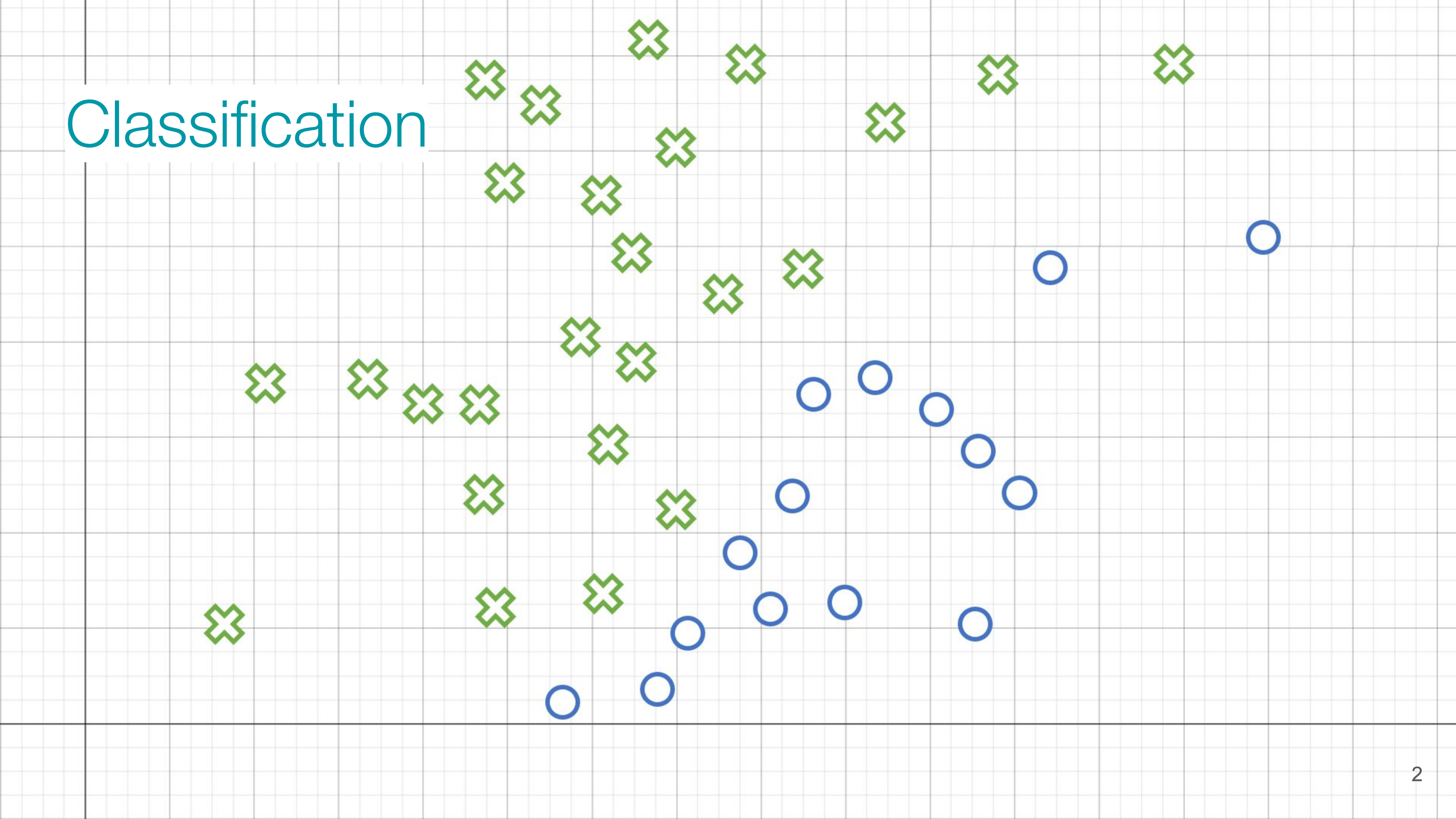
A **Short Overview** of Neural Networks

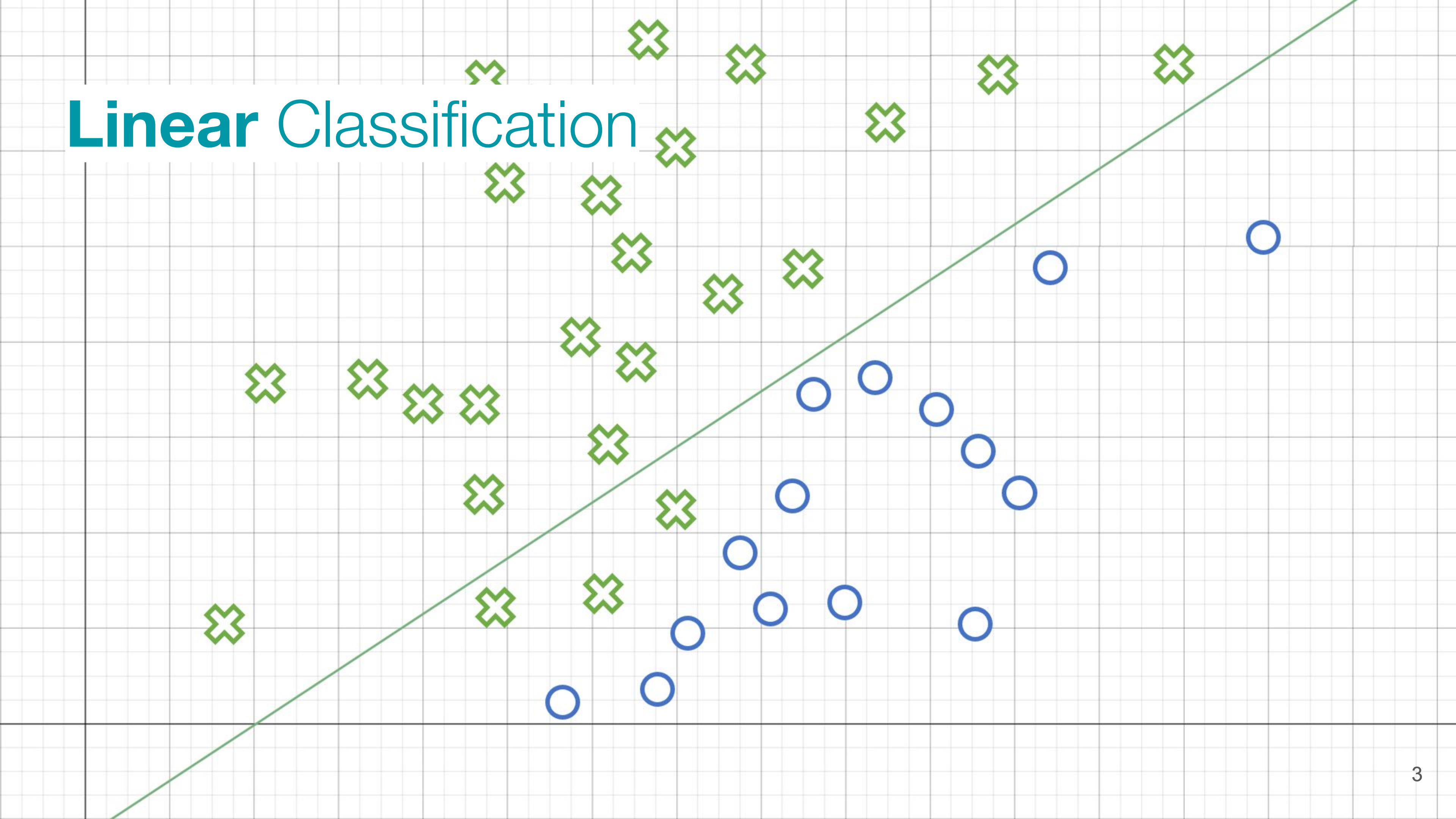
Jens Lehmann, Gaurav Maheshwari, Priyansh Trivedi, Mohnish Dubey, Denis Lukovnikov,

