

# Introduction to Deep Learning

## Chapter 2: Single Neuron / Preceptron

**Bernd Bischl**

Department of Statistics – LMU Munich

WS 2021/2022



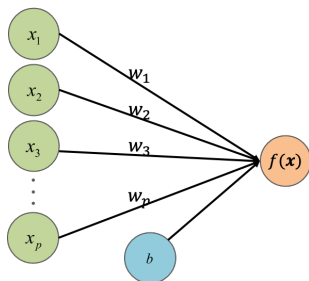
# A SINGLE NEURON

- To illustrate the types of functions that neural networks can represent, let us begin with a simple model: logistic regression.
- The hypothesis space of logistic regression can be written as follows, where  $\tau(z) = (1 + \exp(-z))^{-1}$  is the logistic sigmoid function:

$$\mathcal{H} = \left\{ f : \mathbb{R}^p \rightarrow [0, 1] \mid f(\mathbf{x}) = \tau \left( \sum_{j=1}^p w_j x_j + b \right), \mathbf{w} \in \mathbb{R}^p, b \in \mathbb{R} \right\},$$

- It is straightforward to represent  $f(\mathbf{x})$  graphically as a neuron.
- Note:  $\mathbf{w}$  and  $b$  together constitute  $\theta$ .

# A SINGLE NEURON



Perceptron  $z$ , with **input features**  $x_1, x_2, \dots, x_p$ , **weights**  $w_1, w_2, \dots, w_p$ , **bias term**  $b$  and **activation function**  $\tau$ .

- The perceptron is the basic computational unit for neural networks.
- It is a weighted sum of input values, transformed by  $\tau$ :

$$y = \tau(w_1x_1 + \dots + w_px_p + b) = \tau(w^T x + b)$$

# A SINGLE NEURON

**Choices for  $\tau$ :** a single neuron can represent different functions if we choose suitable activation function for it.

- The identity function gives us the simple **linear regression**:

$$y = \tau(w^T x) = w^T x$$

- The logistic function gives us the **logistic regression**:

$$y = \tau(w^T x) = \frac{1}{1 + \exp(-w^T x)}$$

# A SINGLE NEURON

We consider a logistic regression model for  $p = 3$ , i.e.

$$f(\mathbf{x}) = \tau(w_1 \mathbf{x}_1 + w_2 \mathbf{x}_2 + w_3 \mathbf{x}_3 + b).$$

- First, features of  $\mathbf{x}$  are represented by nodes in the “input layer”.

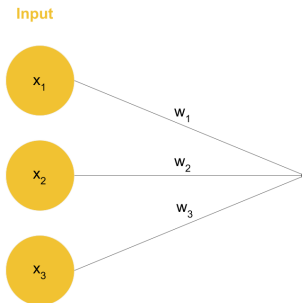
Input



- In general, a  $p$ -dimensional input vector  $\mathbf{x}$  will be represented by  $p$  nodes in the input layer.

# A SINGLE NEURON

- Next, weights  $\mathbf{w}$  are represented by edges from the input layer.



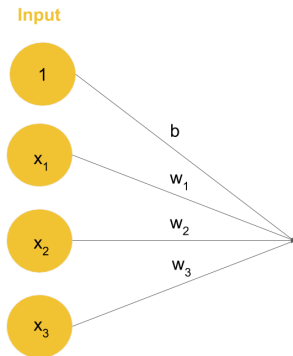
- The bias term  $b$  is implicit. It is not shown visually.

# A SINGLE NEURON

For an explicit graphical representation, we do a simple trick:

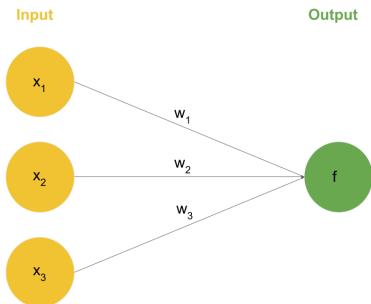
- Add a constant feature to the inputs  $\tilde{\mathbf{x}} = (1, x_1, \dots, x_p)^T$
- and the bias term to the weight vector  $\tilde{\mathbf{w}} = (b, w_1, \dots, w_p)$ .

The graphical representation is then:



# A SINGLE NEURON

- Finally, the computation  $\tau(w_1x_1 + w_2x_2 + w_3x_3 + b)$  is represented by the neuron in the “output layer”.

