COM 3240

REINFORCEMENT LEARNING

OPTIMISATION

GRADIENT DESCENT

Desirable: A system (an agent) that performs well a task

The system has parameters that we need to select appropriately (optimise)

$$f(x_1, x_2, \ldots, x_n)$$

parameters

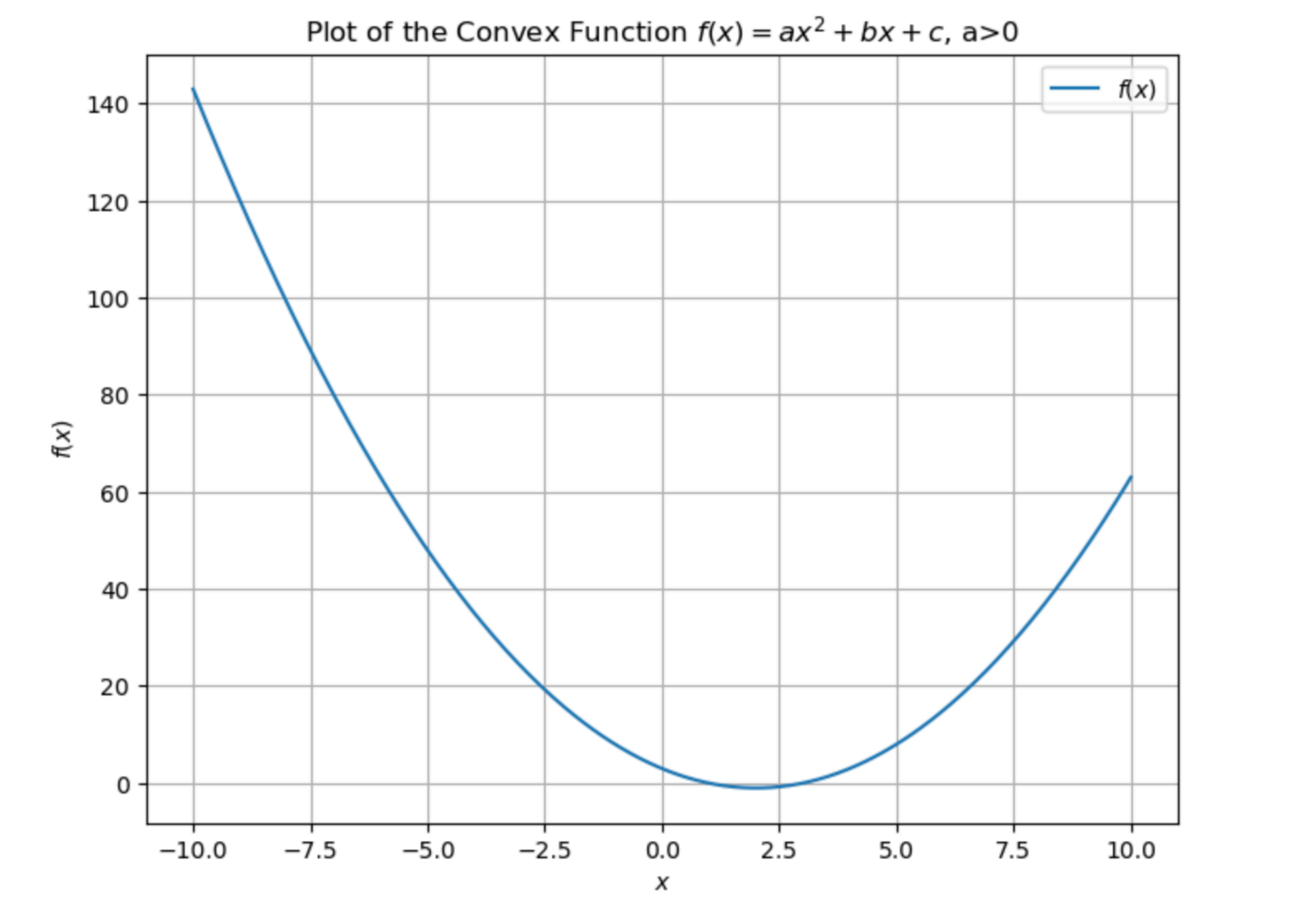
f

System's performance: maximise

System's error : minimise

OPTIMISATION

SINGLE (GLOBAL) MINIMUM

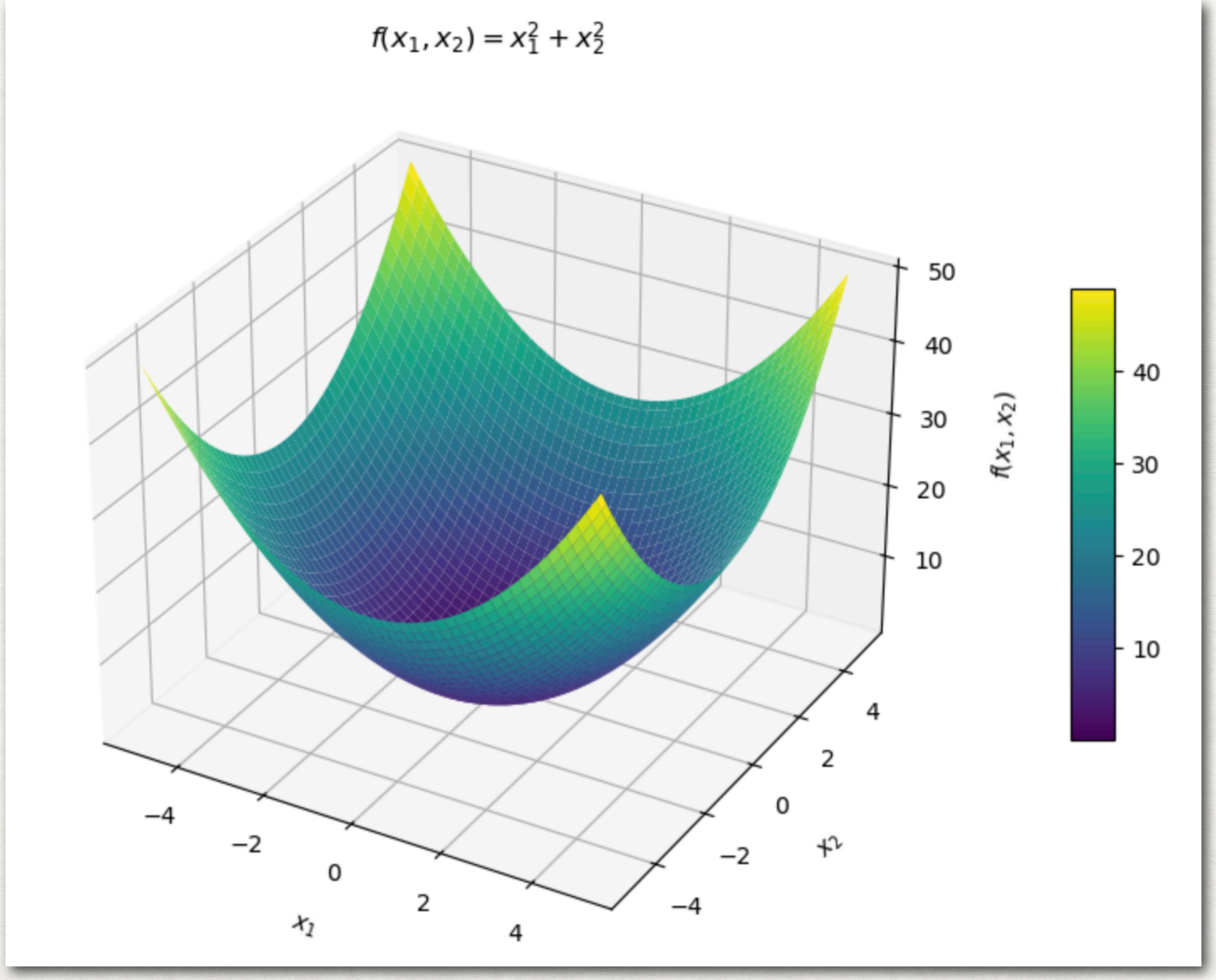


