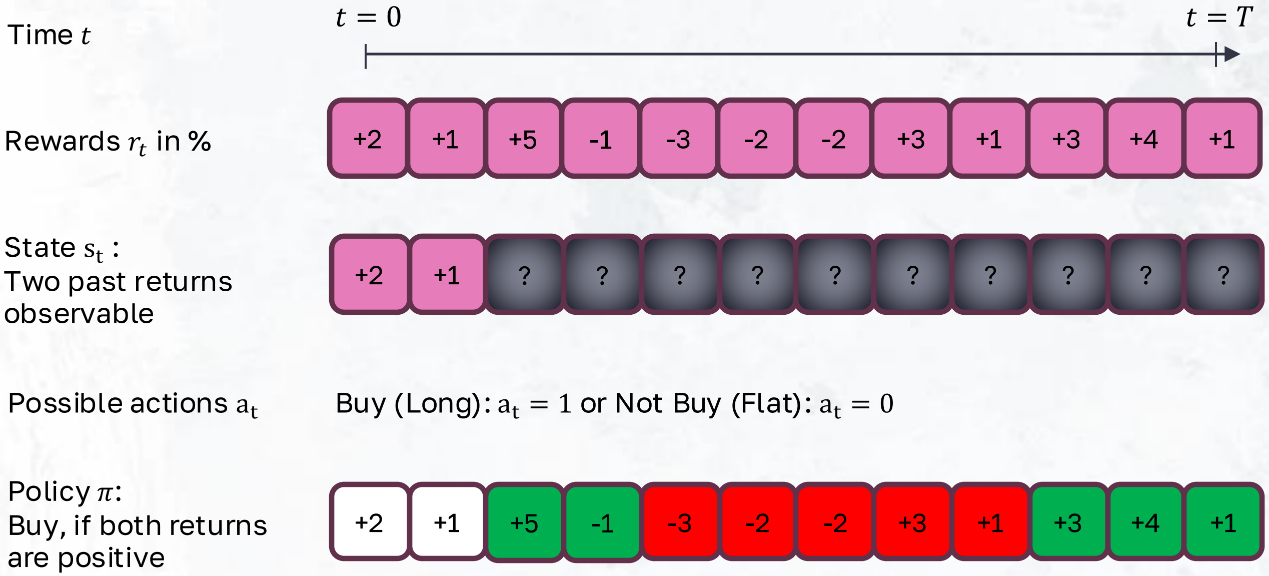


Source: AlMahamid, Grolinger, 2022

Remember this slide? Let‘s update!

Time t	$t = 0$											$t = T$
Returns r_t in %	+2	+1	+5	-1	-3	-2	-2	+3	+1	+3	+4	+1
Situation: Two past returns observable	+2	+1	?	?	?	?	?	?	?	?	?	?
Possible trades	Buy (Long): $a_t = 1$ or Not Buy (Flat): $a_t = 0$											
Strategy: Buy, if both returns are positive	⬜ +2	⬜ +1	🟩 +5	🟩 -1	🟥 -3	🟥 -2	🟥 -2	🟥 +3	🟥 +1	🟩 +3	🟩 +4	🟩 +1

Remember this slide? Let‘s update!



Time t	$t = 0$											$t = T$
Rewards r_t in %	+2	+1	+5	-1	-3	-2	-2	+3	+1	+3	+4	+1
State $\mathrm{s_t}$: Two past returns observable	+2	+1	?	?	?	?	?	?	?	?	?	?

Possible actions $\mathrm{a_t}$ Buy (Long): $\mathrm{a_t} = 1$ or Not Buy (Flat): $\mathrm{a_t} = 0$

Policy π: Buy, if both returns are positive	+2	+1	🟩 +5	🟩 -1	🟥 -3	🟥 -2	🟥 -2	🟥 +3	🟥 +1	🟩 +3	🟩 +4	🟩 +1

