Lecture 2: Linear Models

Matthew Caldwell

COMP0088 Introduction to Machine Learning • UCL Computer Science • Autumn 2021

Contents

2.1: Background

COMP0088 Introduction to Machine Learning • UCL Computer Science • Autumn 2021

What is a linear model?

- Linear models are simple and common
- All variables are (for the moment) continuous
- Output is just a weighted sum of inputs (plus optional bias):

  → $y = w_0 + w_1x_1 + w_2x_2 + \ldots w_dx_d$

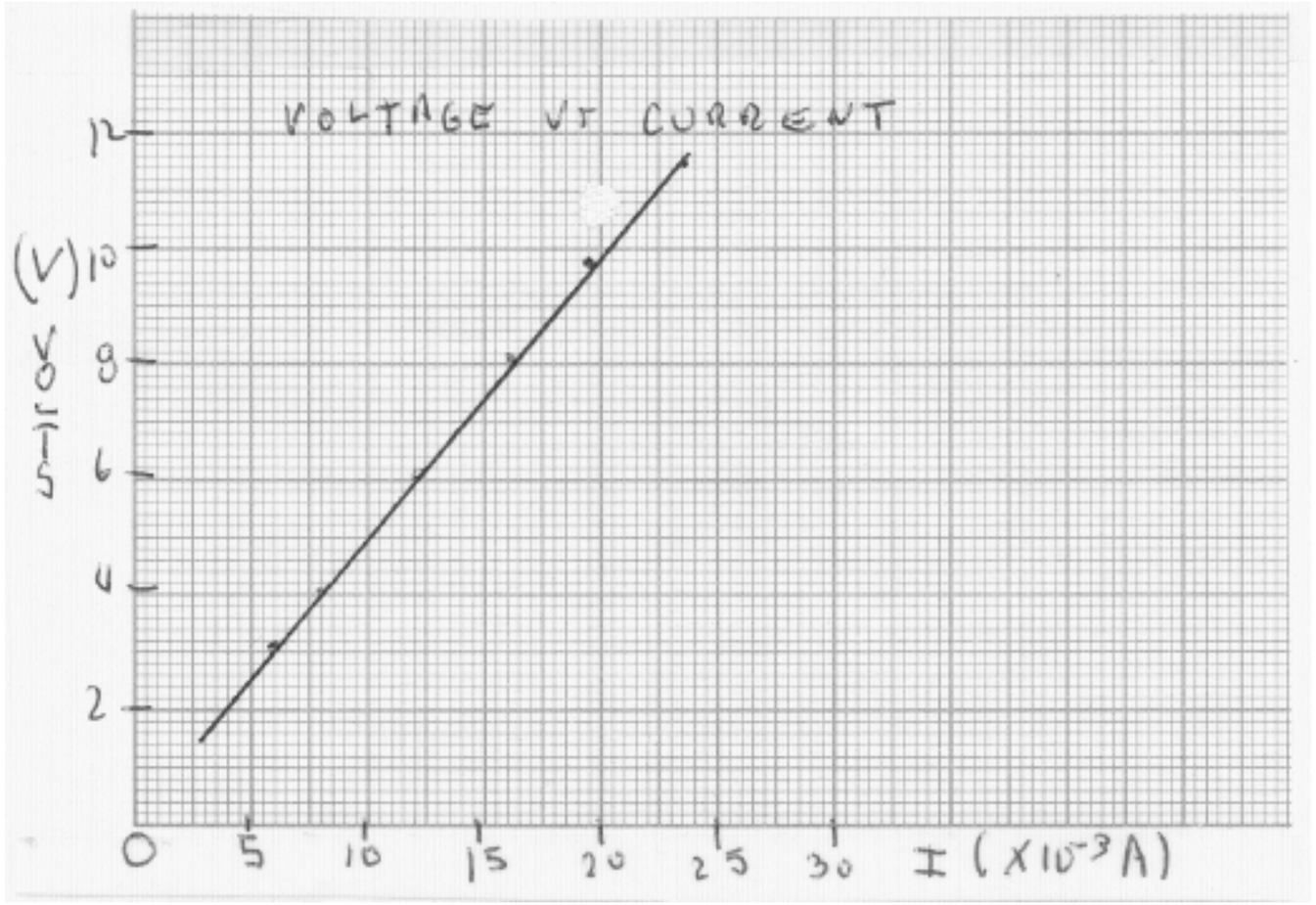
- Question: what are the best weights for given data?

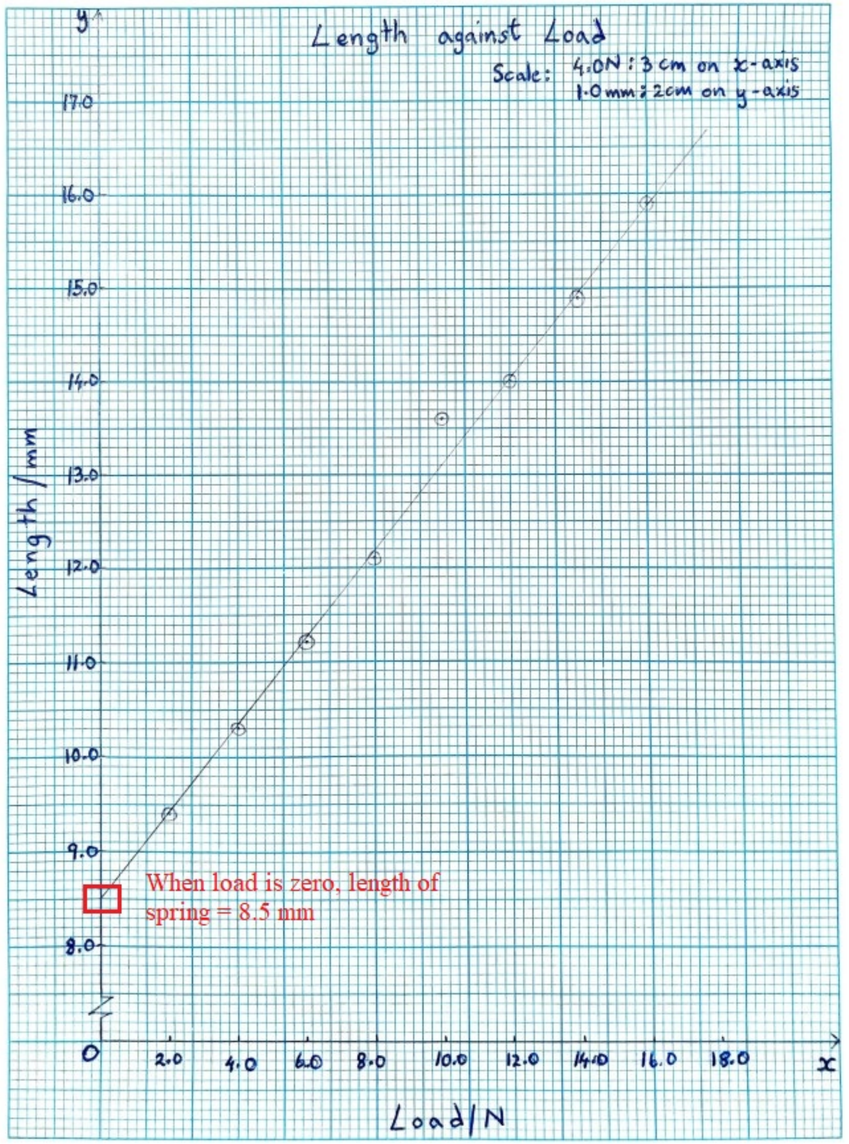
Linear models are everywhere

- Partly because they are easy to frame
- But lots of relationships really **are** linear, to a reasonable approximation
  - → Even complex relationships often have a big linear component, plus some messy other stuff it may be possible to gloss over
- This is why linear algebra is a thing
- Many other problems can be turned into linear ones relatively easily
  - → You've probably seen a lot of **log** plots in the last couple of years
- They involve relatively weak assumptions
- They have a long history and are well-understood
- You can do them in Excel

Examples

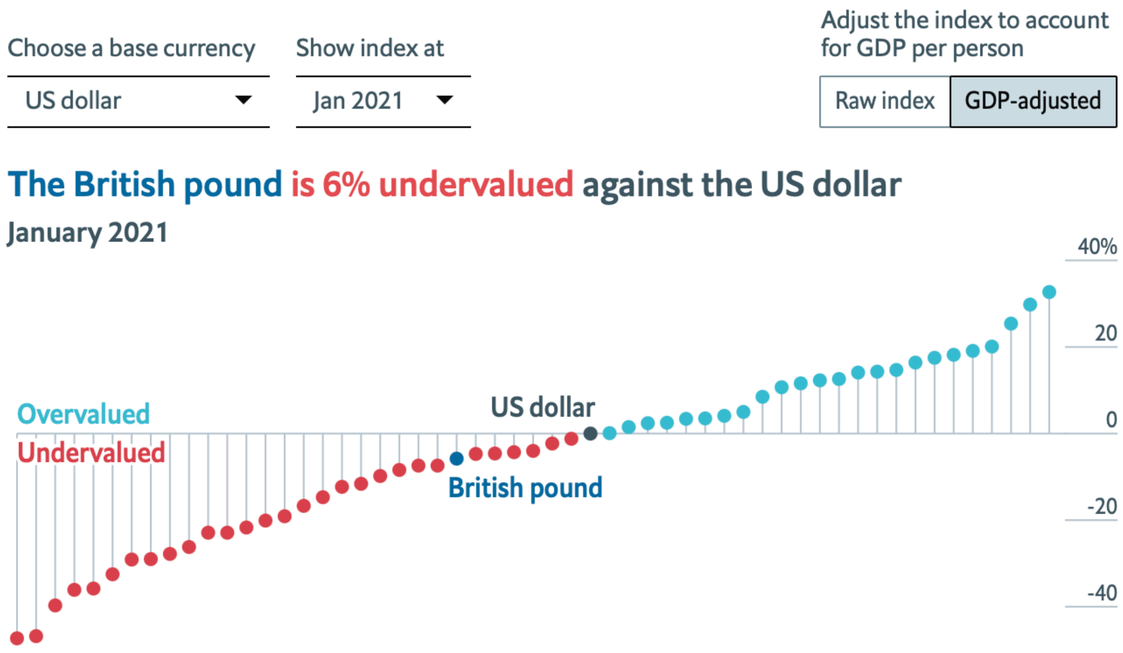
High school physics favourites



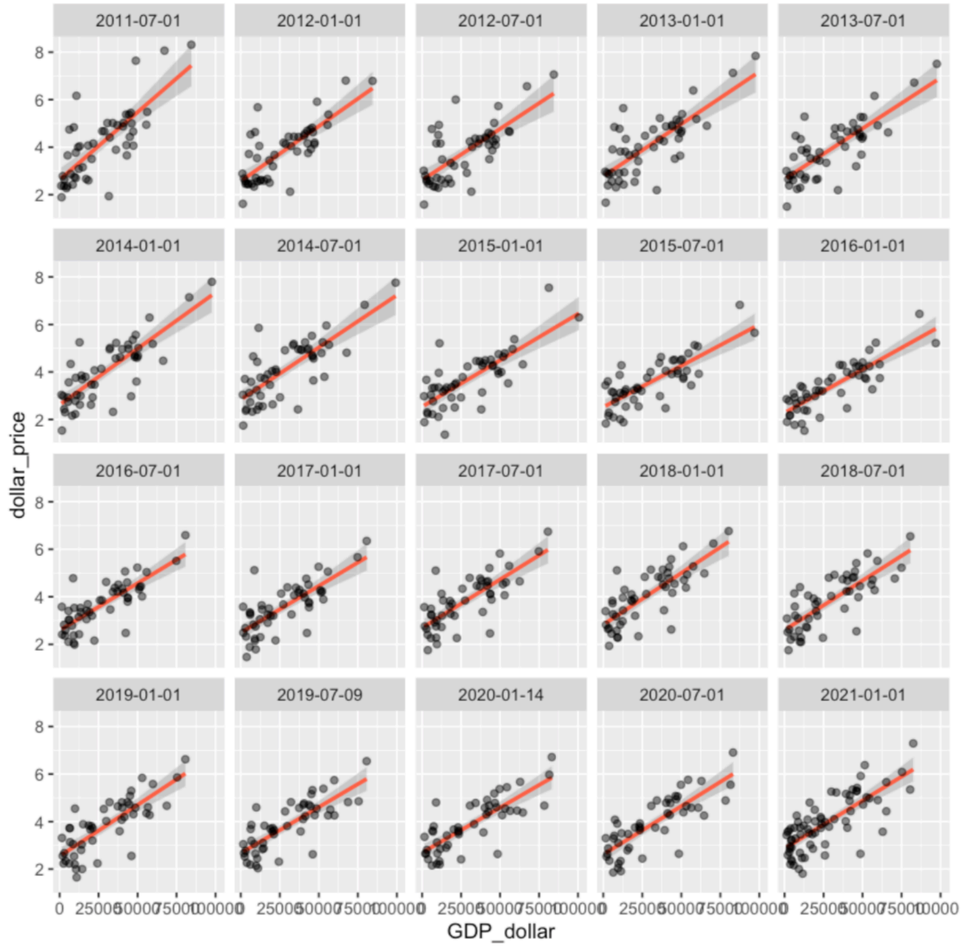


Examples

The Economist Big Mac Index

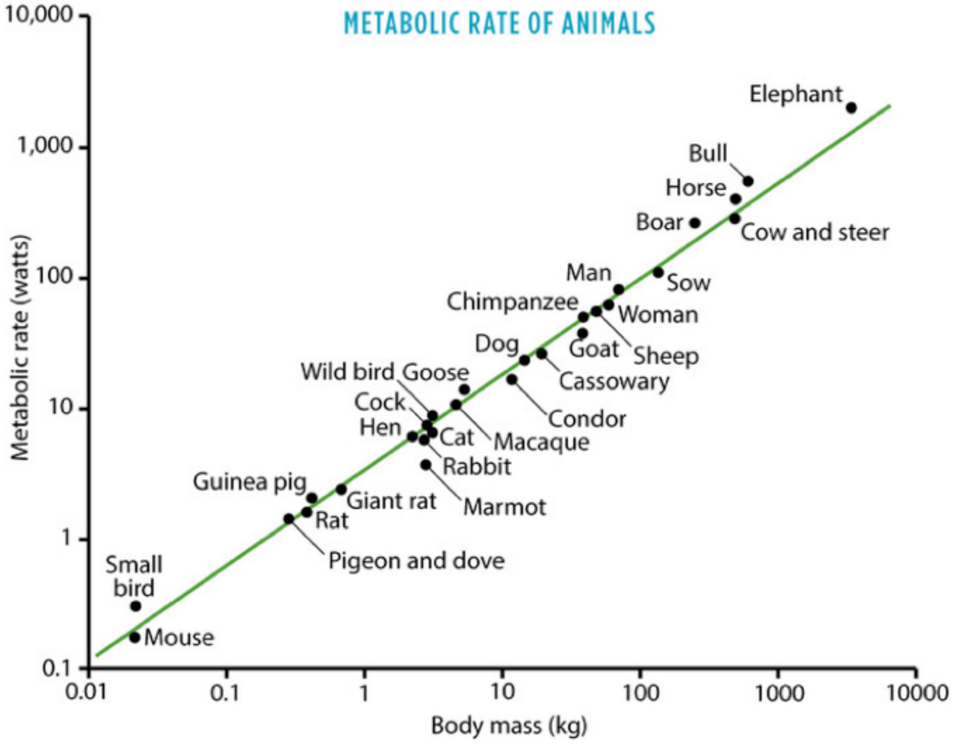


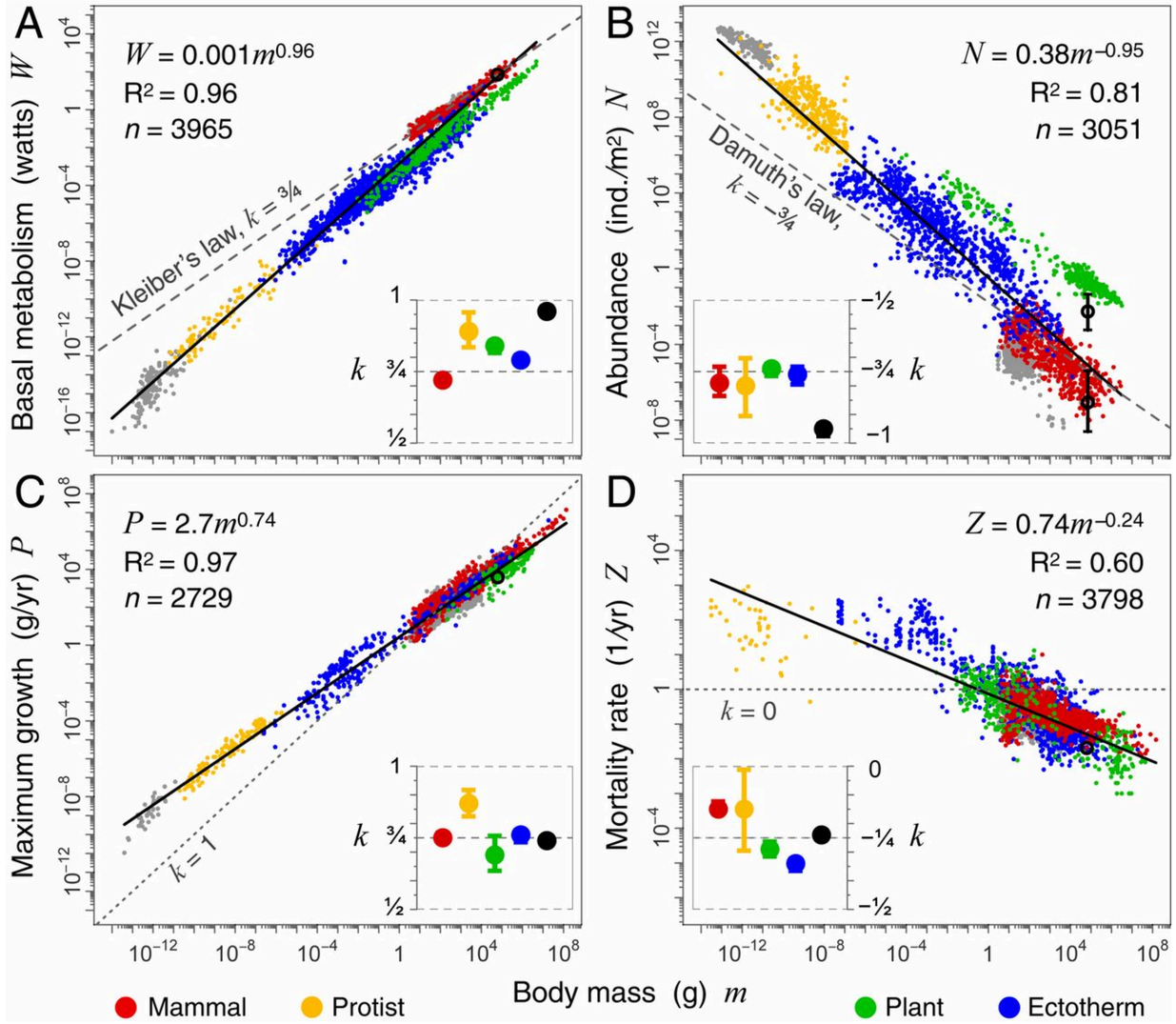
A Big Mac costs 22% less in Britain (US$4.44) than in the United States (US$5.66) at market exchange rates. Based on differences in GDP per person, a Big Mac should cost 17% less. This suggests the pound is 5.8% undervalued



