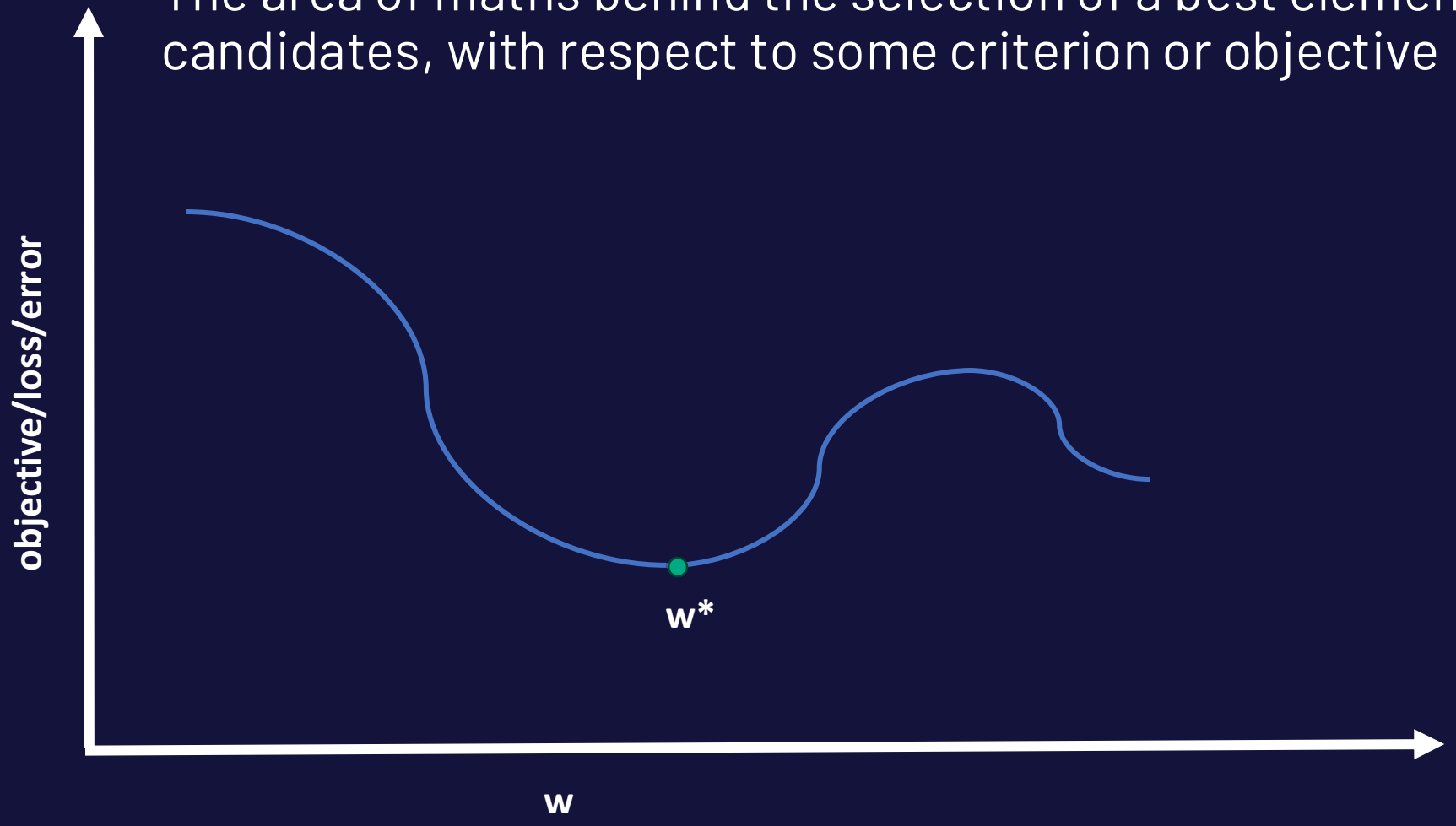

# An Introduction to Optimisation

Alasdair Paren - alasdair.paren@eng.ox.ac.uk

# What is Optimisation?

The area of maths behind the selection of a best element from some set of candidates, with respect to some criterion or objective

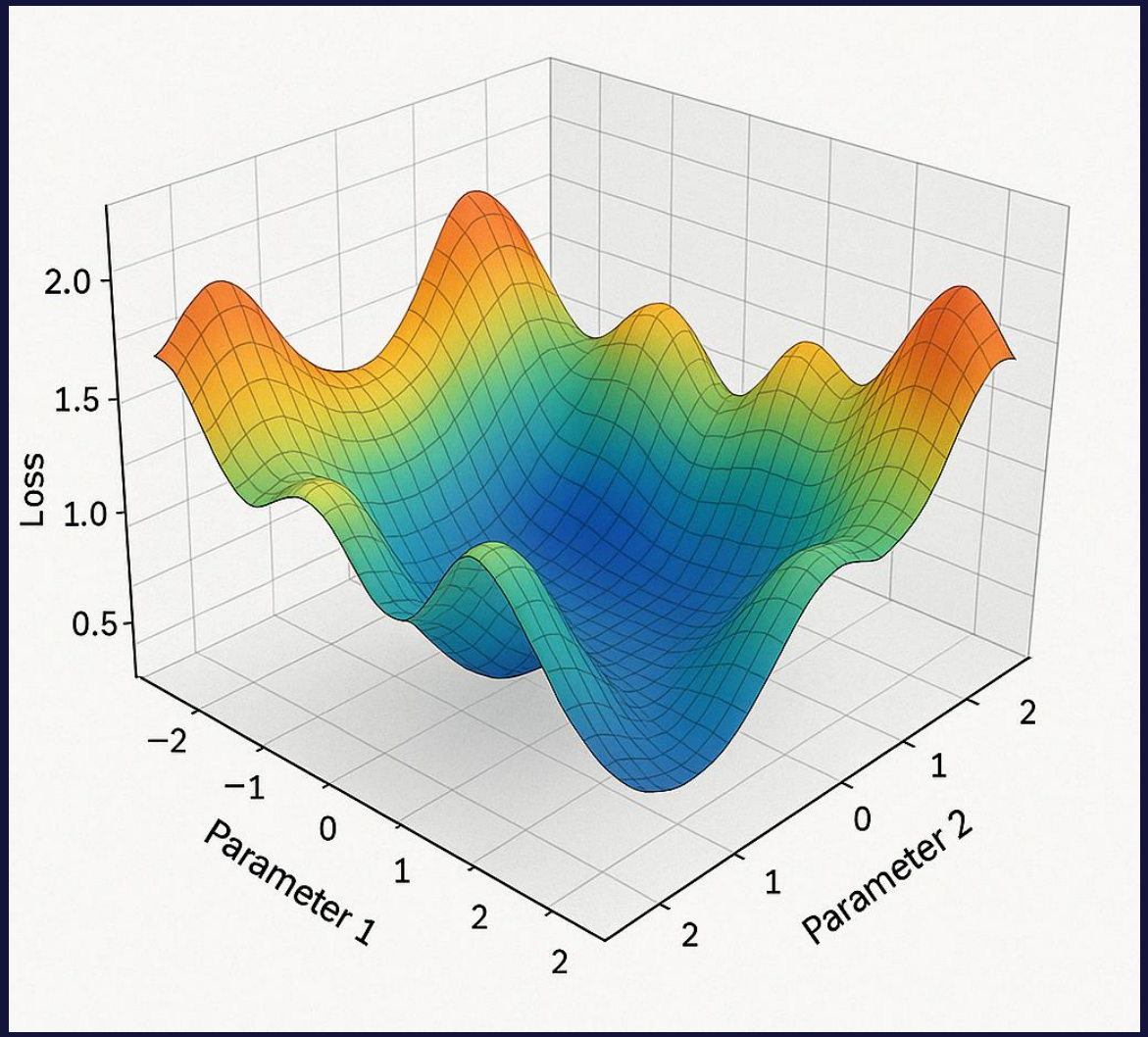


# Why is Optimisation Important to Machine Learning?


- Most State-Of-The-Art (SOTA) AI systems employ large neural networks trained on large data sets

- Training neural networks requires optimising a very high dimensional, non-convex optimisation problem, computed over a very large training set

# Visualization of a Neural Network Loss Function



# Loss Functions


We want to optimise (typically minimise) scalar valued "loss functions" that quantify the error of the model

Desirable properties:

- Bounded below (typically by zero)
- Continuous
- Smooth
- Convex
- Cheap to calculate
- Finite-sum Structure

# Types of Optimisation


- Convex / Non-Convex
- Constrained / Non-Constrained
- Discrete / Continuous / Mixed Integer Programming
- Stochastic / Deterministic
- Gradient Free / First Order / Second Order
- Reinforcement Learning

# Quiz time (1/4)

DEPARTMENT OF ENGINEERING SCIENCE

UNIVERSITY OF OXFORD

If $f(w) = w^2 - 4w + 16$, what is $\text{argmin}_{w \in \mathbb{R}} f(w)$?

1. w = -4
2. w = -2
3. w = 2
4. w = 4

# Quiz time (2/4)

If $f(w) = w^2 - 4w + 16$, whats is min $f(w)$?

1. f(w) = -8
2. f(w) = 0
3. f(w) = 8
4. f(w) = 12

# Quiz time (3/4)

If $f(w) = \max\{-w, w, \frac{1}{2}(w+3)\}$ (point-wise maximum), what is $\operatorname{argmin}_{w \in \mathbb{R}} f(w)$?

1. -2
2. -1
3. 1
4. 2

# Quiz time (4/4)

If $\mathbf{x}^\top A\mathbf{x} \geq 0$ for all none zero $\mathbf{x}$, and a square matrix $A \in \mathbb{R}^{d \times d}$. We call $A$:

1. Full Rank
2. A Negative Definite Matrix
3. A Negative Semi-Definite Matrix
4. A Positive Semi-Definite Matrix

# Positive Definiteness

## 2.1 Positive Definiteness

A square matrix $A \in \mathbb{R}^{d \times d}$ is positive definite if for all none zero $\mathbf{x}$, $\mathbf{x}^\top A\mathbf{x}$ is positive. Formally:

$$\forall\, \mathbf{x} \in \mathbb{R}^d \backslash 0^d, \quad \mathbf{x}^\top A\mathbf{x} > 0.$$

If the above inequality hold in equality $A$ is known as positive semi-definite

# Multivariate Functions


For a scalar output multivariate function with *d* inputs what is the Hessian?

# Multivariate Functions

## 2.1 Scalar Output Multivariate Functions

Let us consider a multivariate function $f(\mathbf{w})$:

$$f : \mathbb{R}^d \to \mathbb{R}.$$

$$\nabla f \triangleq \begin{bmatrix} \frac{\partial f}{\partial w_1} \\ \frac{\partial f}{\partial w_2} \\ \vdots \\ \frac{\partial f}{\partial w_d} \end{bmatrix}$$

$$\boldsymbol{H}_f \triangleq \begin{bmatrix} \frac{\partial^2 f}{\partial w_1}^2 & \frac{\partial^2 f}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 f}{\partial w_1 \partial w_d} \\ \frac{\partial^2 f}{\partial w_2 w_1} & \frac{\partial^2 f}{\partial w_2 \partial w_2} & \cdots & \frac{\partial^2 f}{\partial w_2 \partial w_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial w_d w_1} & \frac{\partial^2 f}{\partial w_d \partial w_2} & \cdots & \frac{\partial^2 f}{\partial w_d^2} \end{bmatrix}$$

# Taking Derivatives w.r.t matrices or vectors:

Suppose we have a scalar function $f = \mathbf{x}^\top \mathbf{y}$, where $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$ then:

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \mathbf{y}.$$

Now suppose we have a scalar function $f = \mathbf{x}^\top \mathbf{x}$, where $\mathbf{x} \in \mathbb{R}^d$ then:

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = 2\mathbf{x}$$

If in doubt calculate the gradient for one element and then construct the vector of partial derivatives:

$$f(\mathbf{x}) = \sum_i x_i^2$$

$$\frac{\partial f(\mathbf{x})}{\partial x_i} = 2x_i$$

$$\nabla_{\mathbf{x}} f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_d} \end{bmatrix} = \begin{bmatrix} 2x_1 \\ 2x_2 \\ \vdots \\ 2x_d \end{bmatrix} = 2\mathbf{x}$$

# Taking Derivatives w.r.t matrices or vectors:

If $\mathbf{x} \in \mathbb{R}^d, H \in \mathbb{R}^{d \times d}$ and symmetrical, What do you think the derivative of the following is?

$$f(\mathbf{x}) = \mathbf{x}^\top H \mathbf{x}, \qquad \nabla_x f(\mathbf{x}) = ?$$

https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf

# Taking Derivatives w.r.t matrices or vectors:

$$f(\mathbf{x}) = \mathbf{x}^\top H\mathbf{x} = \begin{bmatrix} x_1 & \cdots & x_d \end{bmatrix}^T \begin{bmatrix} h_{1,1} & \dots & h_{1,d} \\ \vdots & \ddots & \vdots \\ h_{d,1} & \dots & h_{d,d} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} = \begin{bmatrix} \dots & \sum_i x_i h_{i,j} & \dots \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} = \sum_i \sum_j x_j x_i h_{i,j}$$

$$\begin{aligned} \sum_i \sum_j x_j x_i h_{i,j} =& x_1 x_1 h_{1,1} + x_1 x_2 h_{1,2} + x_1 x_3 h_{1,3} + \cdots + x_1 x_d h_{1,d}, \\ & + x_2 x_1 h_{2,1} + x_2 x_2 h_{2,2} + x_2 x_3 h_{2,3} \cdots + x_2 x_d h_{2,d}, \\ & + x_3 x_1 h_{3,1} + x_3 x_2 h_{3,2} + x_3 x_3 h_{3,3} \cdots + x_3 x_d h_{3,d}, \\ & + \dots, \\ & + x_d x_1 h_{d,1} + x_d x_2 h_{d,2} + x_d x_3 h_{d,3} + \cdots + x_d x_d h_{d,d}. \end{aligned}$$

https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf

# Taking Derivatives w.r.t matrices or vectors:

$$\frac{\partial}{\partial x_i}\left(\sum_i\sum_j x_j x_i h_{i,j}\right) = 2x_i h_{i,i} + x_2 h_{i,2} + x_3 h_{i,3} + \cdots + x_d h_{i,d},$$
$$+ x_2 h_{2,i} + 0 + 0 \cdots + 0,$$
$$+ x_3 h_{3,i} + 0 + 0 \cdots + 0,$$
$$\cdots$$
$$+ x_d h_{d,i} + 0 + 0 + \cdots + 0.$$

