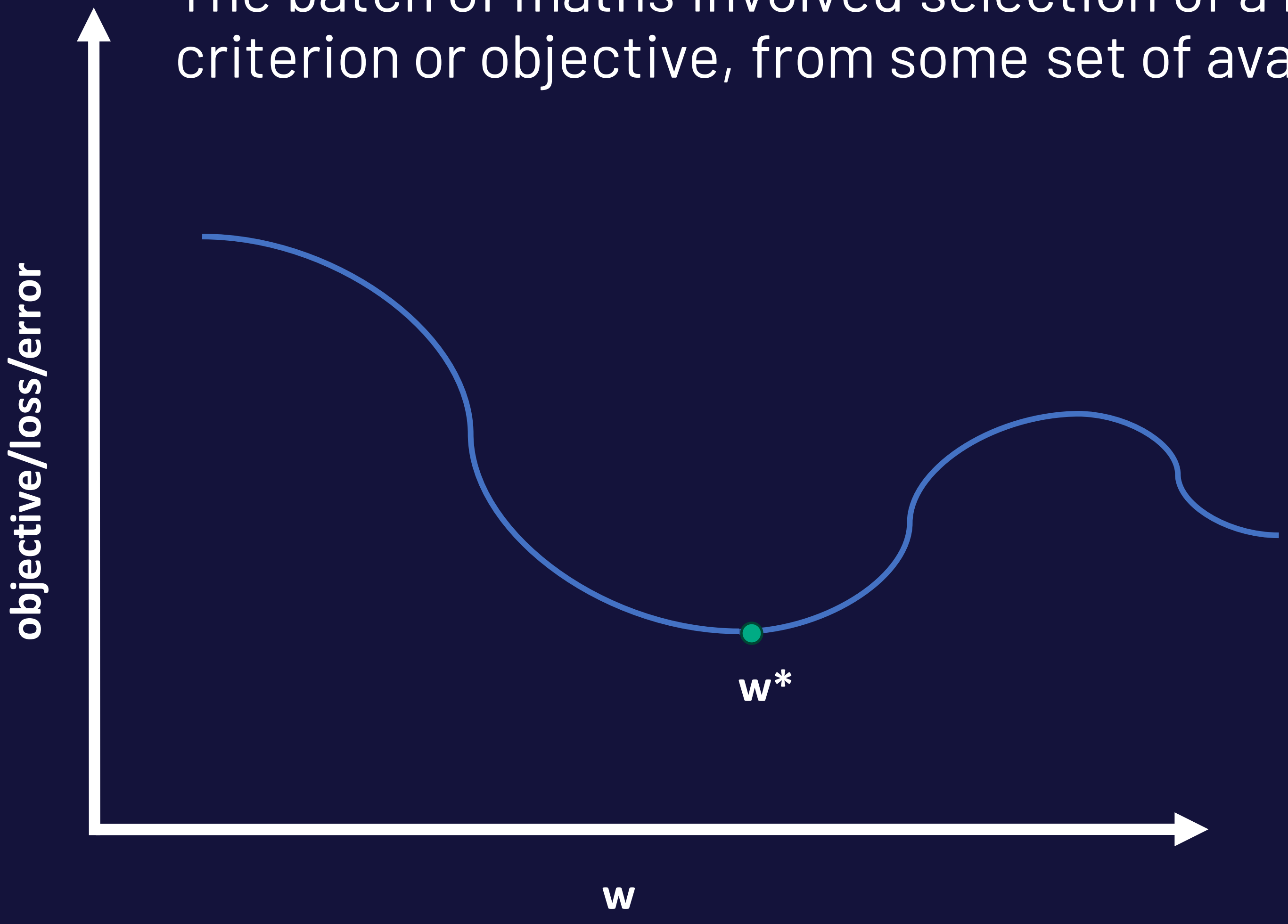
Optimization

Lecturer: Dr. Yangchen Pan, Department of Engineering Science

Acknowledgement: the slides are largely adapted from those designed last year by Dr. Alasdair Paren.

What is Optimisation?