Examples

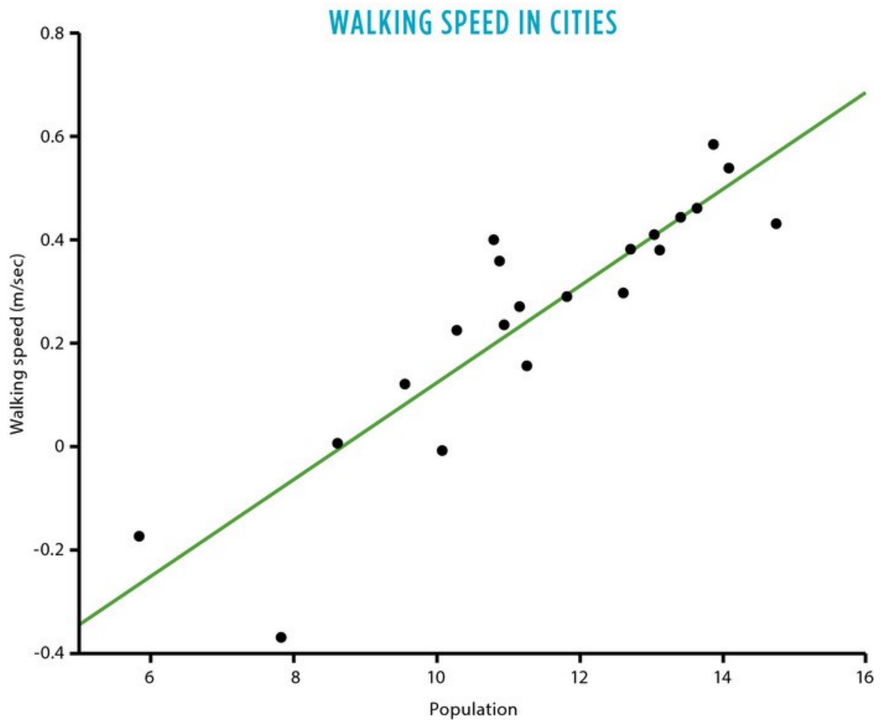
Scaling laws in biology

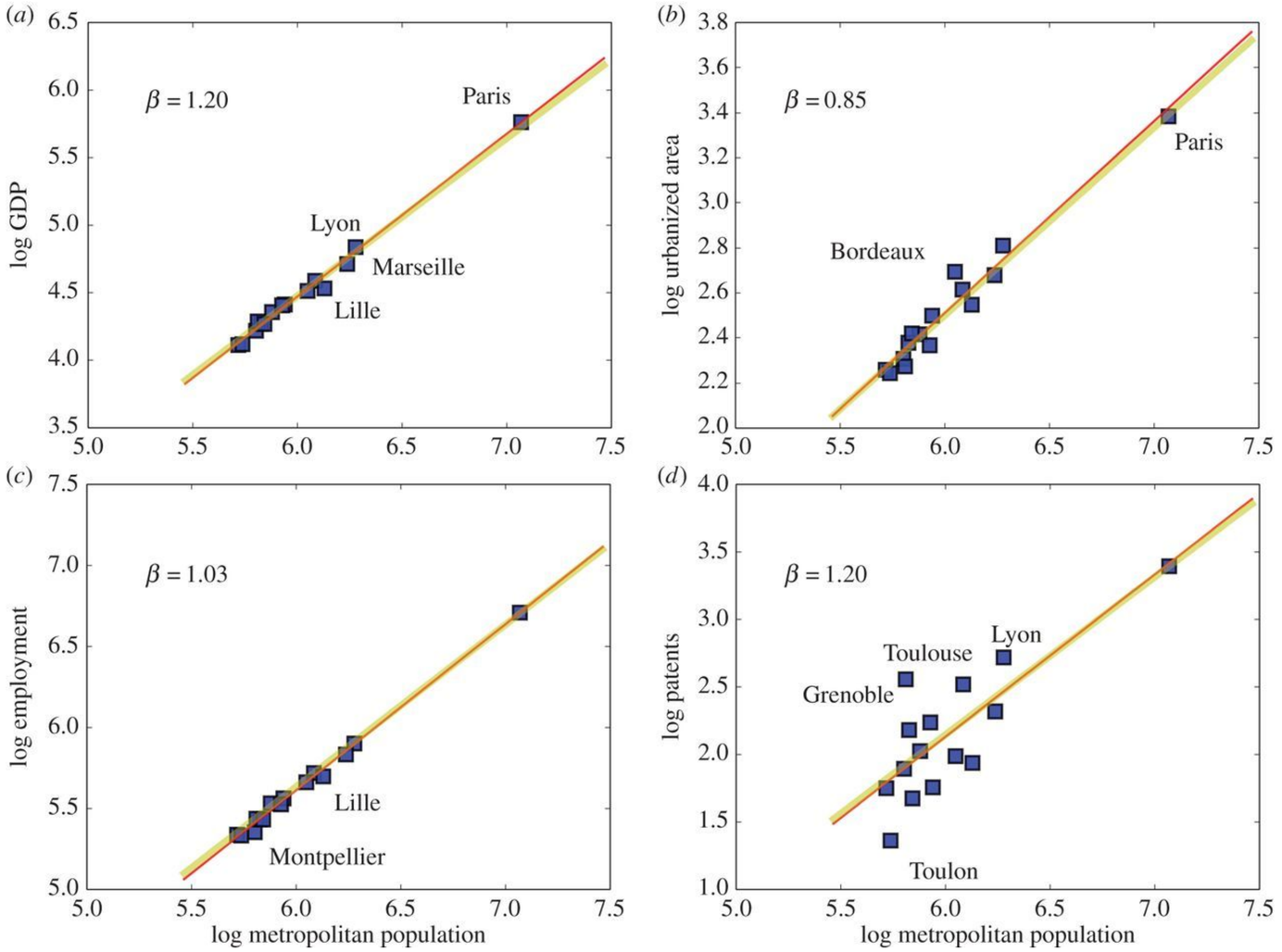




Examples

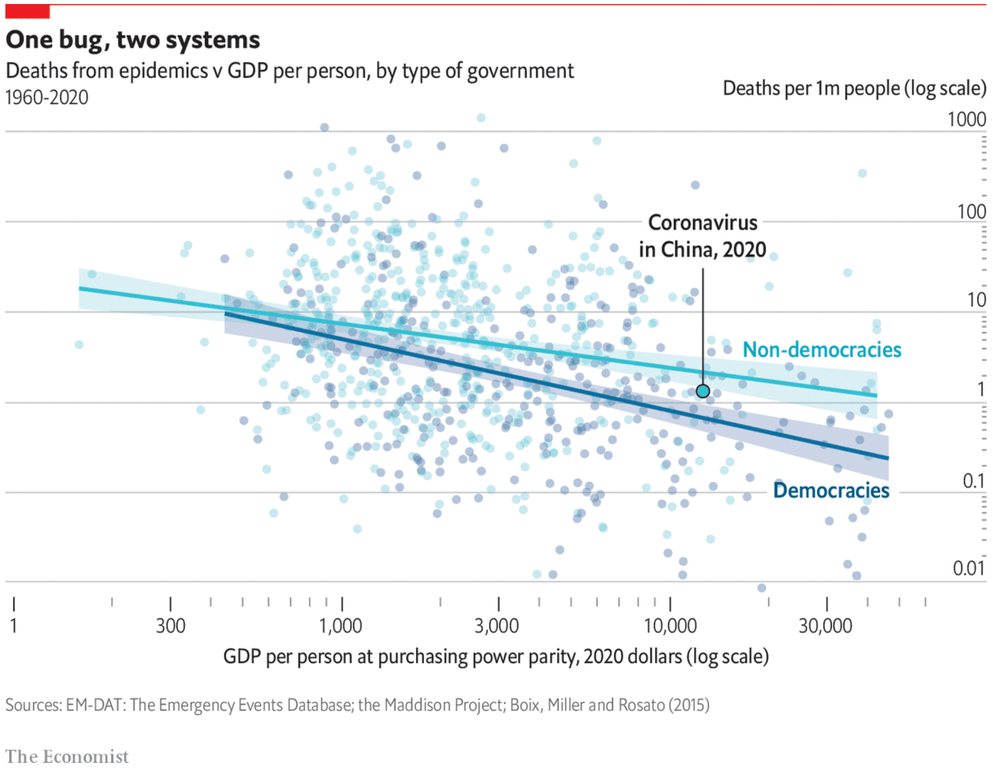
Scaling laws in cities

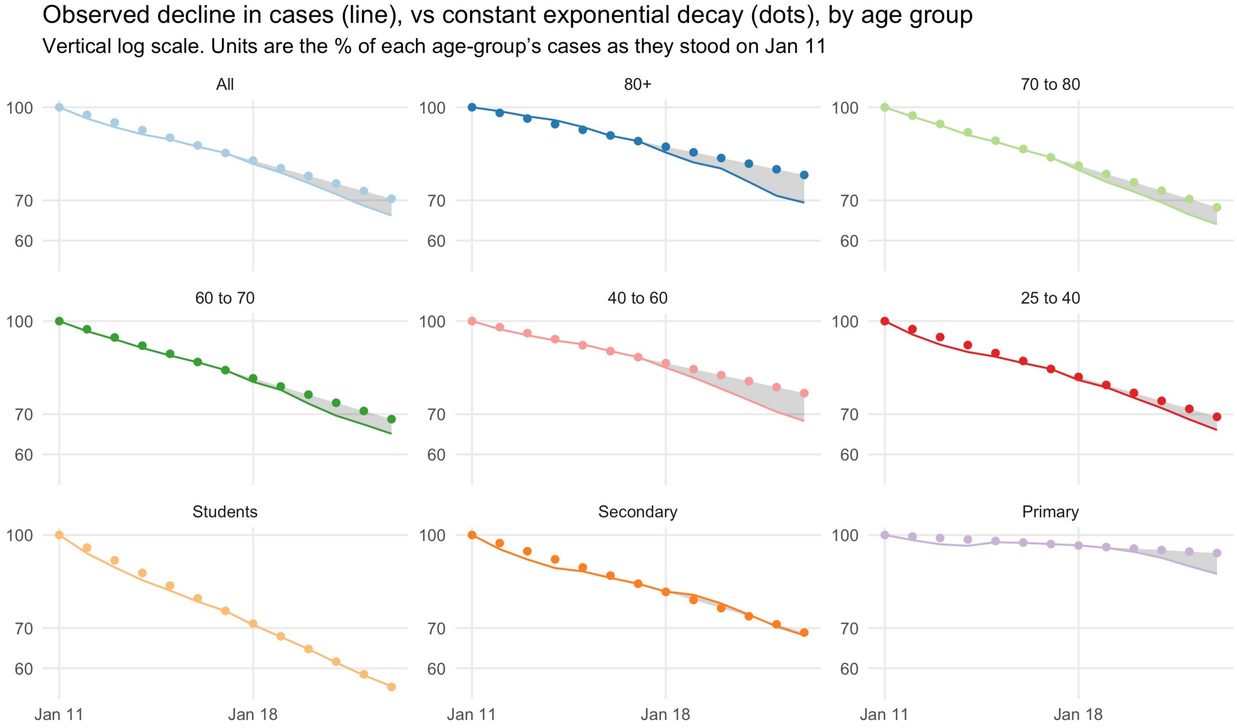




Examples

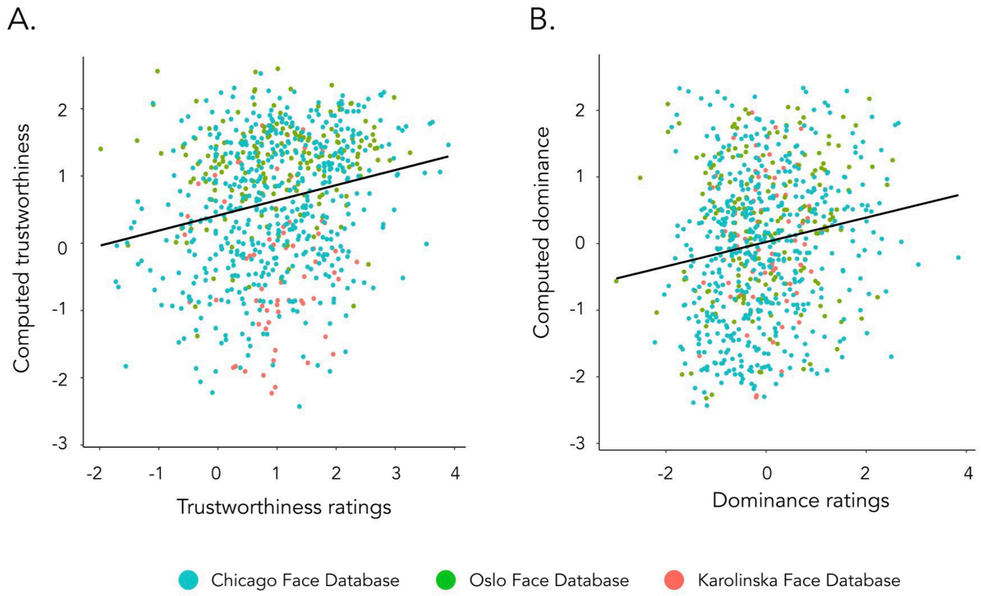
COVID, inevitably

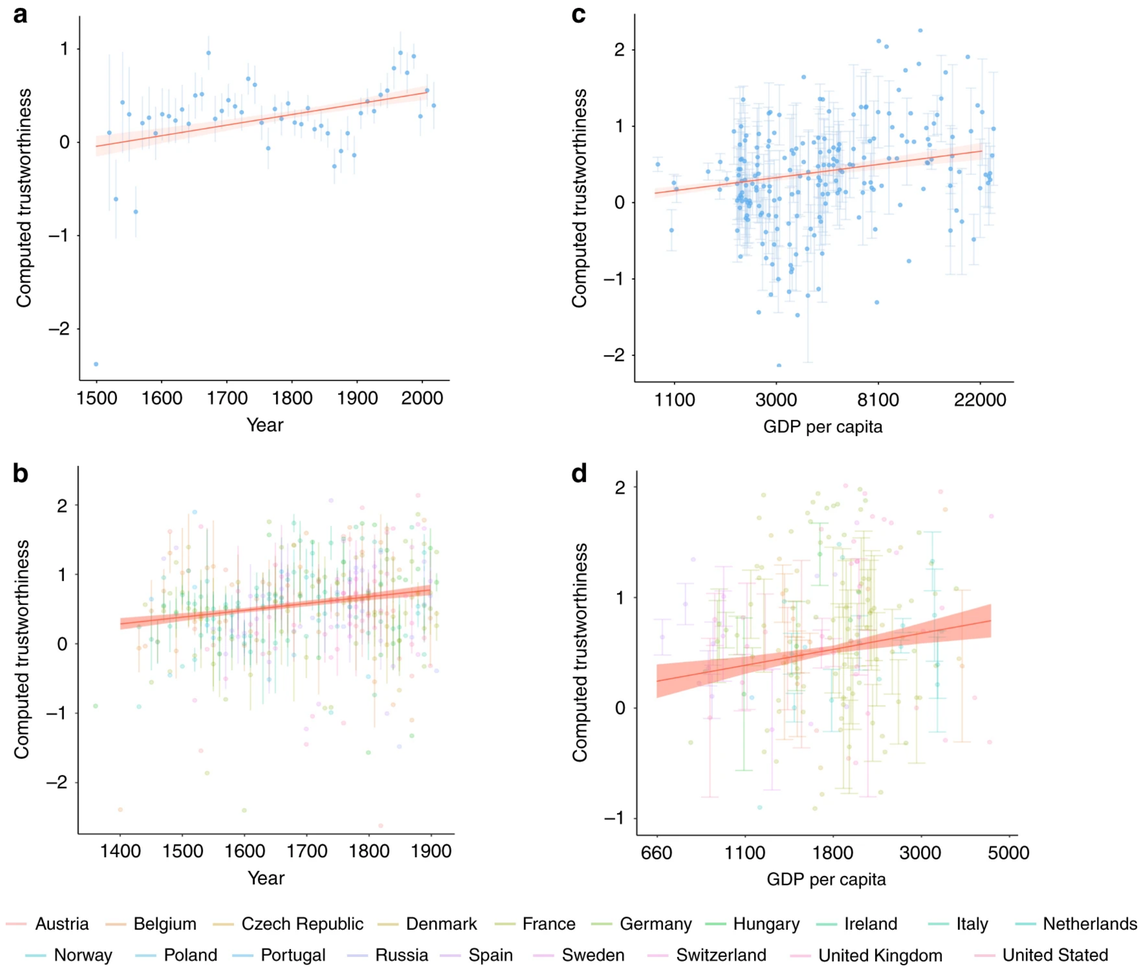




Examples

Terrible Facial Trustworthiness paper somehow published in Nat Comms





Is this even “machine learning” at all?

- Problem and solution long predate digital computers
  → method published by Legendre in 1805, probably known to Gauss earlier
- Solution is closed form and, given certain assumptions, optimal
- Feasible to calculate manually for small datasets
  → involves solving a system of *d* equations, so gets ugly at scale
- Standard statistical tool used routinely by everyone

- Important foundation for later methods
- Extends naturally to more complex, computationally-intensive models

2.2: “Ordinary Least Squares” Regression (part 1)

COMP0088 Introduction to Machine Learning • UCL Computer Science • Autumn 2021

Problem statement

Given *n* training samples $(\mathbf{x}_i, y_i) \qquad i \in \{1, \ldots, n\}$

where each $\mathbf{x}_i \in \mathbb{R}^d = (x_{i,1}, x_{i,2}, ..., x_{i,d})$ and $y_i \in \mathbb{R}$

For the moment we consider only scalar outputs. Method generalises naturally to vector $\mathbf{y}_i$, but is equivalent to having a separate models for each.

Problem statement

Given *n* training samples $(\mathbf{x}_i, y_i) \qquad i \in \{1, \ldots, n\}$

where each $\mathbf{x}_i \in \mathbb{R}^d = (x_{i,1}, x_{i,2}, ..., x_{i,d})$ and $y_i \in \mathbb{R}$

find $\mathbf{w} \in \mathbb{R}^d = (w_1, w_2, ..., w_d)$

such that $\hat{y} = f(\mathbf{x}) = \mathbf{x} \cdot \mathbf{w}$

is a good model of the data — in some to be determined sense.

Design matrix

- Define

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} = \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,d} \end{bmatrix} \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

↓ Feature dimensions ↓

↑ Samples ↑

- Then model for whole training set is compactly expressed as $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w}$

→ We'll be using this data representation a lot throughout the course

Intercept

- Model so far does not include an offset/intercept/bias term
- We can add it explicitly:

$$\hat{\mathbf{y}} = \mathbf{X}\mathbf{w} + b$$

- Alternatively, we can augment the features with a constant $x_0 = 1$ and consider the bias as the corresponding weight w_0

- Alternatively, standardise features and set $b = \bar{\mathbf{y}}$

  → Standardisation is often desirable anyway

Geometric interpretation

- If we consider each training sample as a point in (*d*+1)-space:

$$(\mathbf{x}, y) = (x_1, x_2, \ldots, x_d, y) \in \mathbb{R}^{d+1}$$

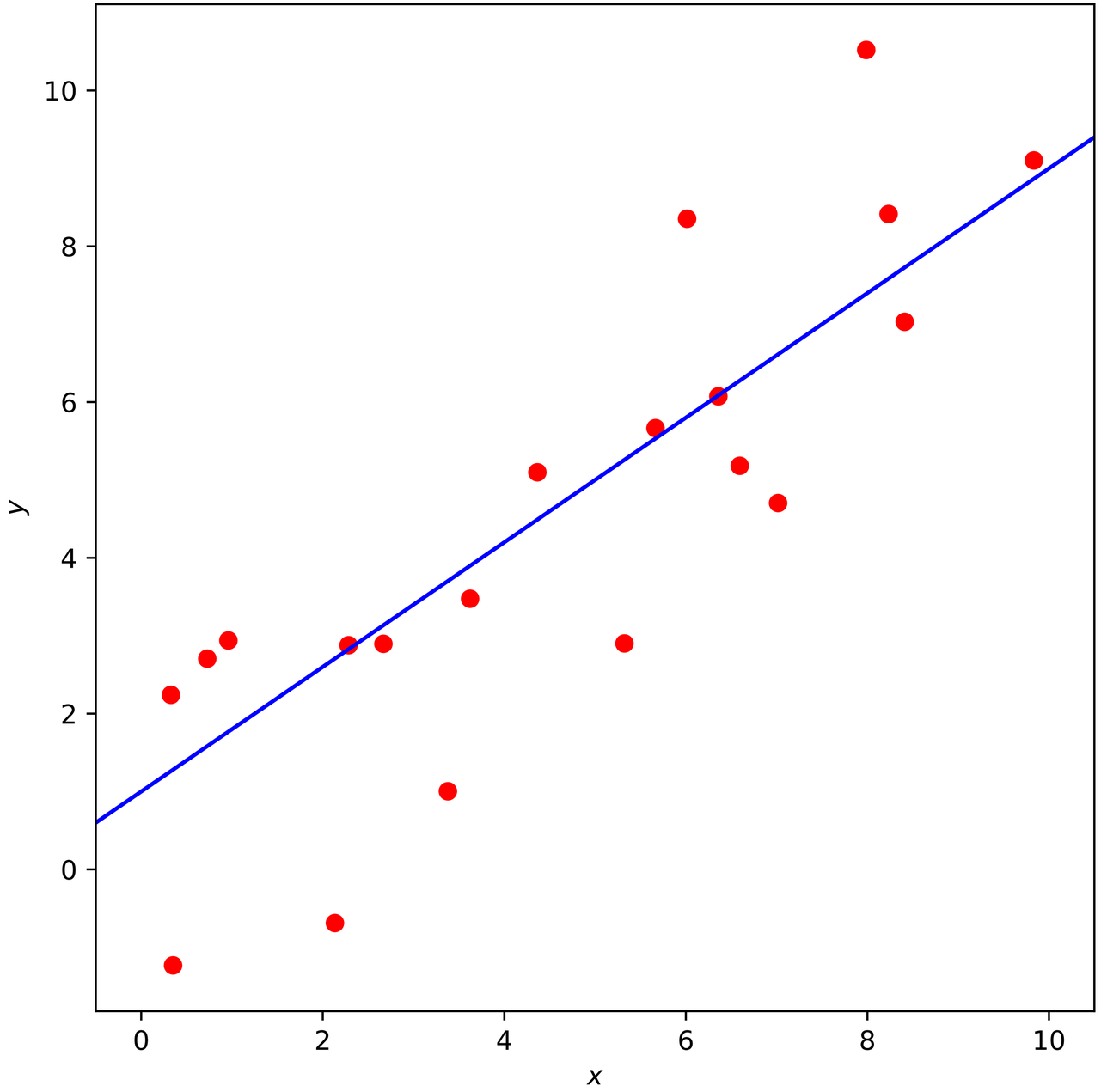
- then the linear model

$$\hat{y} = f(\mathbf{x}) = \mathbf{x} \cdot \mathbf{w}$$

  defines a *d*-dimensional **hyperplane** through that (*d*+1)-space

This is easy to visualise for d=1

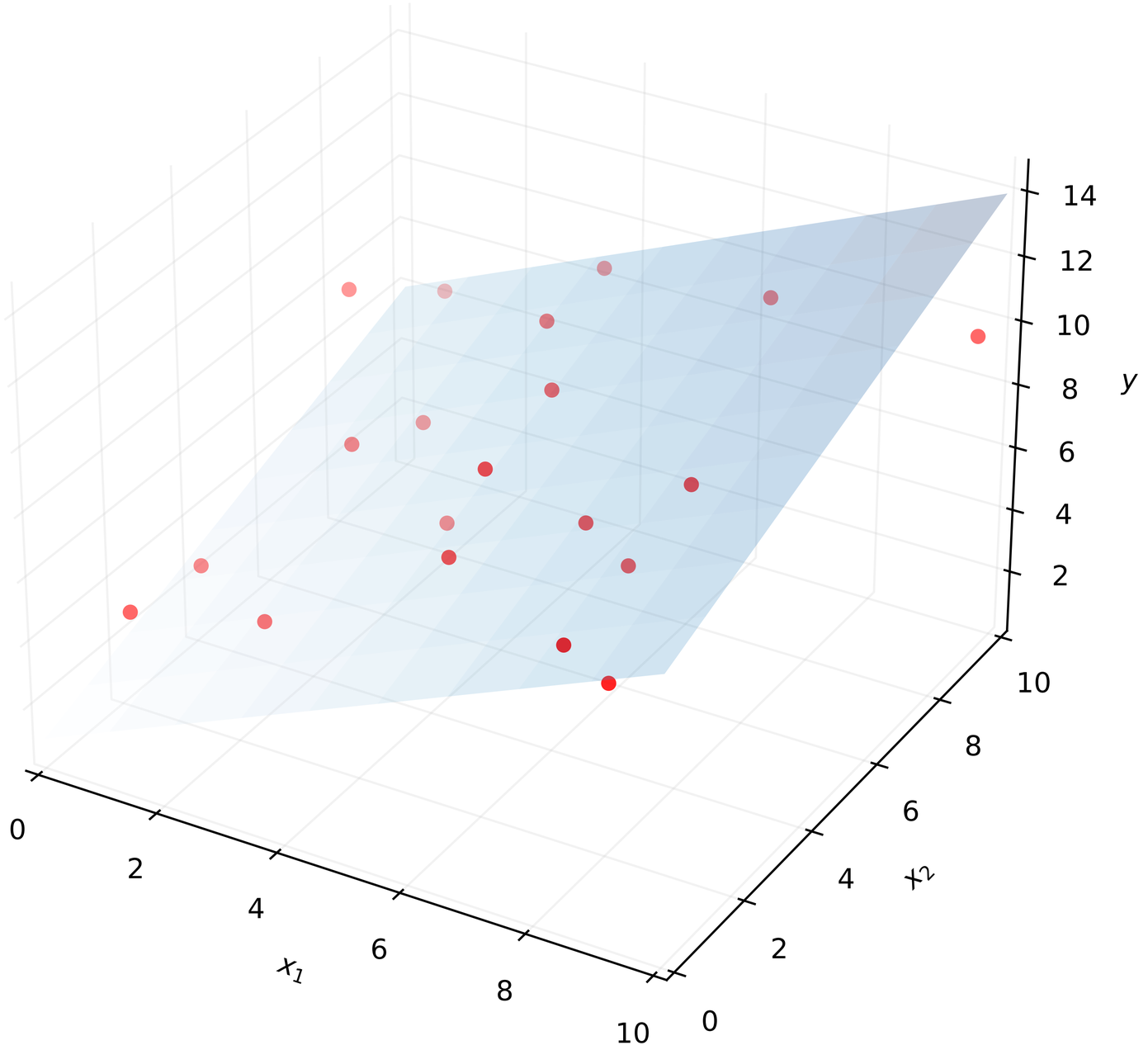
→ we're all familiar with this kind of trend line



It's a little murkier for d=2, because we need to project the (d+1) space onto a flat surface for viewing, but ok

d>2 is pretty much impossible

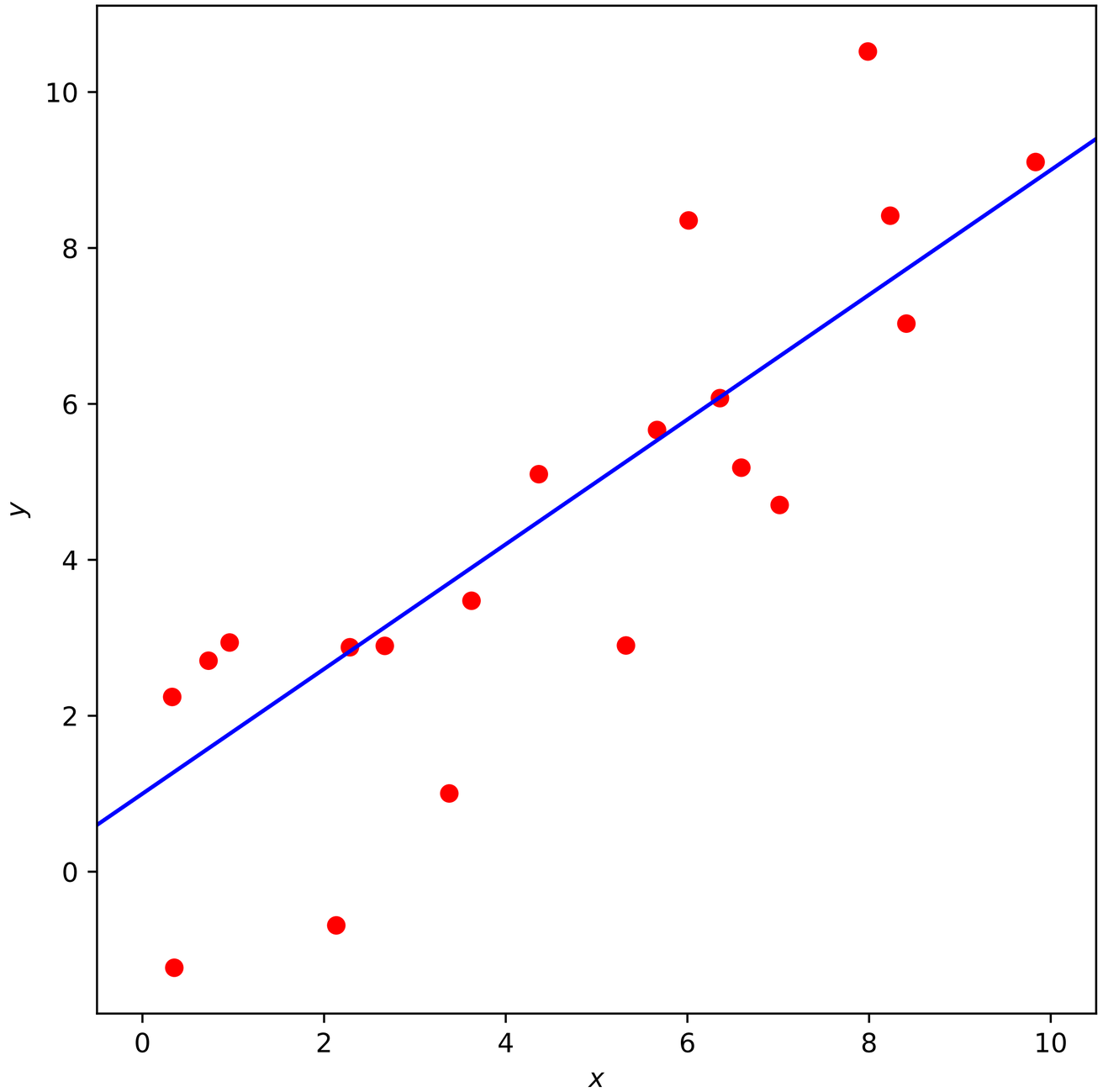
But really, these two cases are enough to understand the model



But *which* hyperplane?

- Any **w** defines a hyperplane, so how do we choose?
- The **best**, obviously!
- By what criteria?
- We’re thinking geometrically, so **distance** seems like a plausible choice.

In particular, the distance between the **true** value for each *y* and the value the model predicts

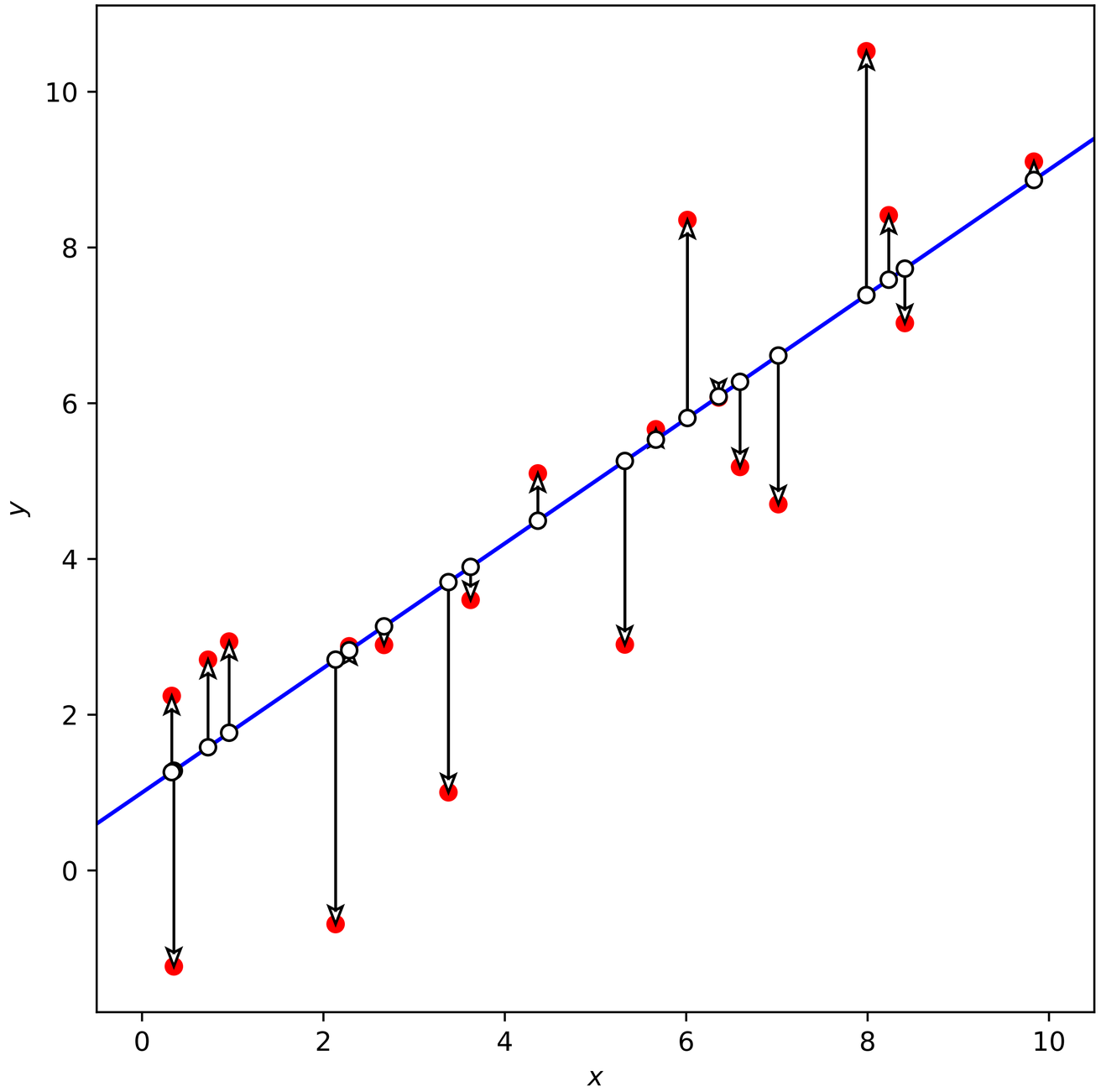


i.e.:

$$y - \hat{y}$$

(often known as the **residual**)

The true value may be above or below the line, so the residual is **signed**. We don't want big errors one way to cancel big errors the other, so we take a norm.