$$\frac{d}{dx}(ax^2 + bx + c) = 2ax + b$$

$$2ax + b = 0$$

OPTIMISATION

SINGLE (GLOBAL) MINIMUM

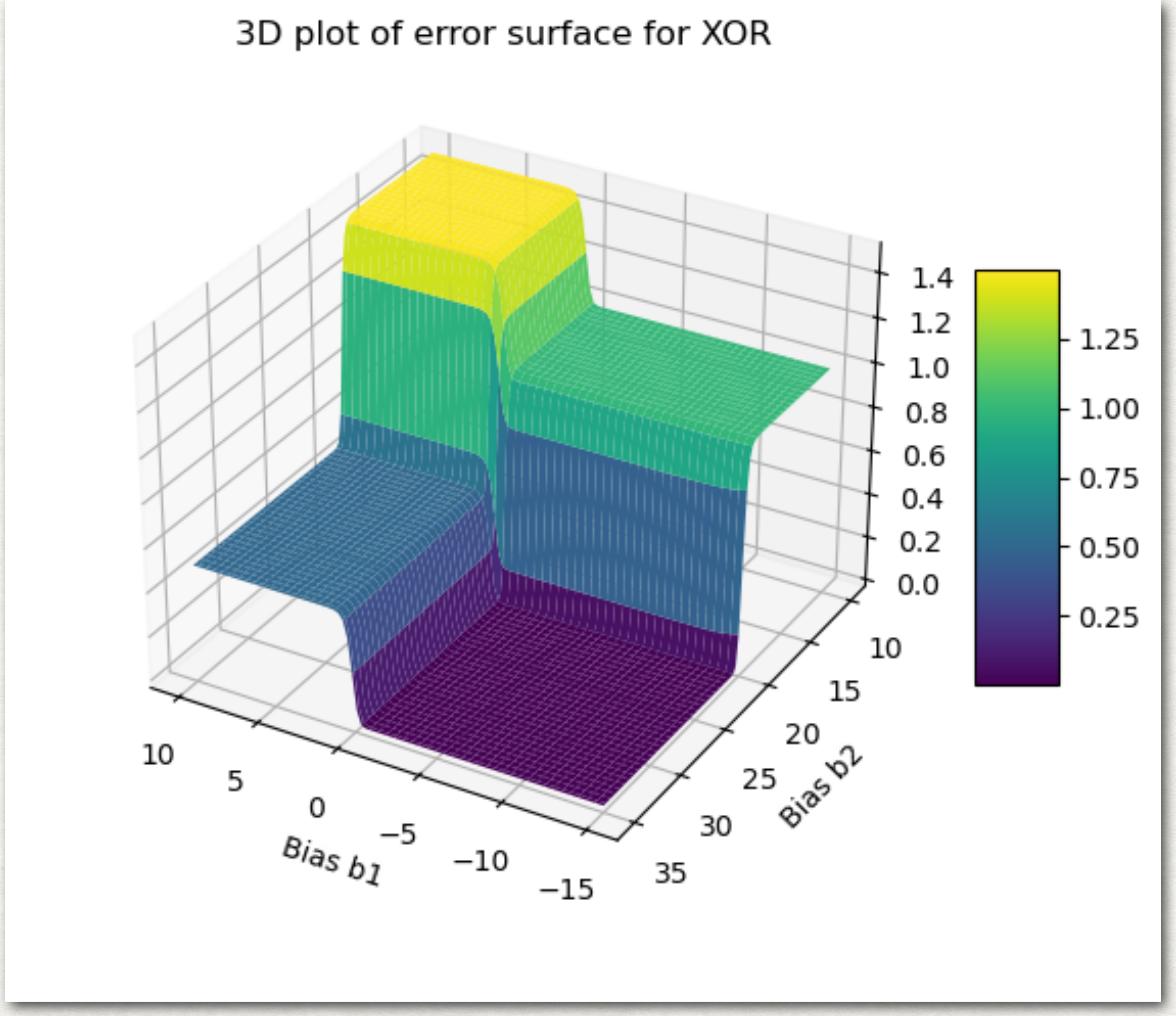


$$\nabla f(\mathbf{x}) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}\right] = 0$$

