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

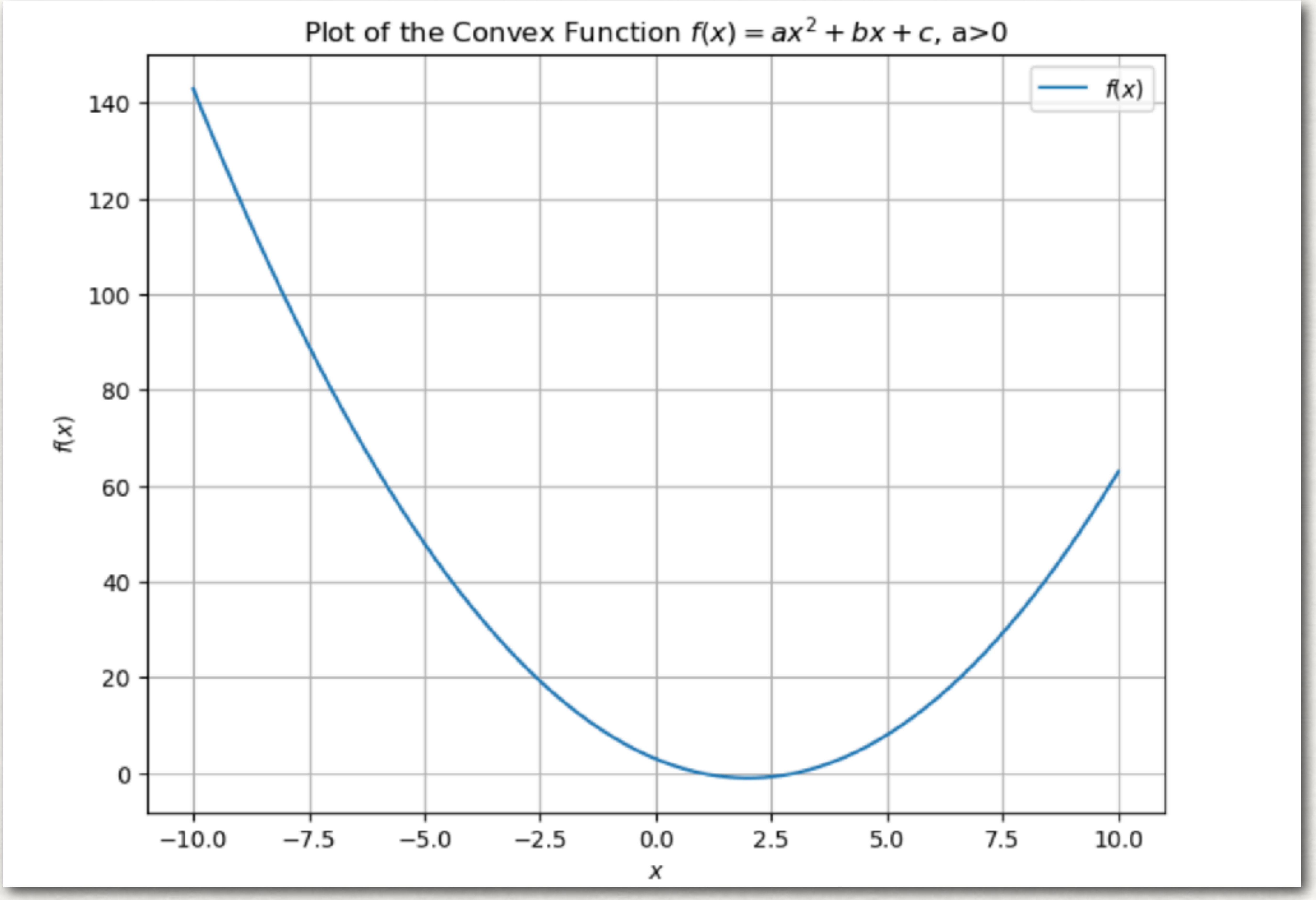
f **System's performance: maximise**

System's error : minimise