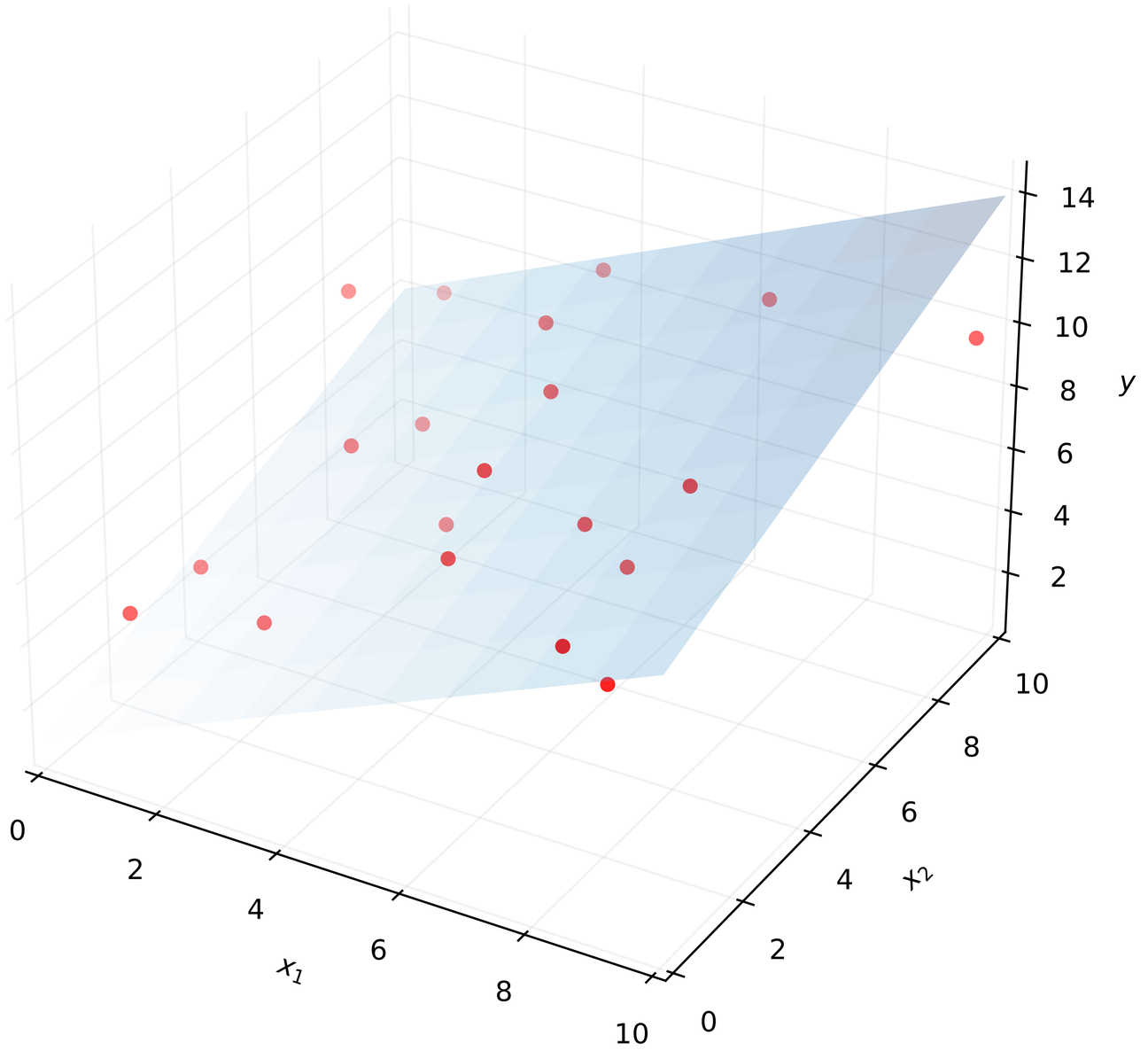
While other choices are possible, it turns out to be mathematically convenient to use the **Euclidean** norm:

$$||y - \hat{y}||$$

Or actually the **square** of that:

$$||y - \hat{y}||^2$$

(We'll revisit this seemingly-arbitrary choice shortly)

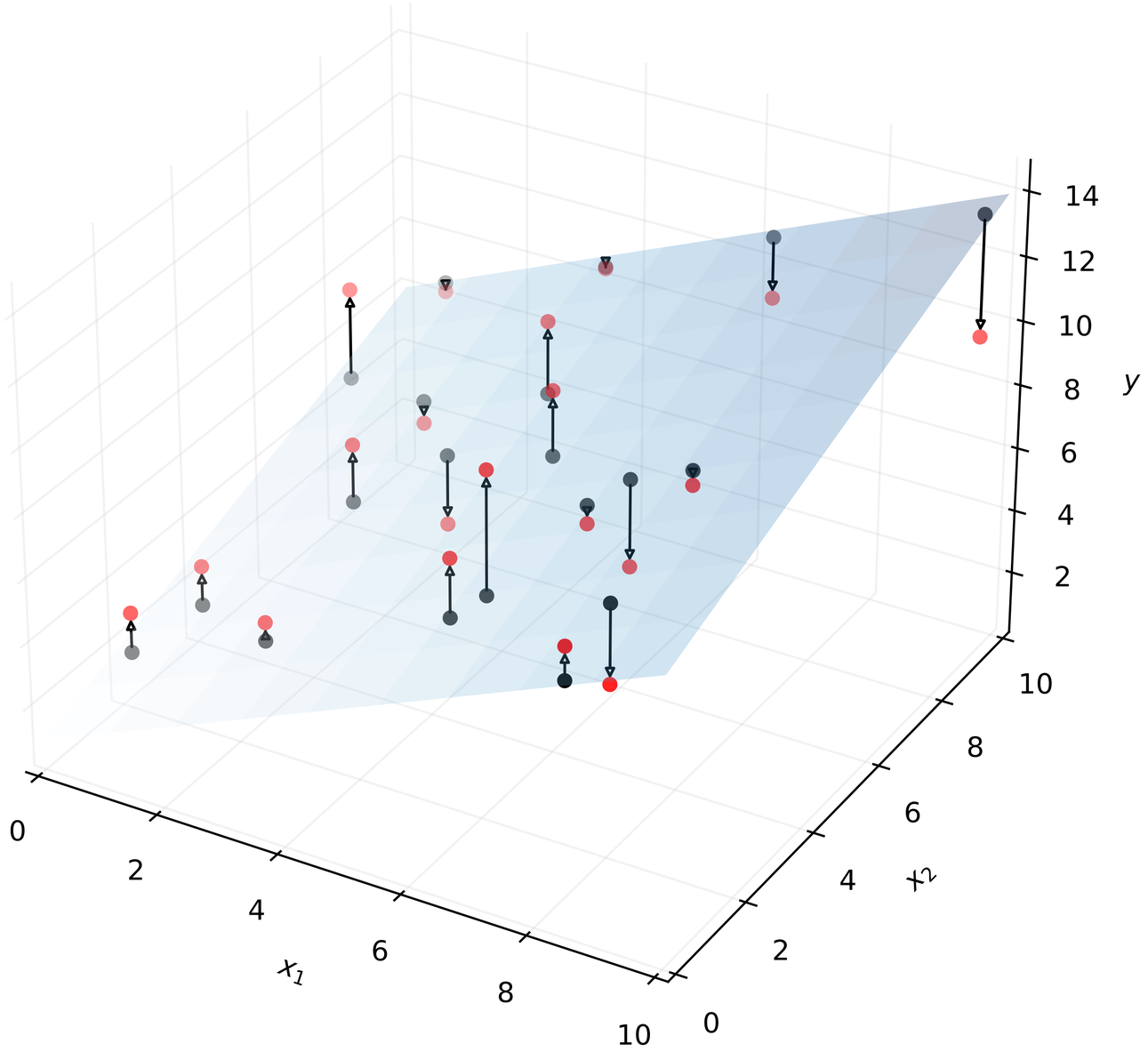


All the points in the training set have their own residual, so we need to aggregate:

$$\sum_{i=1}^{n} \|y_i - \hat{y}_i\|^2$$

Equivalently:

$$\sum_{i=1}^{n} \|y_i - \mathbf{x}_i \cdot \mathbf{w}\|^2$$



Baby's first loss function

- Equivalently:

$$\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$$

- This is our **loss**. All we have to do is minimise it and we're done:

$$\mathbf{w}^\star = \underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$$

- As luck* would have it, this problem is convex and differentiable.

*not actually luck

Notational sidebar

- We need to take derivative with respect to **w**, which is a vector
- What does that even mean?
- Take partial derivatives element by element
- Represent as a vector of partials, matching the shape of the original
- Denote by symbol ∇
- We'll usually call it a "gradient" (it's really a special case of the Jacobian matrix)

$$\nabla_{\mathbf{w}} L = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \\ \vdots \\ \frac{\partial L}{\partial w_m} \end{bmatrix}$$

Useful identities

- Matrix multiplication not commutative but combines with transpose:

$$\mathbf{a}^\mathsf{T}\mathbf{b} = \mathbf{b}^\mathsf{T}\mathbf{a}$$

Ordinary vector inner product

$$(\mathbf{AB})^\mathsf{T} = \mathbf{B}^\mathsf{T}\mathbf{A}^\mathsf{T}$$

- Standard gradients for linear & quadratic forms:

$$\nabla_\mathbf{x}(\mathbf{b}^\mathsf{T}\mathbf{x}) = \mathbf{b}$$

$$\nabla_\mathbf{x}(\mathbf{x}^\mathsf{T}\mathbf{A}\mathbf{x}) = 2\mathbf{A}\mathbf{x}$$

A symmetric

Rearrange

$$\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 = (\mathbf{X}\mathbf{w} - \mathbf{y})^\mathsf{T}(\mathbf{X}\mathbf{w} - \mathbf{y})$$

Rearrange

$$\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 = (\mathbf{X}\mathbf{w} - \mathbf{y})^\mathsf{T}(\mathbf{X}\mathbf{w} - \mathbf{y})$$

Expand term by term

Rearrange

$$\begin{aligned}\|\mathbf{X}\mathbf{w}-\mathbf{y}\|^2 &= (\mathbf{X}\mathbf{w}-\mathbf{y})^\mathsf{T}(\mathbf{X}\mathbf{w}-\mathbf{y}) \\ &= (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{X}\mathbf{w} - (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{y} - \mathbf{y}^\mathsf{T}(\mathbf{X}\mathbf{w}) + \mathbf{y}^\mathsf{T}\mathbf{y}\end{aligned}$$

Rearrange

$$\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 = (\mathbf{X}\mathbf{w} - \mathbf{y})^\mathsf{T}(\mathbf{X}\mathbf{w} - \mathbf{y})$$

$$= (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{X}\mathbf{w} - (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{y} - \mathbf{y}^\mathsf{T}(\mathbf{X}\mathbf{w}) + \mathbf{y}^\mathsf{T}\mathbf{y}$$

$$\mathbf{a}^\mathsf{T}\mathbf{b} = \mathbf{b}^\mathsf{T}\mathbf{a}$$

Rearrange

$$
\begin{aligned}
\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 &= (\mathbf{X}\mathbf{w} - \mathbf{y})^\mathsf{T}(\mathbf{X}\mathbf{w} - \mathbf{y}) \\
&= (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{X}\mathbf{w} - (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{y} - \mathbf{y}^\mathsf{T}(\mathbf{X}\mathbf{w}) + \mathbf{y}^\mathsf{T}\mathbf{y} \\
&= (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{X}\mathbf{w} - 2\mathbf{y}^\mathsf{T}(\mathbf{X}\mathbf{w}) + \mathbf{y}^\mathsf{T}\mathbf{y}
\end{aligned}
$$

Rearrange

$$
\begin{aligned}
\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 &= (\mathbf{X}\mathbf{w} - \mathbf{y})^\mathsf{T}(\mathbf{X}\mathbf{w} - \mathbf{y}) \\
&= (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{X}\mathbf{w} - (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{y} - \mathbf{y}^\mathsf{T}(\mathbf{X}\mathbf{w}) + \mathbf{y}^\mathsf{T}\mathbf{y} \\
&= (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{X}\mathbf{w} - 2\mathbf{y}^\mathsf{T}(\mathbf{X}\mathbf{w}) + \mathbf{y}^\mathsf{T}\mathbf{y}
\end{aligned}
$$

Associative

Rearrange

$$
\begin{aligned}
\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 &= (\mathbf{X}\mathbf{w} - \mathbf{y})^\mathsf{T}(\mathbf{X}\mathbf{w} - \mathbf{y}) \\
&= (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{X}\mathbf{w} - (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{y} - \mathbf{y}^\mathsf{T}(\mathbf{X}\mathbf{w}) + \mathbf{y}^\mathsf{T}\mathbf{y} \\
&= (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{X}\mathbf{w} - 2\mathbf{y}^\mathsf{T}(\mathbf{X}\mathbf{w}) + \mathbf{y}^\mathsf{T}\mathbf{y} \\
&= (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{X}\mathbf{w} - 2(\mathbf{y}^\mathsf{T}\mathbf{X})\mathbf{w} + \mathbf{y}^\mathsf{T}\mathbf{y}
\end{aligned}
$$

Rearrange

$$
\begin{aligned}
\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 &= (\mathbf{X}\mathbf{w} - \mathbf{y})^\mathsf{T}(\mathbf{X}\mathbf{w} - \mathbf{y}) \\
&= (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{X}\mathbf{w} - (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{y} - \mathbf{y}^\mathsf{T}(\mathbf{X}\mathbf{w}) + \mathbf{y}^\mathsf{T}\mathbf{y} \\
&= (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{X}\mathbf{w} - 2\mathbf{y}^\mathsf{T}(\mathbf{X}\mathbf{w}) + \mathbf{y}^\mathsf{T}\mathbf{y} \\
&= (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{X}\mathbf{w} - 2(\mathbf{y}^\mathsf{T}\mathbf{X})\mathbf{w} + \mathbf{y}^\mathsf{T}\mathbf{y}
\end{aligned}
$$

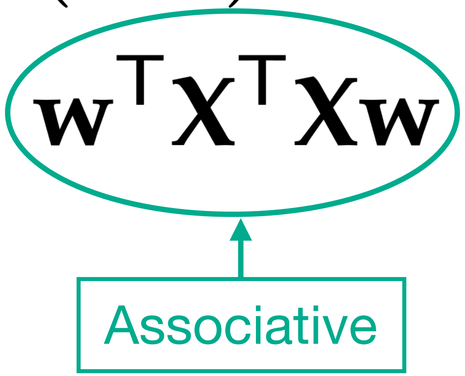
$$(\mathbf{A}\mathbf{B})^\mathsf{T} = \mathbf{B}^\mathsf{T}\mathbf{A}^\mathsf{T}$$

Rearrange

$$
\begin{aligned}
\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 &= (\mathbf{X}\mathbf{w} - \mathbf{y})^\mathsf{T}(\mathbf{X}\mathbf{w} - \mathbf{y}) \\
&= (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{X}\mathbf{w} - (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{y} - \mathbf{y}^\mathsf{T}(\mathbf{X}\mathbf{w}) + \mathbf{y}^\mathsf{T}\mathbf{y} \\
&= (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{X}\mathbf{w} - 2\mathbf{y}^\mathsf{T}(\mathbf{X}\mathbf{w}) + \mathbf{y}^\mathsf{T}\mathbf{y} \\
&= (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{X}\mathbf{w} - 2(\mathbf{y}^\mathsf{T}\mathbf{X})\mathbf{w} + \mathbf{y}^\mathsf{T}\mathbf{y} \\
&= \mathbf{w}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{w} - 2(\mathbf{X}^\mathsf{T}\mathbf{y})^\mathsf{T}\mathbf{w} + \mathbf{y}^\mathsf{T}\mathbf{y}
\end{aligned}
$$

Rearrange

$$
\begin{aligned}
\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 &= (\mathbf{X}\mathbf{w} - \mathbf{y})^\mathsf{T}(\mathbf{X}\mathbf{w} - \mathbf{y}) \\
&= (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{X}\mathbf{w} - (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{y} - \mathbf{y}^\mathsf{T}(\mathbf{X}\mathbf{w}) + \mathbf{y}^\mathsf{T}\mathbf{y} \\
&= (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{X}\mathbf{w} - 2\mathbf{y}^\mathsf{T}(\mathbf{X}\mathbf{w}) + \mathbf{y}^\mathsf{T}\mathbf{y} \\
&= (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{X}\mathbf{w} - 2(\mathbf{y}^\mathsf{T}\mathbf{X})\mathbf{w} + \mathbf{y}^\mathsf{T}\mathbf{y} \\
&= \mathbf{w}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{w} - 2(\mathbf{X}^\mathsf{T}\mathbf{y})^\mathsf{T}\mathbf{w} + \mathbf{y}^\mathsf{T}\mathbf{y}
\end{aligned}
$$



Rearrange

$$
\begin{aligned}
\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 &= (\mathbf{X}\mathbf{w} - \mathbf{y})^\mathsf{T}(\mathbf{X}\mathbf{w} - \mathbf{y}) \\
&= (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{X}\mathbf{w} - (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{y} - \mathbf{y}^\mathsf{T}(\mathbf{X}\mathbf{w}) + \mathbf{y}^\mathsf{T}\mathbf{y} \\
&= (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{X}\mathbf{w} - 2\mathbf{y}^\mathsf{T}(\mathbf{X}\mathbf{w}) + \mathbf{y}^\mathsf{T}\mathbf{y} \\
&= (\mathbf{X}\mathbf{w})^\mathsf{T}\mathbf{X}\mathbf{w} - 2(\mathbf{y}^\mathsf{T}\mathbf{X})\mathbf{w} + \mathbf{y}^\mathsf{T}\mathbf{y} \\
&= \mathbf{w}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{X}\mathbf{w} - 2(\mathbf{X}^\mathsf{T}\mathbf{y})^\mathsf{T}\mathbf{w} + \mathbf{y}^\mathsf{T}\mathbf{y} \\
&= \mathbf{w}^\mathsf{T}(\mathbf{X}^\mathsf{T}\mathbf{X})\mathbf{w} - 2(\mathbf{X}^\mathsf{T}\mathbf{y})^\mathsf{T}\mathbf{w} + \mathbf{y}^\mathsf{T}\mathbf{y}
\end{aligned}
$$

Differentiate

$$\nabla_{\mathbf{w}} L = \nabla_{\mathbf{w}} \left(\mathbf{w}^{\mathsf{T}} (\mathbf{X}^{\mathsf{T}} \mathbf{X}) \mathbf{w} - 2 (\mathbf{X}^{\mathsf{T}} \mathbf{y})^{\mathsf{T}} \mathbf{w} + \mathbf{y}^{\mathsf{T}} \mathbf{y} \right)$$

Differentiate

$$\nabla_{\mathbf{w}} L = \nabla_{\mathbf{w}} \left(\mathbf{w}^\mathsf{T} (\mathbf{X}^\mathsf{T} \mathbf{X}) \mathbf{w} - 2 (\mathbf{X}^\mathsf{T} \mathbf{y})^\mathsf{T} \mathbf{w} + \mathbf{y}^\mathsf{T} \mathbf{y} \right)$$

Doesn't depend on **w**

Differentiate

$$\nabla_{\mathbf{w}} L = \nabla_{\mathbf{w}} \left(\mathbf{w}^\mathsf{T} (\mathbf{X}^\mathsf{T} \mathbf{X}) \mathbf{w} - 2 (\mathbf{X}^\mathsf{T} \mathbf{y})^\mathsf{T} \mathbf{w} + \mathbf{y}^\mathsf{T} \mathbf{y} \right)$$

$$= \nabla_{\mathbf{w}} \left(\mathbf{w}^\mathsf{T} (\mathbf{X}^\mathsf{T} \mathbf{X}) \mathbf{w} - 2 (\mathbf{X}^\mathsf{T} \mathbf{y})^\mathsf{T} \mathbf{w} \right)$$

Differentiate

$$\nabla_{\mathbf{w}} L = \nabla_{\mathbf{w}} \left(\mathbf{w}^\mathsf{T} (\mathbf{X}^\mathsf{T} \mathbf{X}) \mathbf{w} - 2 (\mathbf{X}^\mathsf{T} \mathbf{y})^\mathsf{T} \mathbf{w} + \mathbf{y}^\mathsf{T} \mathbf{y} \right)$$

$$= \nabla_{\mathbf{w}} \left(\mathbf{w}^\mathsf{T} (\mathbf{X}^\mathsf{T} \mathbf{X}) \mathbf{w} - 2 (\mathbf{X}^\mathsf{T} \mathbf{y})^\mathsf{T} \mathbf{w} \right)$$

$\mathbf{X}^\mathsf{T}\mathbf{X}$ is symmetric, quadratic derivative identity applies:

$$\nabla_{\mathbf{x}} (\mathbf{x}^\mathsf{T} \mathbf{A} \mathbf{x}) = 2\mathbf{A}\mathbf{x}$$

Differentiate

$$\nabla_{\mathbf{w}} L = \nabla_{\mathbf{w}} \left(\mathbf{w}^\mathsf{T} (\mathbf{X}^\mathsf{T} \mathbf{X}) \mathbf{w} - 2 (\mathbf{X}^\mathsf{T} \mathbf{y})^\mathsf{T} \mathbf{w} + \mathbf{y}^\mathsf{T} \mathbf{y} \right)$$

$$= \nabla_{\mathbf{w}} \left(\mathbf{w}^\mathsf{T} (\mathbf{X}^\mathsf{T} \mathbf{X}) \mathbf{w} - 2 (\mathbf{X}^\mathsf{T} \mathbf{y})^\mathsf{T} \mathbf{w} \right)$$

$\mathbf{X}^\mathsf{T}\mathbf{y}$ is just a vector, so linear derivative identity applies:

$$\nabla_{\mathbf{x}} (\mathbf{b}^\mathsf{T} \mathbf{x}) = \mathbf{b}$$

Differentiate

$$\nabla_{\mathbf{w}} L = \nabla_{\mathbf{w}} \left(\mathbf{w}^\mathsf{T} (\mathbf{X}^\mathsf{T} \mathbf{X}) \mathbf{w} - 2 (\mathbf{X}^\mathsf{T} \mathbf{y})^\mathsf{T} \mathbf{w} + \mathbf{y}^\mathsf{T} \mathbf{y} \right)$$

$$= \nabla_{\mathbf{w}} \left(\mathbf{w}^\mathsf{T} (\mathbf{X}^\mathsf{T} \mathbf{X}) \mathbf{w} - 2 (\mathbf{X}^\mathsf{T} \mathbf{y})^\mathsf{T} \mathbf{w} \right)$$

$$= 2\mathbf{X}^\mathsf{T} \mathbf{X} \mathbf{w} - 2\mathbf{X}^\mathsf{T} \mathbf{y}$$

Differentiate

$$\nabla_{\mathbf{w}} L = \nabla_{\mathbf{w}} \left(\mathbf{w}^\mathsf{T} (\mathbf{X}^\mathsf{T} \mathbf{X}) \mathbf{w} - 2 (\mathbf{X}^\mathsf{T} \mathbf{y})^\mathsf{T} \mathbf{w} + \mathbf{y}^\mathsf{T} \mathbf{y} \right)$$

$$= \nabla_{\mathbf{w}} \left(\mathbf{w}^\mathsf{T} (\mathbf{X}^\mathsf{T} \mathbf{X}) \mathbf{w} - 2 (\mathbf{X}^\mathsf{T} \mathbf{y})^\mathsf{T} \mathbf{w} \right)$$

$$= 2\mathbf{X}^\mathsf{T} \mathbf{X} \mathbf{w} - 2\mathbf{X}^\mathsf{T} \mathbf{y}$$

Could also write as

$$2\mathbf{X}^\mathsf{T} (\hat{\mathbf{y}} - \mathbf{y})$$

Differentiate

$$\nabla_{\mathbf{w}} L = \nabla_{\mathbf{w}} \left(\mathbf{w}^\mathsf{T} (\mathbf{X}^\mathsf{T} \mathbf{X}) \mathbf{w} - 2 (\mathbf{X}^\mathsf{T} \mathbf{y})^\mathsf{T} \mathbf{w} + \mathbf{y}^\mathsf{T} \mathbf{y} \right)$$

$$= \nabla_{\mathbf{w}} \left(\mathbf{w}^\mathsf{T} (\mathbf{X}^\mathsf{T} \mathbf{X}) \mathbf{w} - 2 (\mathbf{X}^\mathsf{T} \mathbf{y})^\mathsf{T} \mathbf{w} \right)$$

$$= 2\mathbf{X}^\mathsf{T} \mathbf{X} \mathbf{w} - 2\mathbf{X}^\mathsf{T} \mathbf{y}$$

$$= 0 \quad \Longrightarrow \quad \boxed{\mathbf{w}^\star = (\mathbf{X}^\mathsf{T} \mathbf{X})^{-1} \mathbf{X}^\mathsf{T} \mathbf{y}}$$

We've assumed X^TX is invertible

Two important cases where this fails:

- Not enough data = underdetermined, too many unknowns
  - → Get more!
  - → Reduce features
- Features are not linearly independent
  - → Drop redundant features
  - → Regularise (§2.4)
  - → Use pseudo-inverse

2.3: “Ordinary Least Squares” Regression (part 2)

COMP0088 Introduction to Machine Learning • UCL Computer Science • Autumn 2021

Uncertainty

- Our linear model is deterministic
  → For a given **x**, it always predicts the same *y*
- Sampled data is not deterministic, at least not entirely
- For that matter, where do the residuals come from?
  → If the model were correct, it would give the right answers!

- Strategy: distinguish modelled behaviour from uncertainty
- The model is—might be—correct as far it goes, but incomplete.
  → This approach is applicable to any model, not just linear

Additive error model

- Error term captures the deviations between prediction & observation

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\varepsilon}$$

- $\boldsymbol{\varepsilon}$ here covers a multitude of sins:
  - → model specification errors: ways in which the model is wrong
    - approximation errors: behaviour not reproducible by the functional form
    - estimation errors: behaviour that isn't reproduced due to poor parameterisation
  - → irreducible errors: ways in which the world is really not helping
    - things that cannot be measured
    - natural variability in the population
  - → noise

Additive error model

- These distinctions are pretty important!

  → eg, if the true behaviour is sinusoidal, $\boldsymbol{\varepsilon}$ is going to be doing a lot of heavy lifting

- …but we're going to gloss over them anyway ¯_(ツ)_/¯

- For the moment, $\boldsymbol{\varepsilon}$ is just a single cloud of uncertainty over the model
- With a quick handwave at the Central Limit Theorem, we're going to assume it is i.i.d. with a univariate Gaussian distribution:

$$\boldsymbol{\varepsilon} \sim N(0, \sigma^2)$$

Probabilistic interpretation

- This model is no longer deterministic — *y* is now a random variable. Conditioned on **x** and parameterised by **w**, *y* is also Gaussian:

$$y|\mathbf{x};\mathbf{w} \sim N(\mathbf{x}\cdot\mathbf{w}, \sigma^2)$$

- Equivalently:

$$P(y|\mathbf{x};\mathbf{w}) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(y-\mathbf{x}\cdot\mathbf{w})^2}{2\sigma^2}\right)$$

Probabilistic interpretation

- Assumed i.i.d., so probability of the whole training set is product of the individual probabilities of the observations:

$$P(\mathbf{y}|\mathbf{X};\mathbf{w}) = \prod_i^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right)$$

- With **X** and **y** fixed, this is a function of **w**, called the likelihood:

$$L(\mathbf{w}) = \prod_i^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right)$$

Maximum likelihood estimation

- $L(\mathbf{w})$: "How likely is it that we would have seen what we saw (i.e. **X** and **y**) if the parameters were **w**?"
- Maximum likelihood estimation: choose **w** such that the data was most likely.

  → NB: this does not mean they are high probability given **w**, just that they would have been even less likely under any other **w**

- In this case:

$$\mathbf{w}^\star = \underset{\mathbf{w}}{\operatorname{argmax}} \prod_i^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right)$$

Log likelihood

- Probabilities are in [0, 1]. Multiplying many together is numerically unstable, so standard practice is to take logarithms
  - → monotonic non-decreasing transformations do not move optima
    - ▸ eg: log, exp, offset, multiply by positive constant
  - → log also handily cancels exp

Log likelihood

$$\log L(\mathbf{w}) = \log \prod_{i}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right)$$

Log likelihood

$$\log L(\mathbf{w}) = \log \prod_{i}^{n} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right)$$

Product becomes sum

Log likelihood

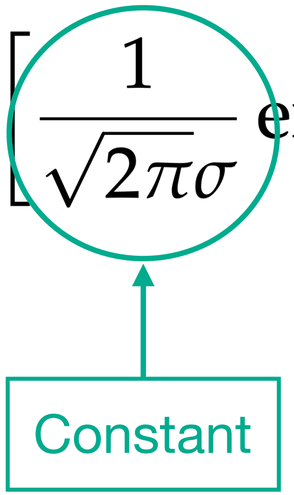
$$\log L(\mathbf{w}) = \log \prod_i^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right)$$

$$= \sum_i^n \log\left[\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right)\right]$$

Log likelihood

$$\log L(\mathbf{w}) = \log \prod_i^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right)$$

$$= \sum_i^n \log \left[\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right)\right]$$



Log likelihood

$$\log L(\mathbf{w}) = \log \prod_i^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right)$$

$$= \sum_i^n \log\left[\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right)\right]$$

exp disappears

Log likelihood

$$
\begin{aligned}
\log L(\mathbf{w}) &= \log \prod_i^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right) \\
&= \sum_i^n \log\left[\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right)\right] \\
&= \sum_i^n \left[\log\left(\frac{1}{\sqrt{2\pi}\sigma}\right) - \frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right]
\end{aligned}
$$

Log likelihood

$$\log L(\mathbf{w}) = \log \prod_i^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right)$$

$$= \sum_i^n \log\left[\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right)\right]$$

$$= \sum_i^n \left[\log\left(\frac{1}{\sqrt{2\pi}\sigma}\right) - \frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right]$$

Constants move outside summation

Log likelihood

$$
\begin{aligned}
\log L(\mathbf{w}) &= \log \prod_i^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right) \\
&= \sum_i^n \log\left[\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right)\right] \\
&= \sum_i^n \left[\log\left(\frac{1}{\sqrt{2\pi}\sigma}\right) - \frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right] \\
&= n \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_i^n (y_i - \mathbf{x}_i \cdot \mathbf{w})^2
\end{aligned}
$$

Log likelihood

$$\log L(\mathbf{w}) = \log \prod_i^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right)$$

$$= \sum_i^n \log\left[\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right)\right]$$

$$= \sum_i^n \left[\log\left(\frac{1}{\sqrt{2\pi}\sigma}\right) - \frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right]$$

$$= n \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_i^n (y_i - \mathbf{x}_i \cdot \mathbf{w})^2$$

Constants

Log likelihood

$$\log L(\mathbf{w}) = \log \prod_i^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right)$$

$$= \sum_i^n \log\left[\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right)\right]$$

$$= \sum_i^n \left[\log\left(\frac{1}{\sqrt{2\pi}\sigma}\right) - \frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right]$$

$$= n \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_i^n (y_i - \mathbf{x}_i \cdot \mathbf{w})^2$$

Sum of squared residuals

Log likelihood

$$
\begin{aligned}
\log L(\mathbf{w}) &= \log \prod_i^n \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right) \\
&= \sum_i^n \log\left[\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right)\right] \\
&= \sum_i^n \left[\log\left(\frac{1}{\sqrt{2\pi}\sigma}\right) - \frac{(y_i - \mathbf{x}_i \cdot \mathbf{w})^2}{2\sigma^2}\right] \\
&= n \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{2\sigma^2} \sum_i^n (y_i - \mathbf{x}_i \cdot \mathbf{w})^2 \\
&= A - B\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2
\end{aligned}
$$

Big reveal!