Overview of core elements in RL

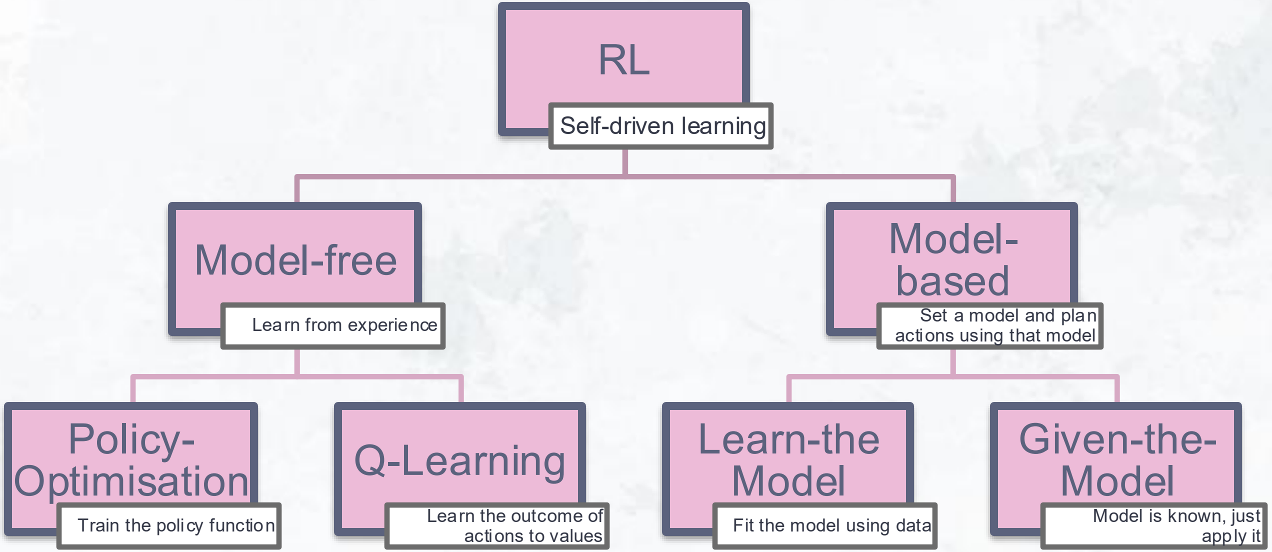
Term	Meaning general	Meaning in our example
Agent	Decision-making entity	Investor
Environment	Setting in which an agent operates	Daily stock market with prices and returns
State s	Representation of the current situation in the environment	Last $k = 2$ returns
Action a	A decision an agent can make	Long (1) or Flat (0)
Reward r	A feedback signal after taking an action	Return on investment
Policy π	Mapping from states to actions	Investment strategy, e.g. long, if both returns positive

04 →

Reinforcement Learning: Q-Learning

(AI)

Overview of RL frameworks



RL

Self-driven learning

Model-free

Learn from experience

Model-based

Set a model and plan actions using that model

Policy-Optimisation

Train the policy function

- REINFORCE
- PPO
- A3C
- DDPG

Q-Learning

Learn the outcome of actions to values

- DQN
- Double-DQN
- SARSA

Learn-the Model

Fit the model using data

- Dyna-Q
- MBPO

Given-the-Model

Model is known, just apply it

- Dynamic Programming
- Policy Iteration

Q-Learning: DQN

$$Q^*(s,a) = E\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \,|s_t = s, a_t = a\right]$$

- Goal: Find parameters θ, such that $Q_\theta(s,a) \approx Q^*(s,a)$; γ is a discount factor
- $Q^*(s,a)$ is the optimal action-value function, as it perfectly maps actions to states
- We achieve that, by optimizing (using simple MSE):

$$L(\theta) = E_{(s,a,r,s')\sim D}[(y - Q_\theta(s,a,))^2]$$

- Where we create the target y as:

$$\left. y = r + \gamma \max_{a'} Q_{\bar{\theta}}(s', a') \quad \right\} \text{Bellman Target}$$

- The Q-function is essentially our NN, but we use it to create targets and estimate!
- To achieve this, we sporadically update, so that $Q_\theta(s,a) = Q_{\bar{\theta}}(s,a)$