As $H$ is symmetrical:

$$\frac{\partial}{\partial x_i}\left(\sum_i\sum_j x_j x_i h_{i,j}\right) = 2x_i h_{i,i} + 2x_2 h_{i,2} + 2x_3 h_{i,3} + \cdots + 2x_d h_{i,d}$$
$$= 2H_{i,:}\mathbf{x}$$

Thus considering the derivative $w.r.t$ $\mathbf{x}$ rather than $x_i$:

$$\frac{\partial}{\partial \mathbf{x}}\left(\sum_i\sum_j x_j x_i h_{i,j}\right) = \begin{bmatrix} 2H_{1,:}\mathbf{x} \\ 2H_{2,:}\mathbf{x} \\ \vdots \\ 2H_{d,:}\mathbf{x} \end{bmatrix} = 2H\mathbf{x}$$

https://www.math.uwaterloo.ca/~hwolkowi/matrixcookbook.pdf

# Taylor Expansions

DEPARTMENT OF ENGINEERING SCIENCE

UNIVERSITY OF OXFORD

## 2.3 Taylor Expansions

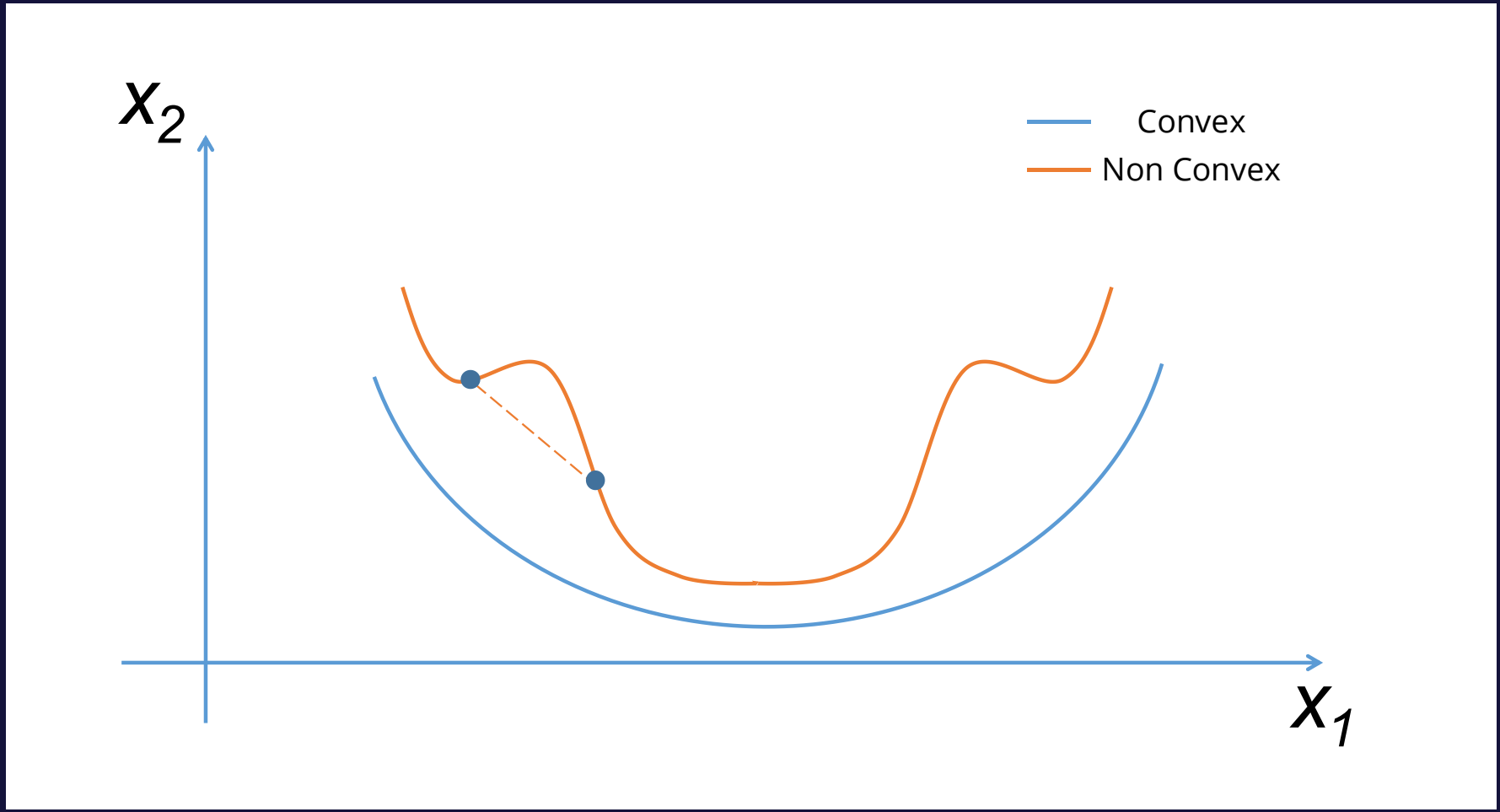
A function $f$ at a a point $\mathbf{w}_t$ can be approximated by its **first** order Taylor expansion around this point:

$$f(\mathbf{w}) \approx f(\mathbf{w}_t) + \nabla f(\mathbf{w}_t)^\top (\mathbf{w} - \mathbf{w}_t)$$

A function $f$ at a a point $\mathbf{w}_t$ can be approximated by its **second** order Taylor expansion around this point:

$$f(\mathbf{w}) \approx f(\mathbf{w}_t) + \nabla f(\mathbf{w}_t)^\top (\mathbf{w} - \mathbf{w}_t) + \frac{1}{2}(\mathbf{w} - \mathbf{w}_t)^T \boldsymbol{H}_f (\mathbf{w} - \mathbf{w}_t)$$

# Convex Functions



# Convex Functions

DEPARTMENT OF ENGINEERING SCIENCE
UNIVERSITY OF OXFORD

## 2.4 Convex Function

A function $f$ is convex if for any $\mathbf{x}, \mathbf{y}$:

$$f(\mathbf{x}) \geq f(\mathbf{y}) + \nabla f(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}),$$

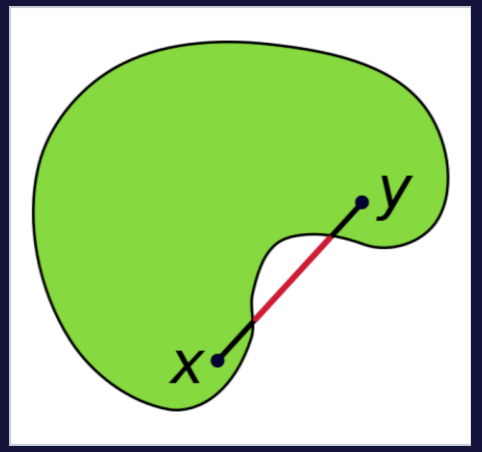
A function $f$ is **strictly convex** if for any $\mathbf{x}, \mathbf{y}$:

$$f(\mathbf{x}) > f(\mathbf{y}) + \nabla f(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}),$$

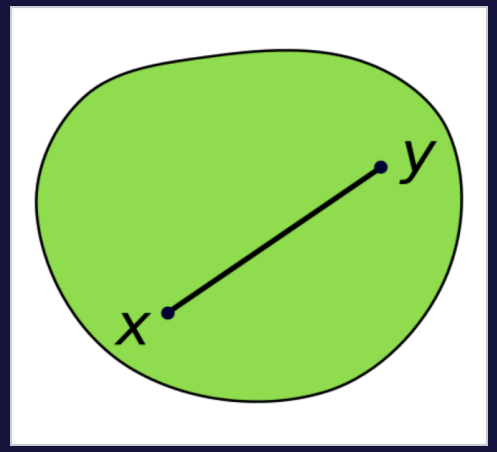
A twice differentiable convex function will have a positive semi-definite Hessian. A twice differentiable strictly convex function will have a positive definite Hessian.

# Convex Sets

A




B




Which sets are convex?

1. A
2. B
3. Both
4. Neither

# Convex Sets

DEPARTMENT OF ENGINEERING SCIENCE
UNIVERSITY OF OXFORD

### 3.6 Convex Set

A set $\Omega$ is convex if for any $\mathbf{x}, \mathbf{y} \in \Omega$ and any $\lambda \in [0, 1]$:

$$\lambda \mathbf{x} + (1 - \lambda)\mathbf{y} \in \Omega$$

Alternatively:
A set $\Omega$ is convex if there exists a convex function $f$ that $\forall \mathbf{x} \in \Omega \ f(\mathbf{x}) \leq k$ and for all $\mathbf{x} \notin \Omega \ f(\mathbf{x}) > k$ where $k$ is some constant.

# Convex Sets

**Euclidean Projection** let us define the euclidean Projection operation onto a set $\Omega$ as:

$$\Pi_{\Omega}(\mathbf{w}_t) = \min_{\mathbf{w} \in \Omega} \|\mathbf{w} - \mathbf{w}_t\|^2.$$

Euclidean Projection onto a convex set decreases the distance to every point in the set, thus it always reduces the distance to an constrain optimum. Formally if we define:

$$\mathbf{w}_\star = \operatorname*{argmin}_{\mathbf{w} \in \omega} f(\mathbf{w})$$

Then we have for all $\mathbf{w}$ in $\mathbb{R}^d$:

$$||\mathbf{w}_\star - \mathbf{w}||_2^2 \geq ||\mathbf{w}_\star - \Pi_{\Omega}(\mathbf{w})||_2^2$$

# Lipschitz Continuity

A function $f$ is $C$-Lipschitz over a set $\Omega$ with respect to a norm $||\cdot||$ if for any $\mathbf{x}, \mathbf{y} \in \Omega$:

$$||f(\mathbf{x}) - f(\mathbf{y})|| \leq C||\mathbf{y} - \mathbf{x}||.$$

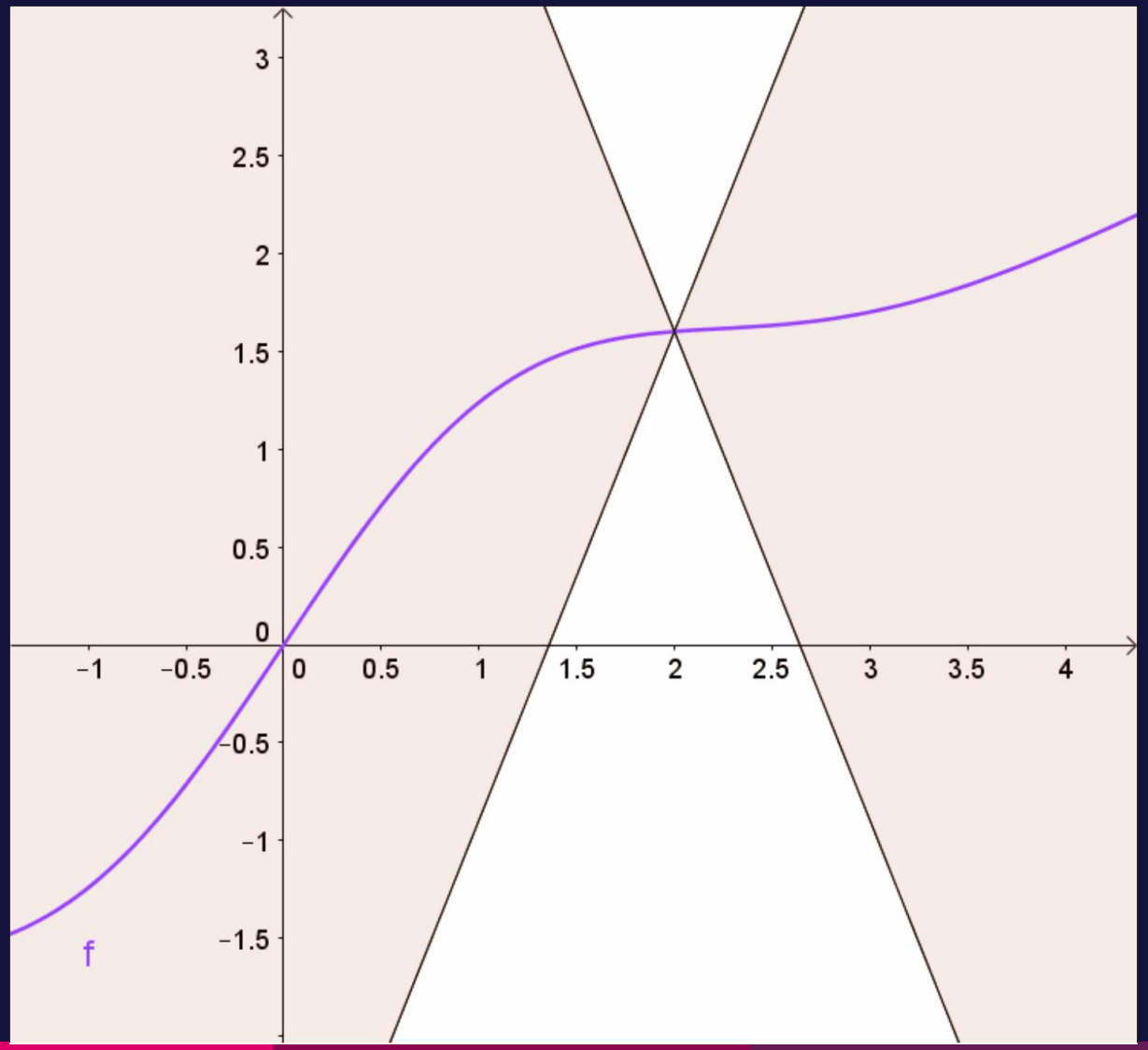
Most commonly started with reference to the $\ell_1$ norm, or:

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq C|\mathbf{y} - \mathbf{x}|.$$

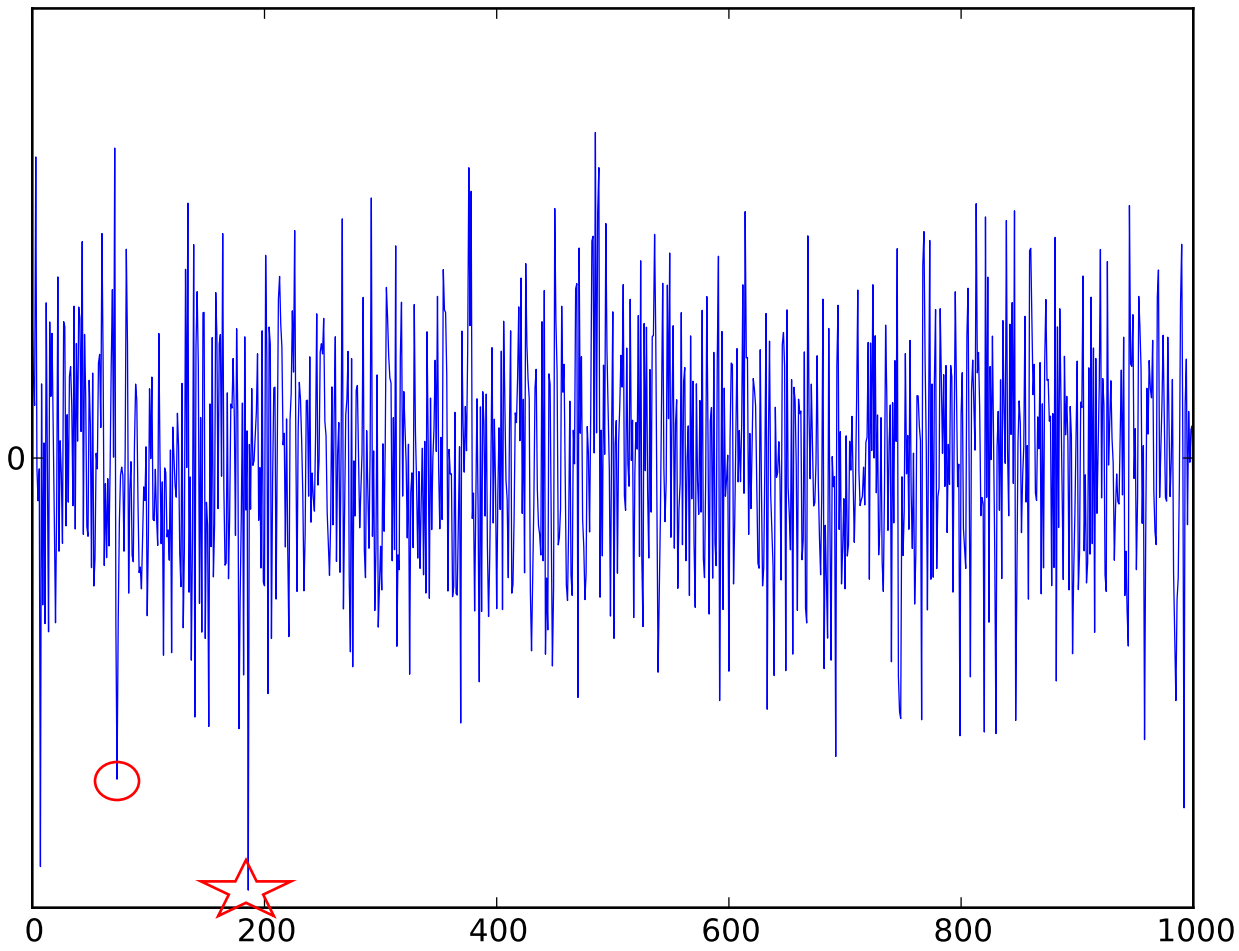
Alternatively:

$$\nabla f(\mathbf{x}) \leq C, \ \forall \, \mathbf{x} \in \Omega$$

# Lipschitz Continuity



# Why is Lipschitz Important?



# Smoothness

A function $f$ is $\beta$-Smooth over a set $\Omega$ with respect to a norm $||\cdot||$ if for any $\mathbf{x}, \mathbf{y} \in \Omega$:

$$||\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})|| \leq \beta ||\boldsymbol{y} - \mathbf{x}||.$$

This is normally defined in terms of the $\ell_1$ norm:

$$|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})| \leq \beta |\mathbf{y} - \mathbf{x}|.$$

Alternatively:

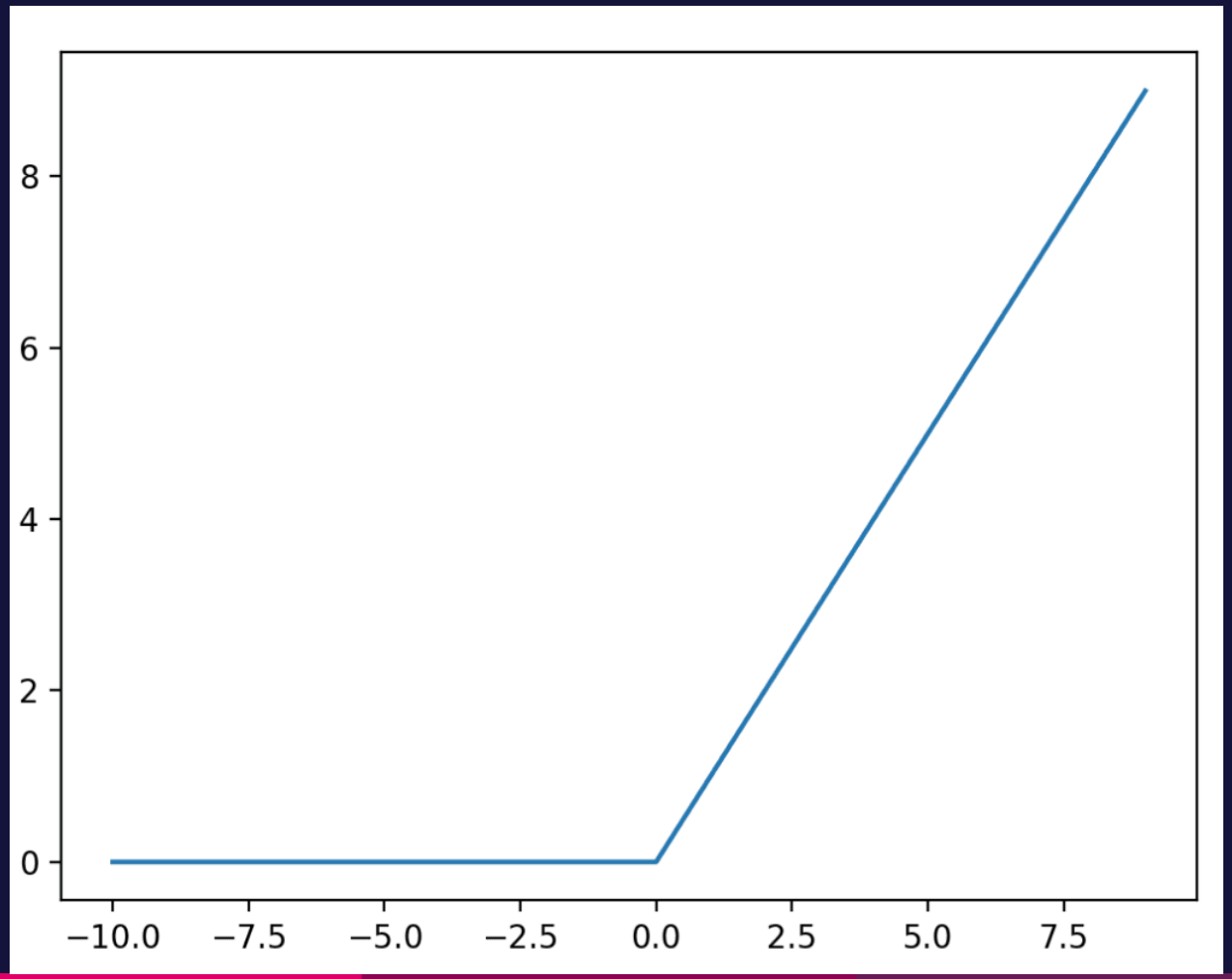
$$\forall\, \mathbf{x}, \mathbf{y} \in \Omega, \quad |f(\mathbf{x}) - f(\mathbf{y}) - \nabla f(\mathbf{y})^\top (\mathbf{x} - \mathbf{y})| \leq \frac{\beta}{2} \|\mathbf{x} - \mathbf{y}\|^2.$$

Finally if $f$ is twice differentiable, then $f$ is $\beta$-smooth if and only if for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^d$:

$$H\mathbf{x}^\top \mathbf{y} \leq \beta ||\mathbf{x}|| ||\mathbf{y}||.$$

# Smoothness

Do you think the below function is smooth?




# Optimisation Problems

# The formulation you will find in the literature

$$
\begin{aligned}
\min \quad & f(\mathbf{w}) \\
\text{s.t.} \quad & g_i(\mathbf{w}) = 0 \quad \text{for } i = 1, \ldots, n \qquad \text{Equality constraints} \\
\text{s.t.} \quad & h_j(\mathbf{w}) \geq 0 \quad \text{for } i = 1, \ldots, m \qquad \text{Inequality constraints}
\end{aligned}
$$

# Optimisation Standard Set Up

$$\begin{aligned} \min \quad & f(\mathbf{w}), \\ \text{s.t.} \quad & \mathbf{w} \in \Omega \end{aligned}$$

$$\Omega \subseteq \mathbb{R}^d,$$

$$f : \Omega \to \mathbb{R}.$$

# Some examples:

DEPARTMENT OF ENGINEERING SCIENCE

UNIVERSITY OF OXFORD

**Convex Optimisation**

$$\begin{aligned} \min \quad & f(\mathbf{w}) \\ \text{s.t.} \quad & \mathbf{w} \in \Omega \end{aligned}$$

$\Omega \subseteq \mathbb{R}^d$,

$\Omega$ is a convex set,

$f : \Omega \to \mathbb{R}$,

$f$ is a convex function.

# Some examples:

DEPARTMENT OF ENGINEERING SCIENCE

UNIVERSITY OF OXFORD

**Linear Programming**

$$\begin{aligned} \min \quad & \boldsymbol{c}^\top \mathbf{w}, \\ \text{s.t.} \quad & A\mathbf{w} \leq \boldsymbol{b}, \\ & \mathbf{w} \geq 0, \\ \text{and} \quad & \mathbf{w} \in \mathbb{R}^d. \end{aligned}$$

$$f(\mathbf{w}) = \boldsymbol{c}^\top \mathbf{w}, \qquad \Omega = \{\mathbf{w} \mid A\mathbf{w} \in \mathbb{R}^d, \ \ \mathbf{w} \leq \boldsymbol{b}, \ \ \text{and} \ \ \mathbf{w} \geq 0\}$$

# Some examples:

DEPARTMENT OF ENGINEERING SCIENCE

UNIVERSITY OF OXFORD

**Discrete Optimisation**

$$\begin{aligned} \min \quad & f(\mathbf{w}) \\ \text{s.t.} \quad & \mathbf{w} \in \Omega \end{aligned}$$

$\Omega \subseteq \mathbb{R}^d$,

Where $\Omega$ is a discrete set. For example $\Omega = \{0, 1\}$,

$f : \Omega \to \mathbb{R}$.

# Some examples:

DEPARTMENT OF ENGINEERING SCIENCE
UNIVERSITY OF OXFORD

**Integer Programming**

$$\begin{aligned} \min \quad & \boldsymbol{c}^\top \mathbf{w} \\ \text{s.t.} \quad & A\mathbf{w} + \mathbf{s} \leq \boldsymbol{b}, \\ & \mathbf{w} \geq 0, \\ & \mathbf{s} \geq 0, \\ \text{and} \quad & \mathbf{w} \in \mathbb{Z}^d. \end{aligned}$$

$$f(\mathbf{w}) = \boldsymbol{c}^\top \mathbf{w}, \qquad \Omega = \{\mathbf{w} \mid A\mathbf{w} + \mathbf{s} \in \mathbb{Z}^d, \;\; \mathbf{w} \leq \boldsymbol{b}, \;\; \text{and} \;\; \mathbf{w} \geq 0, \mathbf{s} \geq 0\},$$

$$\mathbb{Z} \triangleq \{\dots, -3, -2, -1, 0, 1, 2, 3, \dots\}$$

# Focus of this Lecture:

**Unconstrained Optimisation**

$$\begin{aligned} \min \quad & f(\mathbf{w}), \\ \text{s.t.} \quad & \mathbf{w} \in \mathbb{R}^d, \\ & f : \mathbb{R}^d \to \mathbb{R}. \end{aligned}$$

# Finite Sum Loss Functions

DEPARTMENT OF ENGINEERING SCIENCE
UNIVERSITY OF OXFORD

## 5.11 Finite Sum loss functions

$$f(\mathbf{w}) \triangleq \frac{1}{N}\sum_{z=1}^{N}\ell_z(\mathbf{w}) \approx \mathbb{E}_{z\in\mathcal{Z}}[\ell_z(\mathbf{w})]$$

Can you think of a type of loss that would not naturally exhibit a finite sum structure?