DALL·E

OPTIMISATION

SINGLE (GLOBAL) MINIMUM

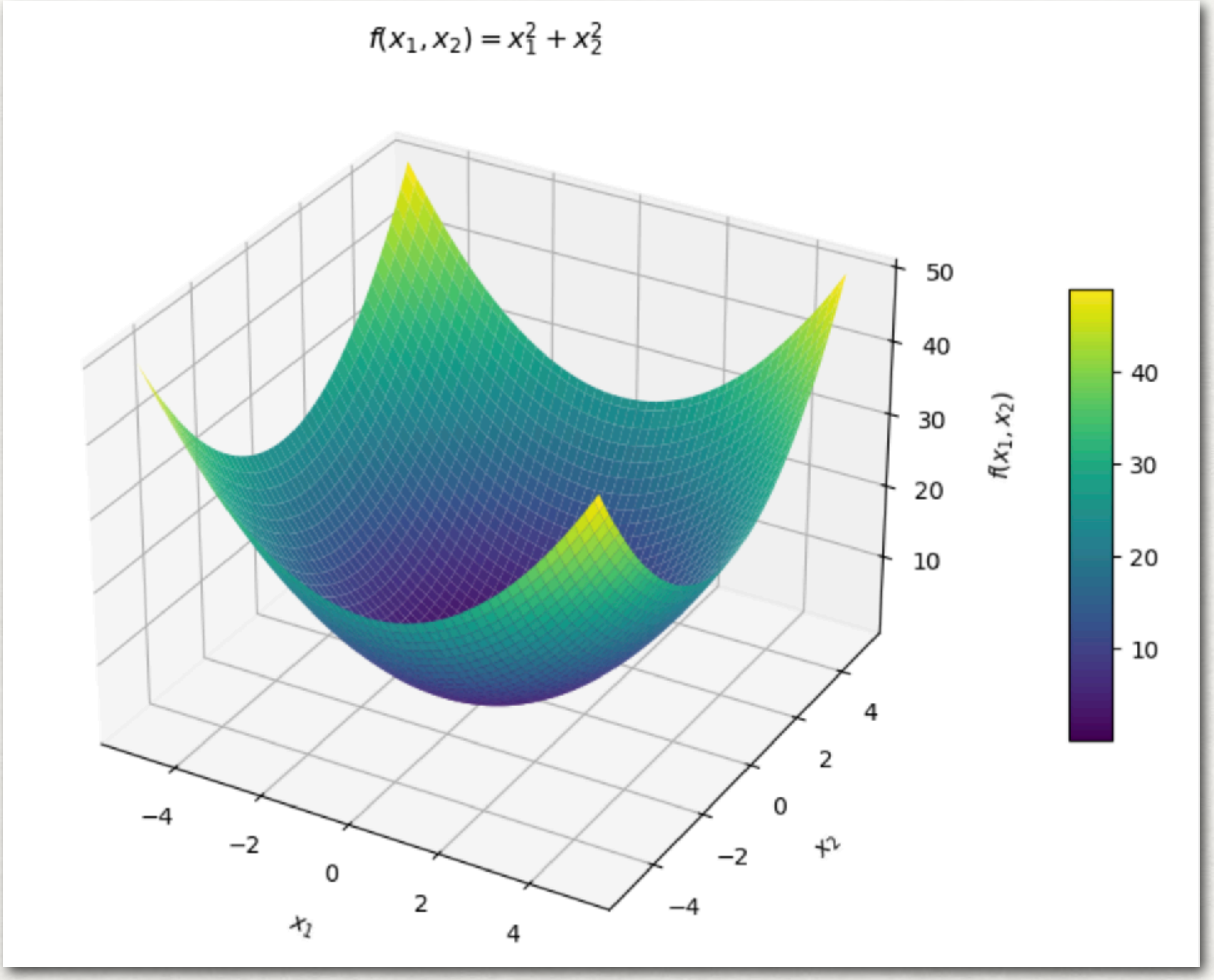


$$\frac{d}{dx}(ax^2 + bx + c) = 2ax + b$$

$$2ax + b = 0$$

OPTIMISATION