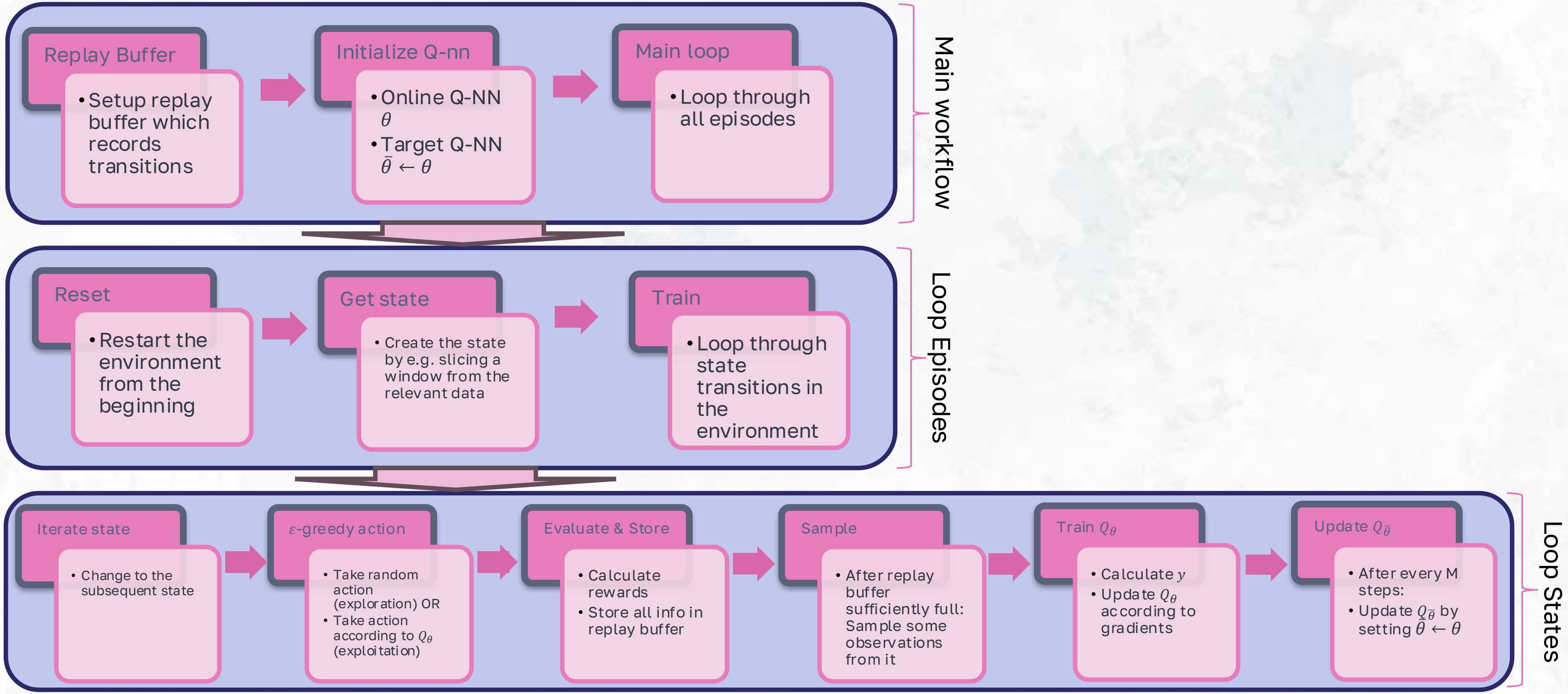
Q-Learning: The Magic

Is the dog catching it's own tail?
- OR -
How can the trainable network learn from targets generated by itself?

In simple terms:

- Training and target network start identical, but we freeze the target network temporarily
- Targets created contain the **reward r**, i.e. the networks are not purely self referential
- When taking actions, we either use the action suggested by the training network (**exploitation**) or a random one (**exploration**)
- Taking random actions helps us to learn what happens alternatively
- We **record** all of those actions and then **learn from the consequences** (gradient descent)
- Finally, we copy the weights of the training network into the target network and keep going

Simplified workflow of DQN



Main workflow

Replay Buffer

- Setup replay buffer which records transitions

Initialize Q-nn

- Online Q-NN θ
- Target Q-NN $\bar{\theta} \leftarrow \theta$

Main loop

- Loop through all episodes

Loop Episodes

Reset

- Restart the environment from the beginning

Get state

- Create the state by e.g. slicing a window from the relevant data

Train

- Loop through state transitions in the environment

Loop States

Iterate state

- Change to the subsequent state

ε-greedy action

- Take random action (exploration) OR
- Take action according to Q_θ (exploitation)

Evaluate & Store

- Calculate rewards
- Store all info in replay buffer

Sample

- After replay buffer sufficiently full: Sample some observations from it

Train Q_θ

- Calculate y
- Update Q_θ according to gradients

Update $Q_{\bar{\theta}}$

- After every M steps:
- Update $Q_{\bar{\theta}}$ by setting $\bar{\theta} \leftarrow \theta$

Thanks! →

Any questions?

Dr. Rick Steinert

steinert@europa-uni.de

CREDITS: This presentation template was created by **Slidesgo** and includes icons by **Flaticon**, infographics & images by **Freepik** and content by **Eliana Delacour**