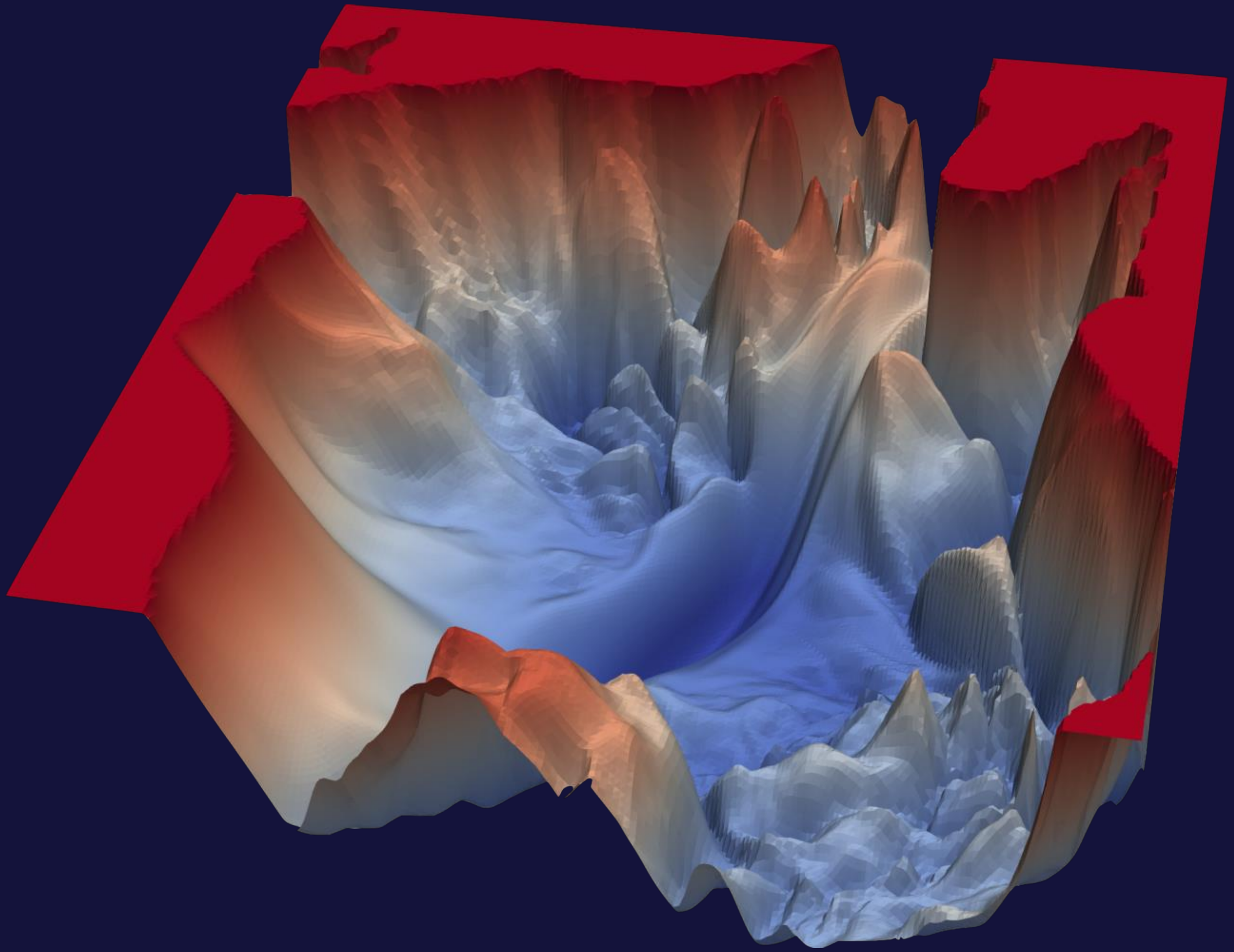
The batch of maths involved selection of a best element, with regard to some criterion or objective, from some set of available alternatives.



Visualization of Neural Network Loss Function

DEPARTMENT OF ENGINEERING SCIENCE

UNIVERSITY OF OXFORD



https://www.cs.umd.edu/~tomg/projects/landscapes/

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, over a very large training set

Loss Functions

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

Desirable properties:

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

Types or Optimisation

- Convex / None Convex
- Constrained / None Constrained
- Discrete / Continuous / Mixed Integer Programming
- Stochastic / None stochastic
- 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)

DEPARTMENT OF ENGINEERING SCIENCE

UNIVERSITY OF OXFORD

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

DEPARTMENT OF ENGINEERING SCIENCE
UNIVERSITY OF OXFORD

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

DEPARTMENT OF ENGINEERING SCIENCE

UNIVERSITY OF OXFORD

2.1 Scalar Output Multivariate Functions

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

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

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

Taking Derivatives w.r.t matrixes or vectors:

DEPARTMENT OF ENGINEERING SCIENCE

UNIVERSITY OF OXFORD

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

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

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

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

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

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

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

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

Taking Derivatives w.r.t matrixes or vectors:

DEPARTMENT OF ENGINEERING SCIENCE
UNIVERSITY OF OXFORD

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

$$f(\mathbf{w}) = \frac{1}{2}\mathbf{x}^\top H\mathbf{x}, \qquad \nabla_{\mathbf{x}} f(\mathbf{w}) = ?$$

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

Taking Derivatives w.r.t matrixes 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 matrixes or vectors:

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

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

2.3 Taylor Expansions

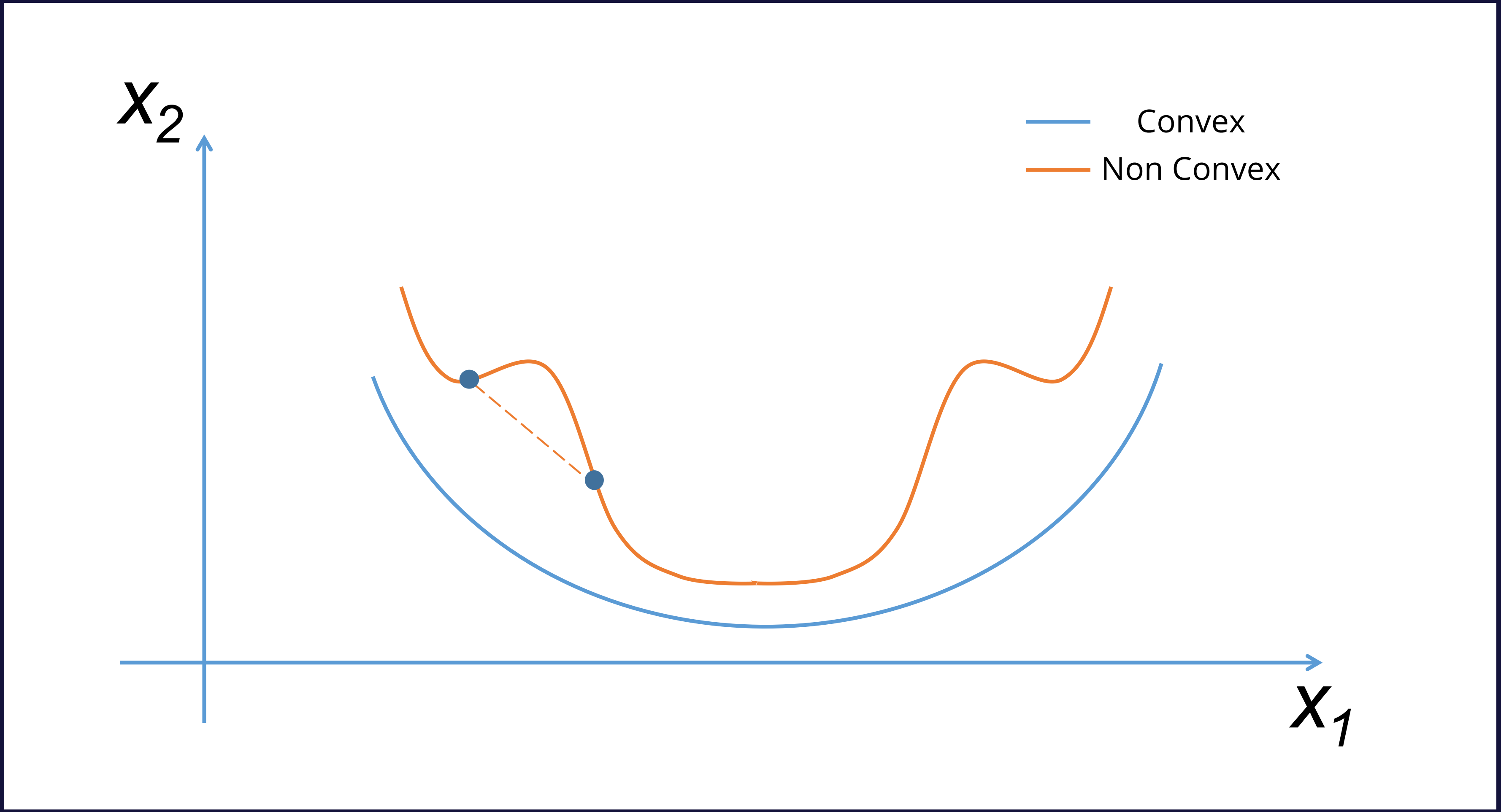
A function f at a a point $\mathbf{w}_t$ can be approximated by its **first** order Taylor expansion around this point:

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

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

2.4 Convex Function

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

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

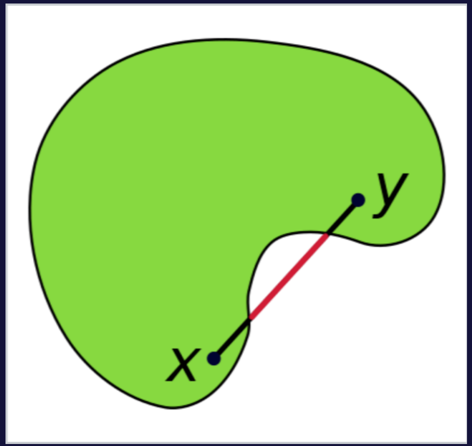
A function f is **strictly convex** if for any $\mathbf{x}, \mathbf{y}$:

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

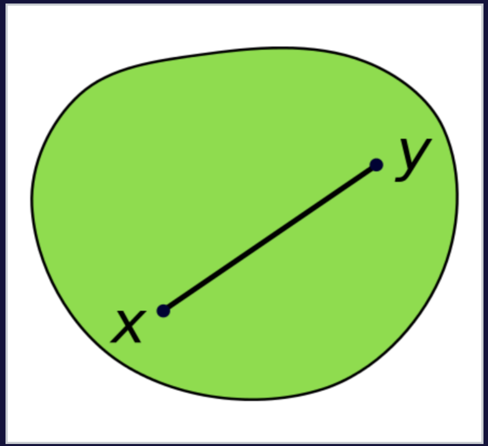
A twice differentiable convex function will have a positive semi-definite Hessian. A twice differentiable strictly convex function will have a positive definite Hessian.

Convex Sets

A



B



Which sets are convex?

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

Convex Sets

3.6 Convex Set

A set Ω 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 Ω 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.

Lipschitz Continuity

A function f is C-Lipschitz over a set Ω 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 ℓ_1 norm, or:

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

Alternatively:

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

Smoothness

A function f is β-Smooth over a set Ω 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 ℓ_1 norm:

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

Alternatively:

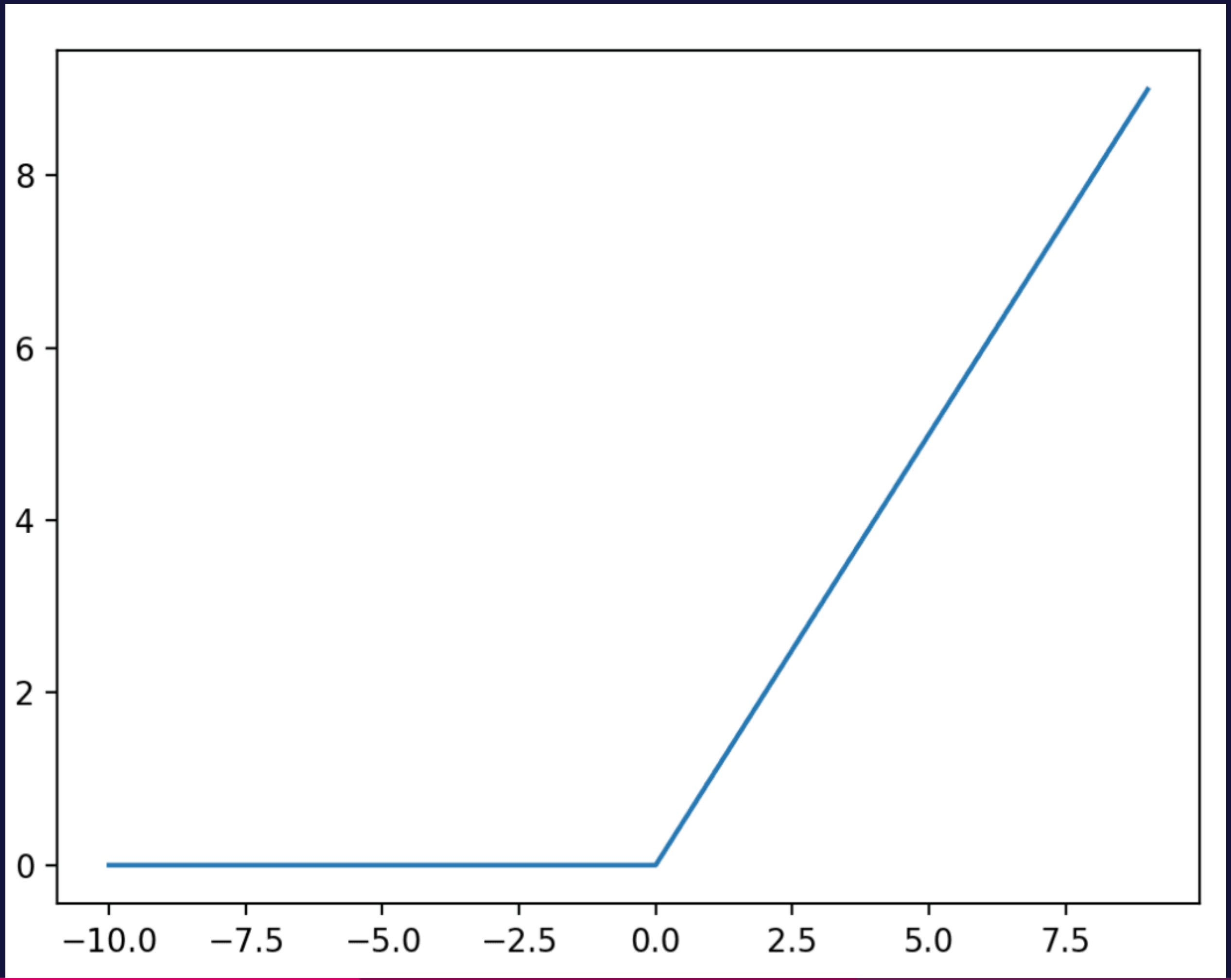
$$\forall\, \mathbf{x}, \mathbf{y} \in \Omega, \;\; |f(\mathbf{x}) - f(\mathbf{y}) - \nabla f(\mathbf{y})^\top(\mathbf{x} - \mathbf{y})| \leq \frac{\beta}{2}\|\mathbf{x} - \mathbf{y}\|^2.$$