SINGLE (GLOBAL) MINIMUM

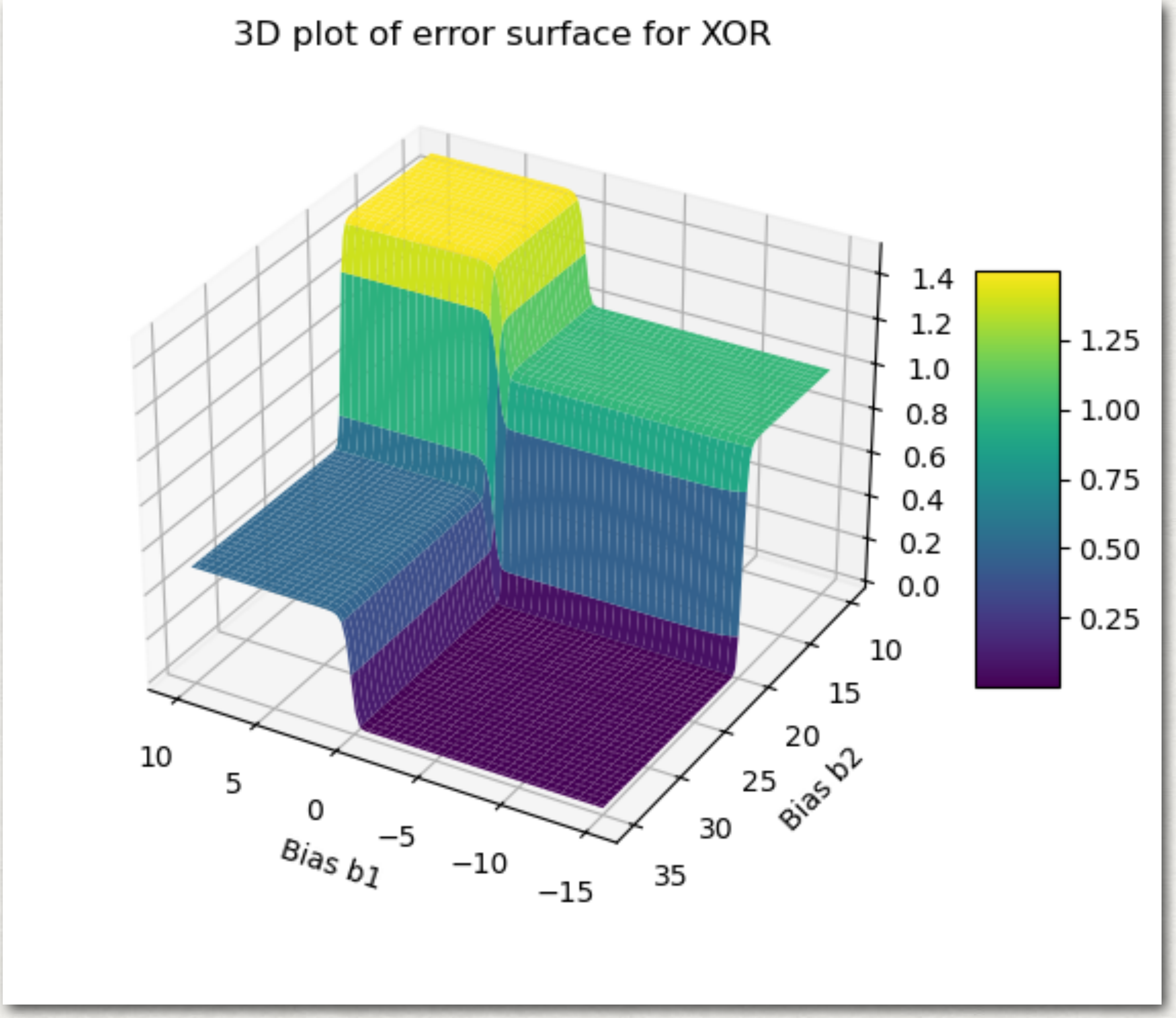


$$\nabla f(\mathbf{x}) = \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}\right] = 0$$

