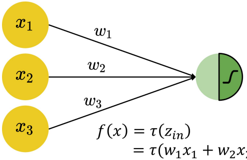
Deep Learning

Single Neuron / Perceptron

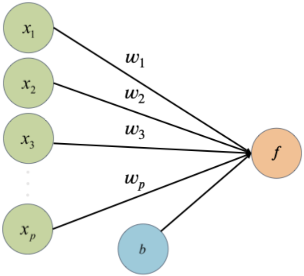


$$f(x) = \tau(z_{in}) \\ = \tau(w_1 x_1 + w_2 x_2 + w_3 x_3 + b)$$

Learning goals

- Graphical representation of a single neuron
- Affine transformations and non-linear activation functions
- Hypothesis spaces of a single neuron
- Typical loss functions

A SINGLE NEURON



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

- The perceptron is a single artificial neuron and the basic computational unit of neural networks.
- It is a weighted sum of input values, transformed by τ:

$$f(x) = \tau(w_1 x_1 + \ldots + w_p x_p + b) = \tau(\mathbf{w}^T \mathbf{x} + b)$$

A SINGLE NEURON

Activation function τ: a single neuron represents different functions depending on the choice of activation function.

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

$$f(x) = \tau(\mathbf{w}^T\mathbf{x}) = \mathbf{w}^T\mathbf{x}$$

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

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

A SINGLE NEURON

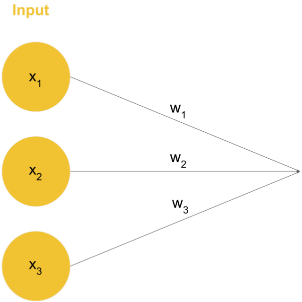
We consider a perceptron with 3-dimensional input, i.e.
$f(\mathbf{x}) = \tau(w_1 x_1 + w_2 x_2 + w_3 x_3 + b)$.

- Input features $\mathbf{x}$ are represented by nodes in the "input layer".

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

A SINGLE NEURON

- Weights $\mathbf{w}$ are connected to edges from the input layer.



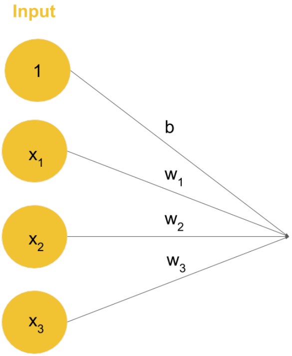
- The bias term b is implicit here. It is often not visualized as a separate node.

A SINGLE NEURON

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

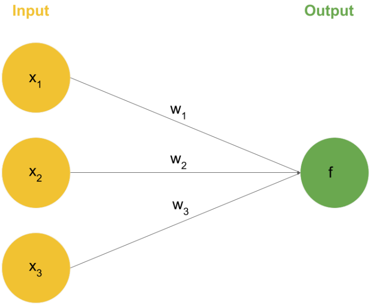
- Add a constant feature to the inputs $\tilde{\mathbf{x}} = (1, x_1, ..., x_p)^T$
- and absorb the bias into the weight vector $\tilde{\boldsymbol{w}} = (b, w_1, ..., w_p)$.

The graphical representation is then:



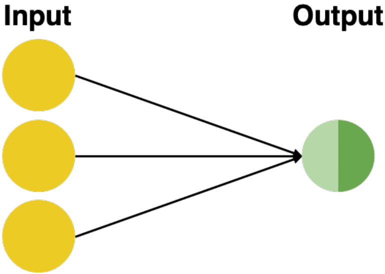
A SINGLE NEURON

- The computation $\tau(w_1 x_1 + w_2 x_2 + w_3 x_3 + b)$ is represented by the neuron in the "output layer".



A SINGLE NEURON

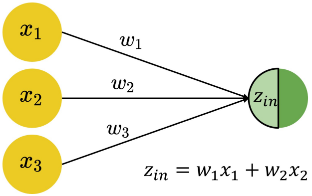
- You can picture the input vector being "fed" to neurons on the left followed by a sequence of computations performed from left to right. This is called a **forward pass**.



A SINGLE NEURON

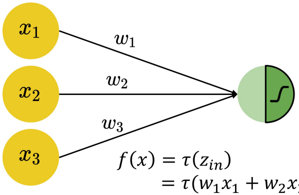
A neuron performs a 2-step computation:

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



$$z_{in} = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

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



$$\begin{aligned} f(x) &= \tau(z_{in}) \\ &= \tau(w_1 x_1 + w_2 x_2 + w_3 x_3 + b) \end{aligned}$$

A SINGLE NEURON: HYPOTHESIS SPACE

- The hypothesis space that is formed by single neuron is

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

- If τ is the logistic sigmoid or identity function, $\mathcal{H}$ corresponds to the hpothesis space of logistic or linear regression, respectively.

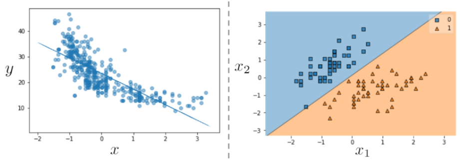


Figure: *Left*: A regression line learned by a single neuron. *Right*: A decision-boundary learned by a single neuron in a binary classification task.

A SINGLE NEURON: OPTIMIZATION

- To optimize this model, we minimize the empirical risk

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

  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 and both choices of τ the loss function is convex.
- The global optimum can be found with an iterative algorithm like gradient descent.
- A single neuron with logistic sigmoid function trained with the Bernoulli loss yields the same result as logistic regression when trained until convergence.
- Note: In the case of regression and the L2-loss, the solution can also be found analytically using the “normal equations”. However, in other cases a closed-form solution is usually not available.