Finally if f is twice differentiable, then f is β-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

DEPARTMENT OF ENGINEERING SCIENCE
UNIVERSITY OF OXFORD

Optimisation Standard Set Up

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

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

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

Focus of this Lecture:

DEPARTMENT OF ENGINEERING SCIENCE

UNIVERSITY OF OXFORD

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.

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

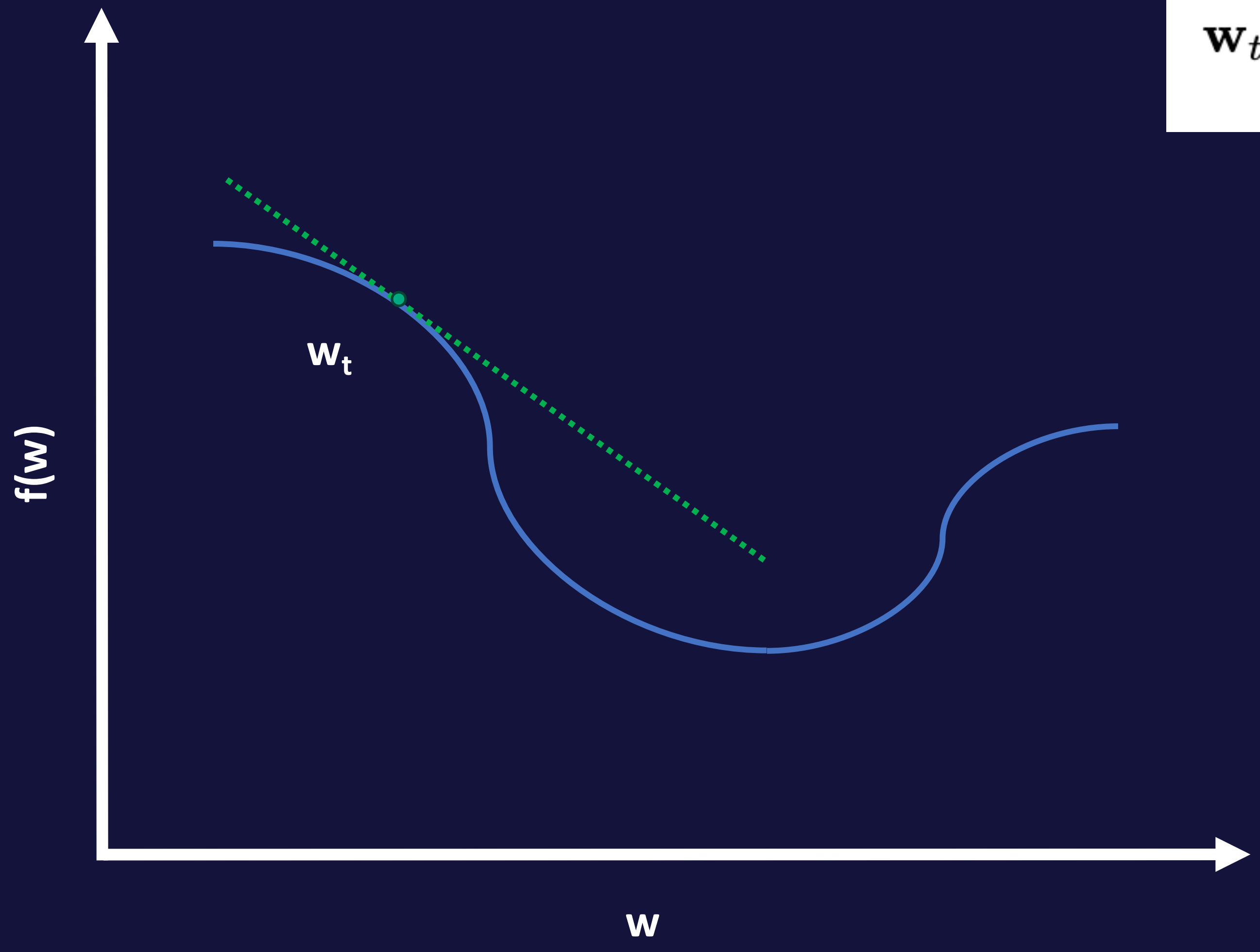
Optimisation Algorithms

Types of Optimiser

- **Gradient Free** – Only use 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 subset of training data set