**Squared Loss**

$$\text{Squared Loss}: \ell_z(\mathbf{y}_z, \mathbf{y}_z^*) = ||\mathbf{y} - \mathbf{y}^*||_2^2$$

**Cross Entropy**

$$\text{Cross Entropy Loss}: \ell_z(\mathbf{y}_z, \mathbf{y}_z^*) = -\sum_{c\in C} y_c^* \log(y_c)$$

For the cross entropy loss both $\mathbf{y}_z$ and $\mathbf{y}_z^*$ should be vectors denoting a probability distribution ($\sum_i y_{z,i} = 1$, $y_{z,i} \geq 0$, $\forall i$). How can we ensure we have this property? One choice is the softmax.

$$\text{Softmax}(\mathbf{x}) = \frac{\exp \mathbf{x}}{\sum_i \exp x_i}$$

# Regularisation

DEPARTMENT OF ENGINEERING SCIENCE

UNIVERSITY OF OXFORD

## 5.12 Regularisation Functions

$$F(\mathbf{w}) = f(\mathbf{w}) + \lambda R(\mathbf{w})$$

**L2 Regularisation - Weight Decay**

$$\text{L2 Regularisation} \ : R_{\ell_1}(\mathbf{w}) = ||\mathbf{w}||_2^2$$

**L1 Regularisation - Lasso**

$$\text{L1 Regularisation} \ : R_{\ell_2}(\mathbf{w}) = ||\mathbf{w}||_1$$

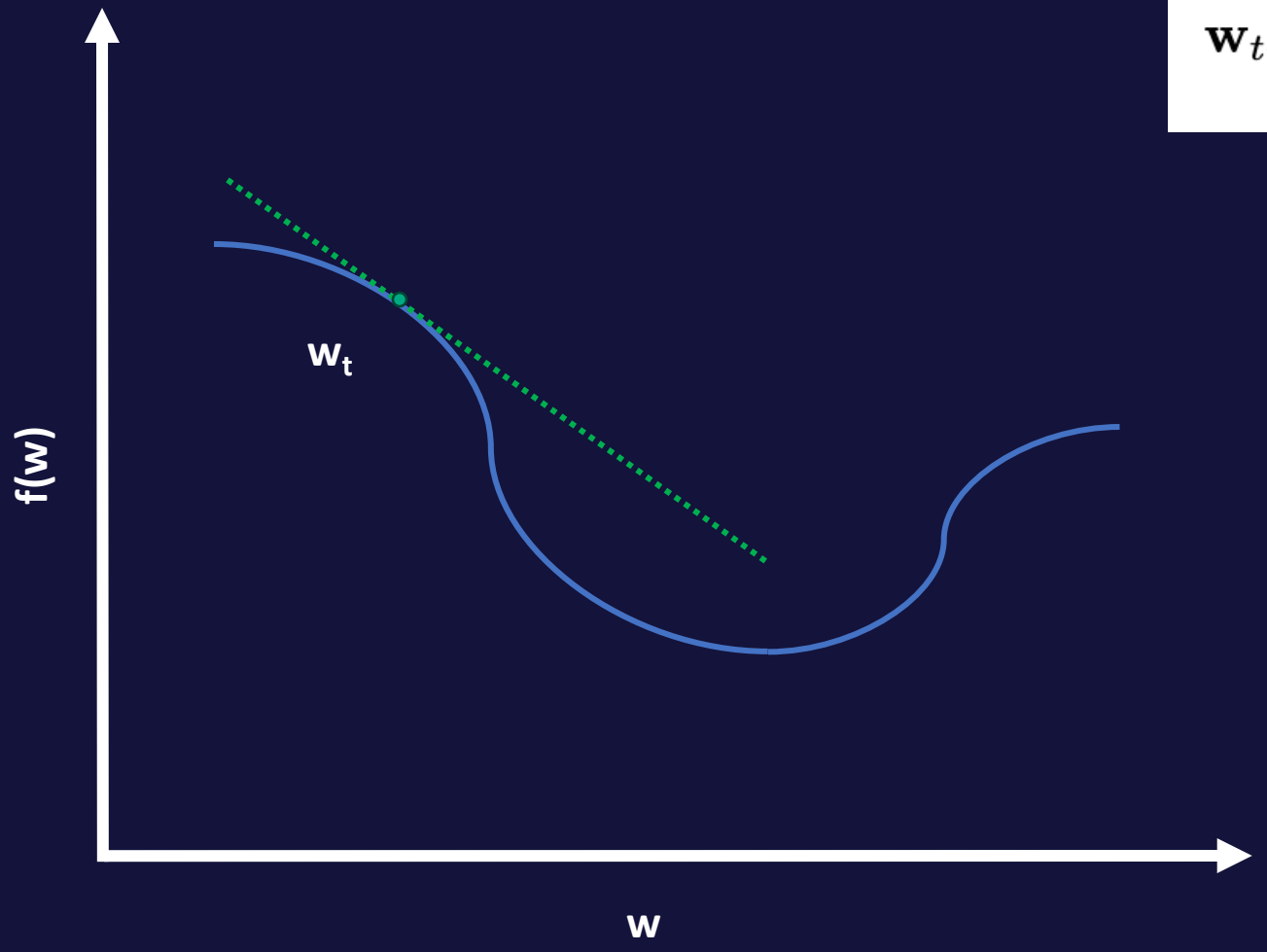

# Optimisation Algorithms

# Types of Optimiser

- **Gradient Free** – Only uses function value information
- **First Order Methods** – use function value and gradient information
- **Second order methods** – use function value, gradient and Hessian information
- **Stochastic Optimisers** – only use approximate function information which has been evaluated on a subset of the training data set

+ Many others

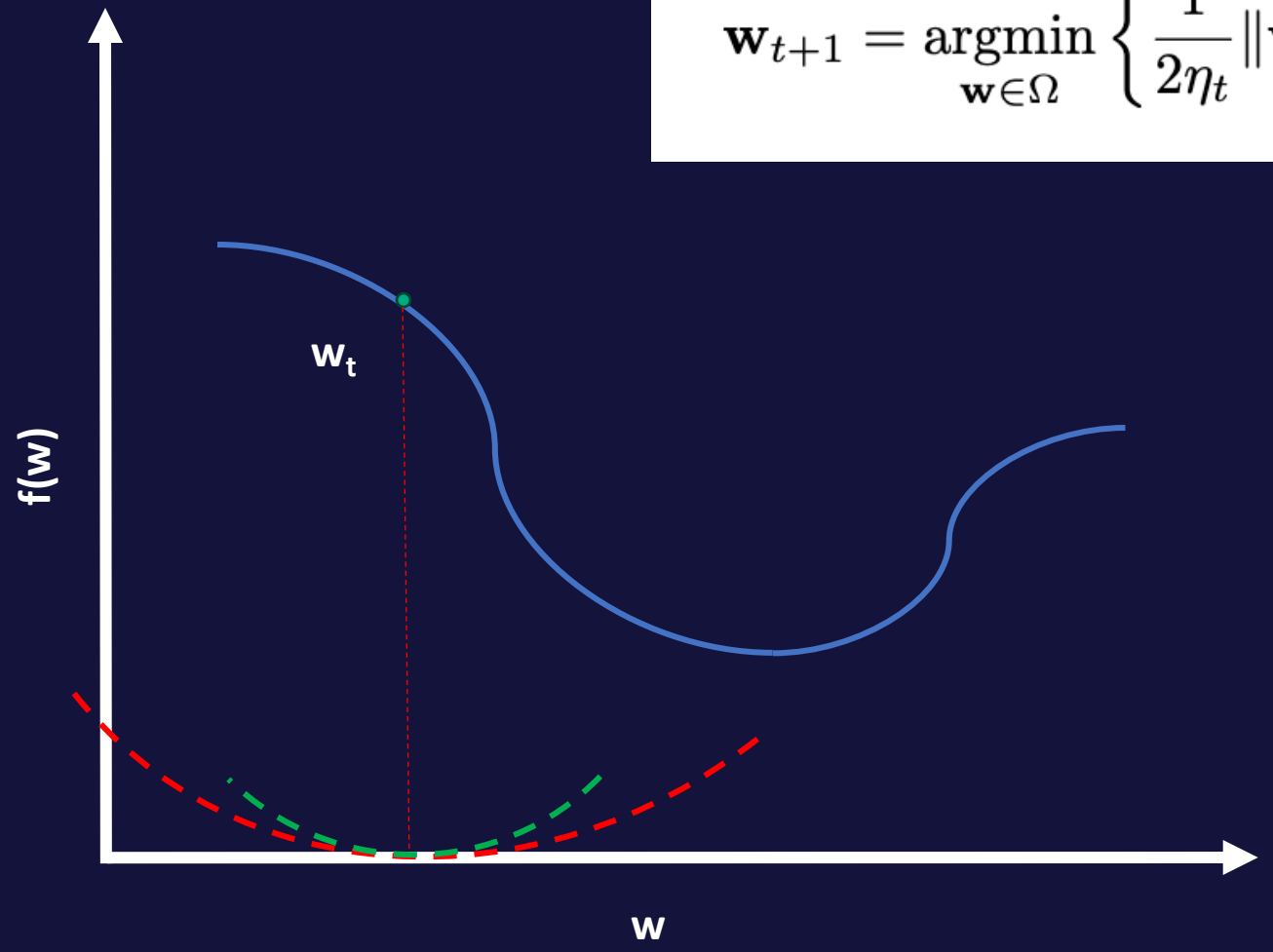
# Gradient Descent (GD)




$$\mathbf{w}_{t+1} = \underset{\mathbf{w}\in\mathbb{R}^d}{\operatorname{argmin}} \left\{ f(\mathbf{w}_t) + \nabla f(\mathbf{w}_t)^\top (\mathbf{w} - \mathbf{w}_t) \right\}.$$

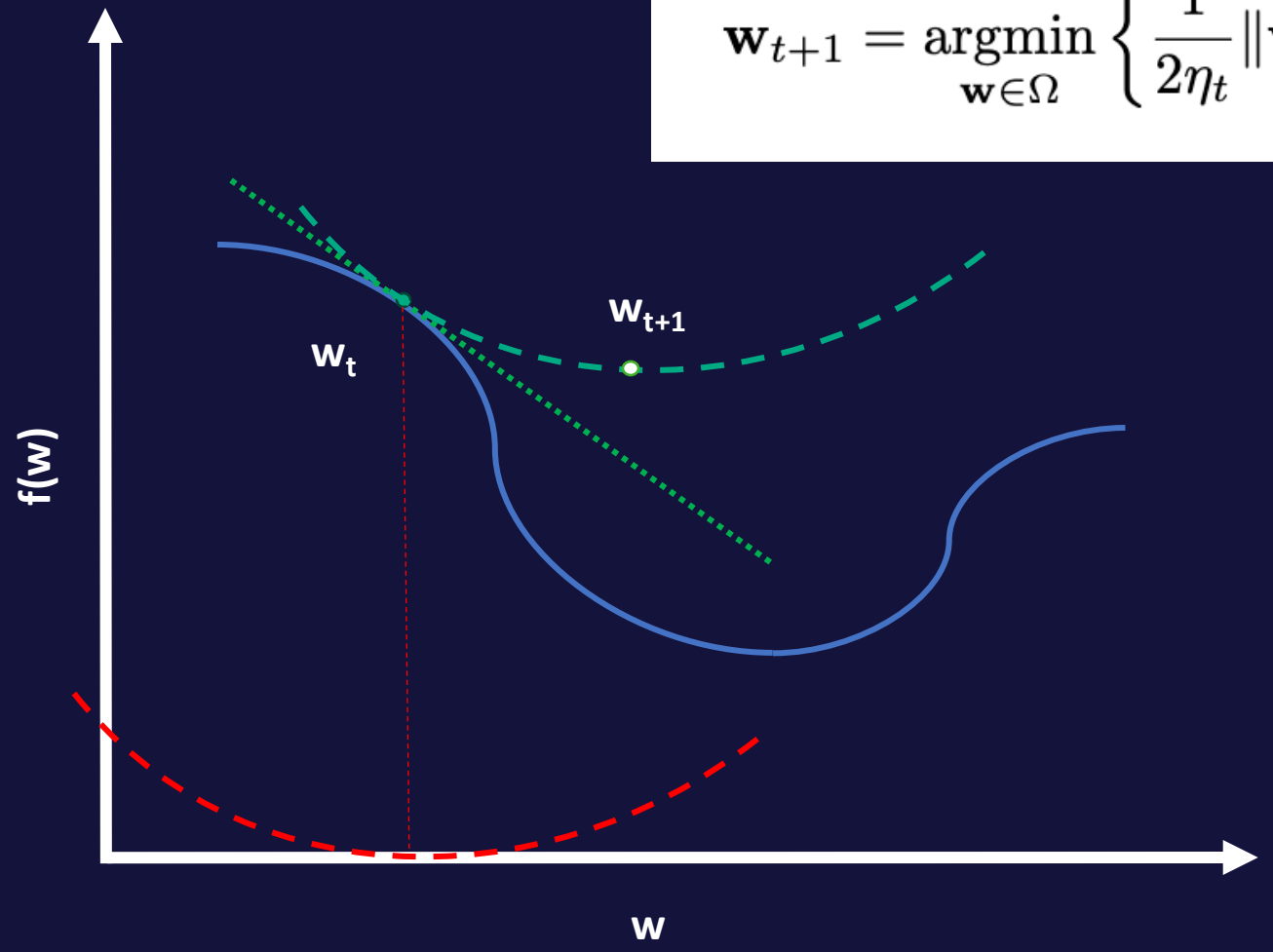
# Gradient Descent (GD)

$$\mathbf{w}_{t+1} = \underset{\mathbf{w}\in\Omega}{\operatorname{argmin}} \left\{ \frac{1}{2\eta_t} \|\mathbf{w} - \mathbf{w}_t\|^2 + f(\mathbf{w}_t) + \nabla f(\mathbf{w}_t)^\top (\mathbf{w} - \mathbf{w}_t) \right\}.$$