ESWC 2019, Portoroz

Classification



Linear Classification



Non Linear Classification

Non linearity

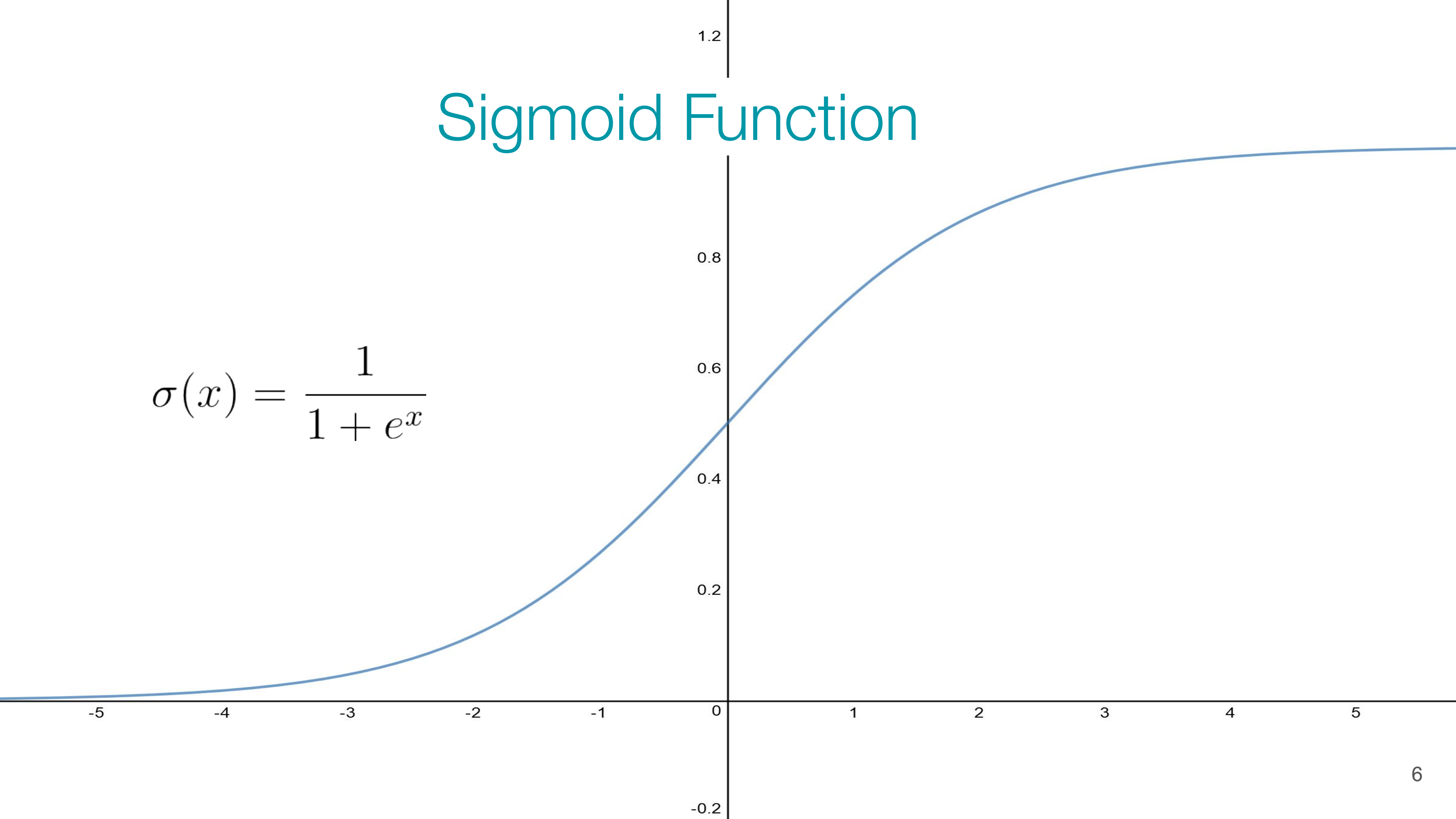
Can be stacked (composite functions) to increase expressivity.

Their gradients are smoother (derivatives of linear transformations are constant, regardless of the input.)

In any case, nonlinear functions are a generalization of linear functions.