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

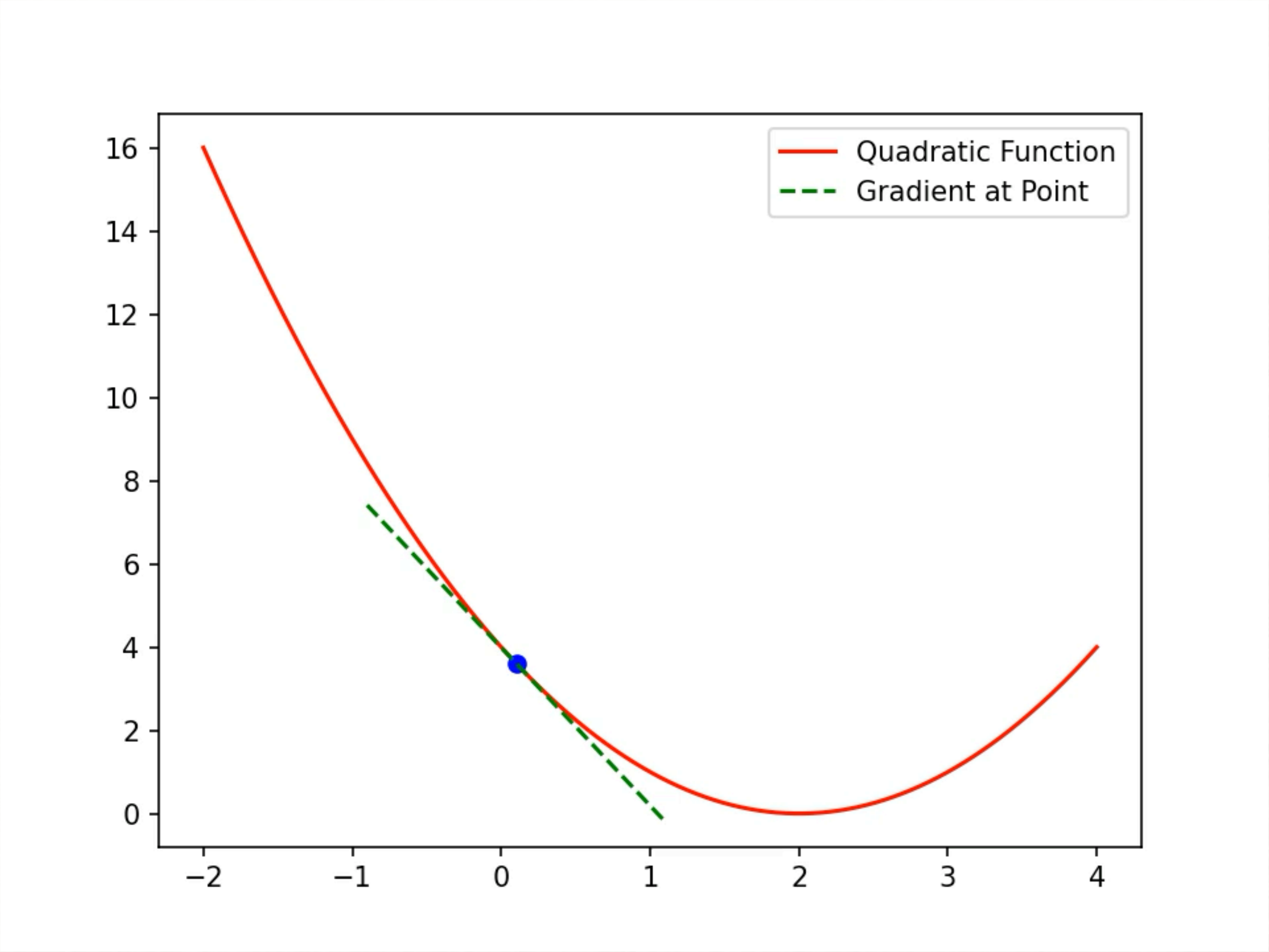
$$\mathbf{x}_{i+1} = \mathbf{x}_i - \eta \nabla f(\mathbf{x}_i)$$

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

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

OPTIMISATION

GRADIENT DESCENT - FINDS A LOCAL MINIMUM



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

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

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

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 learning 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.