# Gradient Descent (GD)

$$\mathbf{w}_{t+1} = \underset{\mathbf{w}\in\Omega}{\operatorname{argmin}} \left\{ \frac{1}{2\eta_t} \|\mathbf{w} - \mathbf{w}_t\|^2 + f(\mathbf{w}_t) + \nabla f(\mathbf{w}_t)^\top (\mathbf{w} - \mathbf{w}_t) \right\}.$$



# Gradient Descent

## 5.1 Gradient Descent

$$\mathbf{w}_{t+1} = \underset{\mathbf{w}\in\Omega}{\operatorname{argmin}} \left\{ \frac{1}{2\eta_t}\|\mathbf{w}-\mathbf{w}_t\|^2 + f(\mathbf{w}_t) + \nabla f(\mathbf{w}_t)^\top(\mathbf{w}-\mathbf{w}_t) \right\}.$$

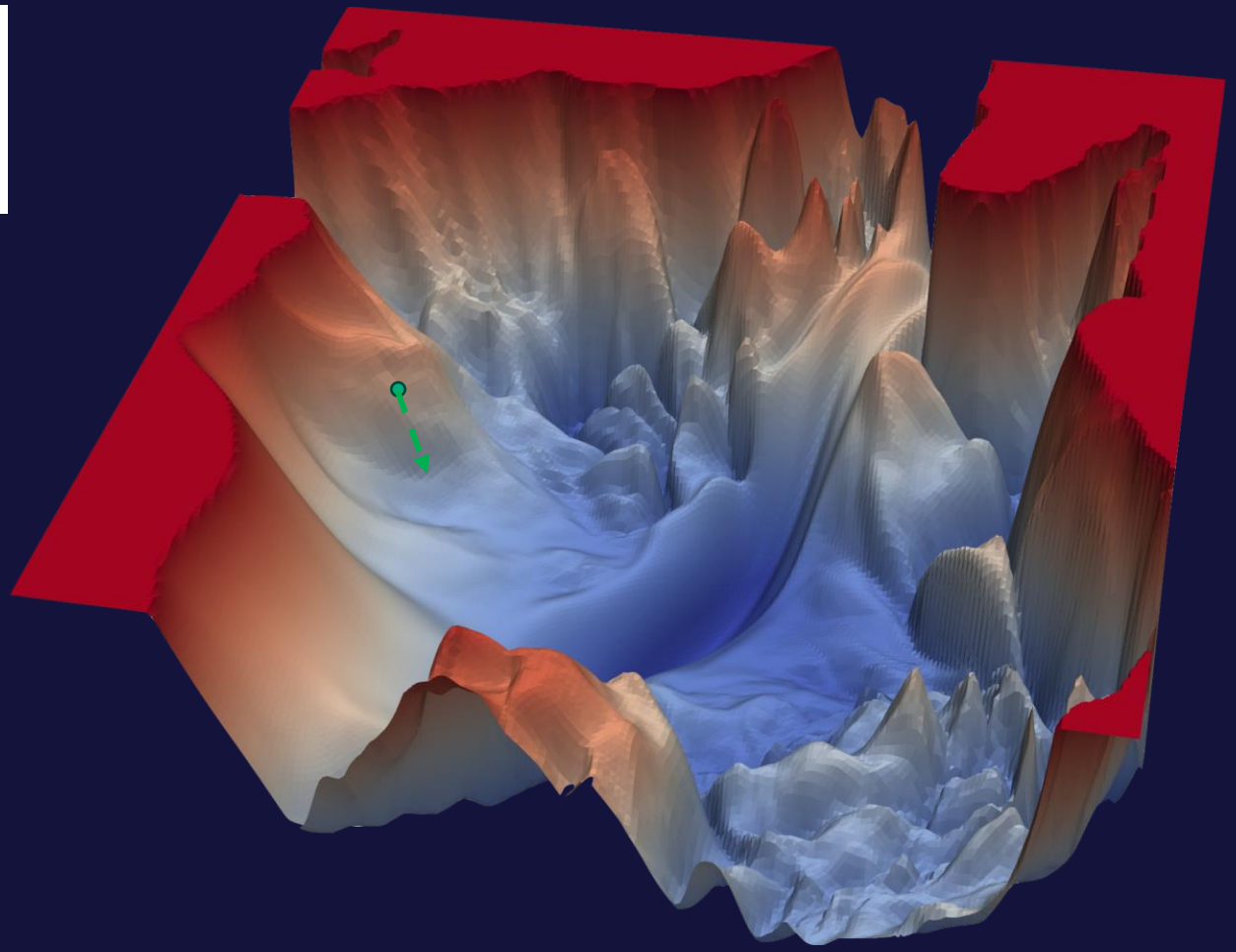
$$\begin{aligned}
&\frac{\partial}{\partial \mathbf{w}} \left( \frac{1}{2\eta_t}\|\mathbf{w}-\mathbf{w}_t\|^2 + f(\mathbf{w}_t) + \nabla f(\mathbf{w}_t)^\top(\mathbf{w}-\mathbf{w}_t) \right), \\
&= \frac{\partial}{\partial \mathbf{w}} \left( \frac{1}{2\eta_t} \left( \|\mathbf{w}\|^2 - 2\mathbf{w}_t^\top\mathbf{w} + \|\mathbf{w}_t\|^2 \right) + f(\mathbf{w}_t) + \nabla f(\mathbf{w}_t)^\top(\mathbf{w}-\mathbf{w}_t) \right), \\
&= \frac{1}{2\eta_t}\left(2\mathbf{w} - 2\mathbf{w}_t + 0\right) + 0 + \nabla f(\mathbf{w}_t), \\
&= \frac{1}{\eta_t}(\mathbf{w}-\mathbf{w}_t) + \nabla f(\mathbf{w}_t).
\end{aligned}$$

Setting the gradient to zero and rearranging give the desired output.