But this is not the case for all machine learning techniques, including multi- layer and deep neural networks

OPTIMISATION - GRADIENT DESCENT

MULTIPLE LOCAL MINIMA



Simple ANN example

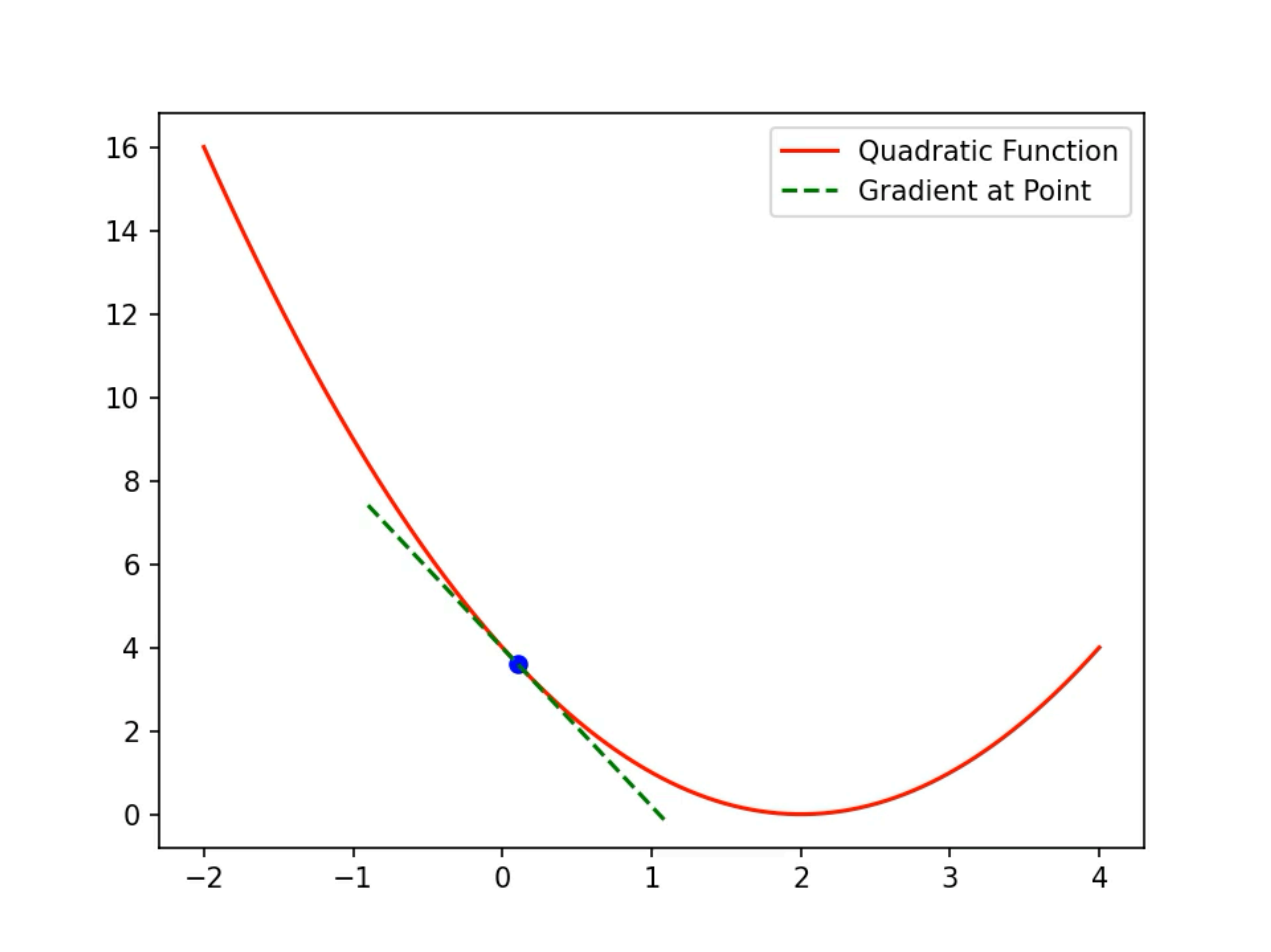
$$\mathbf{x}_{i+1} = \mathbf{x}_i - \eta \nabla f(\mathbf{x}_i)$$