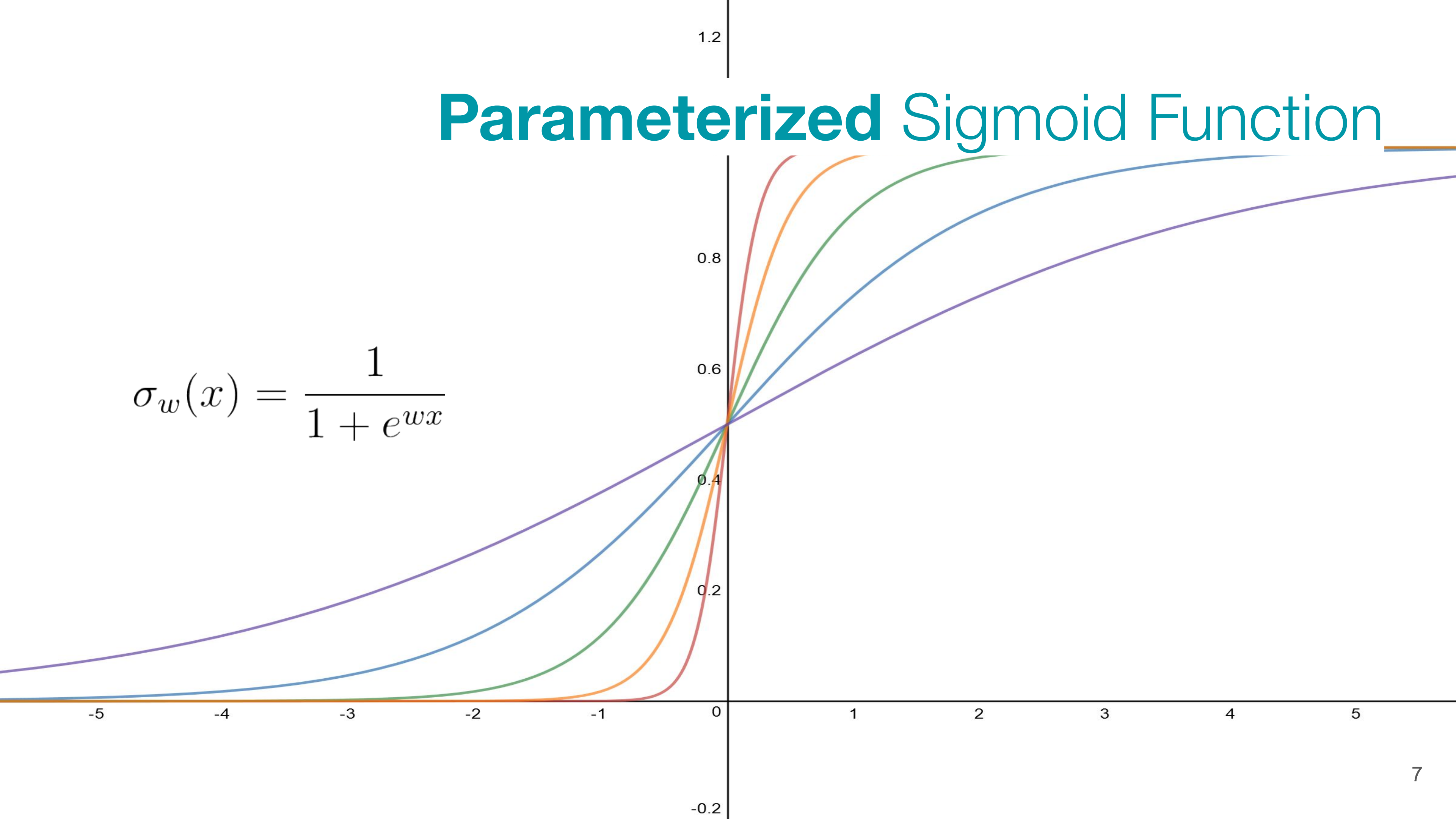
Sigmoid Function

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



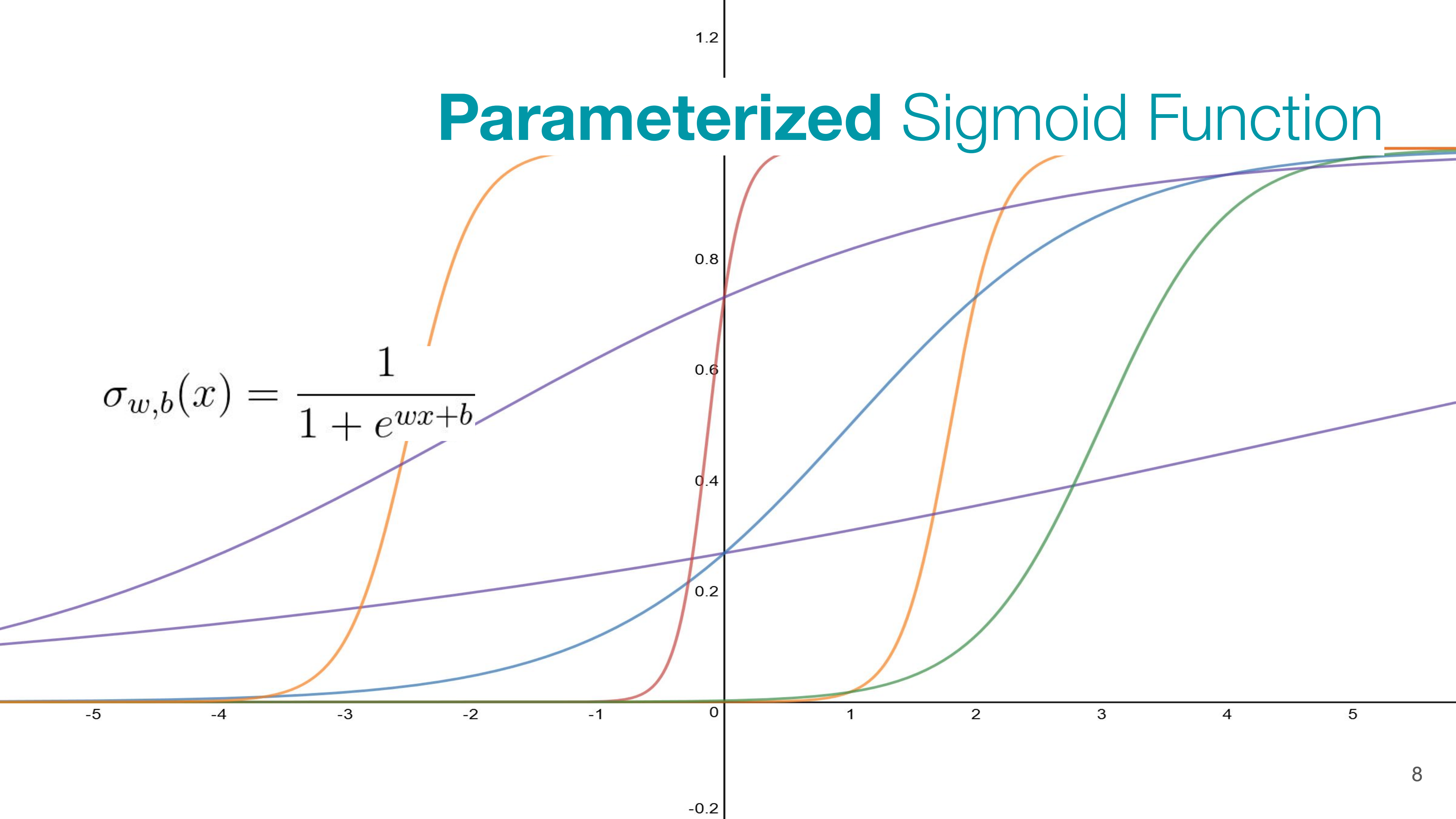
Parameterized Sigmoid Function

$$\sigma_w(x) = \frac{1}{1 + e^{wx}}$$

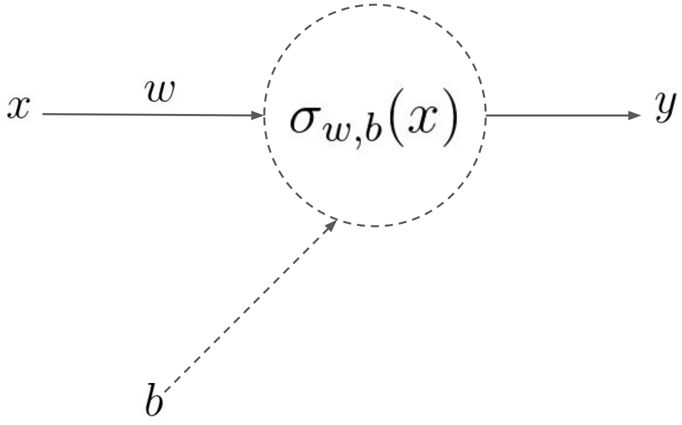


Parameterized Sigmoid Function

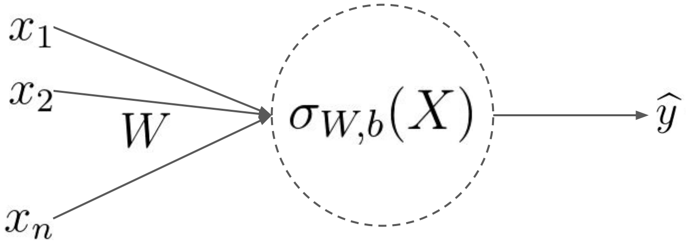
$$\sigma_{w,b}(x) = \frac{1}{1 + e^{wx+b}}$$



Neuron



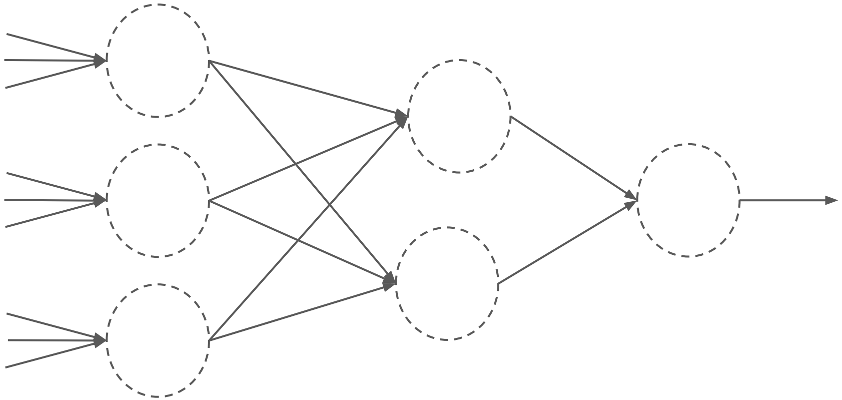
Vector Inputs to Neuron



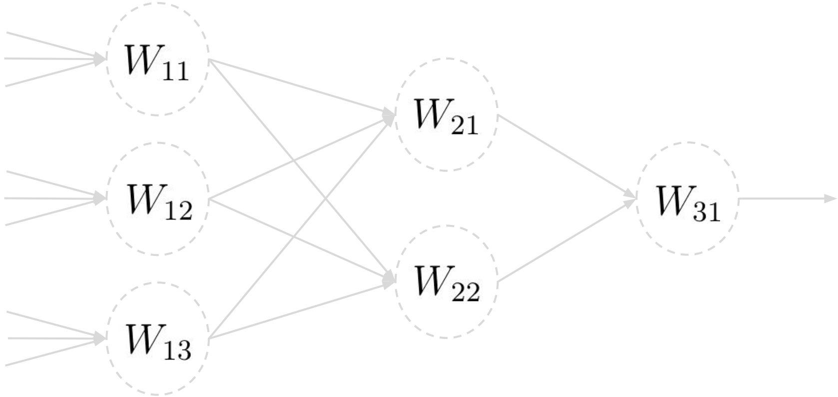
Stacking (Composite Functions)

$$\sigma_{W_2,b_2}(\sigma_{W_1,b_1}(X))$$

Feed Forward Network



Parameters



* omitted biases for brevity's sake

Needles in a Haystack

Can we set them manually?

Weights (above):

No 👼

Recommended?

~~Kill a kitten till it works~~

Needles in a Haystack

Can we set them manually?

Weights (above):

No 👼

Recommended?

~~Kill a kitten till it works~~

Backpropagate

Do you even *backpropagate?*

Idea

Find the effect of every parameter on model's performance.

For that we first need a way to calculate model performance:

Loss Function (Error Function)

Loss Function

Difference between $\underbrace{\textit{model output}}_{\hat{Y} = f_w(X)}$ and $\underbrace{\textit{given label}}_{Y}$ for a data point.

$$E = L(\hat{Y}, Y)$$

Many ways to calculate loss. Eg. Mean Squared Error

$$L(\hat{Y}, Y) = \frac{1}{2n} \sum_{i=1}^{n} (\hat{y} - y)^2$$

Gradient Descent

Gradients of Error w.r.t. model parameters: change the parameter in that direction to *increase* the error.