IMMEDIATE REWARDS

BANDITS

$$G_t = R(A_t)$$

No need to denote a state.

Two or more actions.

Time t here is an index on the “trial”, which constitutes pulling a lever.

Reward is a consequence of the action of trial t.

IMMEDIATE REWARDS

BANDITS

q*: estimate of true expected reward

$$q^*(a) \doteq E[G_t | A_t = a]$$

$$\doteq E[R(A_t) | A_t = a] = E[R(A_t = a)]$$

The expectation is calculated across trials.

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

Rewards and reward distributions are unknown: $r_i, P(R = r_i)$

IMMEDIATE REWARDS

BANDITS

$$q^*(a) \doteq E[G_t \,|\, A_t = a] = E[R(A_t = a)]$$

Q-value : estimate of the expected return

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

In RL we obtain an estimate via sampling

Ideally: $\dfrac{E[\left(Q(a) - R(A_t = a)\right)^2]}{2} = 0$

IMMEDIATE REWARDS

BANDITS

Minimise: $\dfrac{E[\left(Q(a) - R(A_t = a)\right)^2]}{2}$

$$L(a) = \frac{1}{2T_a}\sum_{t=1}^{T}\left(Q(a) - R\right)^2 \mathbf{1}_{A_t=a}$$

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

Indicator function

$$T(a) = \sum_{t=1}^{T}\mathbf{1}_{A_t=a}$$

R=R(t) is the reward at trial t

IMMEDIATE REWARDS

BANDITS

$$\frac{dL(a)}{dQ} = \frac{1}{T(a)} \sum_{t=1}^{T} \left(Q(a) - R\right) \mathbf{1}_{A_t=a}$$

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

$$\frac{1}{T(a)} \sum_{t=1}^{T} \left(Q(a) - R\right) \mathbf{1}_{A_t=a} = 0$$

This leads to: $Q(a) = \frac{1}{T(a)} \sum_{t=1}^{T} R \; \mathbf{1}_{A_t=a} \approx E[R(A_t = a)] = q^*$ **Stationary**

IMMEDIATE REWARDS

BANDITS

$$L = \sum_a L(a) = \sum_a \left[\frac{1}{2T(a)} \sum_{t=1}^{T} \left(Q(a) - R\right)^2 \mathbf{1}_{A_t = a} \right] \qquad T = \sum_a T(a)$$

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

BATCH AND ONLINE UPDATES

$$\frac{dL(a)}{dQ} = \frac{1}{T(a)} \sum_{t=1}^{T} \left(Q(a) - R\right) \mathbf{1}_{A_t=a}$$

We can also write a gradient rule:

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

η: learning rate

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

IMMEDIATE REWARDS

BATCH AND ONLINE UPDATES

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

At convergence ΔQ(a)=0, hence:

$$Q(a) = \frac{1}{T(a)} \sum_{t=1}^{T} R \; \mathbf{1}_{A_t=a}$$ **exactly as calculated earlier**

IMMEDIATE REWARDS

BATCH AND ONLINE UPDATES

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

Note a: action,
α: learning rate!

We can suppress the sum. This suggests we now update our estimate after each trial (sample) known as online learning. We then write:

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

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

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

IMMEDIATE REWARDS

THE ONLINE RULE MAKES SENSE

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

$$Q_{i+1}(a) = Q_i(a) + \frac{1}{i+1}\left(R_{i+1} - Q_i(a)\right).$$

We can use induction to show that it converges to the numerical average.