- Optimisation is unchanged by positive constants, flipped by negation
- So:

$$\begin{aligned}\mathbf{w}^\star &= \underset{\mathbf{w}}{\operatorname{argmax}} \quad A - B\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 \\ &= \underset{\mathbf{w}}{\operatorname{argmax}} \quad -\|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 \\ &= \underset{\mathbf{w}}{\operatorname{argmin}} \quad \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 \\ &= (\mathbf{X}^\mathsf{T}\mathbf{X})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}\end{aligned}$$

- It's the same answer as our geometric least squares model

Take home

- **Least squares estimation is equivalent to maximising the likelihood under the i.i.d Gaussian noise assumption**
- Provides motivation for loss function beyond mathematical convenience
- OLS does not make an explicit noise distribution assumption. It does not require a particular noise model. But it makes the assumption that $\|\mathbf{y} - \hat{\mathbf{y}}\|^2$ is a good distance metric.
- If errors are not normally distributed, this may not be true. Eg, if they are asymmetric, the model will be biased.

2.4: Basis Expansion

COMP0088 Introduction to Machine Learning • UCL Computer Science • Autumn 2021

Non-linear linear models

- Linear models are linear in the parameters, **w**
  → Parameter fitting is predicated on this
- But you can apply any transformations you like to the data beforehand
  → Remember those **log-log** plots?
- Any model that can be expressed as a weighted sum of arbitrary fixed functions of **x** can be fitted as a linear model

$$\hat{y} = \sum_{i}^{k} w_i h_i(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \cdot \mathbf{w} = \mathbf{x}' \cdot \mathbf{w}$$

Basis expansion

- Model is still just fitting a hyperplane to data, but to different data:

$$\mathbf{x}' = \mathbf{h}(\mathbf{x}) = [\, h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_k(\mathbf{x}) \,]$$

- Transformed vector **x′** is in a different feature space, with a different **basis**
- The new space is typically higher dimensional (though it doesn't have to be), so this is known as basis expansion
- It can be useful to think of the basis as a dictionary of elements that the model can combine to build the output

Intercept

- We've already seen this in action. Adding a constant x_0 "feature" to capture the intercept is tantamount to defining:

$$h_1(\mathbf{x}) = 1$$
$$h_2(\mathbf{x}) = x_1$$
$$\vdots$$
$$h_{d+1}(\mathbf{x}) = x_d$$

- The extra basis function is independent of $\mathbf{x}$, allowing the model to add a global offset

Examples

- Polynomial fitting is the standard teaching example for basis expansion.
- In 1D:

$$h_1(x) = 1$$
$$h_2(x) = x$$
$$h_3(x) = x^2$$
$$\vdots$$
$$h_k(x) = x^{k-1}$$

- …but you do this in the lab exercises, so we won't go into detail here.

Frequency decomposition

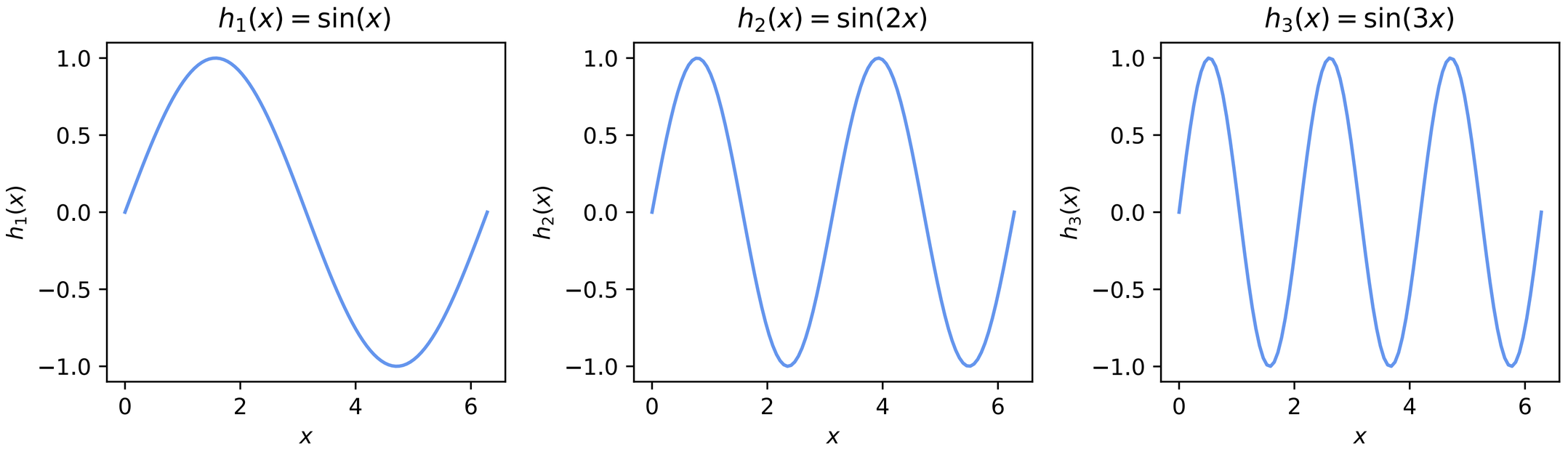
- Useless but illustrative toy example:

$$h_1(x) = \sin(x)$$
$$h_2(x) = \sin(2x)$$
$$h_3(x) = \sin(3x)$$

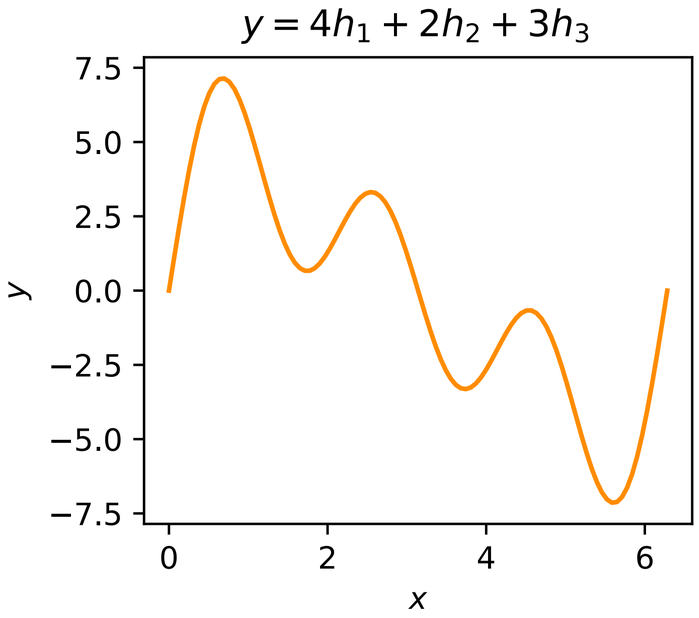
→ Just 3 basis functions, for simplicity

→ We can imagine using much larger basis of many frequencies at multiple phases, but it would be hopelessly inefficient — in real life you'd use an FFT

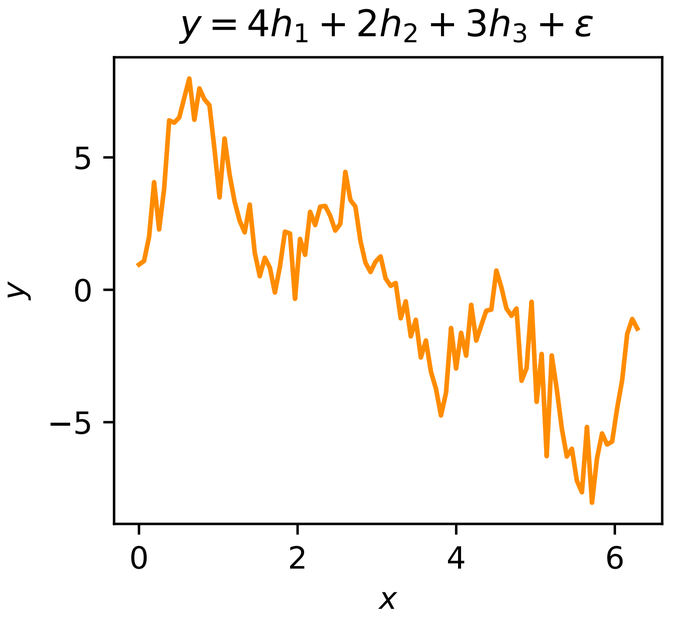
Basis functions



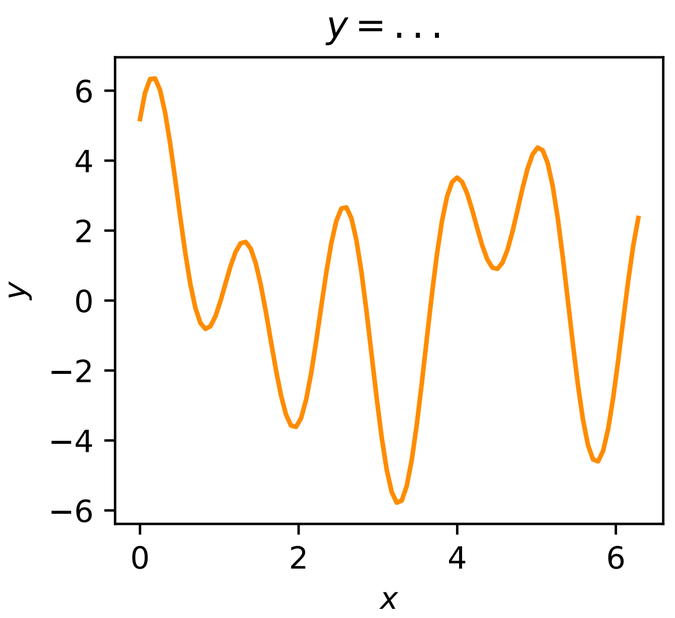
Example data



A clean combination of the basis functions

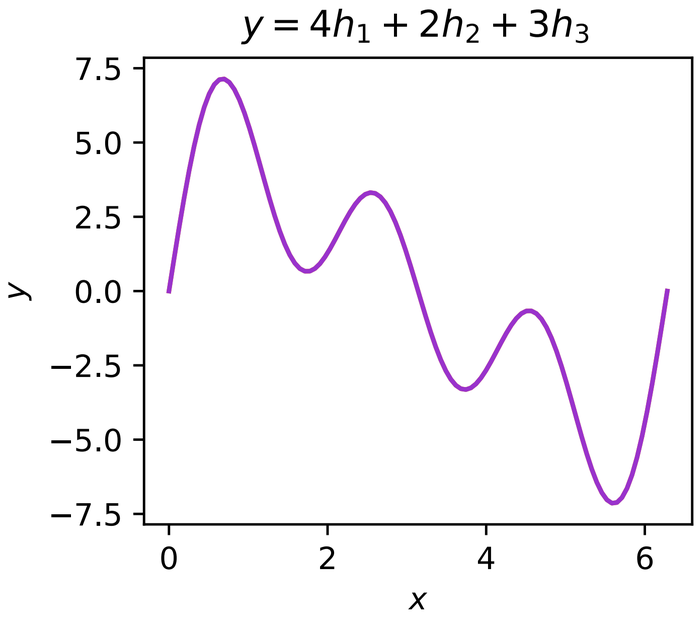


Same combination with added noise



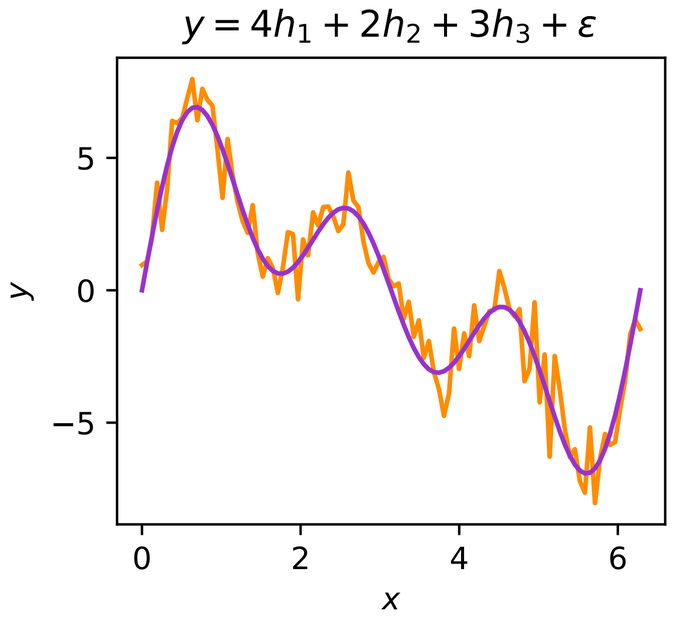
Clean, but phases and frequencies not in basis

Fit results



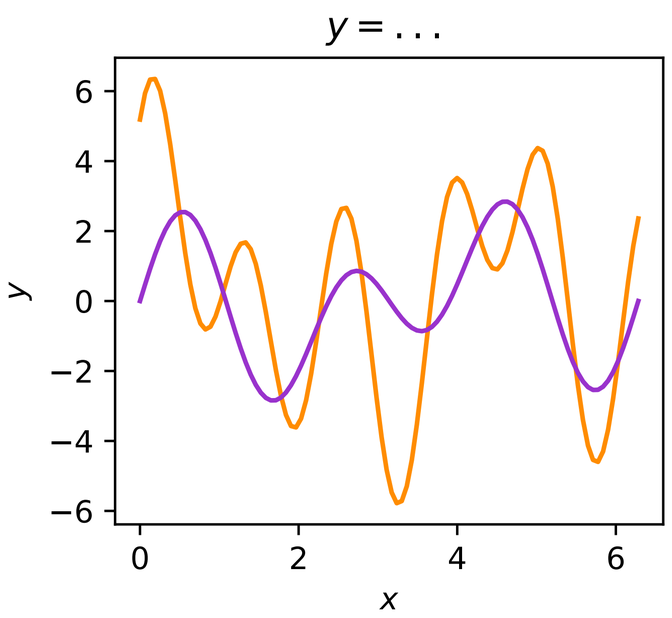
Perfect recovery of weights

(4, 2, 3)



Decent recovery of weights

(3.85, 1.99, 2.87)



No sensible weights to recover

(-0.72, 1.03, 2.02)

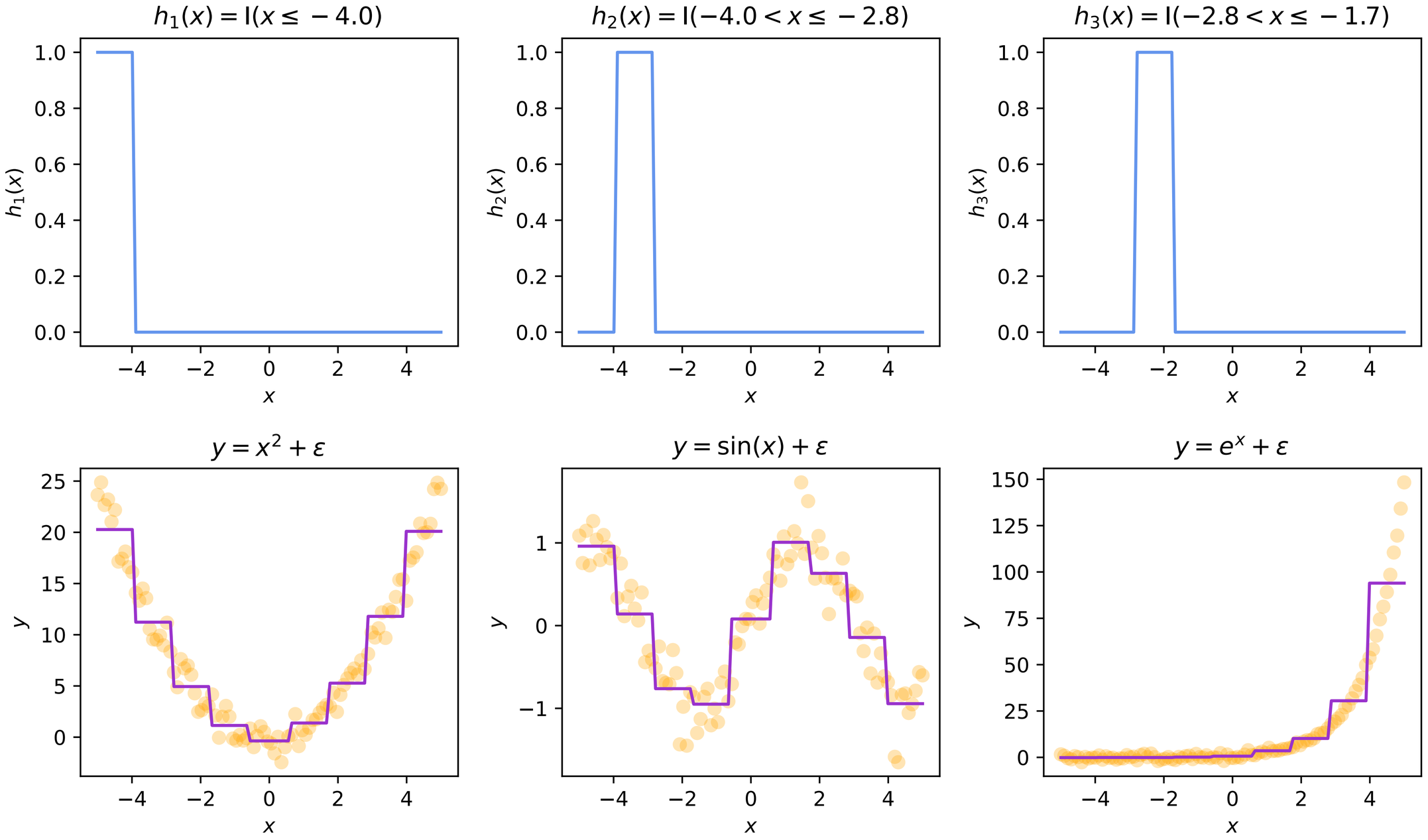
Piecewise constant

- Partition the feature space into regions
- In 1D:

  → Choose *m* boundary values ("knots") $t_1, t_2, \ldots, t_m$
- Then:

$$h_i(x) = \mathbf{1}(t_i < x \leq t_{i+1}), \qquad i \in \{1, 2, \ldots, m-1\}$$

- **x′** is just a one-hot vector indicating which region the data point is in
- Fitting learns the mean of the samples in each region


$h_1(x) = \mathrm{I}(x \leq -4.0)$
$h_2(x) = \mathrm{I}(-4.0 < x \leq -2.8)$
$h_3(x) = \mathrm{I}(-2.8 < x \leq -1.7)$
$h_1(x)$
$h_2(x)$
$h_3(x)$
x
$y = x^2 + \varepsilon$
$y = \sin(x) + \varepsilon$
$y = e^x + \varepsilon$
y

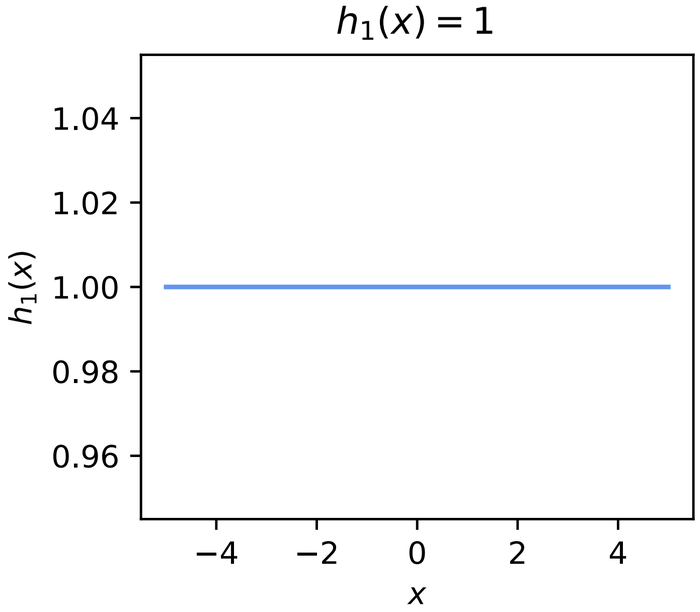
Piecewise linear spline

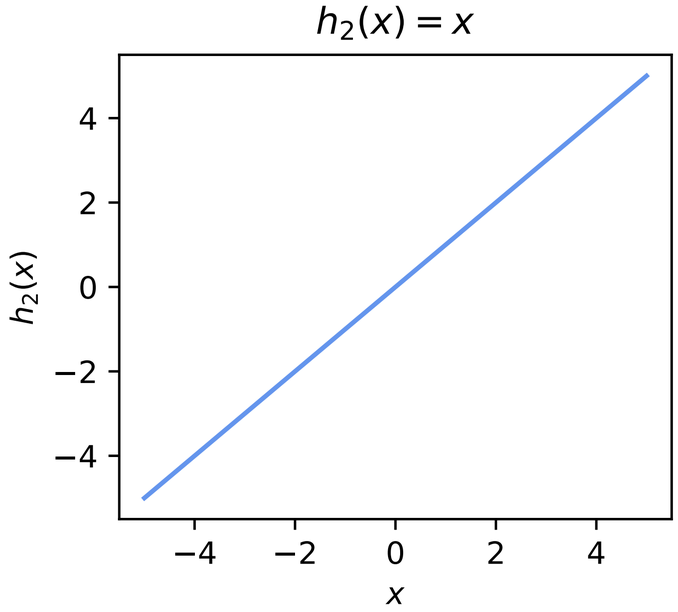
- Partition the feature space into regions, fit (joined up) linear segments
- In 1D:

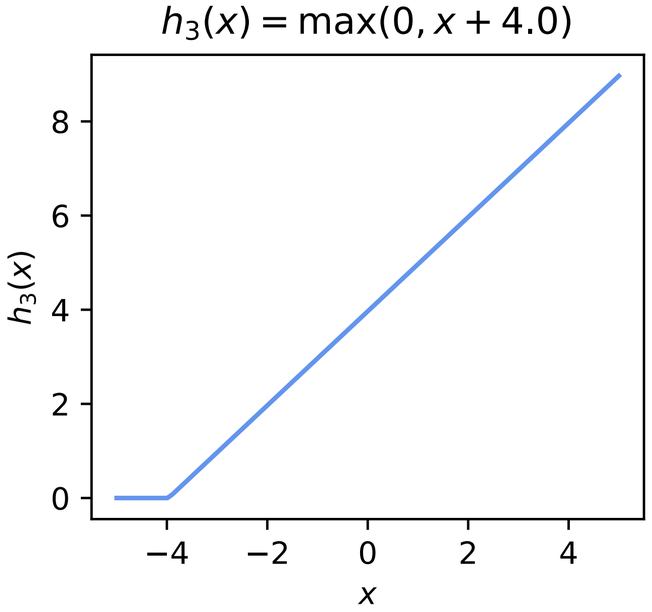
  → Choose *m* boundary values ("knots") $t_1, t_2, \ldots, t_m$

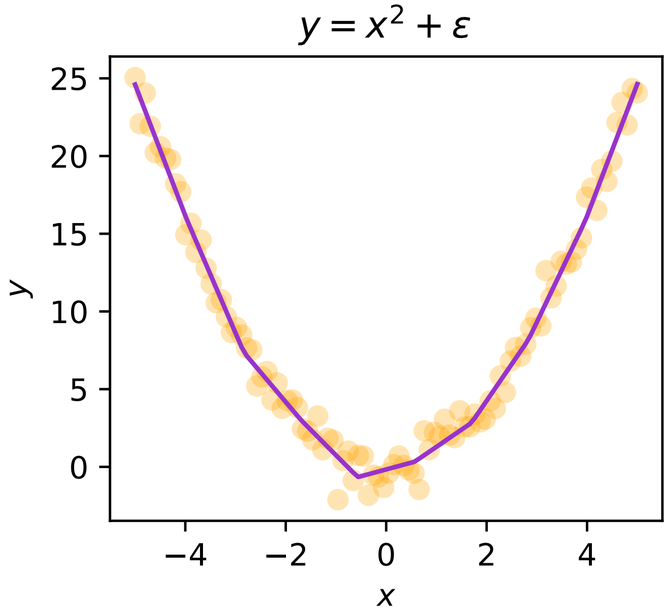
- Then:

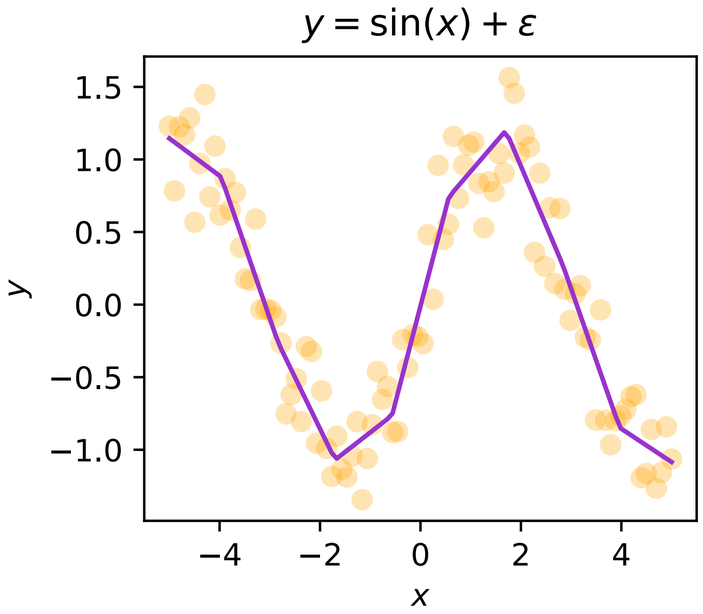
$$h_1(x) = 1$$
$$h_2(x) = x$$
$$h_{i+2}(x) = \max(0, x - t_i), \qquad i \in \{1, 2, \ldots, m\}$$