+ Many others

Gradient Descent (GD)



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

Gradient Descent

5.1 Gradient Descent

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

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

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

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

Provable Progress

DEPARTMENT OF ENGINEERING SCIENCE
UNIVERSITY OF OXFORD

Theorem 5.1. *Let us assume f is convex and smooth with constant β 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 β 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. $\square$

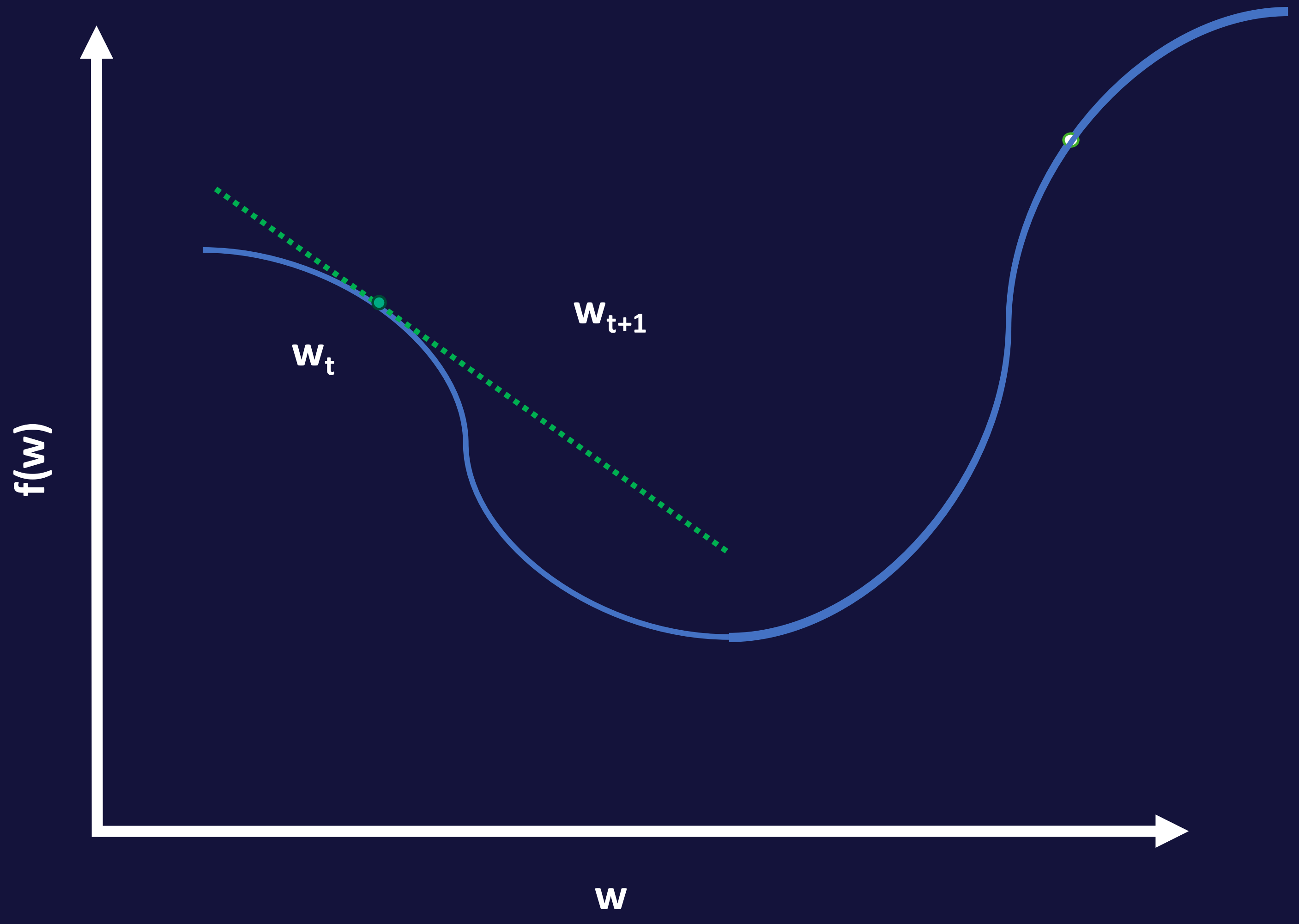
5.3 What if we don't know what the smoothness constant is?

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

Basic idea: start from a large stepsize, according to some condition, either decrease the stepsize or accept the stepsize and execute the update.