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

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

POLICIES

THE EXPLORATION EXPLOITATION DILEMMA

POLICIES

THE EXPLORATION EXPLOITATION DILEMMA

Greedy $a_t = \underset{a}{\text{argmax}} \; Q_t(a)$

Optimistic Greedy: initialise Q-values unrealistically high

Epsilon-Greedy : explore with probability epsilon, greedy otherwise

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

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

IMMEDIATE REWARDS

BANDITS

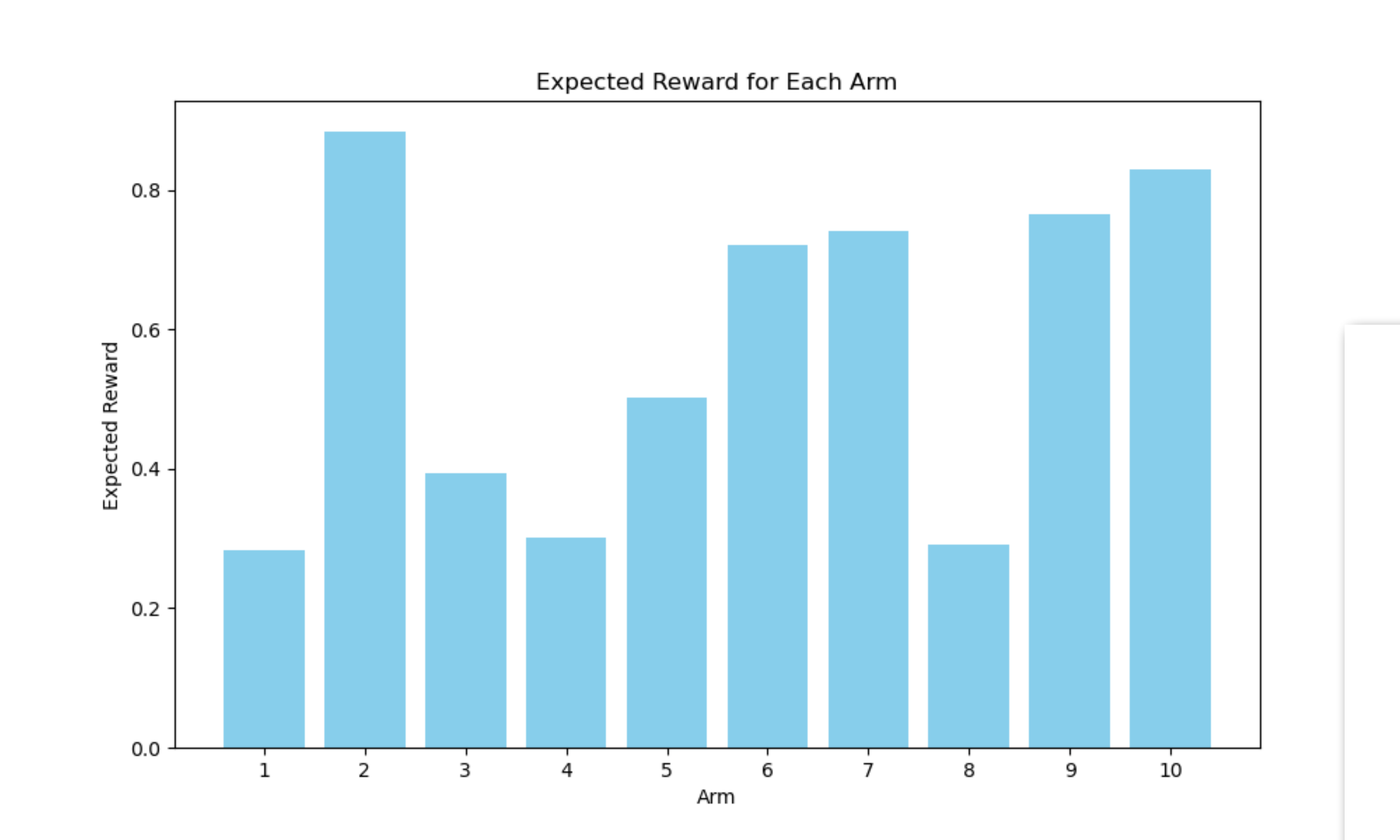
- Initialise Q-values, and count T(a) for all actions a
- For each time step
  - Select an action a using policy
  - Increase the count T(a)
  - Take action a and observe reward R
  - Update Q (a)
    - For a stationary environment: $Q(a) = Q(a) + (R - Q(a))/T(a)$
    - For non-stationary environment: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