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla f(\mathbf{w}_t).$$

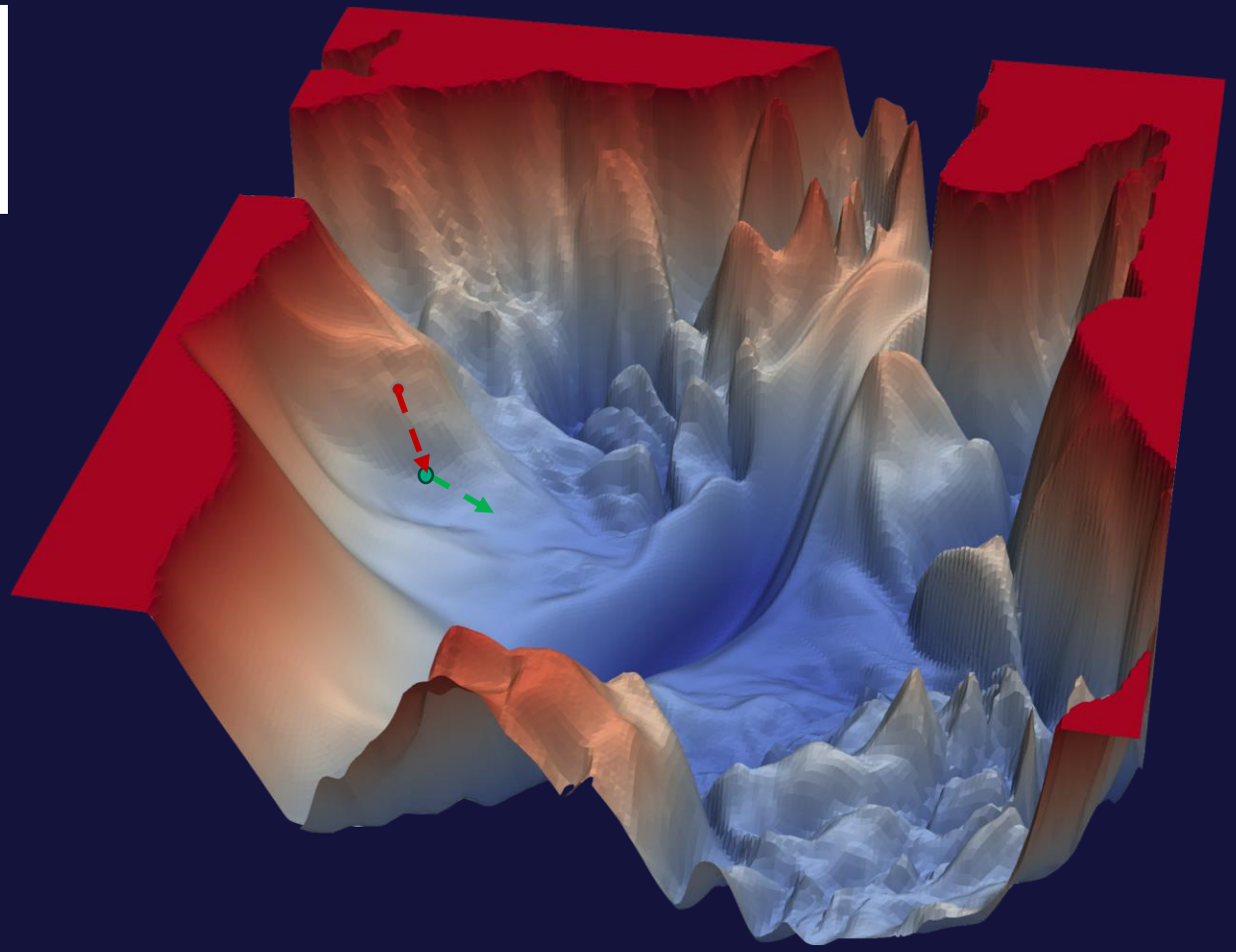
# Gradient Descent

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla f(\mathbf{w}_t).$$

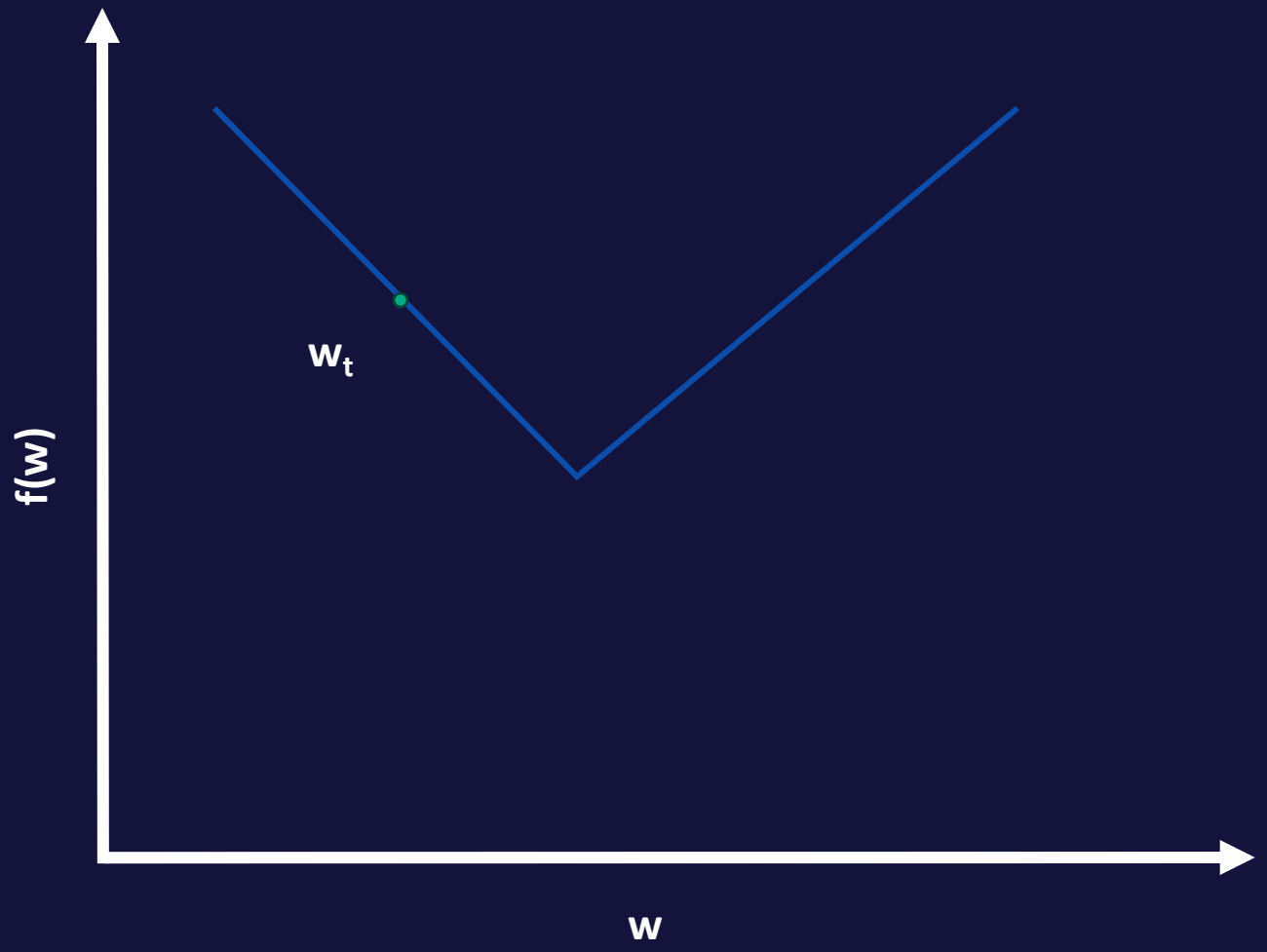


# Gradient Descent

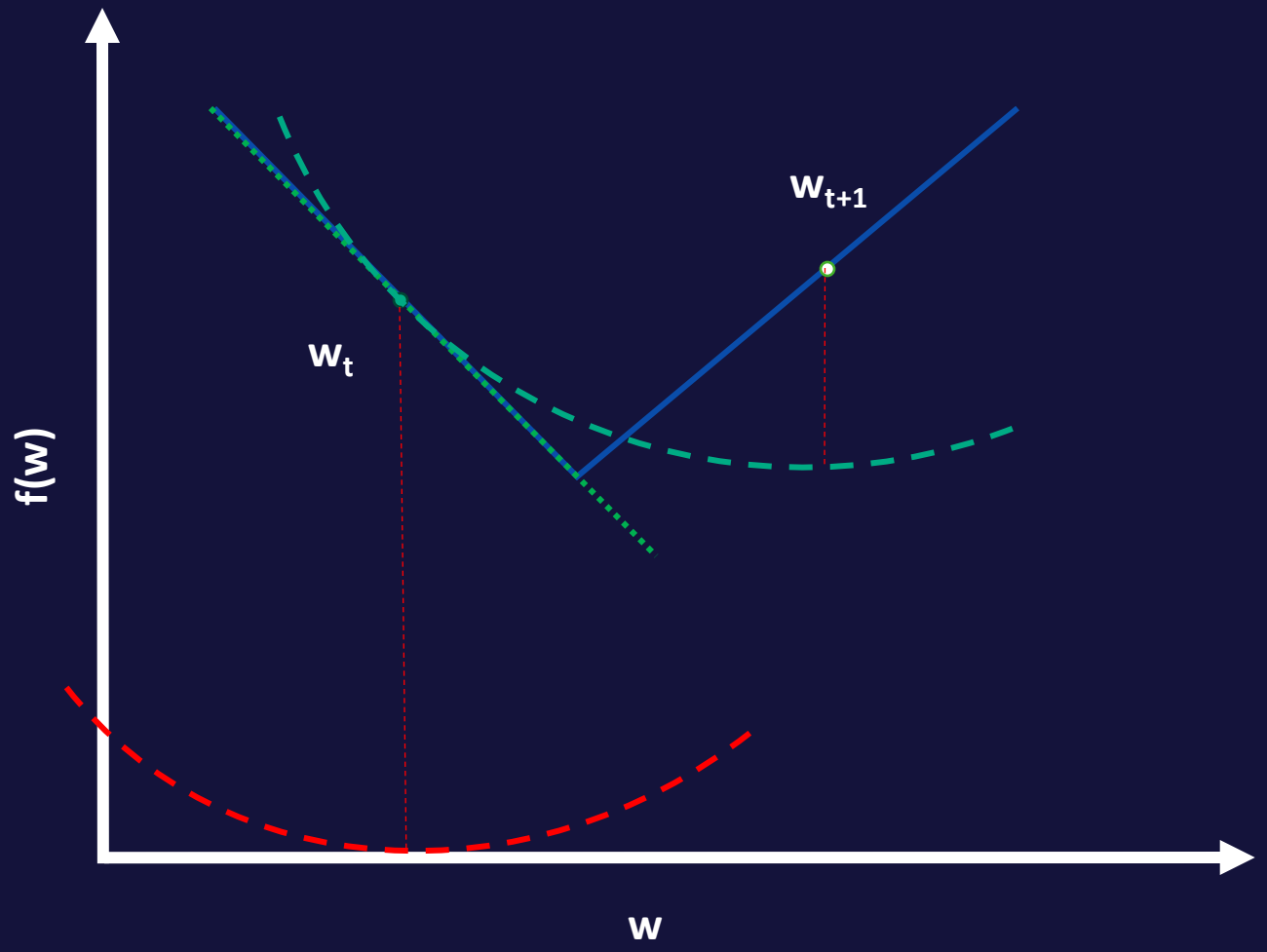
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla f(\mathbf{w}_t).$$



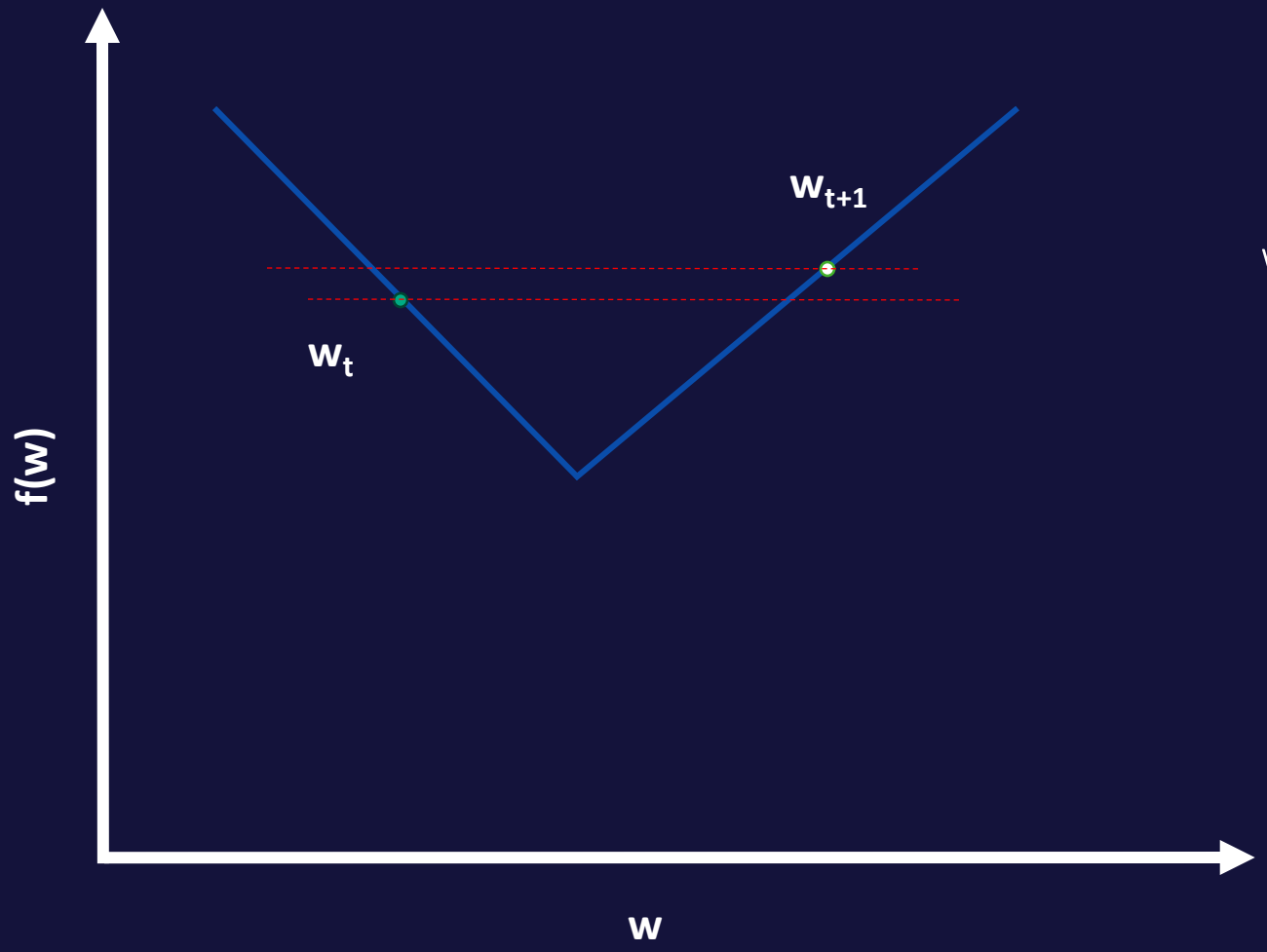
# Gradient Descent (GD)



# Gradient Descent (GD)



# Gradient Descent (GD)




What went wrong here?

# Provable Progress

**Theorem 5.1.** *Let us assume $f$ is convex and smooth with constant $\beta$ then if we select $\eta \leq \frac{2}{\beta}$ we will have monotonic decrease in function value after each step.*

*Proof.* From our assumption that $f$ is convex and smooth with constant $\beta$ then by the definition of the smoothness we have:

$$\forall\, \mathbf{x}, \mathbf{y}, \quad |f(\mathbf{x}) - f(\mathbf{y}) - \nabla f(\mathbf{y})^\top (\mathbf{x} - \mathbf{y})| \leq \frac{\beta}{2} \|\mathbf{x} - \mathbf{y}\|^2.$$

Rearranging gives:

$$f(\mathbf{x}) \leq f(\mathbf{y}) + \nabla f(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}) + \frac{\beta}{2} \|\mathbf{x} - \mathbf{y}\|^2.$$

Let $\mathbf{x} = \mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla f(\mathbf{w}_t)$, $\mathbf{y} = \mathbf{w}_t$ and hence $\mathbf{x} - \mathbf{y} = -\eta \nabla f(\mathbf{w}_t)$. Plugging this in gives:

$$f(\mathbf{w}_{t+1}) \leq f(\mathbf{w}_t) + \eta \left( -\|\nabla f(\mathbf{w}_t)\|^2 + \frac{\beta\eta}{2} \|\nabla f(\mathbf{w}_t)\|^2 \right),$$

$$f(\mathbf{w}_{t+1}) \leq f(\mathbf{w}_t) - \eta \left( 1 - \frac{\beta\eta}{2} \right) \|\nabla f(\mathbf{w}_t)\|^2.$$

Hence if $1 - \frac{\beta\eta}{2} \geq 0$ we will have a decrease. Rearranging this condition gives the desired result. □

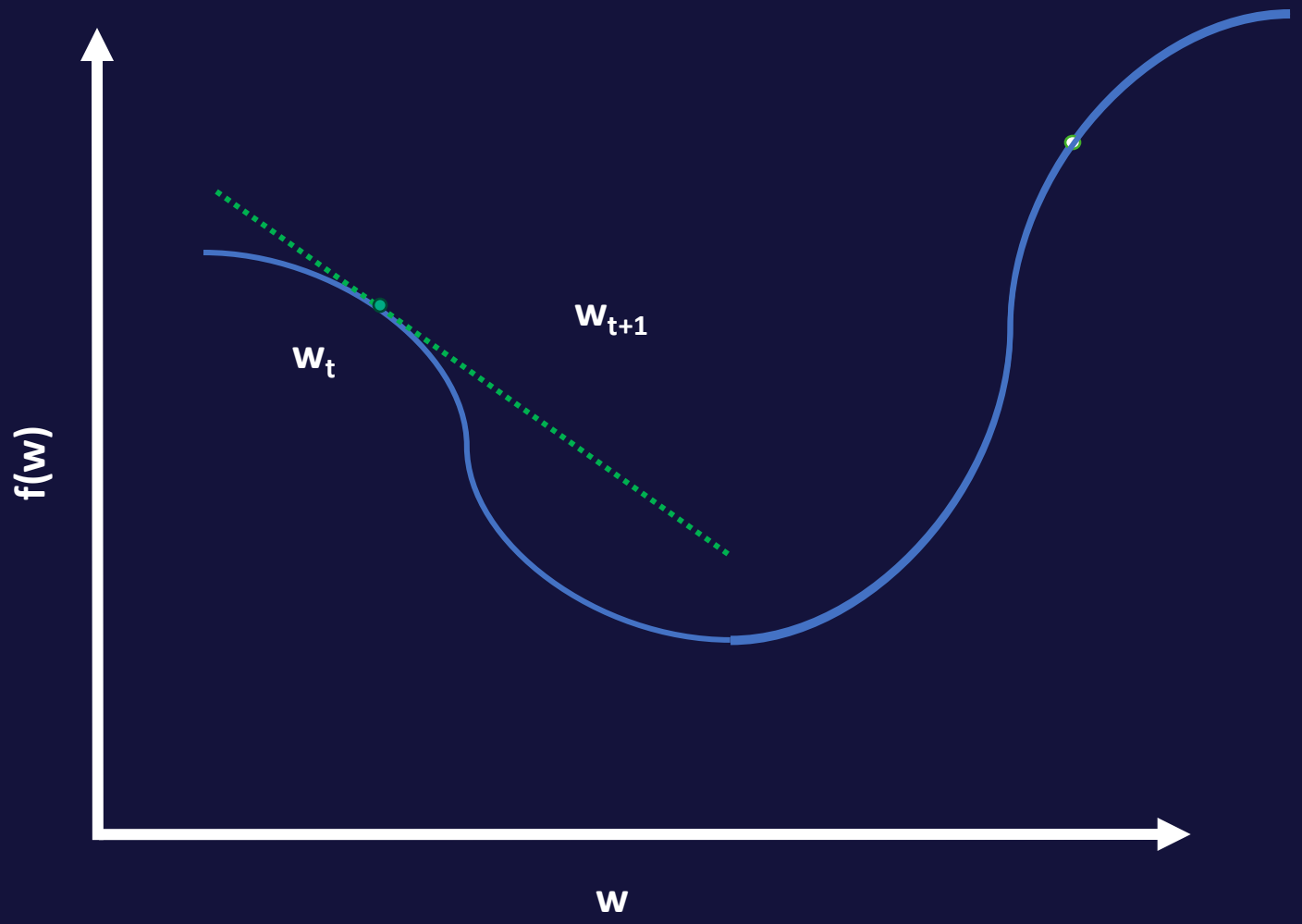
## 5.3 What if we don't know what the smoothness constant is?

Simply try a bunch of different values, and keep the one that works best. This process is know as cross validation or hyperparameter tuning.