Armijo-Goldstein Conditions

The Armijo-Goldstein conditions, also known simply as the Armijo condition, focus on ensuring sufficient decrease in the function value. This condition is defined as follows:

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f(x_k)^T p_k$$

where:

- α_k is the step size.
- p_k is the search direction.
- $f(x)$ is the objective function.
- $\nabla f(x_k)$ is the gradient of the function at x_k.
- c_1 is a constant such that $0 < c_1 < 1$, typically a small number like 0.01.

The Armijo condition ensures that the step size α_k results in a sufficient decrease of the function f, relative to the decrease predicted by the first-order Taylor expansion.

Line search

Basic idea: start from a large stepsize, according to some condition, either decrease the stepsize or accept the stepsize and execute the update.

Wolfe Conditions

The Wolfe conditions are a set of two criteria: the Armijo condition (sufficient decrease condition) and the curvature condition. The Wolfe conditions can be stated as follows:

1. **Sufficient Decrease (same as Armijo condition):**

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f(x_k)^T p_k$$

2. **Curvature Condition:**

$$\nabla f(x_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f(x_k)^T p_k$$

   where c_2 is a constant such that $c_1 < c_2 < 1$, usually closer to 1, like 0.9.

The curvature condition ensures that the derivative in the direction of the step is reduced sufficiently, indicating that the minimum is not just in the immediate vicinity but also somewhat aligned in the search direction, which prevents excessively small updates and zigzagging.

Secord Order Methods

Second Order Methods

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

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

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)

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

Gradient Descent

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

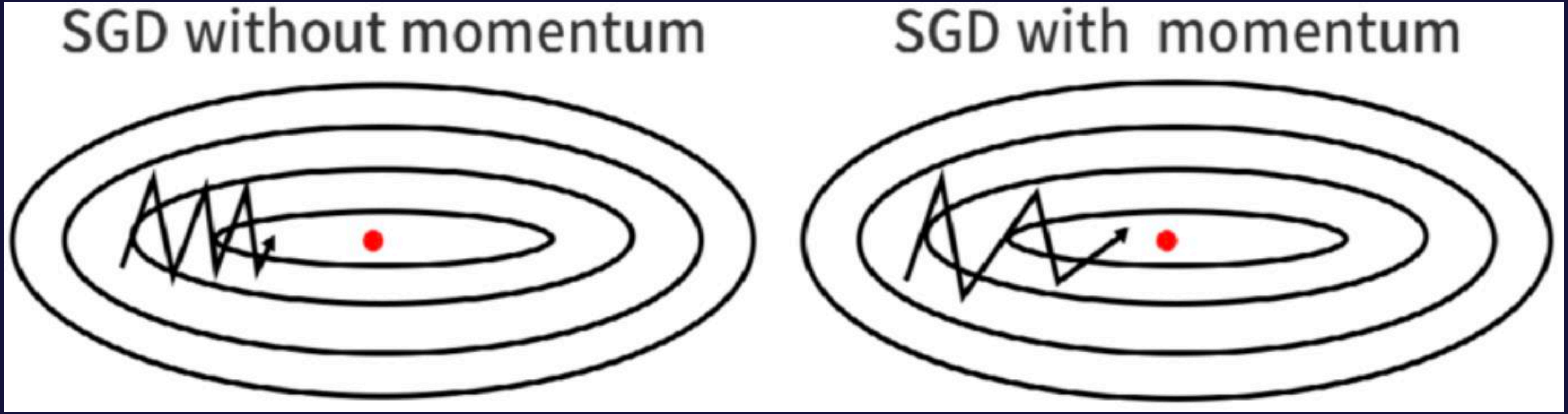
$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \nabla f(\mathbf{w}_t).$$

Stochastic Gradient Descent

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

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

Momentum



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

Momentum methods

Momentum Update Rule

Momentum method is designed to accelerate the convergence of stochastic gradient descent by incorporating the 'momentum' of past updates. Here are the steps for the Momentum update rule:

1. **Velocity Update:**

$$v_{t+1} = \mu v_t - \eta \nabla f(w_t)$$

   - v_t: Velocity at iteration t.
   - μ: Momentum coefficient, typically between 0.9 and 0.99.
   - η: Learning rate.
   - $\nabla f(w_t)$: Gradient of the function f with respect to w at iteration t.

2. **Parameter Update:**

$$w_{t+1} = w_t + v_{t+1}$$

   - w_{t+1}: Updated parameter vector.

This update rule allows to build up speed in directions with persistent gradient, smoothing out oscillations and potentially leading to faster convergence.

Momentum methods

Nesterov Accelerated Gradient (NAG) Update Rule

Nesterov momentum is a variation of the traditional momentum method that calculates the gradient of the function at a lookahead position based on the current momentum. It is particularly effective because it makes a more informed update, reducing overshooting and improving convergence rates. Here's how the NAG update rule works:

1. **Lookahead:**

$$w_{\text{lookahead}} = w_t + \mu v_t$$

    - This step computes the "lookahead" position, where you might expect to be in the next step, based on the current momentum.