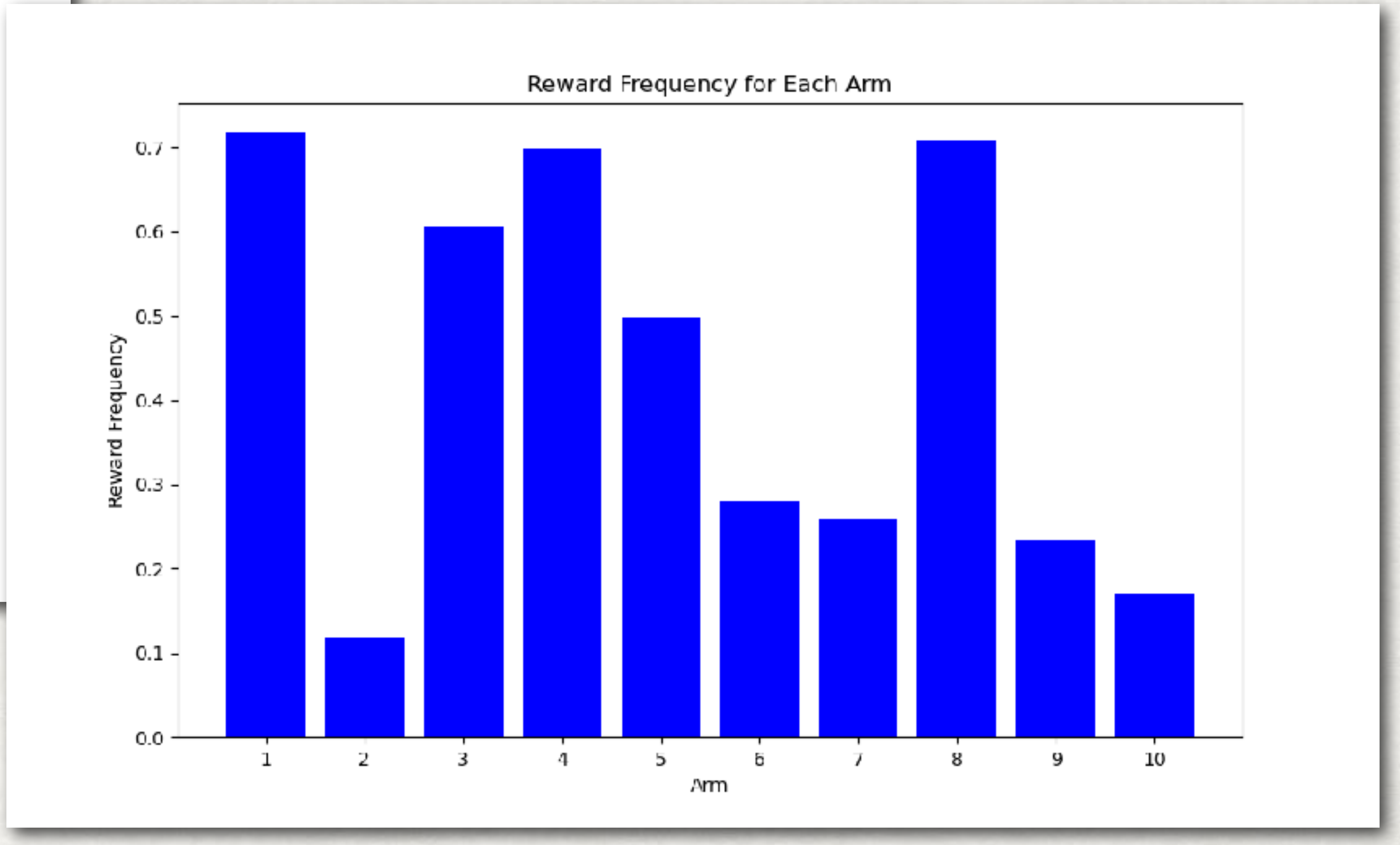
IMMEDIATE REWARDS

BANDITS

IMMEDIATE REWARDS

BANDITS





BANDITS

SUMMARY

- Optimisation
- Immediate Rewards
- Batch Learning
- Online Learning
- Bandit algorithm for stationary and non-stationary environments

THANK YOU!