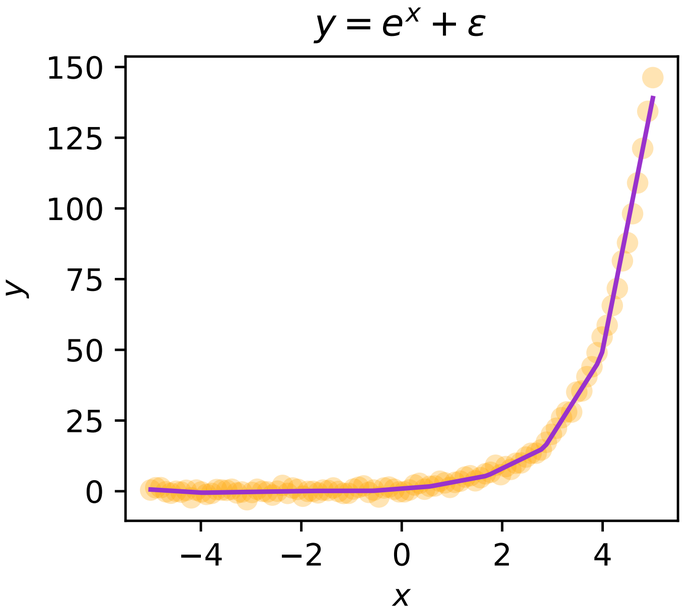

$h_1(x) = 1$
1.04
1.02
1.00
0.98
0.96
$h_1(x)$
−4
−2
0
2
4
x


$h_2(x) = x$
4
2
0
−2
−4
$h_2(x)$
−4
−2
0
2
4
x


$h_3(x) = \max(0, x + 4.0)$
8
6
4
2
0
$h_3(x)$
−4
−2
0
2
4
x


$y = x^2 + \varepsilon$
25
20
15
10
5
0
y
−4
−2
0
2
4
x


$y = \sin(x) + \varepsilon$
1.5
1.0
0.5
0.0
−0.5
−1.0
y
−4
−2
0
2
4
x


$y = e^x + \varepsilon$
150
125
100
75
50
25
0
y
−4
−2
0
2
4
x

Issues with basis expansion

- Basis is not learned, has to be chosen
  → Potentially many hyperparameters
- Inappropriate basis will produce garbage
  → (eg third frequency decomposition example)
- Too specific or flexible a basis may encourage overfitting
- Noise distribution may be distorted

Additive error model revisited

- We assumed errors only in **y**, independent of **X**

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\varepsilon} \qquad \boldsymbol{\varepsilon} \sim N(0, \sigma^2)$$

- In practice, inputs will be noisy
  - → Sort of OK in linear case, assuming noise in **X** also Gaussian
- Transformations will act on that noise and can:
  - → Amplify it
  - → Change its shape — in particular increasing asymmetry
  - → Make it more correlated
- These can all mess up the fitting

2.5: Overfitting & Regularisation

COMP0088 Introduction to Machine Learning • UCL Computer Science • Autumn 2021

Overfitting recap

- A model is overfitted when its parameters are too specialised for the training set and do not generalise well to new data

  → cf. underfitting, when the model does not even capture the behaviour of the training set

- May be thought of as fitting the noise, rather than the shared structure
- Models may be prone to overfitting in various ways, but one common theme is to offset contributions from different features with large opposing weights, amplifying random differences for marginal loss improvements

  → Especially when features are correlated

Regularisation

- Encourage fitting regularities rather than irregularities
  - → But if we actually knew which were which we wouldn't have this problem!
  - → Instead we have to appeal to heuristics
- Occam's Razor suggests we should prefer less extreme parameterisations
  - → this is an intuition, not an ironclad rule
- Shrinkage methods:
  - → Reduce the number of features
  - → Reduce the overall scale of the parameters
  - → Encourage sparsity — set weights for unimportant parameters to zero

Feature selection

- Explicitly choose subsets of available features to use
- Fit models to each subset and select according to performance
  - → Estimate performance on validation set or using cross-validation
  - → Apply some trade-off criterion of parameter size vs performance (eg AIC, BIC)
- Best subset: test all possible subsets of each given size
  - → Exhaustive but expensive
- Stepwise selection
  - → Forward: add one feature at a time
  - → Backward: drop one feature at a time
  - → Greedy algorithms: may find suboptimal subset

Penalty

- Loss function defines the goal of the fit
- Add term(s) to encourage desirable properties/discourage undesirable:

$$L(\mathbf{X}, \mathbf{y}; \boldsymbol{\theta}) = D(f_{\boldsymbol{\theta}}(\mathbf{X}, \mathbf{y}), \mathbf{y}) + \lambda P(\boldsymbol{\theta})$$

- *D* is our original loss function, distance, representing badness of predictions
- P is a new penalty term, representing badness of parameters
- λ is a hyperparameter specifying trade off between *D* and *P*
- For OLS:

$$L(\mathbf{X}, \mathbf{y}; \mathbf{w}) = \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 + \lambda P(\mathbf{w})$$

Ridge regression

- Penalty on the weights is of same form as on the residuals:

$$P(\mathbf{w}) = \|\mathbf{w}\|^2$$

- Make the weights vector as small as possible
- Updated loss function remains convex and differentiable

$$\mathbf{w}^\star = \underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 + \lambda \|\mathbf{w}\|^2$$

- Most of original derivation remains unchanged, but there's an additional term from the start:

$$\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 + \lambda \|\mathbf{w}\|^2 = (\mathbf{X}\mathbf{w} - \mathbf{y})^\mathsf{T}(\mathbf{X}\mathbf{w} - \mathbf{y}) + \boxed{\lambda \mathbf{w}^\mathsf{T}\mathbf{w}}$$

- The gradient of this term (from the quadratic identity) is:

$$\nabla_{\mathbf{w}} \lambda \mathbf{w}^\mathsf{T}\mathbf{w} = 2\lambda \mathbf{I}\mathbf{w}$$

- This adds another factor to $\mathbf{w}$, which goes into the inverse term, giving:

$$\mathbf{w}^\star = (\mathbf{X}^\mathsf{T}\mathbf{X} + \lambda \mathbf{I})^{-1}\mathbf{X}^\mathsf{T}\mathbf{y}$$

Notes

- Adding λ to the diagonal entries of $\mathbf{X}^T\mathbf{X}$ decorrelates the rows/columns, improving invertibility
- Here (and for all shrinkage methods) we don't want to regularise the intercept term, so don't include it as w_0
- Shrinkage is also sensitive to scale differences in feature dimensions

  → Standardize!

Lasso

- “Least absolute shrinkage and selection operator”
- Very similar form to ridge regression, but using L_1 penalty instead of L_2:

$$P(\mathbf{w}) = \|\mathbf{w}\|_1$$

- Loss is still convex but no longer strictly differentiable
- Optimisation looks nearly identical:

$$\mathbf{w}^\star = \underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2 + \lambda \|\mathbf{w}\|_1$$

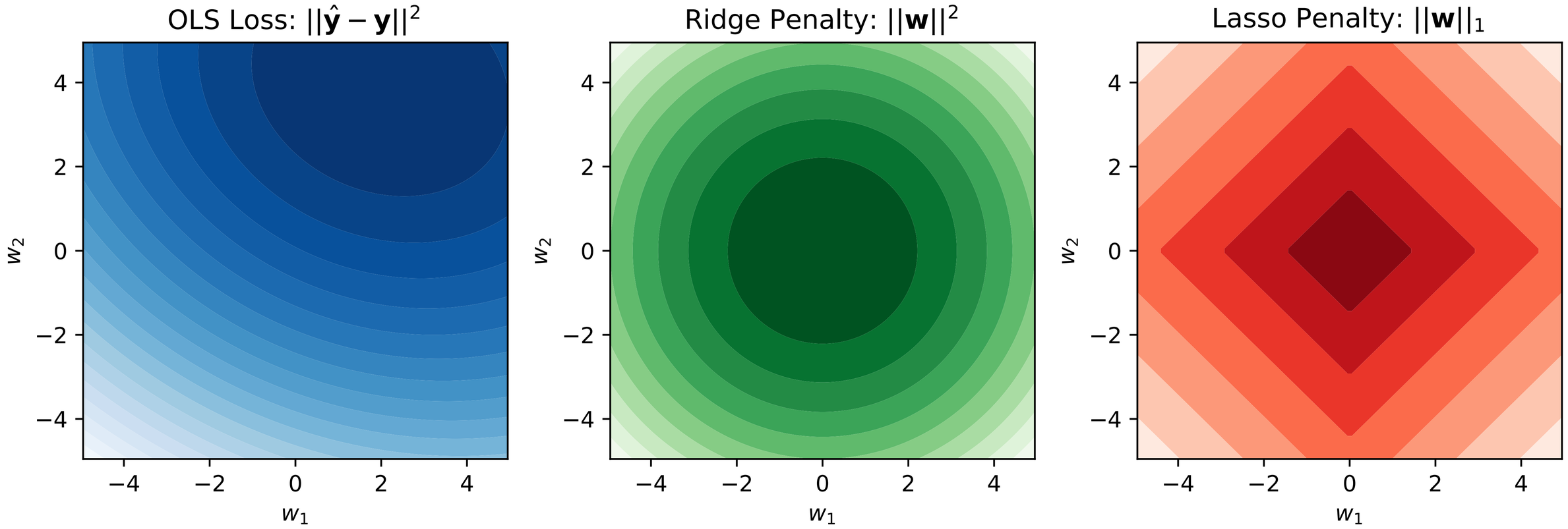
- Can be posed as a Quadratic Program

  → but we’re going to gloss over the details here

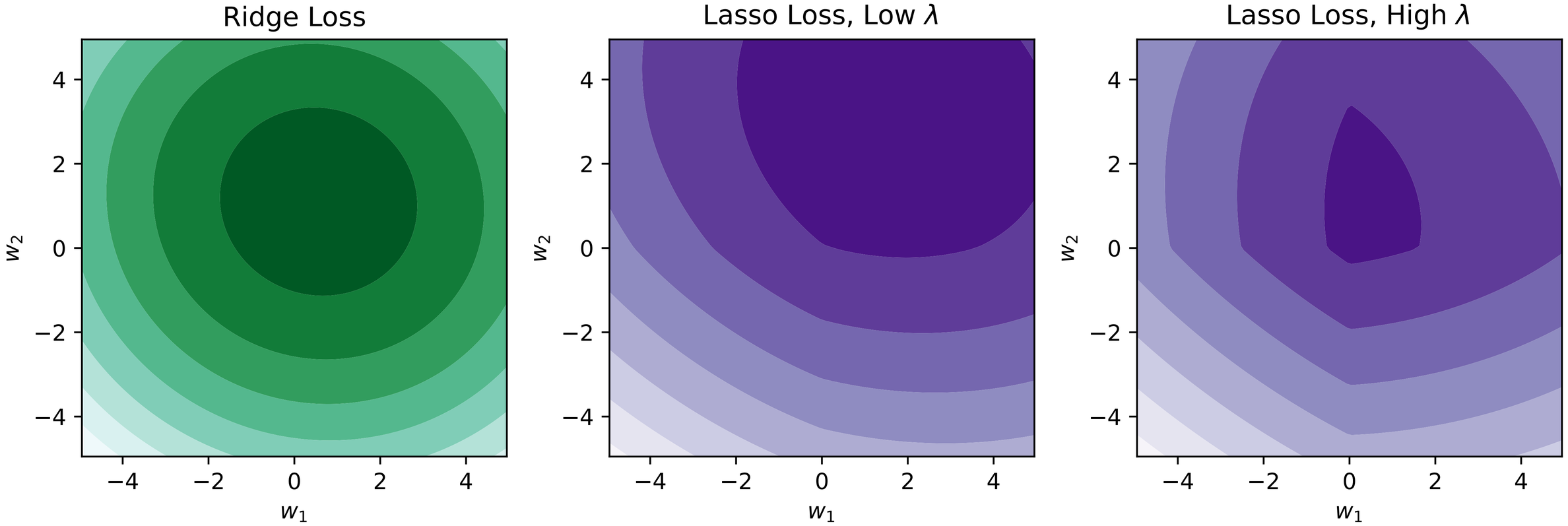
Notes

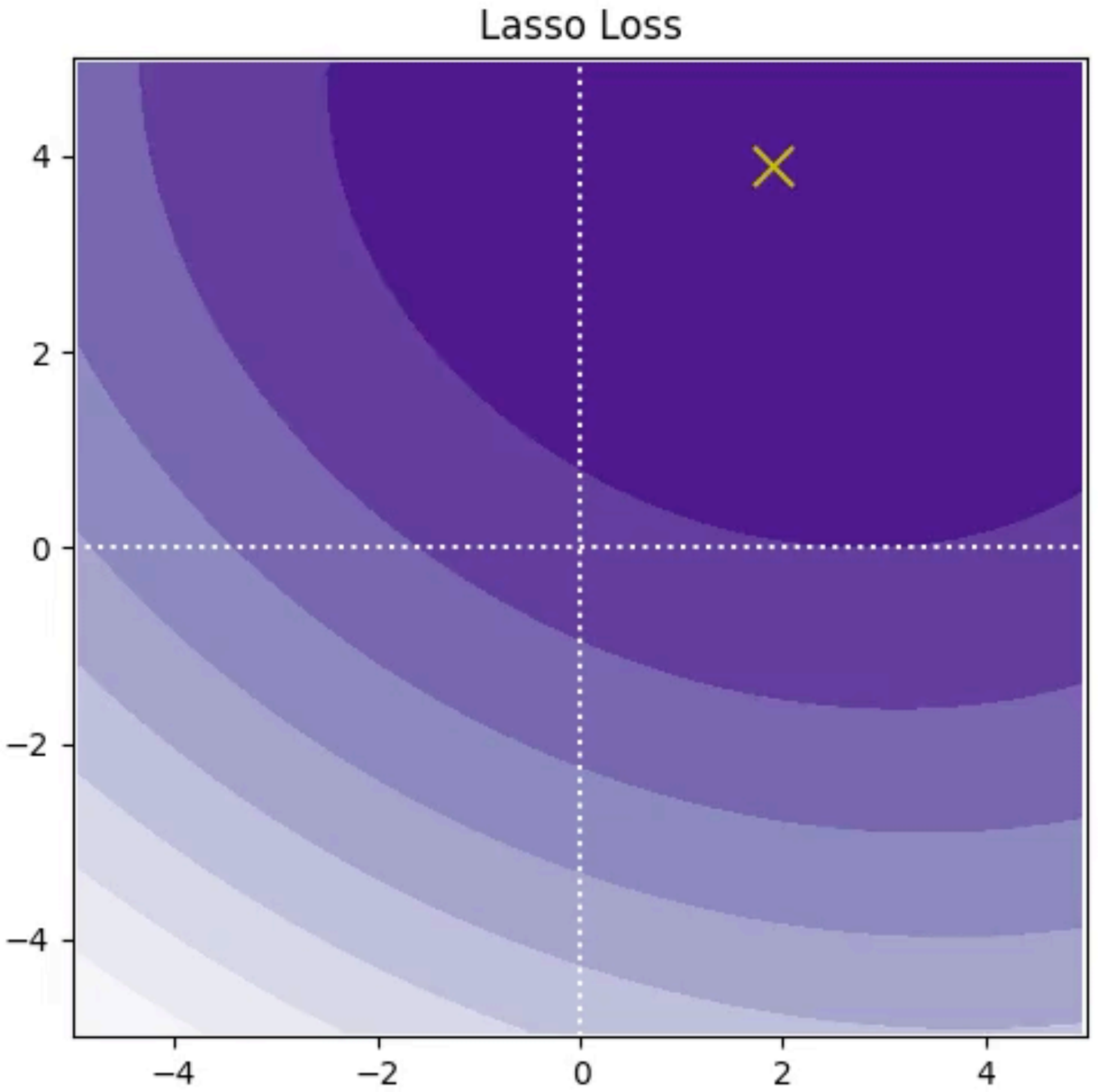
- Although not differentiable, can define a subgradient for optimisation
- Crucial feature: will set weights to zero
- Can be considered as automatic feature selection
- Encourages sparsity
- A proxy for the intractable L_0 pseudo-norm minimisation

Penalty landscapes



Penalty landscapes




Lasso Loss
4
2
0
−2
−4
−4
−2
0
2
4

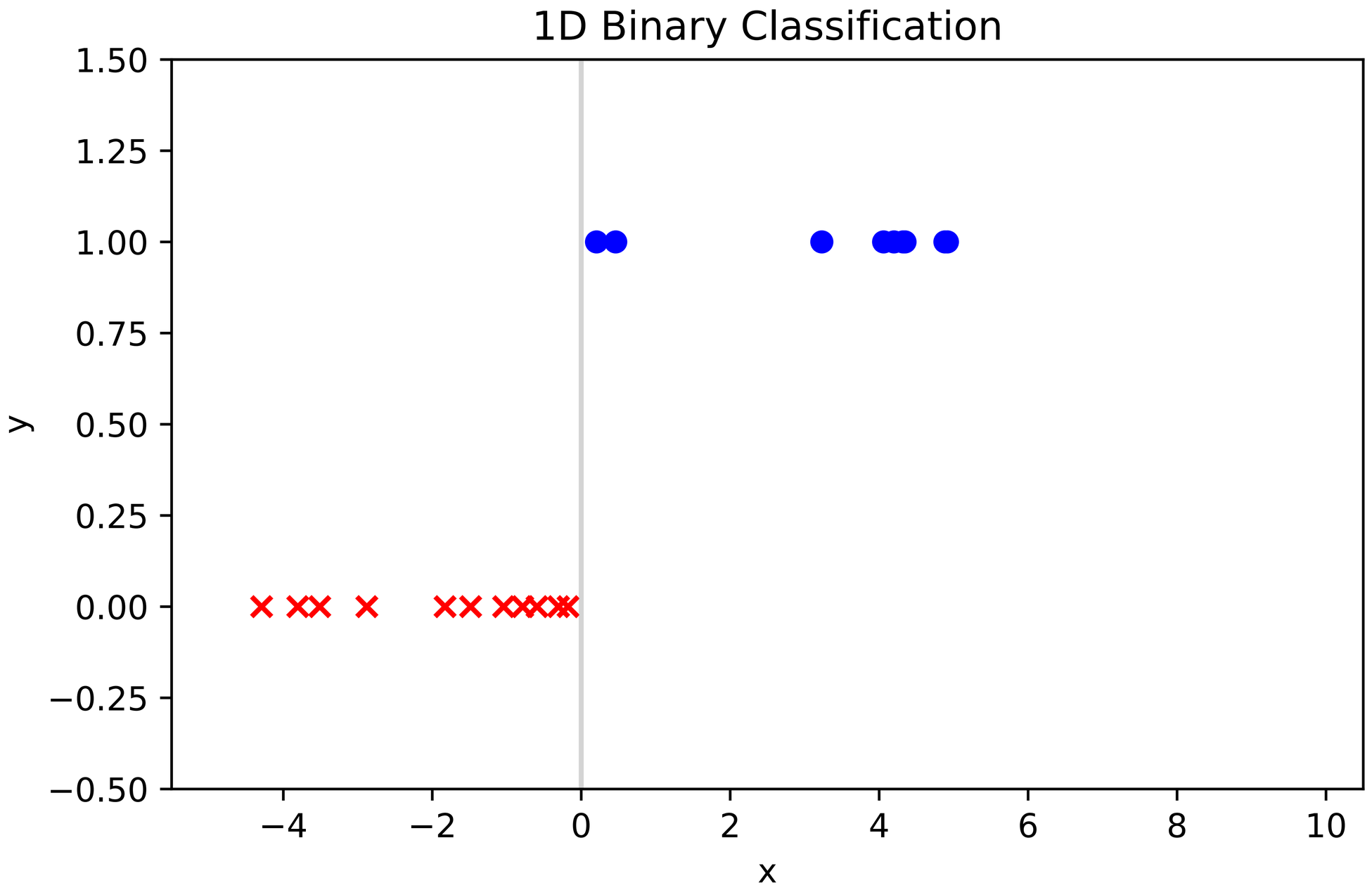
2.6: Linear Classifiers — Logistic Regression

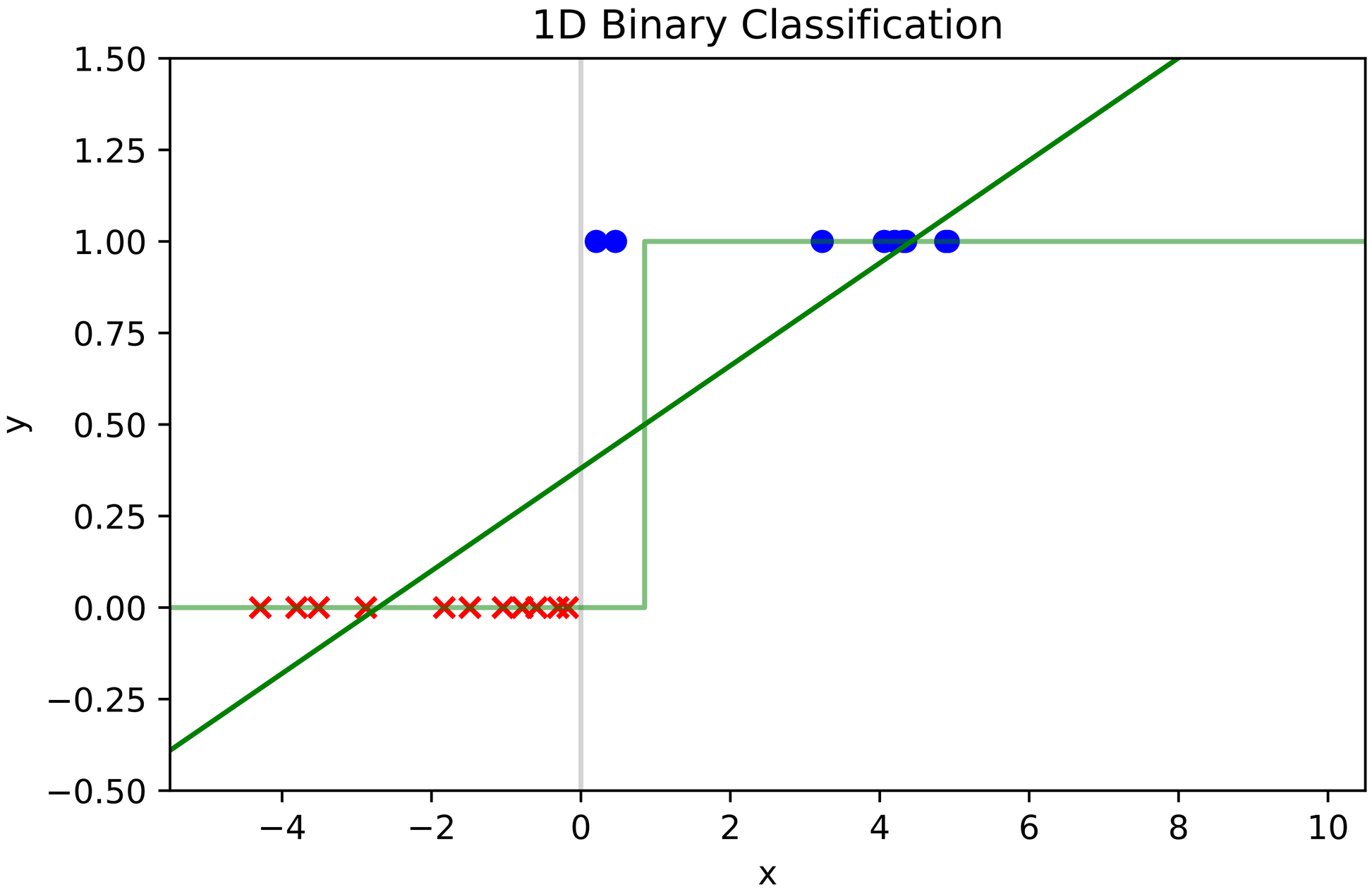
COMP0088 Introduction to Machine Learning • UCL Computer Science • Autumn 2021

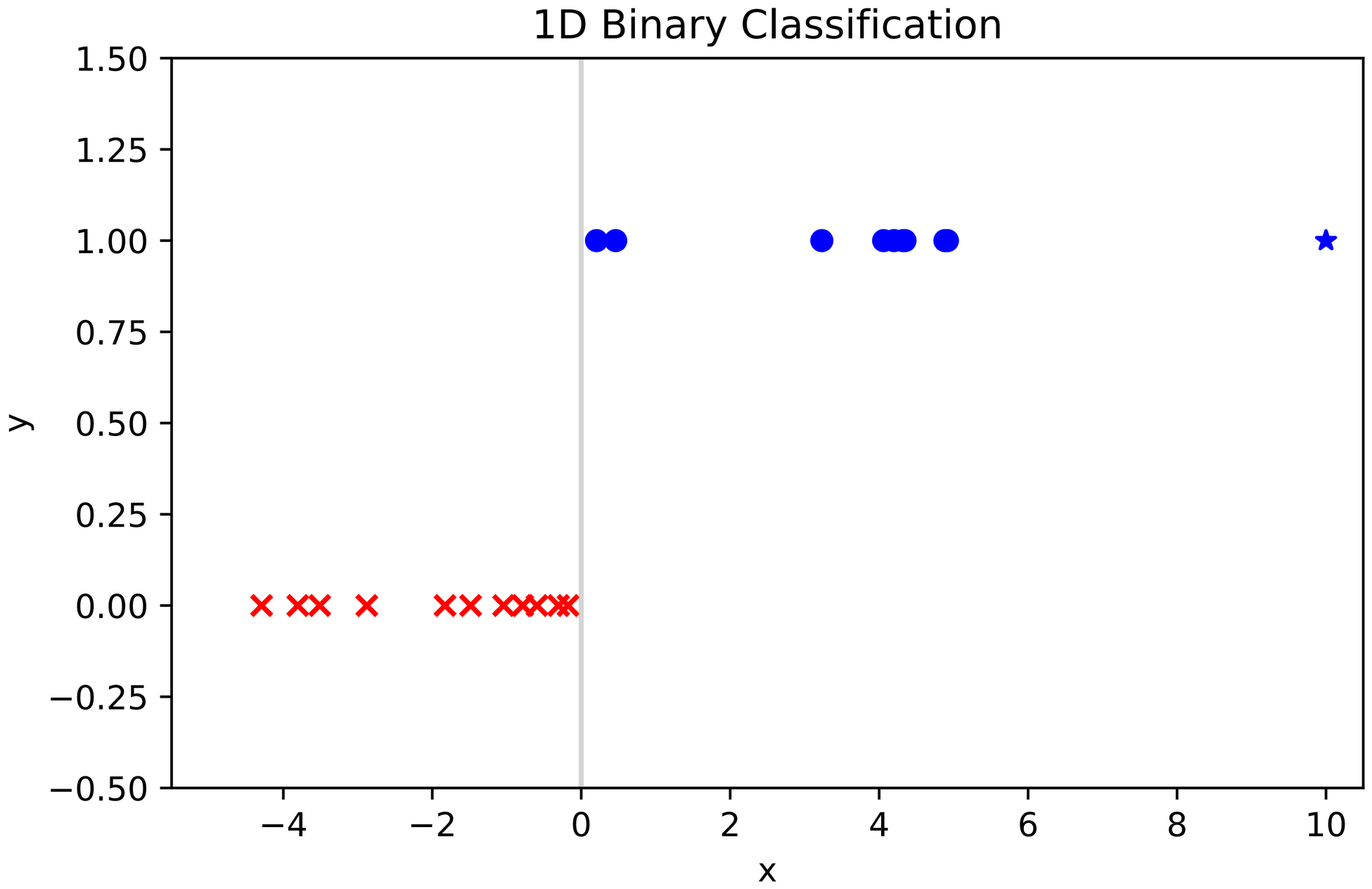
Can’t we just use linear regression?