Update $w_n = w_n - \eta \frac{dL}{dw_n}$

Summing it up

- input data: x

x

y

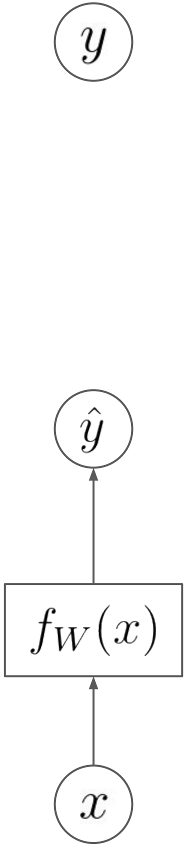
Summing it up

- input data: x
- input data label: y

x

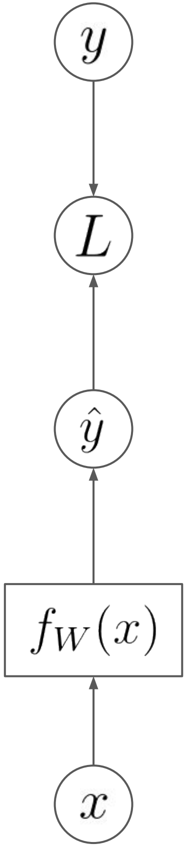
Summing it up

- input data: x
- input data label: y
- model: $f_W(x)$



Summing it up

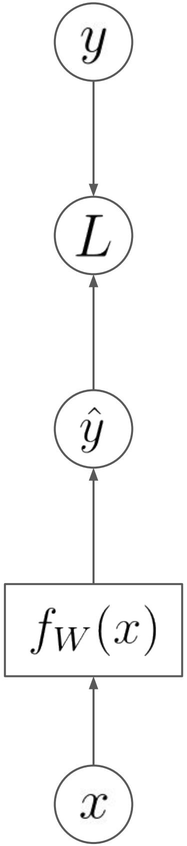
- input data: x
- input data label: y
- model: $f_W(x)$
- model output: $\hat{y} = f_W(x)$



Summing it up

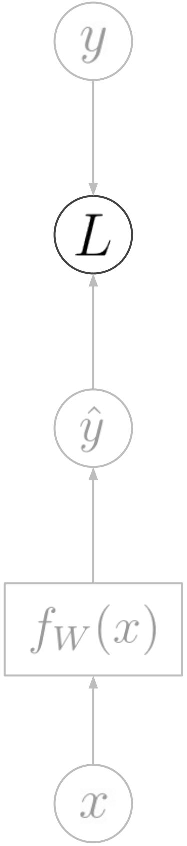
- input data: x
- input data label: y
- model: $f_W(x)$
- model output: $\hat{y} = f_W(x)$
- loss: $L(\hat{y}, y)$

Summing it up



- input data: x
- input data label: y
- model: $f_W(x)$
- model output: $\hat{y} = f_W(x)$
- loss: $L(\hat{y}, y)$

Forward Propagation

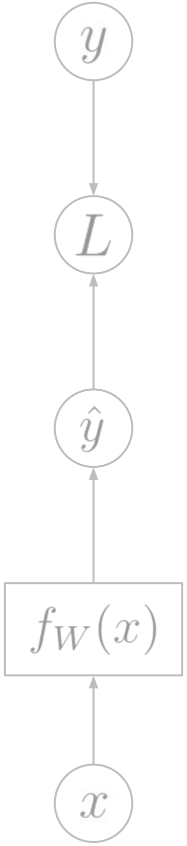


Summing it up

- input data: x
- input data label: y
- model: $f_W(x)$
- model output: $\hat{y} = f_W(x)$
- loss: $L(\hat{y}, y)$

Forward Propagation

- gradients: $\dfrac{dL}{dW_{1:n}}$



Summing it up

- input data: x
- input data label: y
- model: $f_W(x)$
- model output: $\hat{y} = f_W(x)$
- loss: $L(\hat{y}, y)$

Forward Propagation

- gradients: $\dfrac{dL}{dW_{1:n}}$
- parameter update: $w_n = w_n - \eta \dfrac{dL}{dW_{1:n}}$

Backward Propagation