## 5.4 What if we don't have a smooth function?

What if our function is only Lipschitz continuous and not smooth? well we can still prove GD asymptotically converges to the optimum for convex function but only with a decreasing step size such as $\eta_t = \frac{1}{\sqrt{t}}$.

# Line Search Methods



# Line Search Methods


## 5.5 Line Search Methods

Line search methods contain two key components:

1. A method for proposing points, typically backtracking procedure.
2. A condition to determine where a point is accepted.

# Line Search Methods

**Backtracking Line Search** At time $t$ a point we first select a point $\mathbf{w}_t - \eta_{t,0}\nabla f(\mathbf{w}_t)$ check the point meets the acceptance conditions, if not $\eta_{t,k+1} = \gamma \cdot \eta_{t,k}$ where $\gamma \in (0, 1)$.

**Armijo-Goldstein Conditions**

$$f(\mathbf{w}_t - \eta_{t_k}\nabla f(\mathbf{w}_t)) \leq f(\mathbf{w}_t) - c\eta_{t_k}\|\nabla f(\mathbf{w}_t)\|^2, \qquad 0 < c.$$

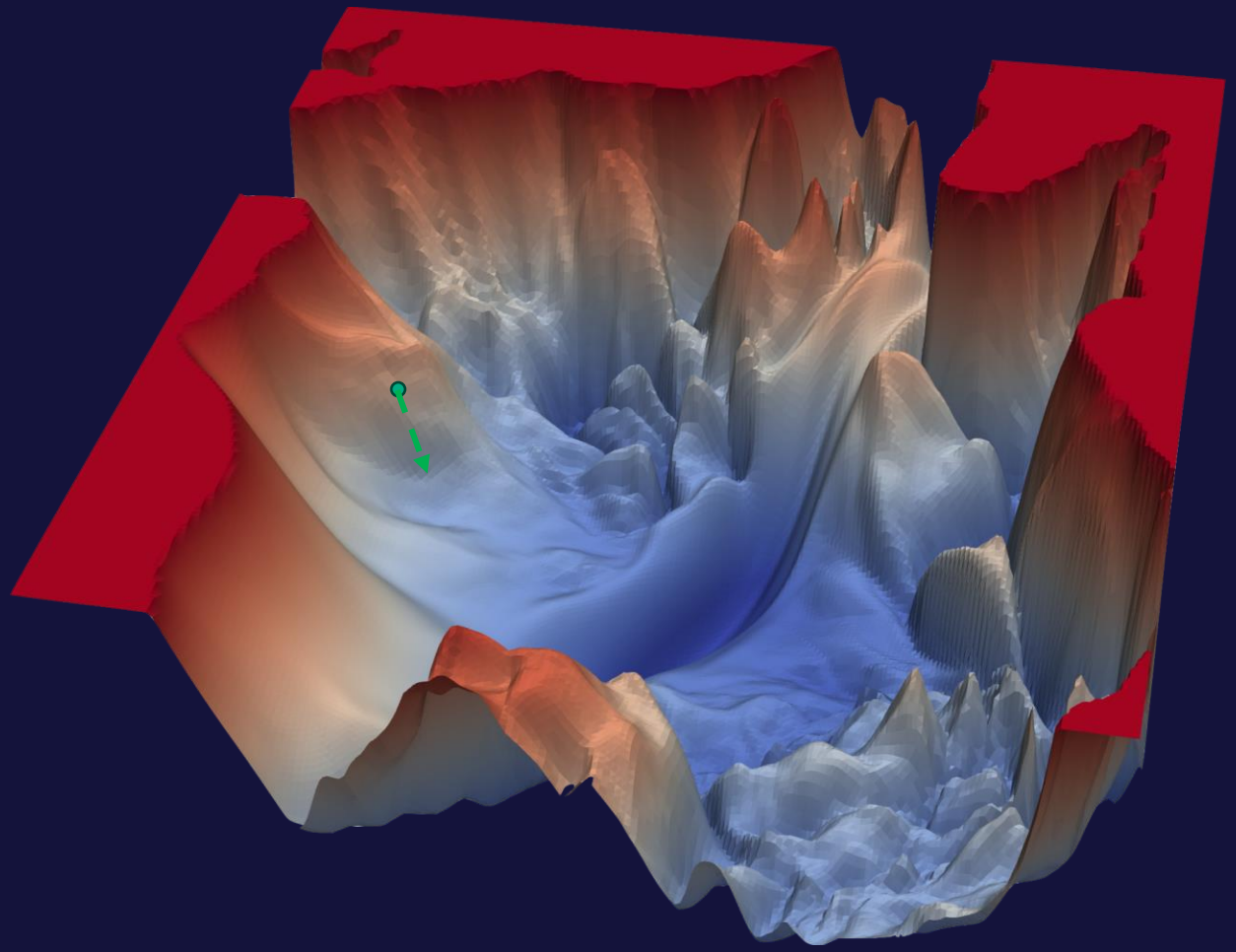
**Wolfe Conditions**

$$f(\mathbf{w}_t - \eta_{t_k}\nabla f(\mathbf{w}_t)) - f(\mathbf{w}_t) \leq c\eta_{t_k}\|\nabla f(\mathbf{w}_t)\|^2,$$
$$\nabla f(\mathbf{w}_t - \eta_{t_k}\nabla f(\mathbf{w}_t))^\top \nabla f(\mathbf{w}_t) \geq c\|\nabla f(\mathbf{w}_t)\|^2 \qquad 0 < c_1 < c_2 < 1$$

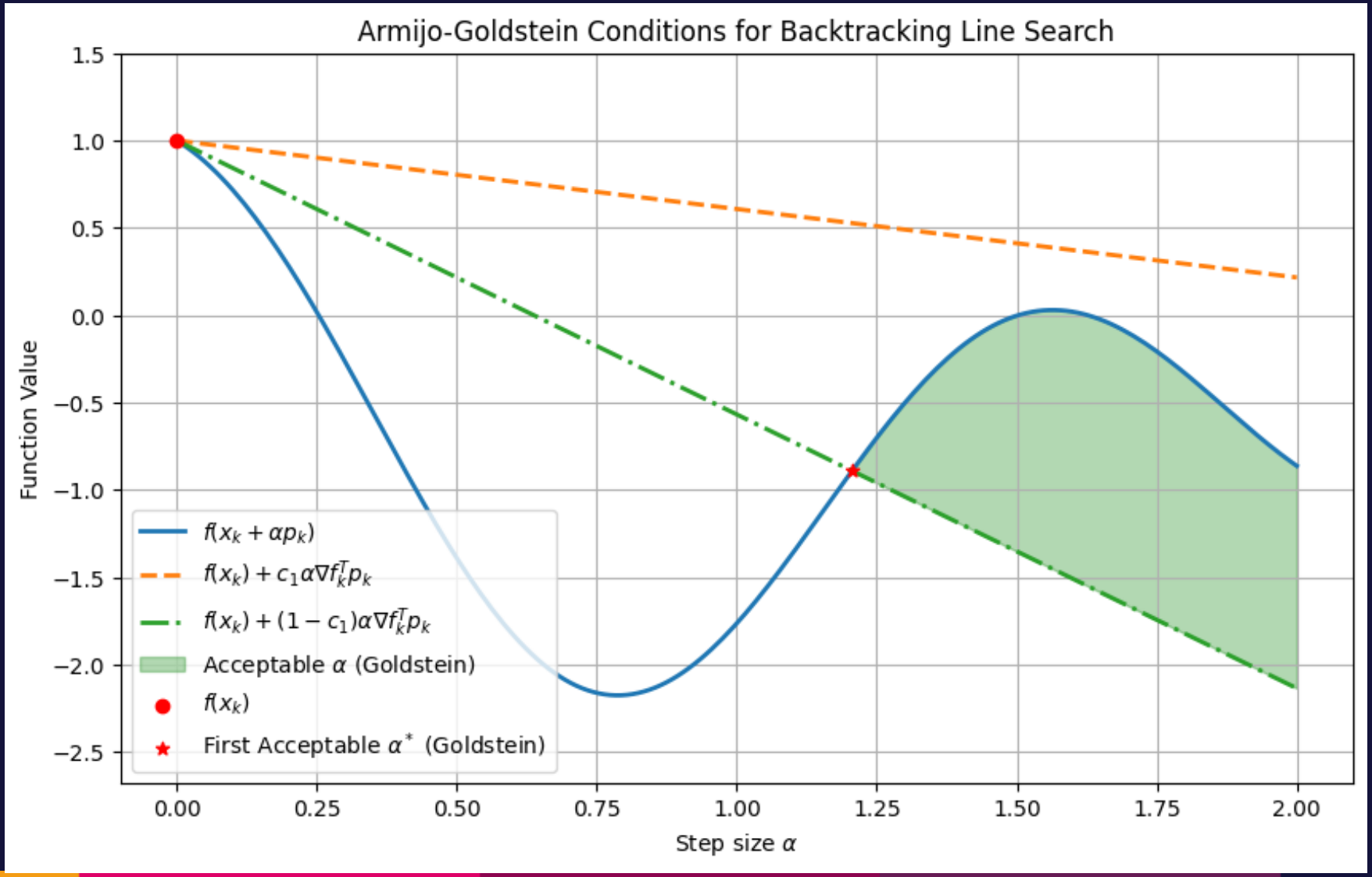
Question: Why might one prefer the Armijo-Goldstein Conditions?

# Line Seach Methods

DEPARTMENT OF ENGINEERING SCIENCE
UNIVERSITY OF OXFORD



# Line Search Methods



# Line Search Methods

**Algorithm 2** Back Tracking Line Search

**Require:** $\gamma$: learning rate
**Require:** $\alpha$: scaling factor
**Require:** $f$: objective function
**Require:** $\mathbf{w}_0$: initial parameter vector
**Require:** $c$: Armijo-Goldstein Hyper-parameter
**Require:** $k_{max}$: max number of inner iterations

**for** $t = 1, \cdots, T$ **do**
  $\eta_{t_0} = \gamma$
  **for** $k$ in range $k_{max}$ **do**
    **if** $f(\mathbf{w}_t - \eta_{t_k} \nabla f(\mathbf{w}_t)) \leq f(\mathbf{w}_t) - c\eta_{t_k} \|\nabla f(\mathbf{w}_t)\|^2$ **then**
      $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_{t_k} \nabla f(\mathbf{w}_t)$
      **Break**
    **else**
      $\eta_{t_k} = \alpha \cdot \eta_{t_{k-1}}$
    **end if**
  **end for**
**end for**
**return** $\mathbf{w}_T$

# Secord Order Methods

DEPARTMENT OF ENGINEERING SCIENCE
UNIVERSITY OF OXFORD

## Second Order Methods

$$\mathbf{w}_{t+1} = \underset{\mathbf{w}\in\Omega}{\operatorname{argmin}} \left\{ f(\mathbf{w}_t) + \nabla f(\mathbf{w}_t)^\top (\mathbf{w} - \mathbf{w}_t) + \frac{1}{2}(\mathbf{w} - \mathbf{w}_t)^T \boldsymbol{H}_f (\mathbf{w} - \mathbf{w}_t) \right\}.$$

$$\begin{aligned}
&\frac{\partial}{\partial \mathbf{w}} \left( f(\mathbf{w}_t) + \nabla f(\mathbf{w}_t)^\top (\mathbf{w} - \mathbf{w}_t) + \frac{1}{2}(\mathbf{w} - \mathbf{w}_t)^T \boldsymbol{H}_f (\mathbf{w} - \mathbf{w}_t) \right), \\
&= \frac{\partial}{\partial \mathbf{w}} \left( f(\mathbf{w}_t) + \nabla f(\mathbf{w}_t)^\top (\mathbf{w} - \mathbf{w}_t) + \frac{1}{2} \left( \mathbf{w}^T \boldsymbol{H}_f \mathbf{w} - 2\mathbf{w}^\top \boldsymbol{H}_f \mathbf{w}_t + \mathbf{w}_t^T \boldsymbol{H}_f \mathbf{w}_t \right) \right), \\
&= \left( 0 + \nabla f(\mathbf{w}_t) + \frac{1}{2} \left( 2\boldsymbol{H}_f \mathbf{w} - 2\boldsymbol{H}_f \mathbf{w}_t + 0 \right) \right) \\
&= \nabla f(\mathbf{w}_t) + \boldsymbol{H}_f \mathbf{w} - \boldsymbol{H}_f \mathbf{w}_t
\end{aligned}$$

# Secord Order Methods

Setting the gradient equal to zero:

$$
\begin{aligned}
0 &= \nabla f(\mathbf{w}_t) + \boldsymbol{H}_f \mathbf{w} - \boldsymbol{H}_f \mathbf{w}_t, \\
-\nabla f(\mathbf{w}_t) &= \boldsymbol{H}_f \mathbf{w} - \boldsymbol{H}_f \mathbf{w}_t, \\
-\boldsymbol{H}_f^{-1} \nabla f(\mathbf{w}_t) &= \mathbf{w} - \mathbf{w}_t, \\
\mathbf{w} &= \mathbf{w}_t - \boldsymbol{H}_f^{-1} \nabla f(\mathbf{w}_t).
\end{aligned}
$$

What do you notice about this update?