2. **Velocity Update:**

$$v_{t+1} = \mu v_t - \eta \nabla f(w_{\text{lookahead}})$$

    - $\nabla f(w_{\text{lookahead}})$: Gradient of the function f evaluated at the lookahead position.

3. **Parameter Update:**

$$w_{t+1} = w_t + v_{t+1}$$

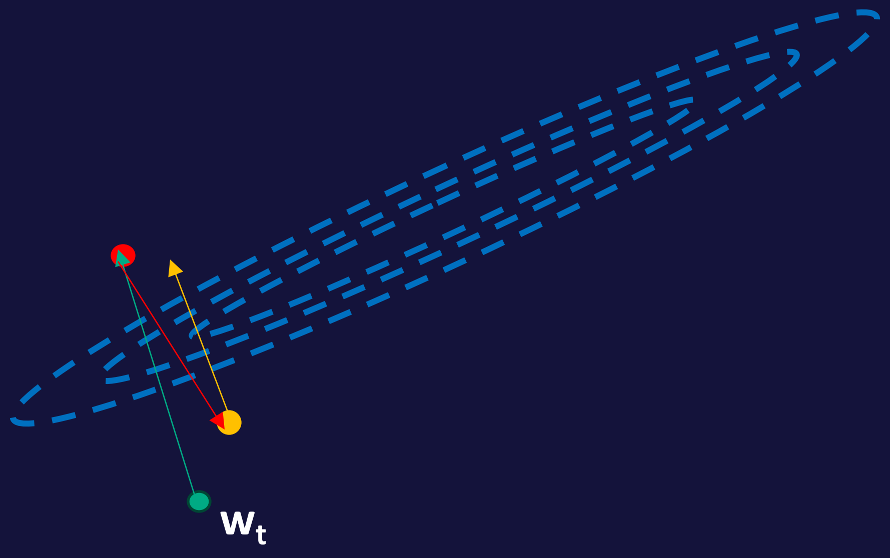
What is causing this zig-zagging?

Dimension with very different scales!

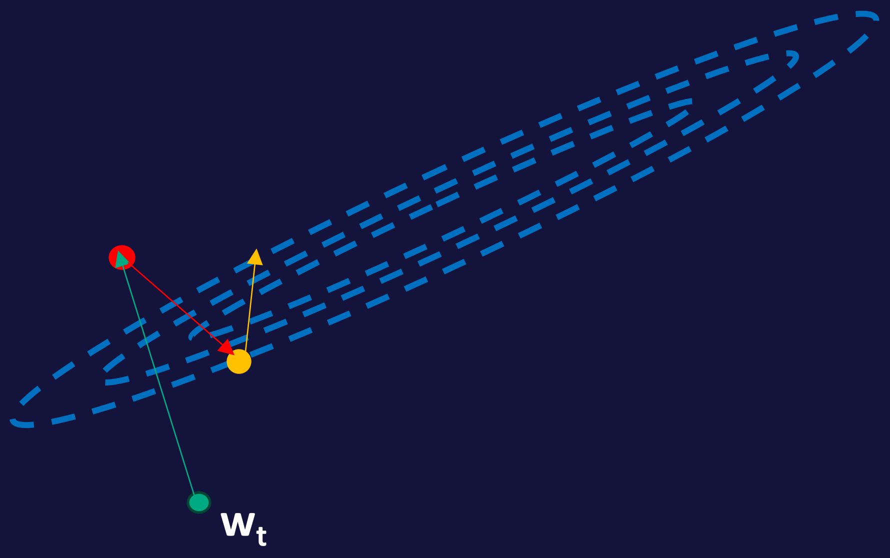
RMSprop

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

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 do we do if we do not have access to the gradient?
- We must use optimisers that only use function values
- many gradient free optimisation algorithms exist in this class we look at the "3 point" method as an illustrative example

Gradient Free Optimisation

DEPARTMENT OF ENGINEERING SCIENCE

UNIVERSITY OF OXFORD

Algorithm 1 Stochastic Three Points Method

Require: η: learning rate
Require: f: objective function
Require: $\mathbf{w}_0$: initial parameter vector
  $t \leftarrow 0$
  while $\mathbf{w}_t$ not converged **do**
    $\boldsymbol{p}_t \sim \mathcal{D}$ for example $\mathcal{D} = \{e_1, e_2, \ldots, e_d\}$
    $\mathbf{w}_{t+1} \leftarrow \text{argmin}\{f(\mathbf{w}_t - \eta \boldsymbol{p}_t), f(\mathbf{w}_t), f(\mathbf{w}_t + \eta \boldsymbol{p}_t)\}$
    $t \leftarrow t + 1$
  end while
  return $\boldsymbol{\theta}_t$

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