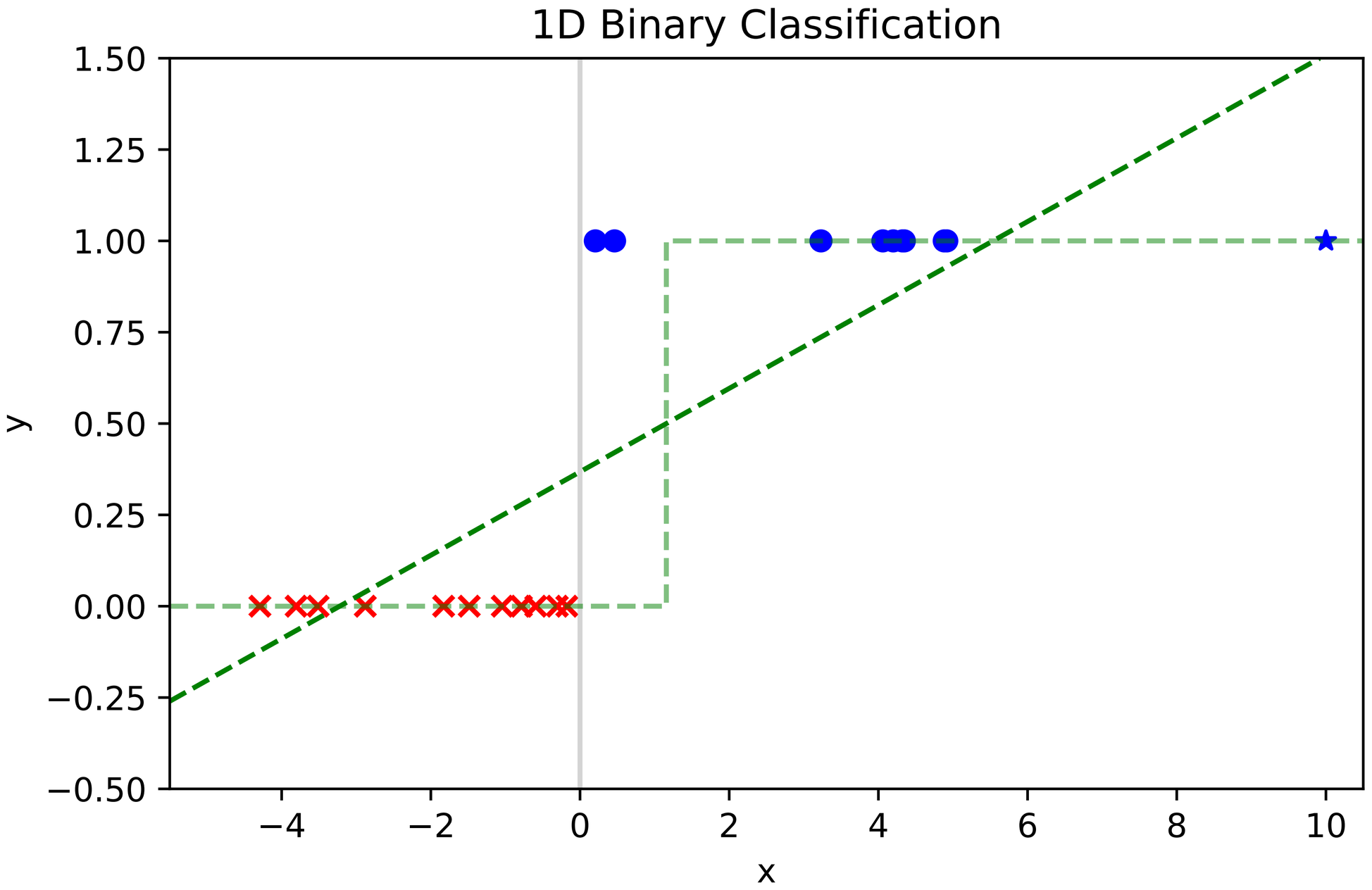
- Treat our {0,1} labels as numbers and predict them
- People do sometimes do this
- OLS outputs are continuous and unbounded

  → threshold at 0.5?


1D Binary Classification
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
y
−4
−2
0
2
4
6
8
10
x


1D Binary Classification
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
y
−4
−2
0
2
4
6
8
10
x


1D Binary Classification
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
y
−4
−2
0
2
4
6
8
10
x


1D Binary Classification
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
y
−4
−2
0
2
4
6
8
10
x

Classification ≠ Regression

- For regression we want to find the hyperplane (or more generally, manifold) that unites the data — that it all more-or-less occupies
- In classification, we want to find the boundary that divides the data — that more-or-less none of it occupies
- The two problems are in a sense reciprocal or dual
  - → In linear regression we are interested in the hyperplane itself
  - → In classification, we are interested in the two half-spaces the hyperplane creates

Issues

- MSE/RSS is not a good loss for this task
  - → symmetric
  - → distance based — not very meaningful for {0,1} outputs
  - → susceptible to outliers — gives more weight to greater distances
  - → we want to be on correct side of boundary, rather than close to it
- How do we interpret continuous unbounded outputs?
  - → what does that mean as a prediction?
  - → something like a confidence score, maybe — greater distance better?

Linear decision boundary

- Separating hyperplane in feature space:

$$\mathbf{x} \cdot \mathbf{w} = 0$$

→ **w** is normal to the hyperplane

- Decision function:

$$\hat{y} = \begin{cases} 1 & \text{if } \mathbf{x} \cdot \mathbf{w} \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Linear decision boundary

- Nice properties
  - → Interpretable
  - → Intuitive
  - → Easy to compute

- Can we find a more appropriate loss to use to fit it?

0-1 loss

$$L(y, \hat{y}) = \mathbf{1}(y \neq \hat{y})$$

- This is often what we really care about: classification accuracy
- But it's non-convex and not differentiable so hard to optimise
  - → sum of step functions
  - → zero gradient everywhere except where it matters, where it's undefined
- (We'll come back to this in future, but it's no use today)

Probability to the rescue?

- Make discrete variables continuous by casting as probabilities
- Here we have only binary outcomes — like a Bernoulli trial
- As in linear case, probabilities should depend on **x** and **w**:

$$P(y = 1 \mid \mathbf{x}; \mathbf{w}) = f_{\mathbf{w}}(\mathbf{x})$$
$$P(y = 0 \mid \mathbf{x}; \mathbf{w}) = 1 - f_{\mathbf{w}}(\mathbf{x})$$

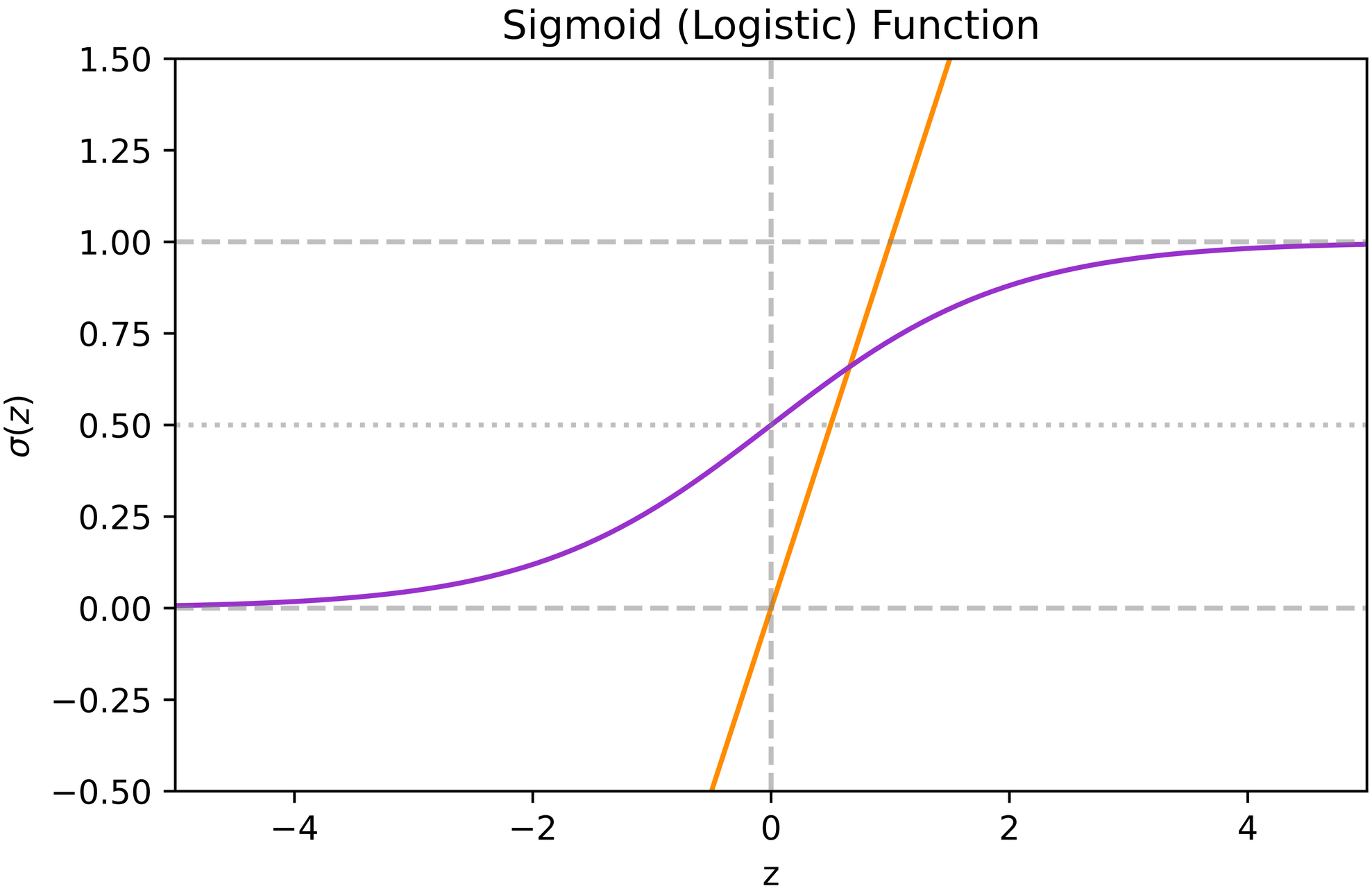
- We would like $f_{\mathbf{w}}$ to be linear

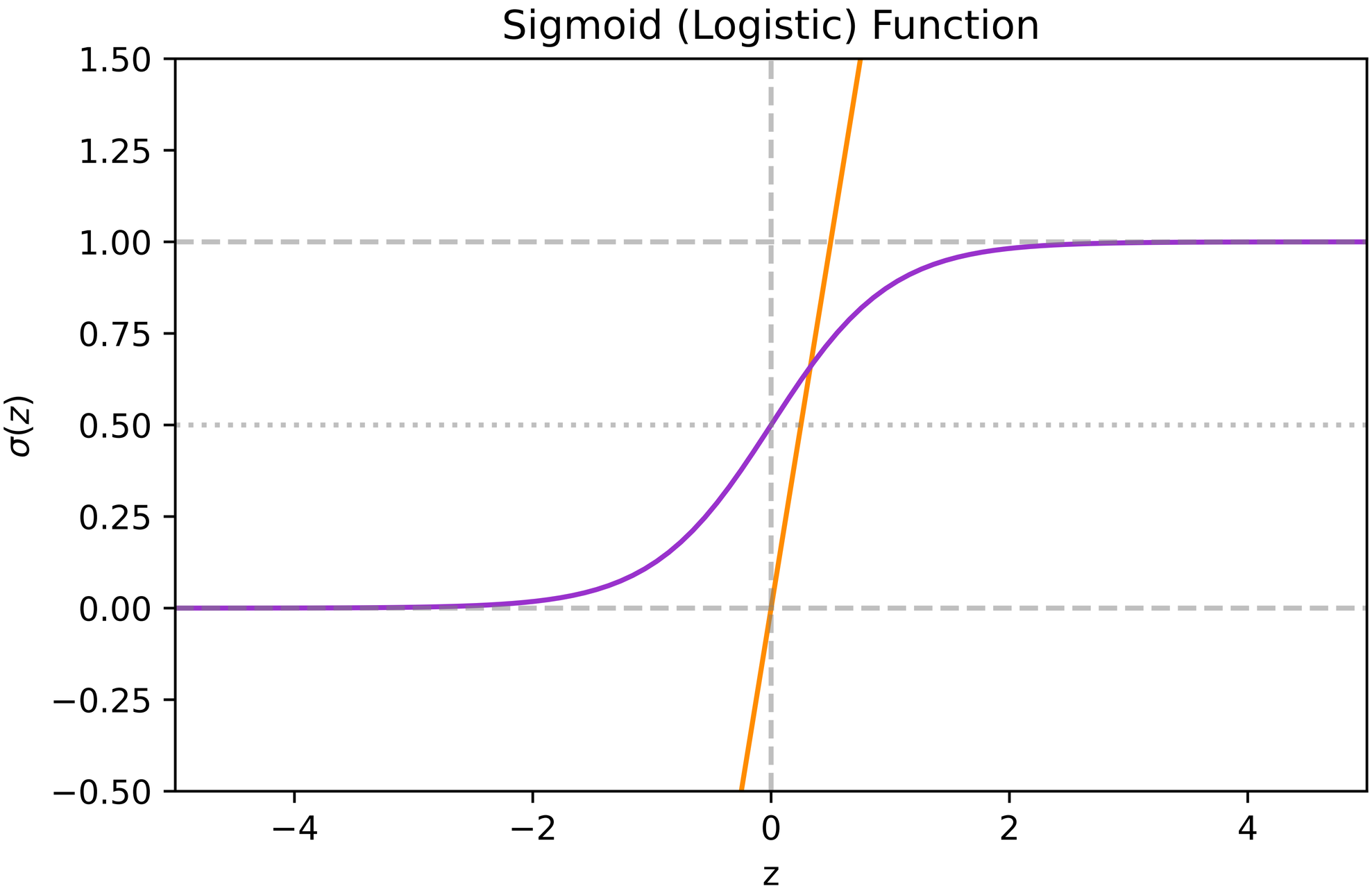
  → but probabilities need to be constrained to [0, 1]

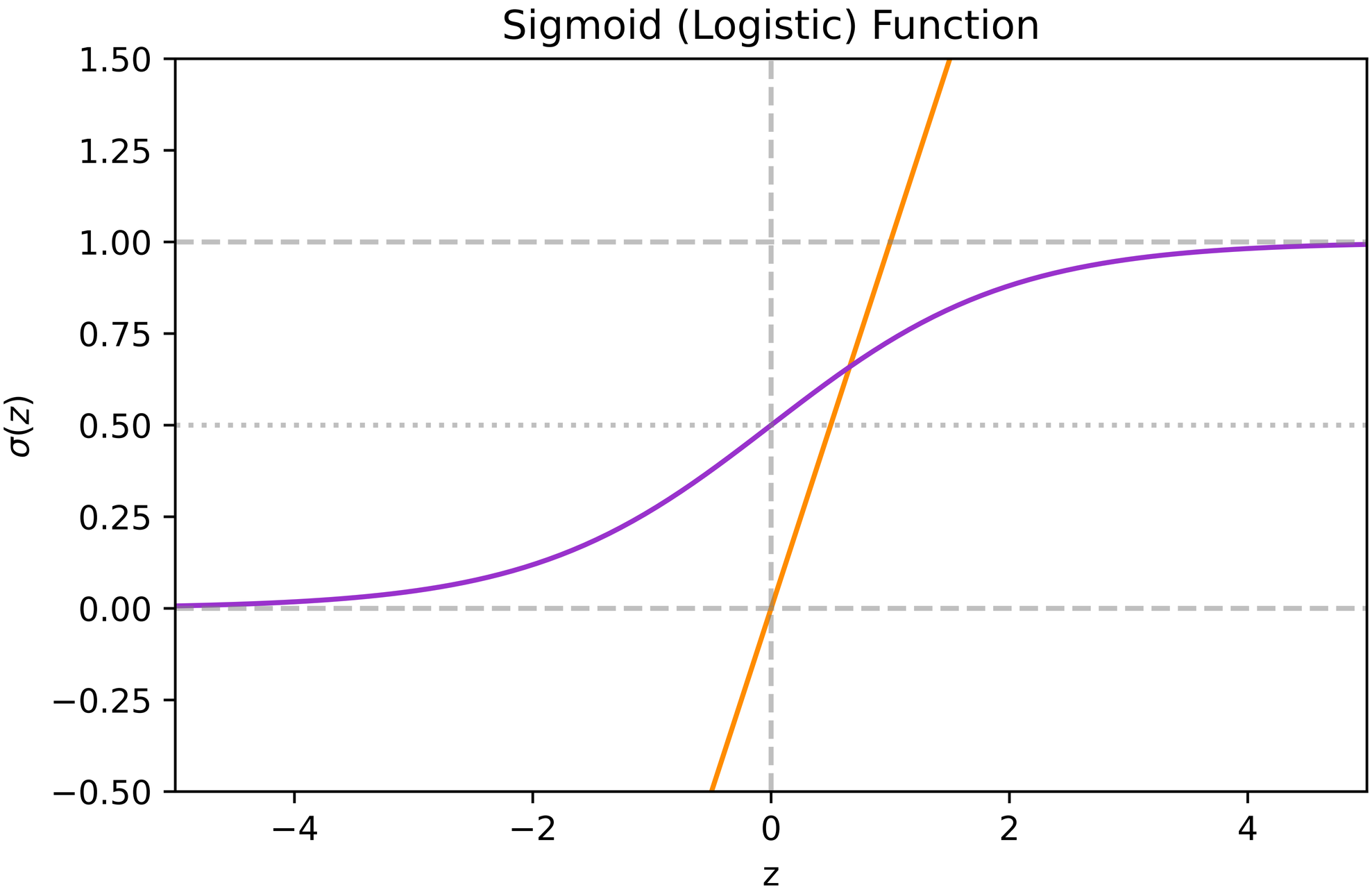
“Squashing” — the sigmoid function

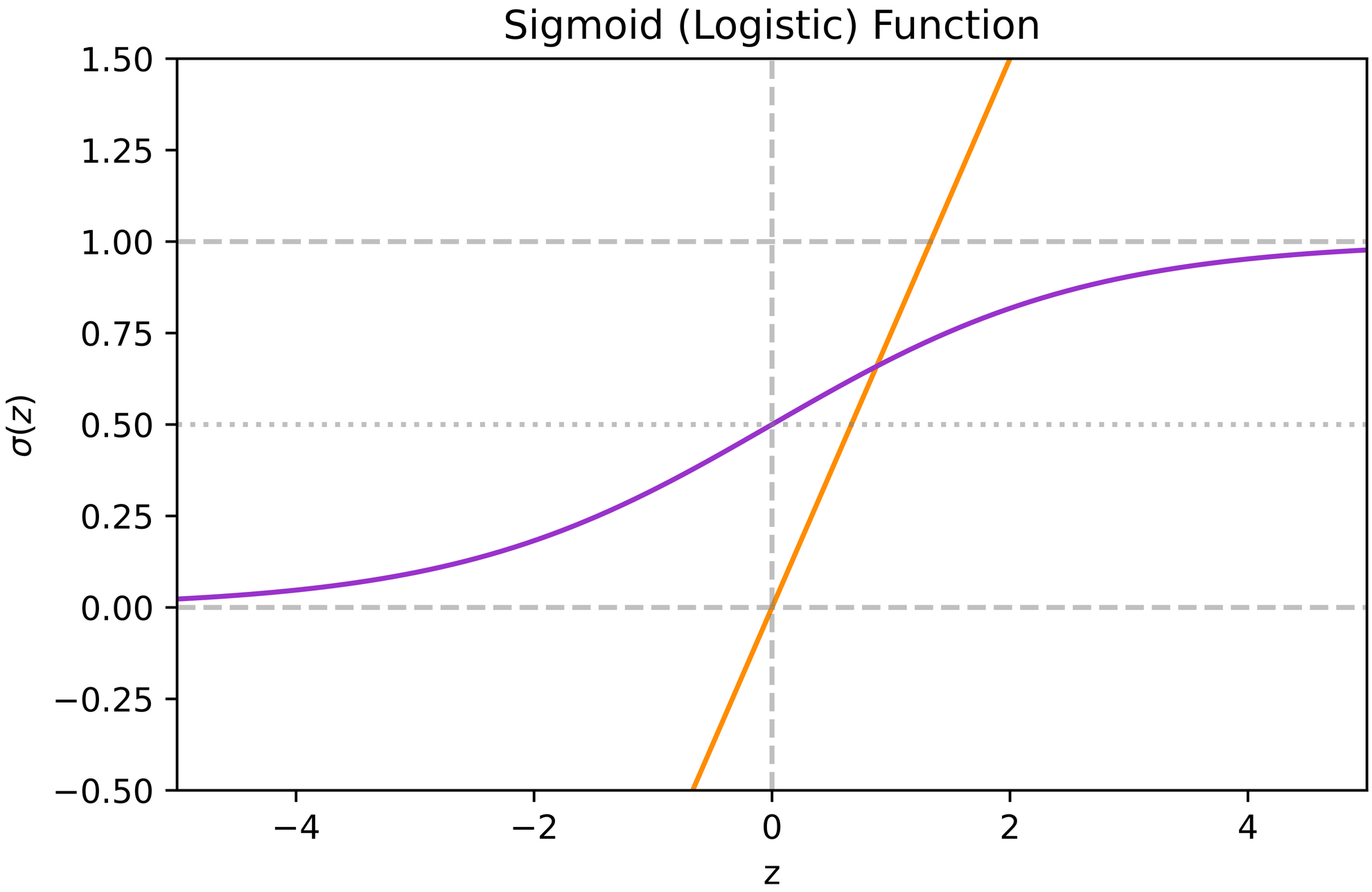
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

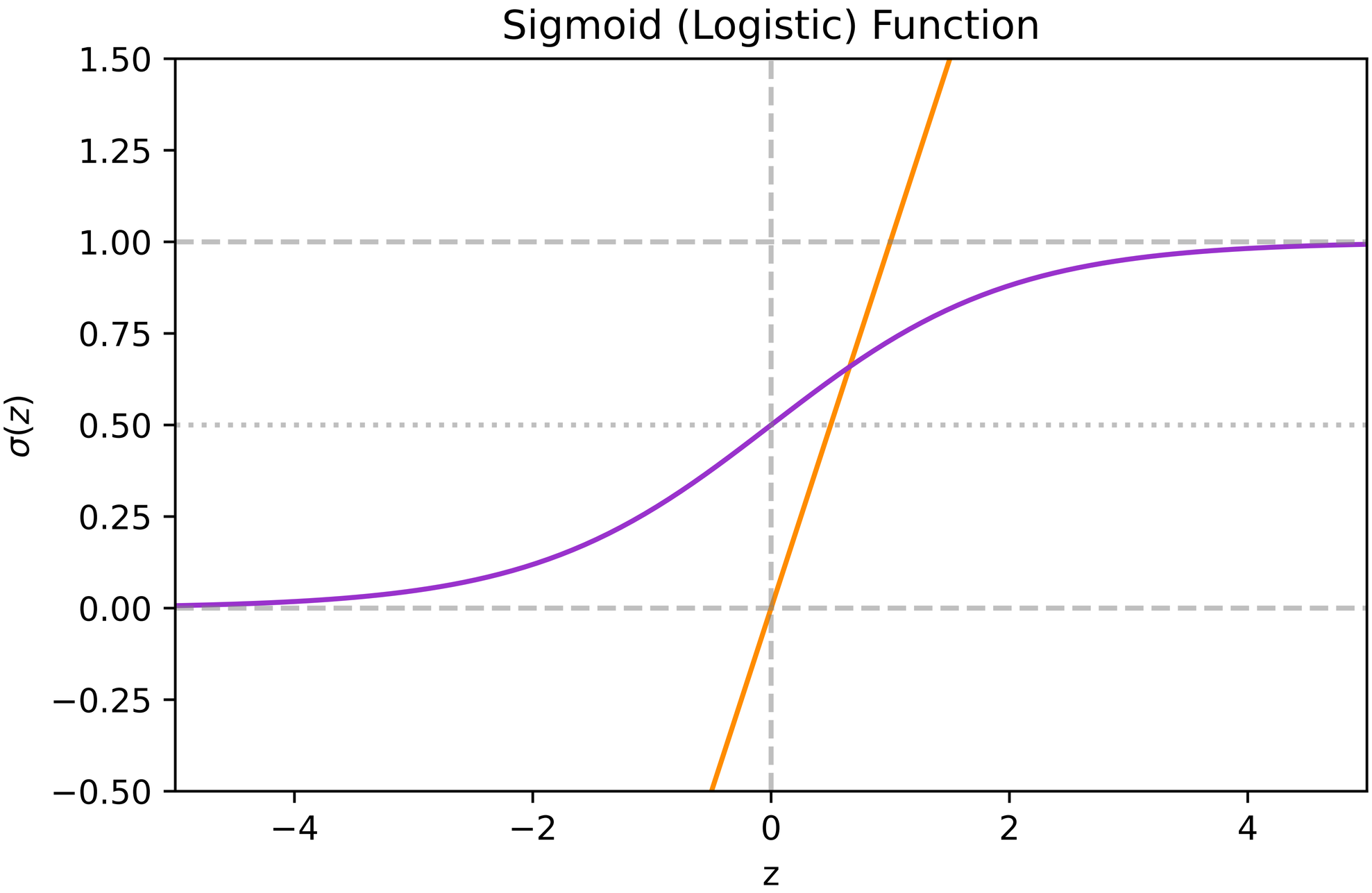
- Also called the logistic function
- Smoothly compresses the output into the required range $\mathbb{R} \mapsto (0, 1)$
- Zero threshold of linear model maps to probability of 0.5
- Linear slope determines sharpness of curve
- Linear offset shifts decision boundary

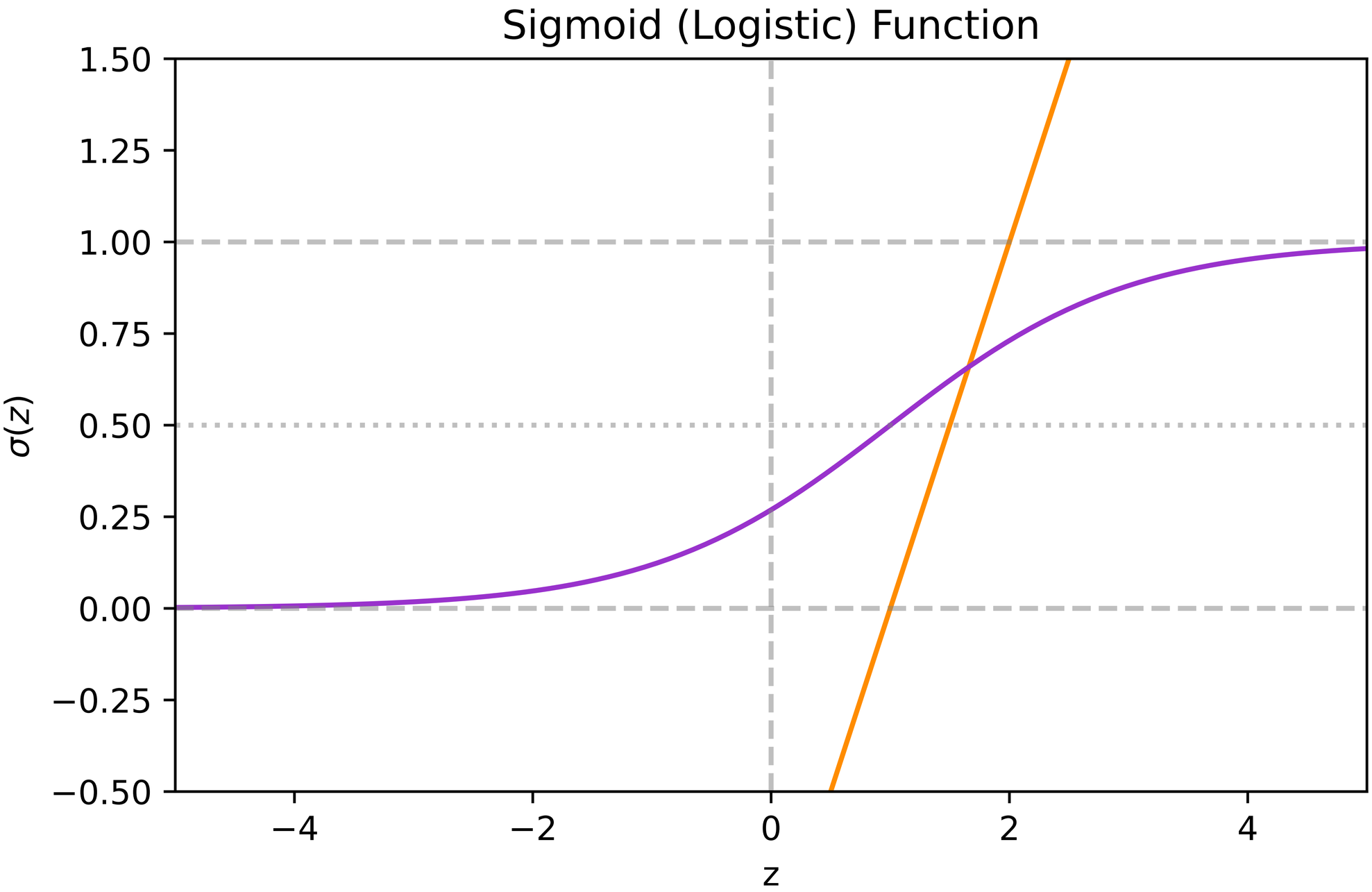

Sigmoid (Logistic) Function
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
σ(z)
−4
−2
0
2
4
z