$$\Delta \mathbf{x} = -\eta \nabla f(\mathbf{x})$$

$$\Delta \mathbf{x} = \mathbf{x}_{i+1} - \mathbf{x}_i$$

OPTIMISATION

GRADIENT DESCENT - FINDS A LOCAL MINIMUM



$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x}$$

$$\Delta \mathbf{x} = -\eta \nabla f(\mathbf{x})$$

$$\Delta \mathbf{x} = \mathbf{x}_{i+1} - \mathbf{x}_i$$

OPTIMISATION

IMPROVING GRADIENT DESCENT: KEY STRATEGIES

- **Different starting point (initialisation of parameters) may lead to a different solution.**
- **Exploit noise to get out of shallow local minima.**
- **Momentum may also help.**
- **Scaling small gradients.**
- **Adaptive learning rate.**

RL THROUGH THE LENS OF OPTIMISATION

RL THROUGH THE LENS OF OPTIMISATION

REWARD MAXIMISATION

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \ldots + R_{t+N}$$

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \ldots$$

$$0 \leq \gamma < 1$$

BREAKING DOWN RL TO ITS INGREDIENTS

LEARN VALUES, CHOOSE ACTIONS

s_t a_t G_t

I am in state s and chose action a.

I would like to learn the total future rewards G resulting from that action.

Knowing the expected future reward will allow me to choose a “good” action.

BREAKING DOWN RL TO ITS INGREDIENTS

LEARN VALUES, CHOOSE ACTIONS

How do we learn the value of the state-actions?

How do we pick an action (aka policy)?

RL THROUGH THE LENS OF OPTIMISATION

MAXIMISE R - MINIMISE PREDICTION ERROR

Maximise Reward.

Minimise prediction error for the expected future reward.


DALL·E

IMMEDIATE REWARDS

BANDITS

$$G = R(A)$$

No need to denote a state.

Two or more actions.

A “trial” constitutes pulling a lever.

Reward is a consequence of the action of trial n.

IMMEDIATE REWARDS

BANDITS

q*: true expected return

$$q^*(a) \doteq E[G \mid A_n = a] = E[R \mid A_n = a]$$

The expectation is calculated across trials.

$$E[R \mid A_n = a] = \sum_i r_i \, P\left(R = r_i \mid A_n = a\right)$$

Alternatively, use an empirical estimate:

$$E[R \mid A_n = a] \approx \frac{1}{N(a)} \sum_{n=1}^{N} R_n \mathbf{1}_{\{A_n = a\}}$$

IMMEDIATE REWARDS

BANDITS

$$q^*(a) \doteq E[R \mid A_n = a]$$

Q-value : estimate of the expected return

$$Q(a) \approx q^*(a)$$

Idea: construct a loss function of Q with minimum at:

$$Q(a) = E[R \mid A_n = a]$$

Hint: think quadratic

IMMEDIATE REWARDS

BATCH LOSS FUNCTION

Minimise: $E[(Q(a) - R)^2 \mid A_n = a]$

$$L(a) = \frac{1}{2N(a)} \sum_{n=1}^{N} (Q(a) - R_n)^2 \mathbf{1}_{A_n=a} \qquad N = \sum_a N(a)$$

$$\mathbf{1}_{A_n=a} = \begin{cases} 1 & \textbf{if } A_n = a \\ 0 & \textbf{otherwise.} \end{cases}$$

Indicator function

$$N(a) = \sum_{n=1}^{N} \mathbf{1}_{A_n=a}$$

R_n is the reward at trial n

IMMEDIATE REWARDS

UPDATE RULE VIA DIRECT MINIMISATION

$$\frac{dL(a)}{dQ(a)} = \frac{d}{dQ(a)}\left(\frac{1}{2N(a)}\sum_{n=1}^{N}(Q(a)-R_n)^2\mathbf{1}_{A_n=a}\right)$$