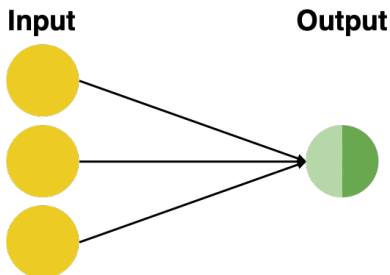


- Because this single neuron represents exactly the same hypothesis space as logistic regression, it can only learn linear decision boundaries.



# A SINGLE NEURON

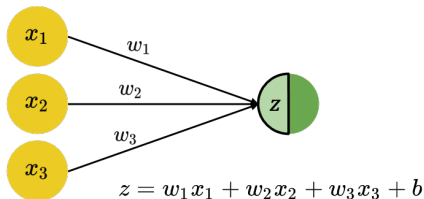
- A nice thing about this graphical representation of functions is that you can picture the input vector being "fed" to the neuron on the left followed by a sequence of computations being performed from left to right. This is called a **forward pass**.



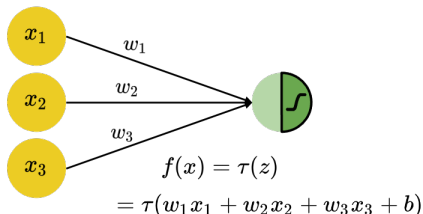
# A SINGLE NEURON

Therefore, a neuron performs a 2-step computation:

- 1 **Affine Transformation:** weighted sum of inputs plus bias.

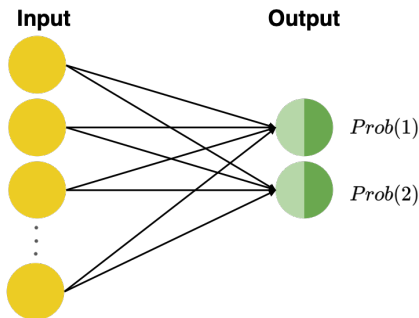


- 2 **Non-linear Activation:** a non-linear transformation applied to the weighted sum.



# A SINGLE NEURON

- Even though all neurons compute a weighted sum in the first step, there is considerable flexibility in the type of activation function used in the second step.
- For example, setting the activation function to logistic sigmoid function allows a neuron to represent logistic regression. The following neuron represents a logistic regression with two outputs.

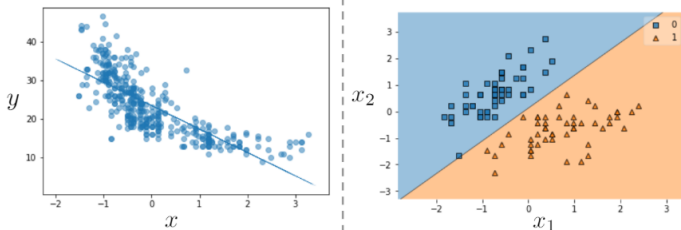


# A SINGLE NEURON

- The hypothesis space that is formed by single neuron architectures is

$$\mathcal{H} = \left\{ f : \mathbb{R}^p \rightarrow \mathbb{R} \mid f(\mathbf{x}) = \tau \left( \sum_{j=1}^p w_j x_j + b \right), \mathbf{w} \in \mathbb{R}^p, b \in \mathbb{R} \right\}.$$

- Both logistic regression and linear regression are subspaces of  $\mathcal{H}$  (if  $\tau$  is the logistic sigmoid / identity function).



**Figure:** *Left:* A regression line learned by a single neuron. *Right:* A decision-boundary learned by a single neuron in a binary classification task.

# A SINGLE NEURON: OPTIMIZATION

- To optimize this model, we minimize the empirical risk

$$\mathcal{R}_{\text{emp}} = \sum_{i=1}^n L\left(y^{(i)}, f\left(\mathbf{x}^{(i)}\right)\right),$$

where  $L(y, f(\mathbf{x}))$  is a loss function. It compares the network's predictions  $f(\mathbf{x})$  to the ground truth  $y$ .

- For regression, we typically use the L2 loss (rarely L1):

$$L(y, f(\mathbf{x})) = \frac{1}{2}(y - f(\mathbf{x}))^2$$

- For binary classification, we typically apply the cross entropy loss (also known as bernoulli loss):

$$L(y, f(\mathbf{x})) = y \log f(\mathbf{x}) + (1 - y) \log(1 - f(\mathbf{x}))$$

# A SINGLE NEURON: OPTIMIZATION

- For a single neuron, in both cases, the loss function is convex and the global optimum can be found with an iterative algorithm like gradient descent.
- In fact, a single neuron with logistic sigmoid function trained with the bernoulli loss does not only have the same hypothesis space as a logistic regression and is therefore the same model, but will also yield to the very same result when trained until convergence.