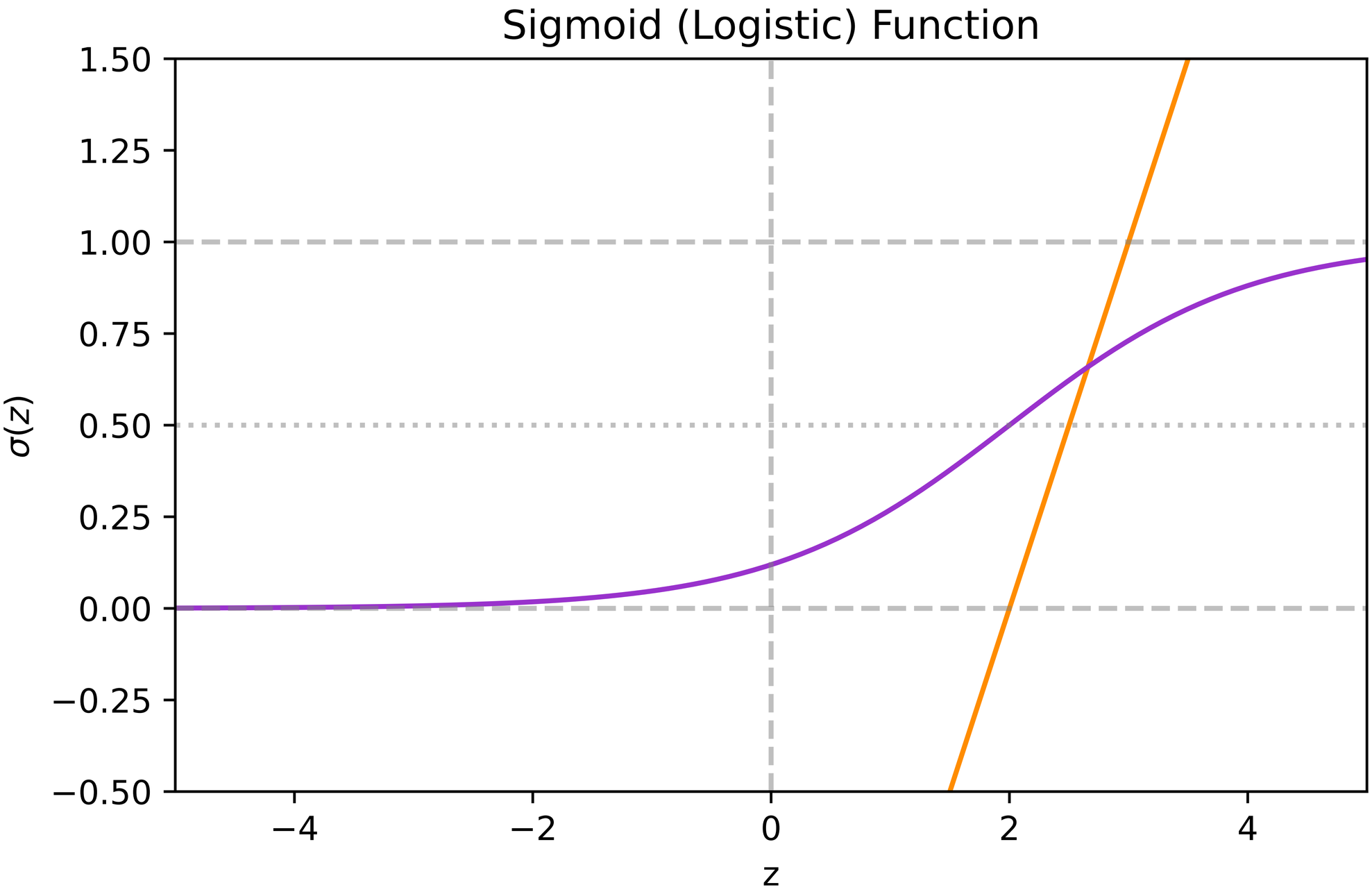

Sigmoid (Logistic) Function
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
σ(z)
−4
−2
0
2
4
z

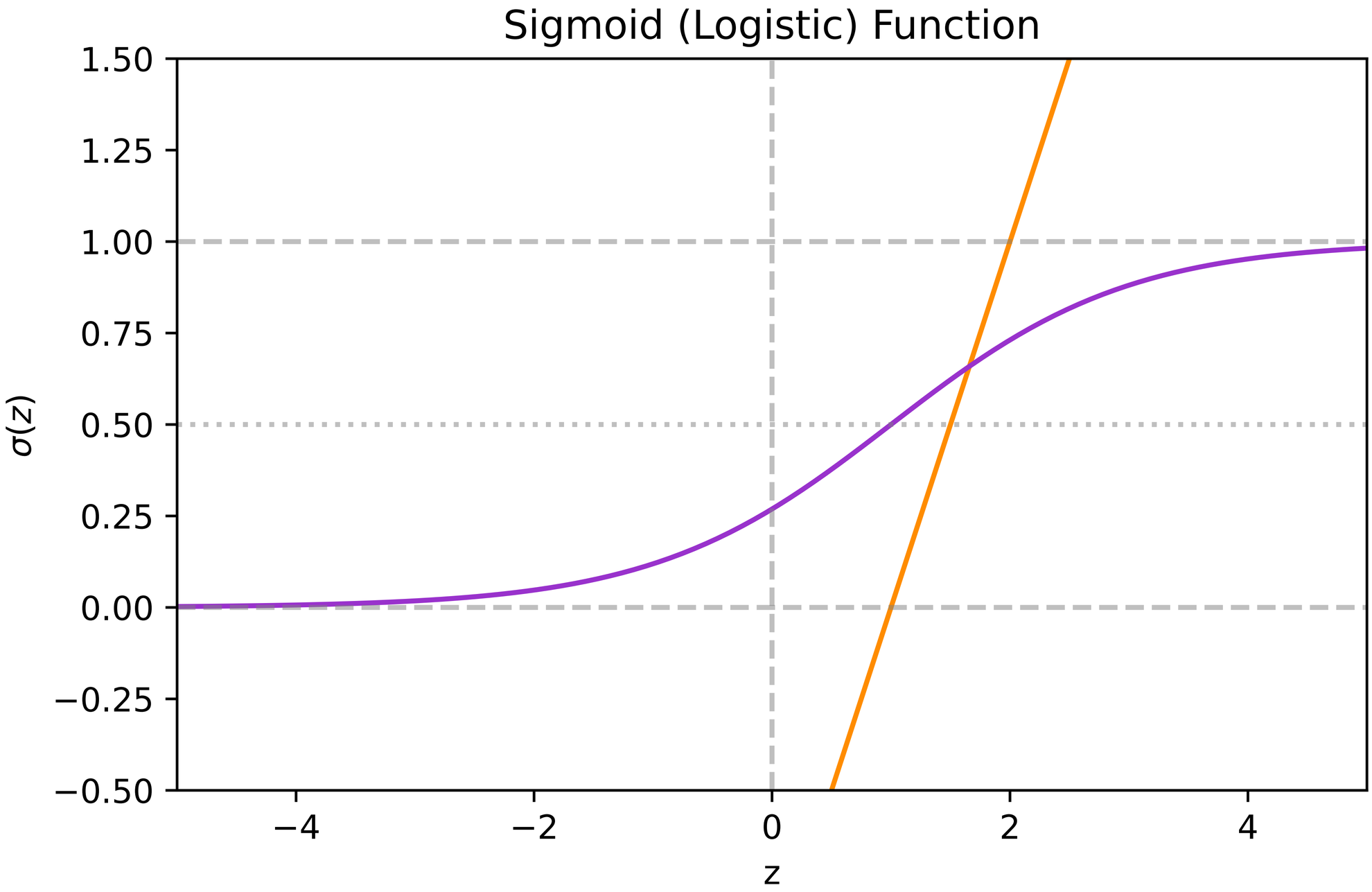

Sigmoid (Logistic) Function
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
σ(z)
−4
−2
0
2
4
z

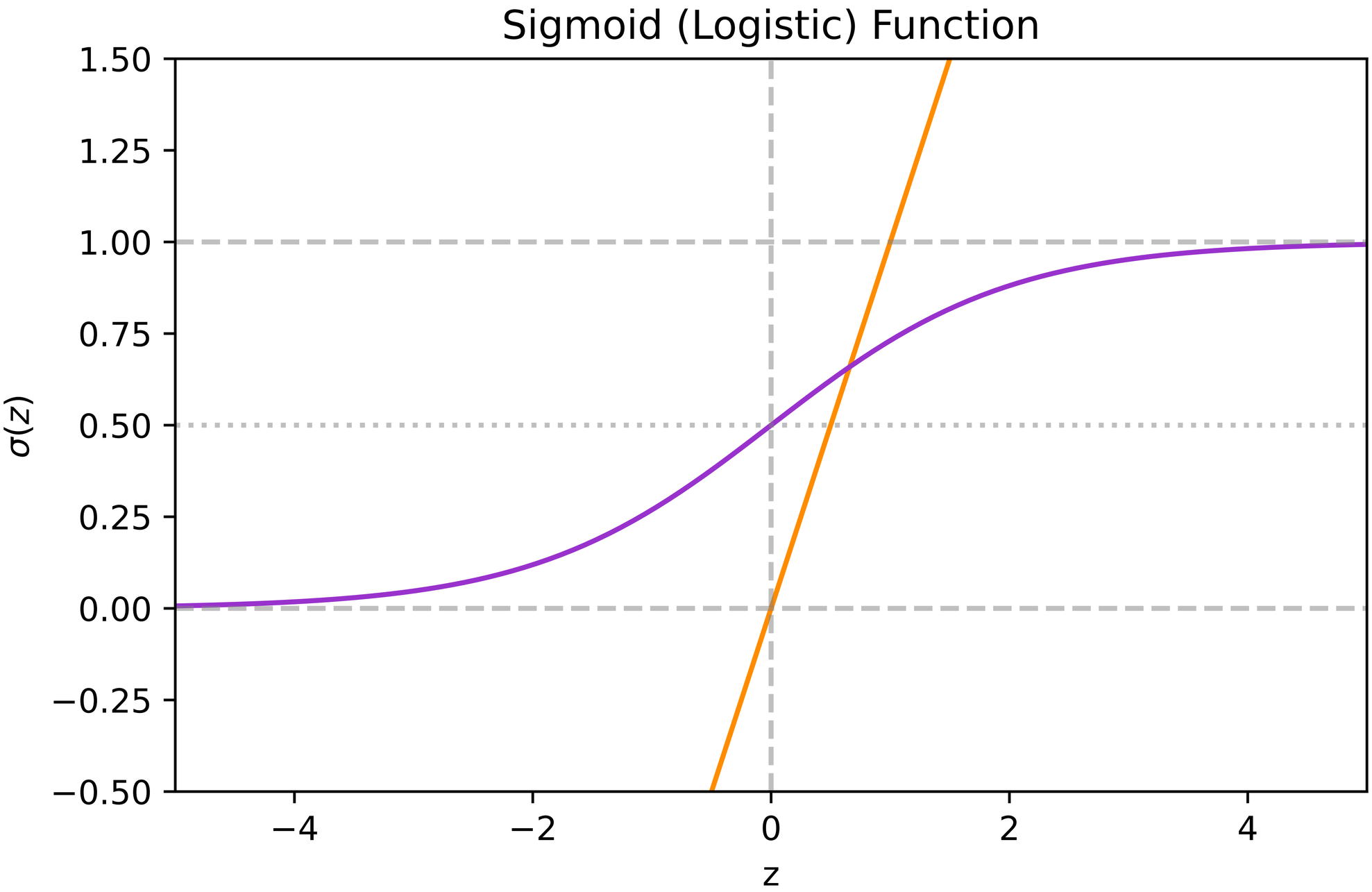

Sigmoid (Logistic) Function
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
σ(z)
−4
−2
0
2
4
z


Sigmoid (Logistic) Function
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
σ(z)
−4
−2
0
2
4
z


Sigmoid (Logistic) Function
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
σ(z)
−4
−2
0
2
4
z


Sigmoid (Logistic) Function
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
σ(z)
−4
−2
0
2
4
z


Sigmoid (Logistic) Function
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
σ(z)
−4
−2
0
2
4
z


Sigmoid (Logistic) Function
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
σ(z)
−4
−2
0
2
4
z

Probabilistic model

- Can now define a “legal” probability

$$P(y = 1 \mid \mathbf{x};\, \mathbf{w}) = \sigma(\mathbf{x} \cdot \mathbf{w})$$
$$P(y = 0 \mid \mathbf{x};\, \mathbf{w}) = 1 - \sigma(\mathbf{x} \cdot \mathbf{w})$$

- *y* is always 0 or 1 — use a notational trick to combine both cases

$$P(y \mid \mathbf{x};\, \mathbf{w}) = \sigma(\mathbf{x} \cdot \mathbf{w})^{y}(1 - \sigma(\mathbf{x} \cdot \mathbf{w}))^{1-y}$$

→ True term remains unchanged, other reduces to 1

Probabilistic model

- Assume *y*s are independent, so probability of whole training set is

$$P(\mathbf{y} \mid \mathbf{X};\, \mathbf{w}) = \prod_{i}^{n} \sigma(\mathbf{x}_i \cdot \mathbf{w})^{y_i} (1 - \sigma(\mathbf{x}_i \cdot \mathbf{w}))^{1-y_i}$$

- Once again, given **X** and **y**, this defines the likelihood of **w**:

$$L(\mathbf{w}) = \prod_{i}^{n} \sigma(\mathbf{x}_i \cdot \mathbf{w})^{y_i} (1 - \sigma(\mathbf{x}_i \cdot \mathbf{w}))^{1-y_i}$$

Maximise (log) likelihood

$$\mathbf{w}^\star = \operatorname*{argmax}_{\mathbf{w}} \; L(\mathbf{w})$$

$$= \operatorname*{argmax}_{\mathbf{w}} \prod_i^n \sigma(\mathbf{x}_i \cdot \mathbf{w})^{y_i} (1 - \sigma(\mathbf{x}_i \cdot \mathbf{w}))^{1-y_i}$$

$$= \operatorname*{argmax}_{\mathbf{w}} \sum_i^n y_i \log(\sigma(\mathbf{x}_i \cdot \mathbf{w})) + (1 - y_i) \log(1 - \sigma(\mathbf{x}_i \cdot \mathbf{w}))$$

We'll be using this expression for the time being

Maximise (log) likelihood

$$\mathbf{w}^\star = \underset{\mathbf{w}}{\operatorname{argmax}}\ L(\mathbf{w})$$

$$= \underset{\mathbf{w}}{\operatorname{argmax}} \prod_i^n \sigma(\mathbf{x}_i \cdot \mathbf{w})^{y_i} (1 - \sigma(\mathbf{x}_i \cdot \mathbf{w}))^{1-y_i}$$

$$= \underset{\mathbf{w}}{\operatorname{argmax}} \sum_i^n y_i \log(\sigma(\mathbf{x}_i \cdot \mathbf{w})) + (1 - y_i) \log(1 - \sigma(\mathbf{x}_i \cdot \mathbf{w}))$$

$$= \underset{\mathbf{w}}{\operatorname{argmax}} \sum_i^n y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$$

But could also write it this way — we'll revisit later

Sigmoid derivative

$$\sigma'(z) = \frac{d}{dz}\left(\frac{1}{1+e^{-z}}\right)$$

Sigmoid derivative

$$\sigma'(z) = \frac{d}{dz}\left(\frac{1}{1+e^{-z}}\right)$$

Apply chain rule

Sigmoid derivative

$$\sigma'(z) = \frac{d}{dz}\left(\frac{1}{1+e^{-z}}\right)$$
$$= \frac{1}{(1+e^{-z})^2}\left(e^{-z}\right)$$

Sigmoid derivative

$$\sigma'(z) = \frac{d}{dz}\left(\frac{1}{1+e^{-z}}\right)$$

$$= \frac{1}{(1+e^{-z})^2}\left(e^{-z}\right)$$

Refactor denominator

Sigmoid derivative

$$
\begin{aligned}
\sigma'(z) &= \frac{d}{dz}\left(\frac{1}{1+e^{-z}}\right) \\
&= \frac{1}{(1+e^{-z})^2}\left(e^{-z}\right) \\
&= \frac{1}{1+e^{-z}} \cdot \left(\frac{e^{-z}}{1+e^{-z}}\right)
\end{aligned}
$$

Sigmoid derivative

$$\sigma'(z) = \frac{d}{dz}\left(\frac{1}{1+e^{-z}}\right)$$

$$= \frac{1}{(1+e^{-z})^2}\left(e^{-z}\right)$$

$$= \frac{1}{1+e^{-z}} \cdot \left(\frac{e^{-z}}{1+e^{-z}}\right)$$

Rearrange fraction

Sigmoid derivative

$$
\begin{aligned}
\sigma'(z) &= \frac{d}{dz}\left(\frac{1}{1+e^{-z}}\right) \\
&= \frac{1}{(1+e^{-z})^2}\left(e^{-z}\right) \\
&= \frac{1}{1+e^{-z}} \cdot \left(\frac{e^{-z}}{1+e^{-z}}\right) \\
&= \frac{1}{1+e^{-z}} \cdot \left(1 - \frac{1}{1+e^{-z}}\right)
\end{aligned}
$$

Sigmoid derivative

$$\sigma'(z) = \frac{d}{dz}\left(\frac{1}{1+e^{-z}}\right)$$

$$= \frac{1}{(1+e^{-z})^2}\left(e^{-z}\right)$$

$$= \frac{1}{1+e^{-z}} \cdot \left(\frac{e^{-z}}{1+e^{-z}}\right)$$

$$= \frac{1}{1+e^{-z}} \cdot \left(1 - \frac{1}{1+e^{-z}}\right)$$

The original sigmoid

Sigmoid derivative

$$
\begin{aligned}
\sigma'(z) &= \frac{d}{dz}\left(\frac{1}{1+e^{-z}}\right) \\
&= \frac{1}{(1+e^{-z})^2}\left(e^{-z}\right) \\
&= \frac{1}{1+e^{-z}} \cdot \left(\frac{e^{-z}}{1+e^{-z}}\right) \\
&= \frac{1}{1+e^{-z}} \cdot \left(1 - \frac{1}{1+e^{-z}}\right) \\
&= \boxed{\sigma(z)(1-\sigma(z))}
\end{aligned}
$$

Derivative of log likelihood

- The log likelihood function looks like this:

$$l(\mathbf{w}) = \sum_{i}^{n} y_i \log(\sigma(\mathbf{x}_i \cdot \mathbf{w})) + (1 - y_i) \log(1 - \sigma(\mathbf{x}_i \cdot \mathbf{w}))$$

- Consider a limited scalar case:
  - → One single example (**x**, *y*) — so no need for sum

$$l(\mathbf{w}) = y \log(\sigma(\mathbf{x} \cdot \mathbf{w})) + (1 - y) \log(1 - \sigma(\mathbf{x} \cdot \mathbf{w}))$$

  - → Find partial derivative with respect to a single weight, w_j
    - (Still need to know all of **w**)

$$\frac{\partial}{\partial w_j} l(\mathbf{w}) = \frac{\partial}{\partial w_j} \left[y \log(\sigma(\mathbf{x} \cdot \mathbf{w})) + (1 - y) \log(1 - \sigma(\mathbf{x} \cdot \mathbf{w})) \right]$$

$$\frac{\partial}{\partial w_j} l(\mathbf{w}) = \frac{\partial}{\partial w_j} \left[y \log(\sigma(\mathbf{x} \cdot \mathbf{w})) + (1 - y) \log(1 - \sigma(\mathbf{x} \cdot \mathbf{w})) \right]$$

Apply chain rule to each term

$$\frac{\partial}{\partial w_j} l(\mathbf{w}) = \frac{\partial}{\partial w_j} \left[y \log(\sigma(\mathbf{x} \cdot \mathbf{w})) + (1-y) \log(1 - \sigma(\mathbf{x} \cdot \mathbf{w})) \right]$$

$$= \left[y \frac{1}{\sigma(\mathbf{x} \cdot \mathbf{w})} \frac{\partial}{\partial w_j} \sigma(\mathbf{x} \cdot \mathbf{w}) \right] + \left[(1-y) \frac{1}{1 - \sigma(\mathbf{x} \cdot \mathbf{w})} \left(- \frac{\partial}{\partial w_j} \sigma(\mathbf{x} \cdot \mathbf{w}) \right) \right]$$

$$\frac{\partial}{\partial w_j} l(\mathbf{w}) = \frac{\partial}{\partial w_j}\left[y \log(\sigma(\mathbf{x}\cdot\mathbf{w})) + (1-y)\log(1-\sigma(\mathbf{x}\cdot\mathbf{w}))\right]$$

$$= \left[y \frac{1}{\sigma(\mathbf{x}\cdot\mathbf{w})} \frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right] + \left[(1-y)\frac{1}{1-\sigma(\mathbf{x}\cdot\mathbf{w})}\left(-\frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right)\right]$$

Move negative outside brackets

$$
\begin{aligned}
\frac{\partial}{\partial w_j} l(\mathbf{w}) &= \frac{\partial}{\partial w_j}\left[y \log(\sigma(\mathbf{x}\cdot\mathbf{w})) + (1-y)\log(1-\sigma(\mathbf{x}\cdot\mathbf{w}))\right] \\
&= \left[y \frac{1}{\sigma(\mathbf{x}\cdot\mathbf{w})} \frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right] + \left[(1-y)\frac{1}{1-\sigma(\mathbf{x}\cdot\mathbf{w})}\left(-\frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right)\right] \\
&= \left[y \frac{1}{\sigma(\mathbf{x}\cdot\mathbf{w})} \frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right] - \left[(1-y)\frac{1}{1-\sigma(\mathbf{x}\cdot\mathbf{w})}\frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right]
\end{aligned}
$$

$$\frac{\partial}{\partial w_j} l(\mathbf{w}) = \frac{\partial}{\partial w_j}\left[y \log(\sigma(\mathbf{x}\cdot\mathbf{w})) + (1-y)\log(1-\sigma(\mathbf{x}\cdot\mathbf{w}))\right]$$

$$= \left[y \frac{1}{\sigma(\mathbf{x}\cdot\mathbf{w})} \frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right] + \left[(1-y)\frac{1}{1-\sigma(\mathbf{x}\cdot\mathbf{w})}\left(-\frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right)\right]$$

$$= \left[y \frac{1}{\sigma(\mathbf{x}\cdot\mathbf{w})} \frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right] - \left[(1-y)\frac{1}{1-\sigma(\mathbf{x}\cdot\mathbf{w})} \frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right]$$

Pull out shared factor

$$
\begin{aligned}
\frac{\partial}{\partial w_j} l(\mathbf{w}) &= \frac{\partial}{\partial w_j}\left[y \log(\sigma(\mathbf{x}\cdot\mathbf{w})) + (1-y)\log(1-\sigma(\mathbf{x}\cdot\mathbf{w}))\right] \\
&= \left[y \frac{1}{\sigma(\mathbf{x}\cdot\mathbf{w})} \frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right] + \left[(1-y)\frac{1}{1-\sigma(\mathbf{x}\cdot\mathbf{w})}\left(-\frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right)\right] \\
&= \left[y \frac{1}{\sigma(\mathbf{x}\cdot\mathbf{w})} \frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right] - \left[(1-y)\frac{1}{1-\sigma(\mathbf{x}\cdot\mathbf{w})}\frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right] \\
&= \left[y \frac{1}{\sigma(\mathbf{x}\cdot\mathbf{w})} - (1-y)\frac{1}{1-\sigma(\mathbf{x}\cdot\mathbf{w})}\right]\frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})
\end{aligned}
$$

$$\frac{\partial}{\partial w_j} l(\mathbf{w}) = \frac{\partial}{\partial w_j}\left[y \log(\sigma(\mathbf{x}\cdot\mathbf{w})) + (1-y)\log(1-\sigma(\mathbf{x}\cdot\mathbf{w}))\right]$$

$$= \left[y \frac{1}{\sigma(\mathbf{x}\cdot\mathbf{w})} \frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right] + \left[(1-y)\frac{1}{1-\sigma(\mathbf{x}\cdot\mathbf{w})}\left(-\frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right)\right]$$

$$= \left[y \frac{1}{\sigma(\mathbf{x}\cdot\mathbf{w})} \frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right] - \left[(1-y)\frac{1}{1-\sigma(\mathbf{x}\cdot\mathbf{w})} \frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right]$$

$$= \left[y \frac{1}{\sigma(\mathbf{x}\cdot\mathbf{w})} - (1-y)\frac{1}{1-\sigma(\mathbf{x}\cdot\mathbf{w})}\right] \frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})$$

Apply chain rule

$$
\begin{aligned}
\frac{\partial}{\partial w_j} l(\mathbf{w}) &= \frac{\partial}{\partial w_j}\left[y \log(\sigma(\mathbf{x}\cdot\mathbf{w})) + (1-y)\log(1-\sigma(\mathbf{x}\cdot\mathbf{w}))\right] \\
&= \left[y \frac{1}{\sigma(\mathbf{x}\cdot\mathbf{w})} \frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right] + \left[(1-y)\frac{1}{1-\sigma(\mathbf{x}\cdot\mathbf{w})}\left(-\frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right)\right] \\
&= \left[y \frac{1}{\sigma(\mathbf{x}\cdot\mathbf{w})} \frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right] - \left[(1-y)\frac{1}{1-\sigma(\mathbf{x}\cdot\mathbf{w})}\frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right] \\
&= \left[y \frac{1}{\sigma(\mathbf{x}\cdot\mathbf{w})} - (1-y)\frac{1}{1-\sigma(\mathbf{x}\cdot\mathbf{w})}\right]\frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w}) \\
&= \left[y \frac{1}{\sigma(\mathbf{x}\cdot\mathbf{w})} - (1-y)\frac{1}{1-\sigma(\mathbf{x}\cdot\mathbf{w})}\right]\sigma'(\mathbf{x}\cdot\mathbf{w})\frac{\partial}{\partial w_j}(\mathbf{x}\cdot\mathbf{w})
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial}{\partial w_j} l(\mathbf{w}) &= \frac{\partial}{\partial w_j}\left[y \log(\sigma(\mathbf{x}\cdot\mathbf{w})) + (1-y)\log(1-\sigma(\mathbf{x}\cdot\mathbf{w}))\right] \\
&= \left[y \frac{1}{\sigma(\mathbf{x}\cdot\mathbf{w})} \frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right] + \left[(1-y)\frac{1}{1-\sigma(\mathbf{x}\cdot\mathbf{w})}\left(-\frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right)\right] \\
&= \left[y \frac{1}{\sigma(\mathbf{x}\cdot\mathbf{w})} \frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right] - \left[(1-y)\frac{1}{1-\sigma(\mathbf{x}\cdot\mathbf{w})} \frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w})\right] \\
&= \left[y \frac{1}{\sigma(\mathbf{x}\cdot\mathbf{w})} - (1-y)\frac{1}{1-\sigma(\mathbf{x}\cdot\mathbf{w})}\right] \frac{\partial}{\partial w_j}\sigma(\mathbf{x}\cdot\mathbf{w}) \\
&= \left[y \frac{1}{\sigma(\mathbf{x}\cdot\mathbf{w})} - (1-y)\frac{1}{1-\sigma(\mathbf{x}\cdot\mathbf{w})}\right] \sigma'(\mathbf{x}\cdot\mathbf{w}) \frac{\partial}{\partial w_j}(\mathbf{x}\cdot\mathbf{w})
\end{aligned}
$$

But wait, there's more!

$$\frac{\partial}{\partial w_j} l(\mathbf{w}) = \left[y \frac{1}{\sigma(\mathbf{x} \cdot \mathbf{w})} - (1 - y) \frac{1}{1 - \sigma(\mathbf{x} \cdot \mathbf{w})} \right] \sigma'(\mathbf{x} \cdot \mathbf{w}) \frac{\partial}{\partial w_j} (\mathbf{x} \cdot \mathbf{w})$$

$$\frac{\partial}{\partial w_j} l(\mathbf{w}) = \left[y \frac{1}{\sigma(\mathbf{x} \cdot \mathbf{w})} - (1 - y) \frac{1}{1 - \sigma(\mathbf{x} \cdot \mathbf{w})} \right] \sigma'(\mathbf{x} \cdot \mathbf{w}) \frac{\partial}{\partial w_j} (\mathbf{x} \cdot \mathbf{w})$$