Because of the convexity of the loss function with respect to Q, we can set directly the derivative to 0 to find the minimum.

$$\frac{dL(a)}{dQ(a)} = \frac{1}{N(a)}\sum_{n=1}^{N}\left(Q(a)-R_n\right)\mathbf{1}_{A_n=a} = 0$$

This leads to: $Q(a) = \dfrac{1}{N(a)}\displaystyle\sum_{n=1}^{N} R_n\ \mathbf{1}_{A_n=a} \approx E[R \mid A_n = a] = q^*$ **Stationary**

IMMEDIATE REWARDS

MULTI-ARM LOSS FUNCTION

$$L = \sum_{a=1}^{|A|} L(a) = \sum_{a=1}^{|A|} \left[\frac{1}{2N(a)} \sum_{n=1}^{N} \left(Q(a) - R_n\right)^2 \mathbf{1}_{A_n=a} \right]$$

In this case, we would need to take partial derivatives

$$\frac{\partial L}{\partial Q(a)}$$

and end up with the same result.

IMMEDIATE REWARDS

GRADIENT BATCH UPDATE

$$\frac{dL(a)}{dQ(a)} = \frac{1}{N(a)} \sum_{n=1}^{N} \left(Q(a) - R_n \right) \mathbf{1}_{A_n = a}$$

We can also write a gradient rule:

$$\Delta Q(a) = -\eta N(a) \frac{dL(a)}{dQ(a)} = -\eta \sum_{n=1}^{N} \left(Q(a) - R_n \right) \mathbf{1}_{A_n = a}$$

η: learning rate

Batch rule, first collect observations (trial data), then update.

IMMEDIATE REWARDS

GRADIENT BATCH UPDATE MATCHES DIRECT OPTIMISATION

$$\Delta Q(a) = -\,\eta \sum_{n=1}^{N} \left(Q(a) - R_n\right) \mathbf{1}_{A_n=a}$$

At convergence ΔQ(a)=0, hence:

$$Q(a) = \frac{1}{N(a)} \sum_{n=1}^{N} R_n \; \mathbf{1}_{A_n=a}$$ **exactly as calculated earlier**

IMMEDIATE REWARDS

ONLINE GRADIENT RULE

$$\Delta Q(a) = -\eta \frac{dL(a)}{dQ(a)} = -\eta \cancel{\sum_{n=1}^{N}} \left(Q(a) - R\right) \mathbf{1}_{A_t=a}$$

We now update our estimate after each trial (sample) known as online learning. We then write:

$$\Delta Q(a) = -\eta \left(Q(a) - R_n\right) \mathbf{1}_{A_n=a}$$

i.e. for trial 1 to N, if action α is taken, we update its Q value by:

$$Q(a) = Q(a) - \eta \left(Q(a) - R_n\right)$$

IMMEDIATE REWARDS

WHEN TO USE THE ONLINE RULE

Lets assume a stationary environment, introduce an index k on the trials where action a is chosen and set η=1/(k+1).

$$Q_{k+1}(a) = Q_k(a) - \frac{1}{k+1}(Q_k(a) - R_{k+1}) .$$

We can use induction to show that it converges to the numerical average, just like the batch version.

Also, for fixed learning rate η, we can easily show that it takes the form:

$Q_{k+1}(a) = (1-\eta)Q_k(a) + \eta R_{k+1}$ Exponential average (forgetful), non-stationary

POLICIES

THE EXPLORATION EXPLOITATION DILEMMA

POLICIES

THE EXPLORATION EXPLOITATION DILEMMA

Greedy $a_n = \underset{a}{\operatorname{argmax}}\ Q(a)$

Optimistic Greedy: initialise Q-values unrealistically high

Epsilon-Greedy : explore with probability epsilon, greedy otherwise

Softmax: $P(a) = \dfrac{e^{Q(a)/\tau}}{\sum_b e^{Q(b)/\tau}}$

What happens if τ grows very large or tends to 0?