# Stochastic Gradient Descent (SGD)

DEPARTMENT OF ENGINEERING SCIENCE
UNIVERSITY OF OXFORD

$$f(\mathbf{w}) \triangleq \frac{1}{N}\sum_{z=1}^{N}\ell_z(\mathbf{w}) \approx \mathbb{E}_{z\in\mathcal{Z}}[\ell_z(\mathbf{w})]$$

**Gradient Descent**

$$\mathbf{w}_{t+1} = \underset{\mathbf{w}\in\Omega}{\operatorname{argmin}} \left\{ \frac{1}{2\eta_t}\|\mathbf{w}-\mathbf{w}_t\|^2 + f(\mathbf{w}_t) + \nabla f(\mathbf{w}_t)^\top(\mathbf{w}-\mathbf{w}_t) \right\}.$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla f(\mathbf{w}_t).$$

**Stochastic Gradient Descent**

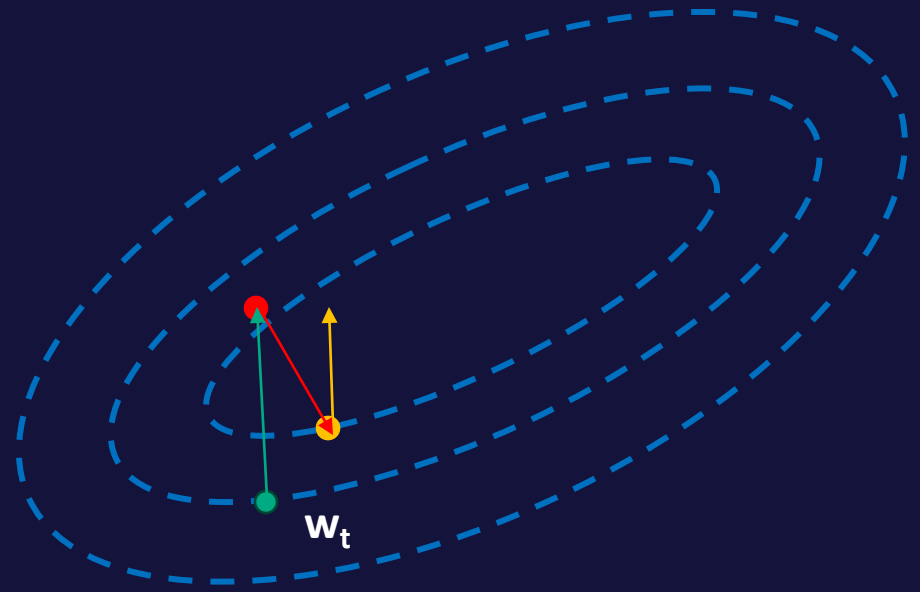
$$\mathbf{w}_{t+1} = \underset{\mathbf{w}\in\Omega}{\operatorname{argmin}} \left\{ \frac{1}{2\eta_t}\|\mathbf{w}-\mathbf{w}_t\|^2 + \ell_{z_t}(\mathbf{w}_t) + \nabla \ell_{z_t}(\mathbf{w}_t)^\top(\mathbf{w}-\mathbf{w}_t) \right\}.$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla \ell_{z_t}(\mathbf{w}_t).$$

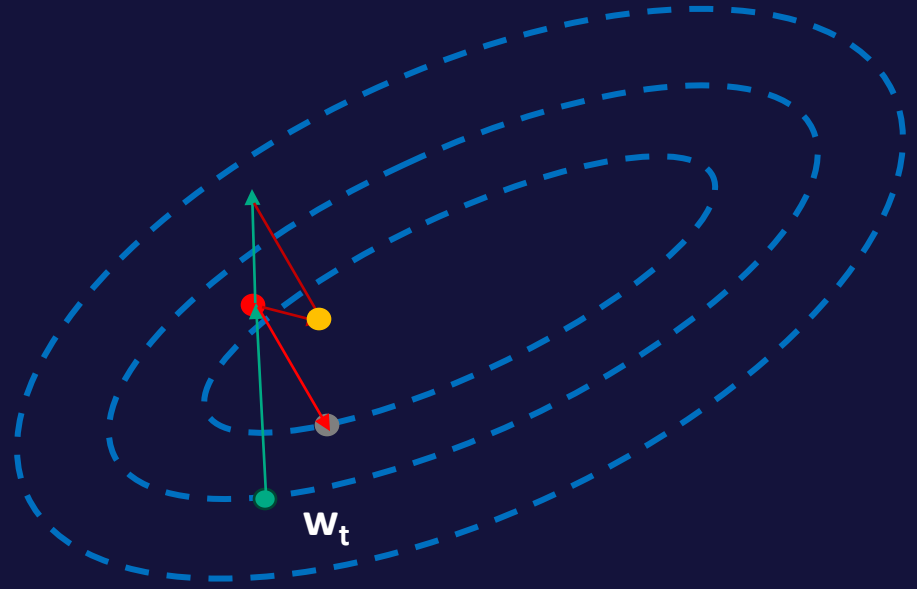
DEPARTMENT OF ENGINEERING SCIENCE

UNIVERSITY OF OXFORD

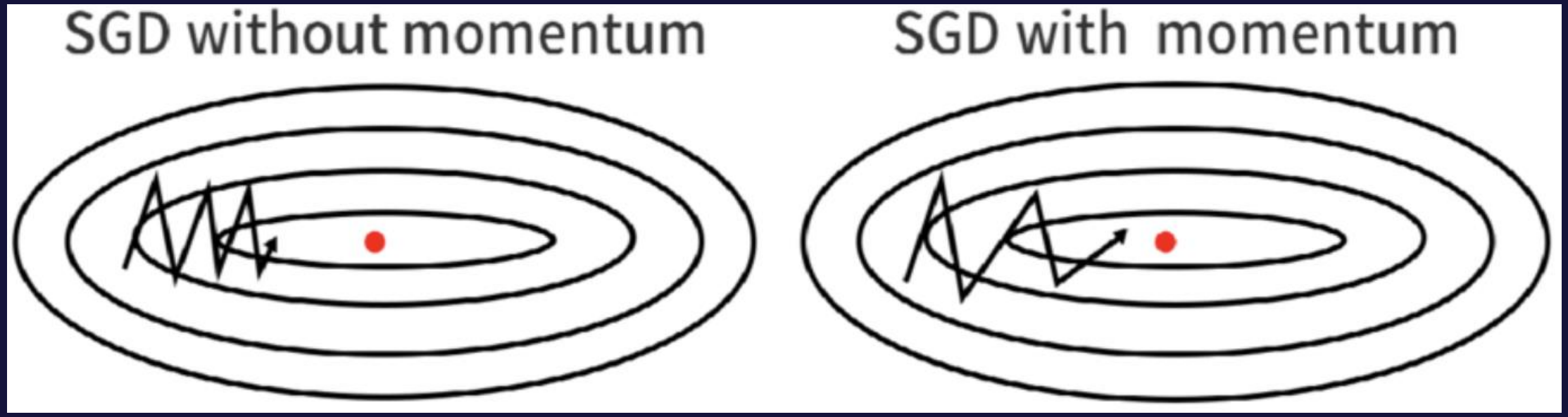
# GD




# Momentum



# Momentum




https://production-media.paperswithcode.com/methods/Screen_Shot_2020-05-28_at_3.25.40_PM_Y687HvA.png

# Momentum

**SGD with momentum**

$$\begin{aligned}\mathbf{m}_0 &= 0,\\ \mathbf{m}_{t+1} &= \mu\mathbf{m}_t - \eta_t \nabla \ell_{z_t}(\mathbf{w}_t),\\ \mathbf{w}_{t+1} &= \mathbf{w}_t + \mathbf{m}_t.\end{aligned}$$

**Nestarov Momentum**

$$\begin{aligned}\mathbf{m}_0 &= 0,\\ \mathbf{m}_{t+1} &= \mu\mathbf{m}_t - \eta_t \nabla \ell_{z_t}(\mathbf{w}_t),\\ \mathbf{w}_{t+1} &= \mathbf{w}_t - \eta_t \nabla \ell_{z_t}(\mathbf{w}_t) + \mu\mathbf{m}_t.\end{aligned}$$

# Momentum

**SGD with momentum**

$$\begin{aligned}\mathbf{m}_0 &= 0,\\ \mathbf{m}_{t+1} &= \mu\mathbf{m}_t - \eta_t \nabla \ell_{z_t}(\mathbf{w}_t),\\ \mathbf{w}_{t+1} &= \mathbf{w}_t + \mathbf{m}_t.\end{aligned}$$

What happens to the update if the gradient is constant?

# What is causing this zig-zagging?

- Poorly scaled dimension – high condition number

# What is causing this zig-zagging?

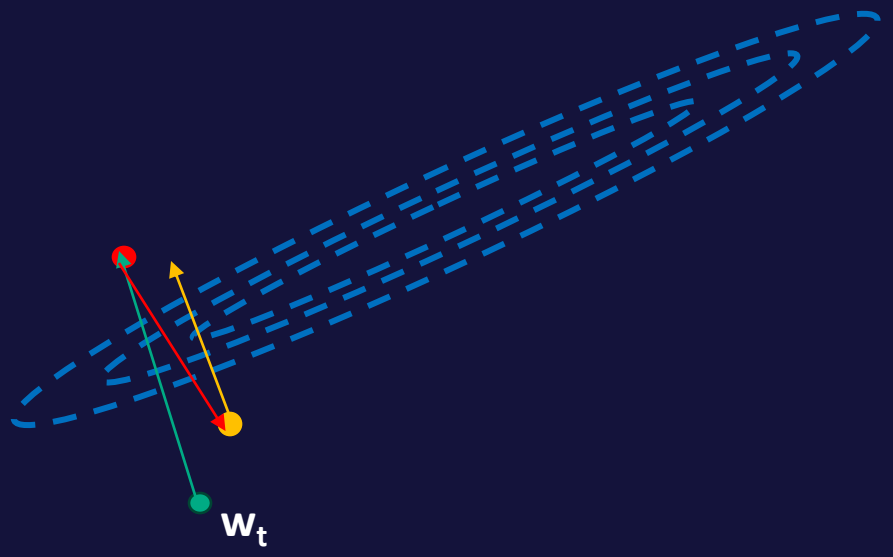
Dimension with very different scales:

**RMSprop**

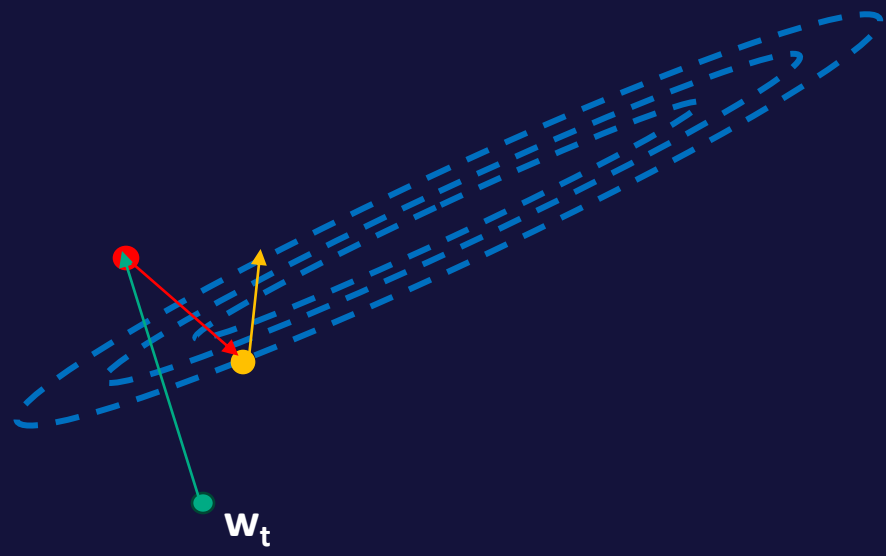
$$\boldsymbol{v}_k \leftarrow 0.9\boldsymbol{v}_{k-1} + 0.1\nabla\ell_{z_t}(\mathbf{w}_t)^2$$

$$\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k - \frac{\eta}{\sqrt{\boldsymbol{v}_k + \epsilon}}\ell_{z_t}(\mathbf{w}_t)$$

# GD




# RMSprop



# Adam



**Adam**

$$\hat{\boldsymbol{m}}_k = \frac{\boldsymbol{m}_k}{1-\beta_1^t}, \ \boldsymbol{m}_k \leftarrow \beta_1 \boldsymbol{m}_{k-1} + (1-\beta_1)\nabla \ell_{z_t}(\mathbf{w}_t)$$

$$\hat{\boldsymbol{v}}_k = \frac{\boldsymbol{v}_k}{1-\beta_2^t}, \ \boldsymbol{v}_k \leftarrow \beta_2 \boldsymbol{v}_{k-1} + (1-\beta_2)\nabla \ell_{z_t}(\mathbf{w}_t)^2$$

$$\mathbf{w}_{k+1} \leftarrow \mathbf{w}_k - \frac{\eta}{\sqrt{\hat{\boldsymbol{v}}_k} + \epsilon}\hat{\boldsymbol{m}}_k$$

# Gradient Free Optimisation


- What if you don't have access to the gradient?
- One must use optimisers that rely on function values
- Many gradient free optimisation algorithms exist
- In this class we use the "3 point" method as an illustrative example

# Gradient Free Optimisation


**Algorithm 1** Non-Stochastic Three Points Method

**Require:** $\eta$: learning rate
**Require:** $f$: objective function
**Require:** $\mathcal{D}$: distribution over trial direction that spans $\mathbb{R}^d$ for example $P(\mathcal{D} = \mathbf{e}_i) = 1/d$
**Require:** $\mathbf{w}_0$: initial parameter vector
  **for** $t = 1, \cdots, T$ **do**
    $\boldsymbol{p}_t \sim \mathcal{D}$
    $\mathcal{W}_t \triangleq \{\mathbf{w}_t - \eta \boldsymbol{p}_t, \mathbf{w}_t, \mathbf{w}_t + \eta \boldsymbol{p}_t\}$
    $\mathbf{w}_{t+1} \leftarrow \text{argmin}_{\mathbf{w} \in \mathcal{W}_t} f(\mathcal{W}_t)$ {try all three values and pick the best}
  **end for**
  **return** $\mathbf{w}_T$

What do you think might be the disadvantages of this approach?


# Time for some exercises