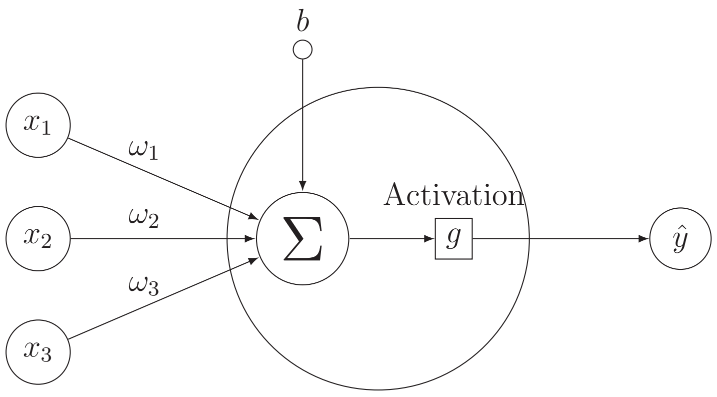

Figure 5.3.: Components of a single neuron.

After providing an brief overview of all components used in a neural network and presenting the idea of how a network learns, the next section focuses on the internals of a single layer.

5.1.1. Single neuron

The so called neurons form the basis of every neural network and are the elements which carry out the data transformations. Each layer of a neural network consists of one or more neurons which are connected differently depending on the structure of the underlying network. The task of a neuron is to take a predefined number of input values and map them to an one dimensional output. Figure (5.3) shows a single neuron with all its associated components needed to perform the data transformation given by:

$$\begin{aligned} \hat{y} &= g\bigg(\sum_{i=1}^{3} \omega_i x_i + b\bigg) \\ &= g(w^\top x + b) \\ &= g(z) \end{aligned} \tag{5.1}$$

In the example shown in figure (5.3) the neuron takes the three input values x_1, x_2 and x_3 and transforms them in several steps into the output value $\hat{y}$. In the first step, a weighted sum is formed from the input values and the corresponding weights w_1, w_2 and w_3. A bias b is then added to this weighted sum and the intermediate result is called z. This intermediate result is then

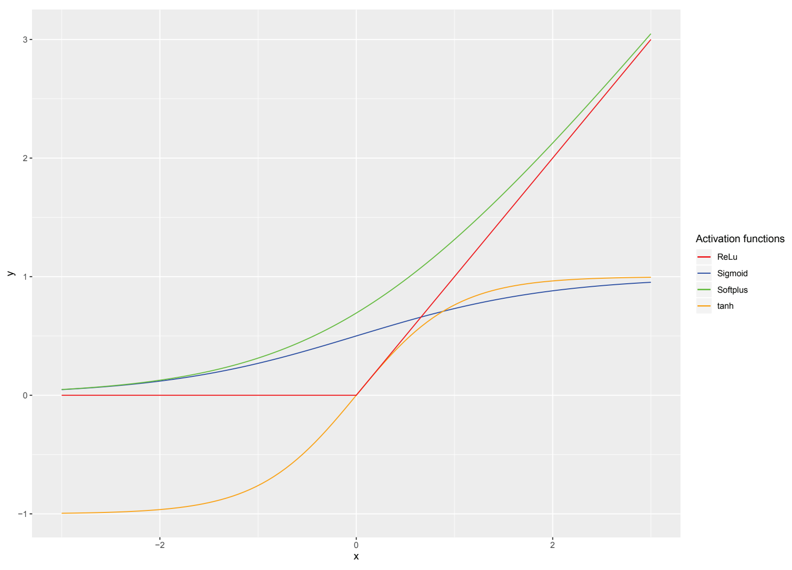

Figure 5.4.: Commonly used activation functions for neural networks.

the input for the activation function g. Applying the activation function to z gives the final output value of the neuron. The purpose of the activation function can be seen as a gate. Since both, the values of the inputs and the values of the weights are unrestricted, the value of z can be in the interval (-∞, +∞). A natural way of determining whether the next neuron should be activated or not is to apply a transformation to z, which is done by an activation function. For this reason neural networks use non-linear activation functions. Those activation functions can limit the value range and also enable the network to learn complex data structures. Some of the most commonly used activation functions in neural networks are shown in figure (5.4):

- Sigmoid function:

$$\begin{aligned} g &: \mathbb{R} \mapsto (0,1) \\ g(x) &= \frac{1}{1+e^{-x}} \end{aligned} \tag{5.2}$$

  The sigmoid function has an order of continuity of C^∞ and is convex for all values less than 0, and it is concave for all values greater than 0. Since the output is always in the range from 0 to 1, one of the use cases of the sigmoid function is to model probabilities. Applying the sigmoid function to strongly negative values results in values close to zero. This means that the next neuron is only activated if the linear transformation has given a value z that is not too negative.

- Hyperbolic tangent function:

$$\begin{aligned} g &: \mathbb{R} \mapsto (-1, 1) \\ g(x) &= \frac{e^x - e^{-x}}{e^x + e^{-x}} \end{aligned} \tag{5.3}$$

  The hyperbolic tangent function is very similar to the sigmoid function and, just like the sigmoid function, also has an order of continuity of C^∞. Since the range of output values lies between -1 and 1, the possibilities of using this activation function as a gate are rather limited compared to the sigmoid function. Possible application scenarios can be found in the area of classification problems where all outputs below zero belong to class A and all outputs greater than zero belong to class B.

- ReLu function:

$$\begin{aligned} g &: \mathbb{R} \mapsto [0, +\infty) \\ g(x) &= max(0, x) \end{aligned} \tag{5.4}$$

  One of the most successful and widely-used activation functions is the Rectified Linear Unit (ReLU). All values greater than zero activate the next neuron and all negative input values result in the next neuron not being activated by setting the output to zero. Because of the maximization the ReLu-function has only an order of continuity of C^0. Although it is non-differentiable, this function is popular because of it's simplicity and reliability as a gate function. [**searchingActivation**], [**nair2010rectified**]

- Softplus function:

$$\begin{aligned} g &: \mathbb{R} \mapsto (0, \infty) \\ g(x) &= ln(1 + e^x) \end{aligned} \tag{5.5}$$

  Since ReLu is non-differentiable at zero a smooth version of it, which also has an order of continuity of C^∞, is given by the softplus function. It can be used as a replacement for the ReLu function. As various analyses have shown the performance between the softplus fuction and the Relu function is negligible in most cases. [**searchingActivation**]

In the example given in figure (5.3), the output value of the neuron is already the prediction $\hat{y}$, but in almost every case the output of one neuron serves as an input for another neuron. The next section is therefore dedicated to the interaction of multiple neurons in multiple layers and also introduces a general notation that allows to describe the mathematical concept of a neural network including an optimization algorithm.