IMMEDIATE REWARDS

BANDITS

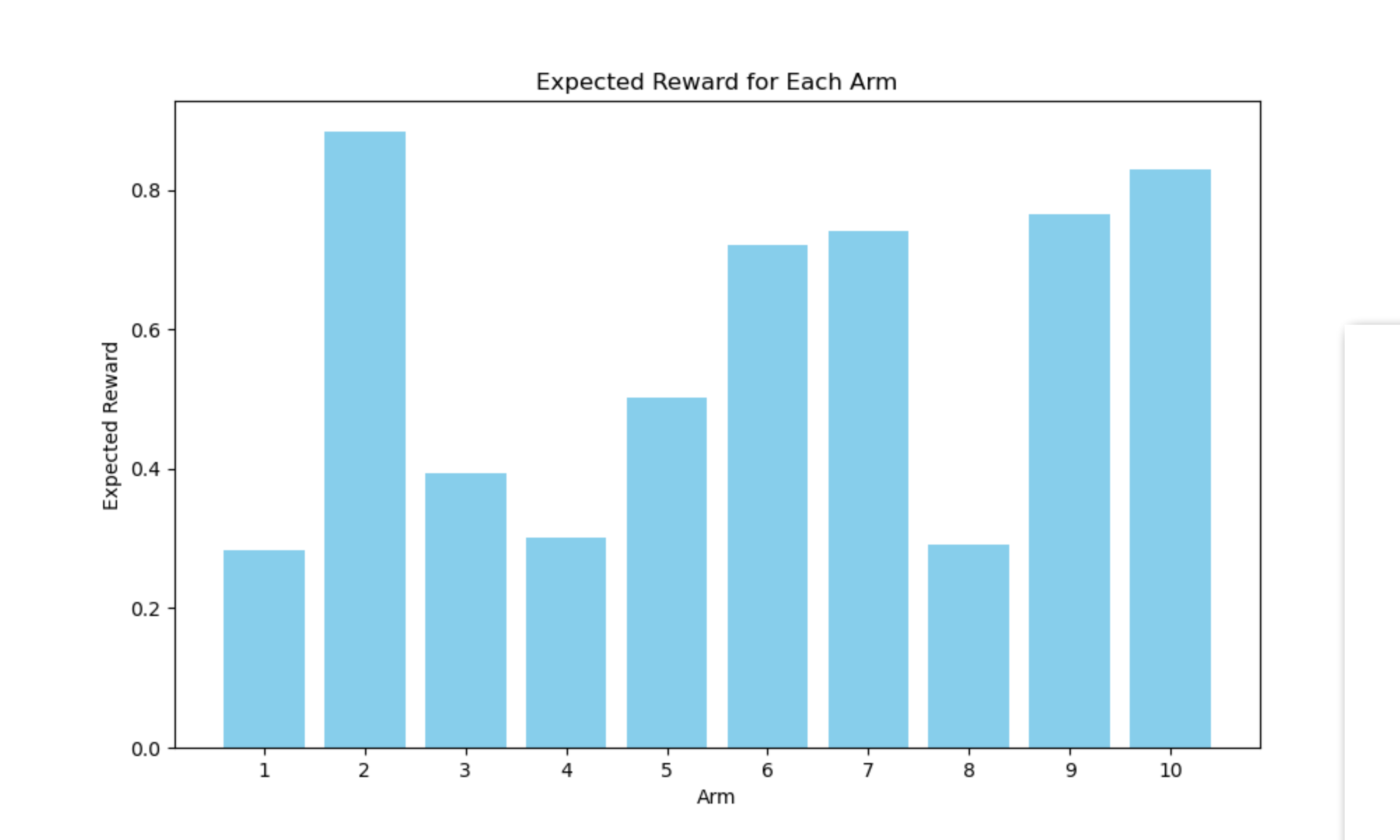
- Initialise Q-values, and count N(a) for all actions a
- For each trial n
  - Select an action a using policy
  - Increase the count N(a)
  - Take action a and observe reward R
  - Update Q (a)
    - For a stationary environment: $Q(a) = Q(a) - (Q(a) - R)/N(a)$
    - For non-stationary environment: $Q(a) = Q(a) - \eta(Q(a) - R)$