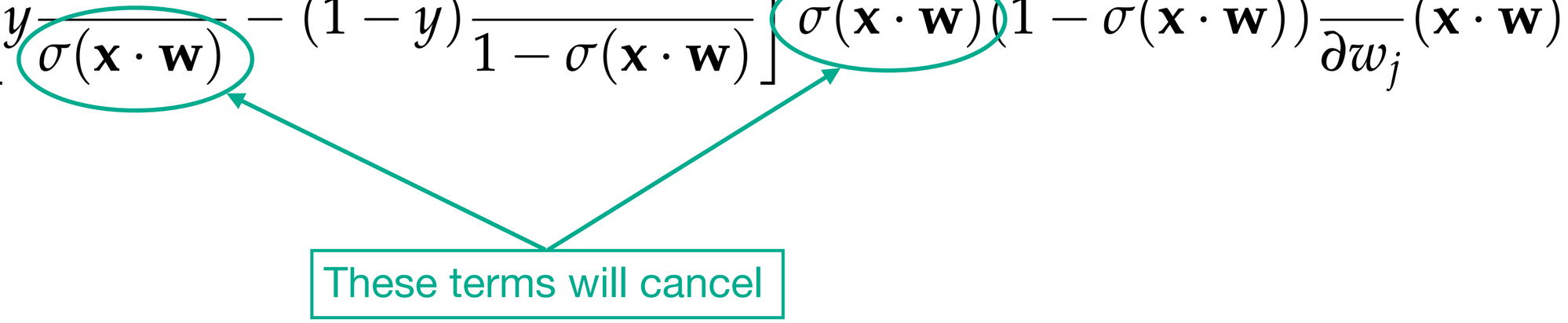
Sigmoid derivative — we computed this a few slides back

$$\frac{\partial}{\partial w_j} l(\mathbf{w}) = \left[y \frac{1}{\sigma(\mathbf{x} \cdot \mathbf{w})} - (1-y) \frac{1}{1-\sigma(\mathbf{x} \cdot \mathbf{w})} \right] \sigma'(\mathbf{x} \cdot \mathbf{w}) \frac{\partial}{\partial w_j} (\mathbf{x} \cdot \mathbf{w})$$

$$= \left[y \frac{1}{\sigma(\mathbf{x} \cdot \mathbf{w})} - (1-y) \frac{1}{1-\sigma(\mathbf{x} \cdot \mathbf{w})} \right] \sigma(\mathbf{x} \cdot \mathbf{w})(1-\sigma(\mathbf{x} \cdot \mathbf{w})) \frac{\partial}{\partial w_j} (\mathbf{x} \cdot \mathbf{w})$$

$$\frac{\partial}{\partial w_j} l(\mathbf{w}) = \left[y \frac{1}{\sigma(\mathbf{x} \cdot \mathbf{w})} - (1 - y) \frac{1}{1 - \sigma(\mathbf{x} \cdot \mathbf{w})} \right] \sigma'(\mathbf{x} \cdot \mathbf{w}) \frac{\partial}{\partial w_j} (\mathbf{x} \cdot \mathbf{w})$$

$$= \left[y \frac{1}{\sigma(\mathbf{x} \cdot \mathbf{w})} - (1 - y) \frac{1}{1 - \sigma(\mathbf{x} \cdot \mathbf{w})} \right] \sigma(\mathbf{x} \cdot \mathbf{w})(1 - \sigma(\mathbf{x} \cdot \mathbf{w})) \frac{\partial}{\partial w_j} (\mathbf{x} \cdot \mathbf{w})$$



$$\frac{\partial}{\partial w_j} l(\mathbf{w}) = \left[y \frac{1}{\sigma(\mathbf{x} \cdot \mathbf{w})} - (1-y) \frac{1}{1-\sigma(\mathbf{x} \cdot \mathbf{w})} \right] \sigma'(\mathbf{x} \cdot \mathbf{w}) \frac{\partial}{\partial w_j} (\mathbf{x} \cdot \mathbf{w})$$

$$= \left[y \frac{1}{\sigma(\mathbf{x} \cdot \mathbf{w})} - (1-y) \frac{1}{1-\sigma(\mathbf{x} \cdot \mathbf{w})} \right] \sigma(\mathbf{x} \cdot \mathbf{w})(1-\sigma(\mathbf{x} \cdot \mathbf{w})) \frac{\partial}{\partial w_j} (\mathbf{x} \cdot \mathbf{w})$$

So will these

$$\frac{\partial}{\partial w_j} l(\mathbf{w}) = \left[y \frac{1}{\sigma(\mathbf{x} \cdot \mathbf{w})} - (1 - y) \frac{1}{1 - \sigma(\mathbf{x} \cdot \mathbf{w})} \right] \sigma'(\mathbf{x} \cdot \mathbf{w}) \frac{\partial}{\partial w_j} (\mathbf{x} \cdot \mathbf{w})$$

$$= \left[y \frac{1}{\sigma(\mathbf{x} \cdot \mathbf{w})} - (1 - y) \frac{1}{1 - \sigma(\mathbf{x} \cdot \mathbf{w})} \right] \sigma(\mathbf{x} \cdot \mathbf{w})(1 - \sigma(\mathbf{x} \cdot \mathbf{w})) \frac{\partial}{\partial w_j} (\mathbf{x} \cdot \mathbf{w})$$

And this is just x_j

$$
\begin{aligned}
\frac{\partial}{\partial w_j} l(\mathbf{w}) &= \left[y \frac{1}{\sigma(\mathbf{x} \cdot \mathbf{w})} - (1-y) \frac{1}{1-\sigma(\mathbf{x} \cdot \mathbf{w})} \right] \sigma'(\mathbf{x} \cdot \mathbf{w}) \frac{\partial}{\partial w_j} (\mathbf{x} \cdot \mathbf{w}) \\
&= \left[y \frac{1}{\sigma(\mathbf{x} \cdot \mathbf{w})} - (1-y) \frac{1}{1-\sigma(\mathbf{x} \cdot \mathbf{w})} \right] \sigma(\mathbf{x} \cdot \mathbf{w})(1-\sigma(\mathbf{x} \cdot \mathbf{w})) \frac{\partial}{\partial w_j} (\mathbf{x} \cdot \mathbf{w}) \\
&= (y(1-\sigma(\mathbf{x} \cdot \mathbf{w})) - (1-y)\sigma(\mathbf{x} \cdot \mathbf{w})) \cdot x_j
\end{aligned}
$$

$$\frac{\partial}{\partial w_j} l(\mathbf{w}) = \left[y \frac{1}{\sigma(\mathbf{x} \cdot \mathbf{w})} - (1 - y) \frac{1}{1 - \sigma(\mathbf{x} \cdot \mathbf{w})} \right] \sigma'(\mathbf{x} \cdot \mathbf{w}) \frac{\partial}{\partial w_j} (\mathbf{x} \cdot \mathbf{w})$$

$$= \left[y \frac{1}{\sigma(\mathbf{x} \cdot \mathbf{w})} - (1 - y) \frac{1}{1 - \sigma(\mathbf{x} \cdot \mathbf{w})} \right] \sigma(\mathbf{x} \cdot \mathbf{w})(1 - \sigma(\mathbf{x} \cdot \mathbf{w})) \frac{\partial}{\partial w_j} (\mathbf{x} \cdot \mathbf{w})$$

$$= (y(1 - \sigma(\mathbf{x} \cdot \mathbf{w})) - (1 - y)\sigma(\mathbf{x} \cdot \mathbf{w})) \cdot x_j$$

Multiply out

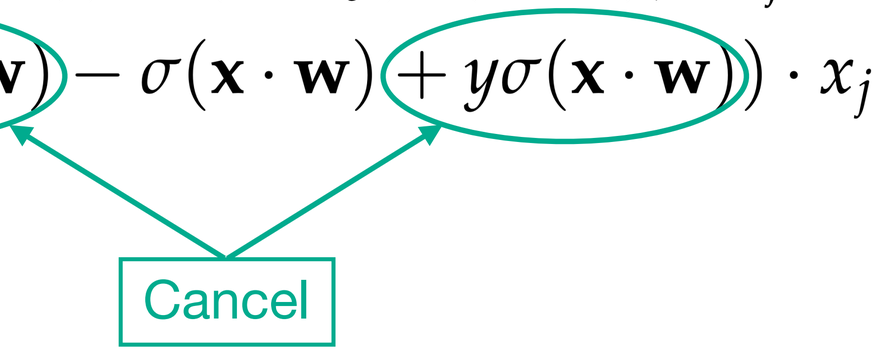
$$
\begin{aligned}
\frac{\partial}{\partial w_j} l(\mathbf{w}) &= \left[y \frac{1}{\sigma(\mathbf{x} \cdot \mathbf{w})} - (1-y) \frac{1}{1-\sigma(\mathbf{x} \cdot \mathbf{w})} \right] \sigma'(\mathbf{x} \cdot \mathbf{w}) \frac{\partial}{\partial w_j} (\mathbf{x} \cdot \mathbf{w}) \\
&= \left[y \frac{1}{\sigma(\mathbf{x} \cdot \mathbf{w})} - (1-y) \frac{1}{1-\sigma(\mathbf{x} \cdot \mathbf{w})} \right] \sigma(\mathbf{x} \cdot \mathbf{w})(1-\sigma(\mathbf{x} \cdot \mathbf{w})) \frac{\partial}{\partial w_j} (\mathbf{x} \cdot \mathbf{w}) \\
&= (y(1-\sigma(\mathbf{x} \cdot \mathbf{w})) - (1-y)\sigma(\mathbf{x} \cdot \mathbf{w})) \cdot x_j \\
&= (y - y\sigma(\mathbf{x} \cdot \mathbf{w}) - \sigma(\mathbf{x} \cdot \mathbf{w}) + y\sigma(\mathbf{x} \cdot \mathbf{w})) \cdot x_j
\end{aligned}
$$

$$\frac{\partial}{\partial w_j} l(\mathbf{w}) = \left[y \frac{1}{\sigma(\mathbf{x} \cdot \mathbf{w})} - (1 - y) \frac{1}{1 - \sigma(\mathbf{x} \cdot \mathbf{w})} \right] \sigma'(\mathbf{x} \cdot \mathbf{w}) \frac{\partial}{\partial w_j} (\mathbf{x} \cdot \mathbf{w})$$

$$= \left[y \frac{1}{\sigma(\mathbf{x} \cdot \mathbf{w})} - (1 - y) \frac{1}{1 - \sigma(\mathbf{x} \cdot \mathbf{w})} \right] \sigma(\mathbf{x} \cdot \mathbf{w})(1 - \sigma(\mathbf{x} \cdot \mathbf{w})) \frac{\partial}{\partial w_j} (\mathbf{x} \cdot \mathbf{w})$$

$$= (y(1 - \sigma(\mathbf{x} \cdot \mathbf{w})) - (1 - y)\sigma(\mathbf{x} \cdot \mathbf{w})) \cdot x_j$$

$$= (y - y\sigma(\mathbf{x} \cdot \mathbf{w}) - \sigma(\mathbf{x} \cdot \mathbf{w}) + y\sigma(\mathbf{x} \cdot \mathbf{w})) \cdot x_j$$



$$
\begin{aligned}
\frac{\partial}{\partial w_j} l(\mathbf{w}) &= \left[y \frac{1}{\sigma(\mathbf{x} \cdot \mathbf{w})} - (1-y) \frac{1}{1-\sigma(\mathbf{x} \cdot \mathbf{w})} \right] \sigma'(\mathbf{x} \cdot \mathbf{w}) \frac{\partial}{\partial w_j} (\mathbf{x} \cdot \mathbf{w}) \\
&= \left[y \frac{1}{\sigma(\mathbf{x} \cdot \mathbf{w})} - (1-y) \frac{1}{1-\sigma(\mathbf{x} \cdot \mathbf{w})} \right] \sigma(\mathbf{x} \cdot \mathbf{w})(1-\sigma(\mathbf{x} \cdot \mathbf{w})) \frac{\partial}{\partial w_j} (\mathbf{x} \cdot \mathbf{w}) \\
&= (y(1-\sigma(\mathbf{x} \cdot \mathbf{w})) - (1-y)\sigma(\mathbf{x} \cdot \mathbf{w})) \cdot x_j \\
&= (y - y\sigma(\mathbf{x} \cdot \mathbf{w}) - \sigma(\mathbf{x} \cdot \mathbf{w}) + y\sigma(\mathbf{x} \cdot \mathbf{w})) \cdot x_j \\
&= (y - \sigma(\mathbf{x} \cdot \mathbf{w})) \cdot x_j
\end{aligned}
$$

$$\frac{\partial}{\partial w_j} l(\mathbf{w}) = \left[y \frac{1}{\sigma(\mathbf{x} \cdot \mathbf{w})} - (1 - y) \frac{1}{1 - \sigma(\mathbf{x} \cdot \mathbf{w})} \right] \sigma'(\mathbf{x} \cdot \mathbf{w}) \frac{\partial}{\partial w_j} (\mathbf{x} \cdot \mathbf{w})$$

$$= \left[y \frac{1}{\sigma(\mathbf{x} \cdot \mathbf{w})} - (1 - y) \frac{1}{1 - \sigma(\mathbf{x} \cdot \mathbf{w})} \right] \sigma(\mathbf{x} \cdot \mathbf{w})(1 - \sigma(\mathbf{x} \cdot \mathbf{w})) \frac{\partial}{\partial w_j} (\mathbf{x} \cdot \mathbf{w})$$

$$= (y(1 - \sigma(\mathbf{x} \cdot \mathbf{w})) - (1 - y)\sigma(\mathbf{x} \cdot \mathbf{w})) \cdot x_j$$

$$= (y - y\sigma(\mathbf{x} \cdot \mathbf{w}) - \sigma(\mathbf{x} \cdot \mathbf{w}) + y\sigma(\mathbf{x} \cdot \mathbf{w})) \cdot x_j$$

$$= (y - \sigma(\mathbf{x} \cdot \mathbf{w})) \cdot x_j$$

This is our predicted value, $\hat{y}$

$$
\begin{aligned}
\frac{\partial}{\partial w_j} l(\mathbf{w}) &= \left[y \frac{1}{\sigma(\mathbf{x} \cdot \mathbf{w})} - (1-y) \frac{1}{1-\sigma(\mathbf{x} \cdot \mathbf{w})} \right] \sigma'(\mathbf{x} \cdot \mathbf{w}) \frac{\partial}{\partial w_j} (\mathbf{x} \cdot \mathbf{w}) \\
&= \left[y \frac{1}{\sigma(\mathbf{x} \cdot \mathbf{w})} - (1-y) \frac{1}{1-\sigma(\mathbf{x} \cdot \mathbf{w})} \right] \sigma(\mathbf{x} \cdot \mathbf{w})(1-\sigma(\mathbf{x} \cdot \mathbf{w})) \frac{\partial}{\partial w_j} (\mathbf{x} \cdot \mathbf{w}) \\
&= (y(1-\sigma(\mathbf{x} \cdot \mathbf{w})) - (1-y)\sigma(\mathbf{x} \cdot \mathbf{w})) \cdot x_j \\
&= (y - y\sigma(\mathbf{x} \cdot \mathbf{w}) - \sigma(\mathbf{x} \cdot \mathbf{w}) + y\sigma(\mathbf{x} \cdot \mathbf{w})) \cdot x_j \\
&= (y - \sigma(\mathbf{x} \cdot \mathbf{w})) \cdot x_j \\
&= (y - \hat{y}) x_j
\end{aligned}
$$

So…

- For single weight w_j:

$$\frac{\partial}{\partial w_j} l(\mathbf{w}) = (y - \hat{y}) x_j$$

- Repeat for all j:

$$\nabla_{\mathbf{w}} l = (y - \hat{y}) \mathbf{x}$$

- Aggregate over all samples:

$$\nabla_{\mathbf{w}} l = \mathbf{X}^{\mathsf{T}} (\mathbf{y} - \hat{\mathbf{y}})$$

After all that…

- We seem to have a neat expression for the log likelihood gradient:

$$\nabla_{\mathbf{w}} l = \mathbf{X}^{\mathsf{T}}(\mathbf{y} - \hat{\mathbf{y}})$$

- But remember it's non-linear:

$$= \mathbf{X}^{\mathsf{T}}(\mathbf{y} - \sigma(\mathbf{X}\mathbf{w}))$$

$$= \mathbf{X}^{\mathsf{T}}\left(\mathbf{y} - \frac{1}{1 + e^{-\mathbf{X}\mathbf{w}}}\right)$$

- When we set it to 0, we can't solve it!

The good news

- At least we have a gradient
- Also the log likelihood is concave
- So (as we'll see in the next chapter) we're basically sorted

- But first…

Aside #1: There has to be a better way!

- Manually assembling algebraic derivatives for chained functions is tedious and error prone

  → And logistic regression is one of the simplest cases!

- But the essence of the chain rule is simple and mechanical

- We'll come back to this in week 5

Aside #2: Loss

- We never defined a loss function, just jumped straight to likelihood
- We can always flip one to get the other

  → Loss = negative (log) likelihood
- And we more or less will do that

- But recall this “alternative” expression from our maximisation:

$$\mathbf{w}^\star = \underset{\mathbf{w}}{\operatorname{argmax}} \sum_i^n y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$$

- This turns out to have a useful interpretation in information theory

Entropy

- Difficulty of sending a message depends on how surprising it is

  → If all messages are the same, you only need ditto: 〃

- This is quantified as its information:

$$h(z) = -\log P(z)$$

- The surprisingness of the whole distribution is given by its entropy:

$$H(Z) = -\sum_{z \in Z} P(z) \log P(z)$$

Cross entropy

- An encoding scheme characterises the distribution of messages
  - → Ideally, it should make it cheap to send messages you send often
- If the distribution implied by the encoding doesn't match the true distribution, communication will be inefficient
- Distributional mismatch is quantified by cross entropy:

$$H(P,Q) = -\sum_{z \in Z} P(z) \log Q(z)$$

  - → Note asymmetry: *H(P,Q) ≠ H(Q,P)*. Reality and model are not interchangeable!

Binary cross entropy

- In the binary case:

$$P(z=0) = 1 - P(z=1) \qquad Q(z=0) = 1 - Q(z=1)$$

- So:

$$H(P,Q) = -(P(z)\log Q(z) + (1-P(z))\log(1-Q(z))$$

- This should look familiar
- P is truth, Q an approximation — just like y and $\hat{y}$

→ Remember we've defined model outputs as probabilities

→ Sum or average over training set to estimate empirical distribution

Cross entropy = negative log likelihood

$$\mathbf{w}^\star = \operatorname*{argmin}_{\mathbf{w}} \; -\frac{1}{n} \sum_{i}^{n} y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$$

- This is probably the most common loss function for classification
  - → As we'll see in §2.8 it readily generalises to multiclass problems
  - → Does not depend on any particular functional form for $\hat{y}$
- Intuitively, it measures a "distance" between our prediction distribution and the true distribution of the data
  - → What happens if prediction is perfect? What if it's wrong with 100% confidence?

2.7: Gradient Descent

COMP0088 Introduction to Machine Learning • UCL Computer Science • Autumn 2021

The laboured hill-walking metaphor

- You’re walking in the hills
- But it’s foggy and you can only see a few metres each way
- And you don’t know where you are
- But you know base camp is right down in the valley

- What do you do?

It’s all downhill from here!

- If we can’t find the minimum analytically, we’ll have too sneak up on it
  - → or down on it
- Provided we can calculate the gradient at all, we can search incrementally
- As long as we’re going downhill, we must be getting closer
  - → at least to some minimum
- If the problem is convex, we have to find the global minimum eventually
  - → or anyway get arbitrarily close

(Batch) gradient descent

- Start somewhere
- Calculate the local loss gradient with respect to the training set
- Travel a short distance down that gradient
- Are we there yet?
- If so, stop
- If not, repeat

Algorithm 1 Iterative minimisation by gradient descent

function GRADIENTDESCENT($f, \mathbf{z}_0, \alpha$)

f : *the function to be minimised*

$\mathbf{z}_0$: *an initial value for the function parameters* $\mathbf{z}$

α : *some small step size, known as the learning rate*

$\mathbf{z} \leftarrow \mathbf{z}_0$

repeat

$\mathbf{z} \leftarrow \mathbf{z} - \alpha \nabla_{\mathbf{z}} f$

until *some stopping criterion reached*

return z

end function

Issues

- Need to choose step size (learning rate)
- Need to choose initial position
- May be slow to converge
- For non-convex problems, can get stuck in local minima
- For large data sets, updates may be prohibitively expensive

Stochastic gradient descent

- Start somewhere
- Calculate the local loss gradient for a single training example
- Travel a short distance down that gradient
- Are we there yet?
- If so, stop
- If not, repeat with the next example

Notes

- Still need to choose the learning rate & initial position
- Does not require processing whole training set before updating
- Progress is usually faster but also noisier
- May waste a lot of time updating

Mini-batch gradient descent

- Start somewhere
- Calculate the local loss gradient for a subset of training examples
- Travel a short distance down that gradient
- Are we there yet?
- If so, stop
- If not, repeat with the next subset

Momentum

- In plain SGD, each update is fully determined by single sample gradient
- For noisy data, majority should point roughly the right way — but any individual might be drastically wrong

  → Even for clean data there can be issues when steepness varies across dimensions

- Momentum instead keeps a weighted tally of recent updates — a velocity
- We retain some (usually lots) of the consensus from update to update, and add only a modest contribution from each individual
- Reduces noise & oscillations of trajectory, speeding up convergence
- May also help escape local minima

Adaptive & multiple LR methods

- Numerous variants: Adagrad, RMSProp, Adam etc
- Adaptively scale learning rates according to gradient variance
- Maintain separate scaling for each feature dimension
- Can increase the number of parameters substantially

Second order methods

- Newton-Raphson:
  - → Use second derivative to set learning rate
  - → Requires computation and inversion of the full Hessian matrix
  - → Individual updates are much more expensive
  - → But convergence is much faster

- Quasi-Newton (eg BGFS)
  - → Avoid directly calculating and inverting the Hessian
  - → Instead, iteratively approximate the inverted matrix from gradients at each step
  - → More complex than Newton, but cheaper

Take home

- Gradient descent methods are the bread-and-butter of ML
  - → Get to know them, you'll see them a lot!
  - → This week's lab exercises include implementing vanilla batch gradient descent
- For the specific case of logistic regression, practical implementations (eg `scikit-learn`) tend to use second-order methods
- For more complex models (in particular neural networks) the adaptive variants of SGD with momentum (eg, Adam) are common

2.8: Multiclass Problems

COMP0088 Introduction to Machine Learning • UCL Computer Science • Autumn 2021

Aggregating binary classifiers

- Given we already have a working binary classifier, can we combine?
- One-vs-rest
  - → O(c) classifiers, each rating a single class against all samples not in that class
  - → highest prediction score wins
- One-vs-one
  - → O(c^2) classifiers, each rating a single class against another single class
  - → all pairwise combinations
  - → class with most votes wins

Categorical distribution

- Multi-class equivalent of Bernoulli
  → sometimes actually called multinoulli, but don't do that
- Each outcome has a probability in [0,1] and they all sum to 1
- All draws independent

Predicting multiple scores

- For binary classification we only need a single probability output
- Probabilities must sum to one, so only one degree of freedom
- For more than two classes, output must be a vector **y**
  - → Probabilistically there are only (*c*-1) degrees of freedom
  - → But always use *c* score outputs to avoid imposing scale constraints
- Ground truth values are likewise represented as a vector
  - → Since true class can only actually be one thing, it's a one-hot vector

Multiclass squashing: the softmax function

- Normalise multiple class scores so total is one
- Also serves to amplify differences
- If c=2, it's directly equivalent to logistic function

$$\varsigma(\mathbf{z}) = \frac{e^{\mathbf{z}}}{\sum e^{\mathbf{z}}}$$

- Vector function ($\mathbb{R}^c \mapsto \mathbb{R}^c$) — all output elements depend on all inputs
- Produces full Jacobian matrix rather than just gradient vector
- See next slide — but we won't actually need this in practice

Softmax Jacobian

$$\nabla_{\mathbf{z}}\, \boldsymbol{\varsigma} = \begin{bmatrix} \frac{\partial \hat{y}_1}{\partial z_1} & \frac{\partial \hat{y}_2}{\partial z_1} & \cdots & \frac{\partial \hat{y}_c}{\partial z_1} \\ \frac{\partial \hat{y}_1}{\partial z_2} & \frac{\partial \hat{y}_2}{\partial z_2} & \cdots & \frac{\partial \hat{y}_c}{\partial z_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \hat{y}_c}{\partial z_c} & \frac{\partial \hat{y}_c}{\partial z_c} & \cdots & \frac{\partial \hat{y}_c}{\partial z_c} \end{bmatrix} = \begin{bmatrix} \hat{y}_1 \cdot (1 - \hat{y}_1) & -\hat{y}_2 \cdot \hat{y}_1 & \cdots & -\hat{y}_c \cdot \hat{y}_1 \\ -\hat{y}_1 \cdot \hat{y}_2 & \hat{y}_2 \cdot (1 - \hat{y}_2) & \cdots & -\hat{y}_c \cdot \hat{y}_2 \\ \vdots & \vdots & \ddots & \vdots \\ -\hat{y}_1 \cdot \hat{y}_c & -\hat{y}_2 \cdot \hat{y}_c & \cdots & \hat{y}_c \cdot (1 - \hat{y}_c) \end{bmatrix}$$

$$\frac{\partial \hat{y}_i}{\partial z_j} = \hat{y}_i \cdot (\mathbf{1}(i = j) - \hat{y}_j) = \begin{cases} \hat{y}_i \cdot (1 - \hat{y}_i) & \text{if } i = j \\ -\hat{y}_j \cdot \hat{y}_i & \text{otherwise} \end{cases}$$

Logits

- Score values prior to softmax squashing ($\mathbf{z}$) are known as logits
- Interpret as log odds of the outcome occurring (ie, sample being of class)
  → Term also sometimes used in logistic regression, with same meaning
- These are what we will want to estimate with a linear model
  → Again, exactly like logistic regression

Categorical cross-entropy

- Recall from §2.6 that cross-entropy compares distributions

$$H(P, Q) = -\sum_{z \in Z} P(z) \log Q(z)$$

- For logistic regression we special-cased two classes, but this is defined for any number:

$$L(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_j y_j \log \hat{y}_j$$

- Gradient w.r.t. prediction is just:

$$\nabla_{\hat{\mathbf{y}}} L = -\frac{\mathbf{y}}{\hat{\mathbf{y}}}$$

y is a one-hot vector, so gradient is mostly zeroes

Combined loss gradients

- Chaining cross-entropy into softmax gives loss gradient w.r.t. logits:

$$\nabla_{\mathbf{z}} L = \hat{\mathbf{y}} - \mathbf{y}$$

  → Zeroes dispose of the off-diagonals in the softmax Jacobian

- And further chaining to linear step:

  → For single sample (with **x** and **y** both as rows, as in design matrix)

$$\nabla_{\mathbf{W}} L = \mathbf{x}^{\mathsf{T}}(\hat{\mathbf{y}} - \mathbf{y})$$

  → And for whole training set:

$$\nabla_{\mathbf{W}} L = \mathbf{X}^{\mathsf{T}}(\hat{\mathbf{Y}} - \mathbf{Y})$$

Multinomial logistic regression

$$\nabla_{\mathbf{W}} L = \mathbf{X}^{\mathsf{T}}(\hat{\mathbf{Y}} - \mathbf{Y})$$

- Suspiciously similar to logistic regression loss for the binary case
  → Differences:
  - **Y** and **W** are matrices
  - softmax ($\boldsymbol{\varsigma}$) replaces sigmoid (σ)

- No closed form (of course) but similarly solvable by gradient descent

Multinomial logistic regression

- Also known as softmax classification
  - → But often not distinguished as a separate model at all
  - → eg, in `scikit-learn` it's just part of the LogisticRegression model
- Combination of categorical cross-entropy loss and softmax squashing is very common as the final step for other classification models, in particular neural networks
  - → The rest of the model can focus on the job of scoring the options
  - → Using whatever arbitrary scale it happens to converge on