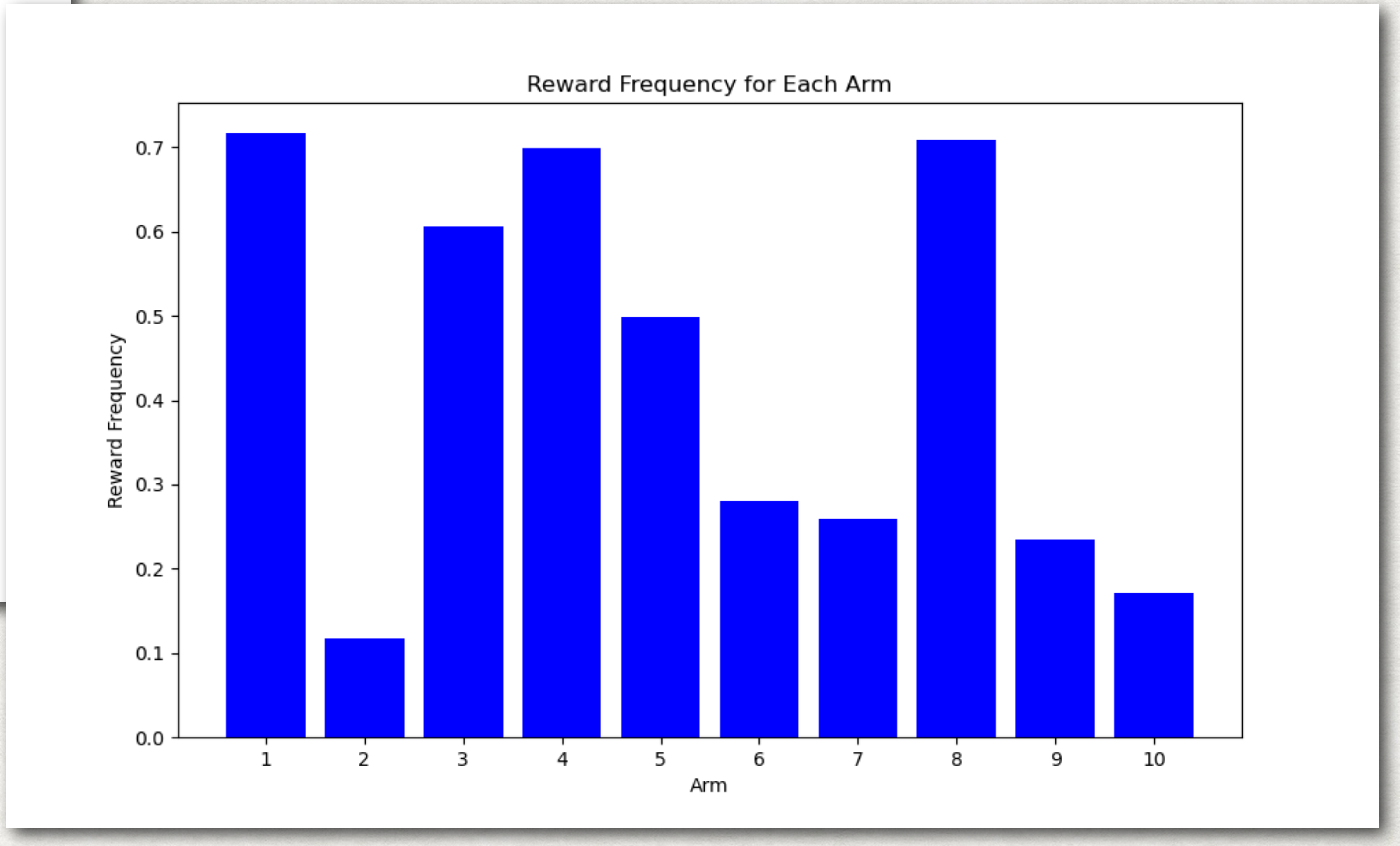
IMMEDIATE REWARDS

BANDITS

IMMEDIATE REWARDS

BANDITS





BANDITS

SUMMARY

- Optimisation
- Immediate Rewards (bandits)
- Batch update rule
- Online update rule
- Bandit algorithm for stationary and non-stationary environments

THANK YOU!