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. [38], [31]

- 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. [38]

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 based on [11], that allows to describe the mathematical concept of a neural network including an optimization algorithm.

5.1.2. Multiple neurons

Having described the basic function of a single neuron in the previous section, the focus is now on how multiple neurons interact in order to form a neural

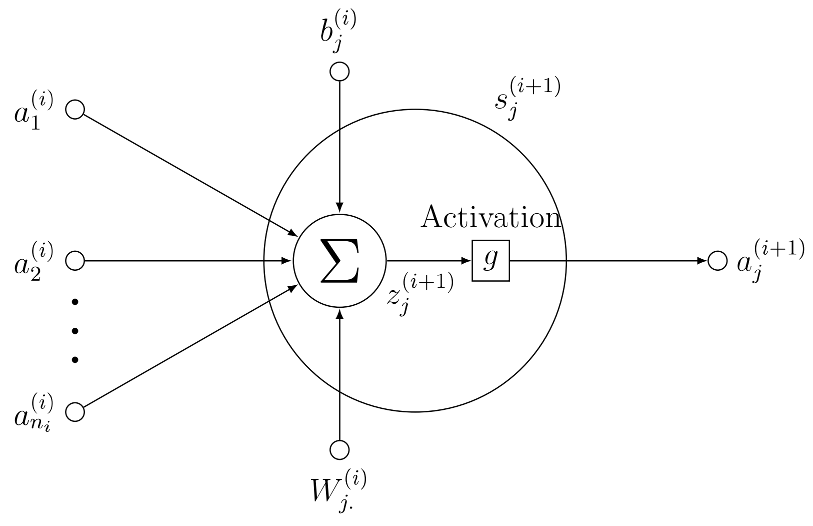

Figure 5.5.: Components of a single neuron.

network. Figure (5.5) shows the same single neuron as figure (5.3) but with a slightly adapted and extended notation. While in figure (5.3) the weights ω_i are directly assigned to the individual inputs x_i, in figure (5.5) these weights are represented by a weight vector $W_{j.}^{(i)}$. To be able to refer to a single neuron in a neural network with a large number of neurons, a single neuron is referred to in the form of $s_j^{(i)}$. Where s stands for single neuron, i for the layer in which the neuron is used and j for a consecutive number over all neurons in this layer i. The notation of the inputs and the output of a neuron have also changed, so that the notation $a_j^{(i)}$ is now used in both cases. The a refers to activation and the two indices i and j refer to the layer and respectively the neuron that generated the activation value. The value $a_1^{(i)}$ from figure (5.5) therefore identifies both, the output of the first neuron in the i-th layer and the input for the j-th neuron in the i+1-th layer. Depending on which and how many layers, and how the individual neurons in the individual layers or between the individual layers are connected to each other, different types of neural networks result. So-called feedforward networks, for example, are characterized by the fact that neurons from one layer are only connected to neurons from the next higher layer. The information flow therefore takes place only in one direction, namely from the input side to the output side. On the other hand there are so-called recurrent nets where feedback between different layers is also allowed. There are three different categories of recurrent neural networks which can be distinguished depending on how they use their feedback signals [9]:

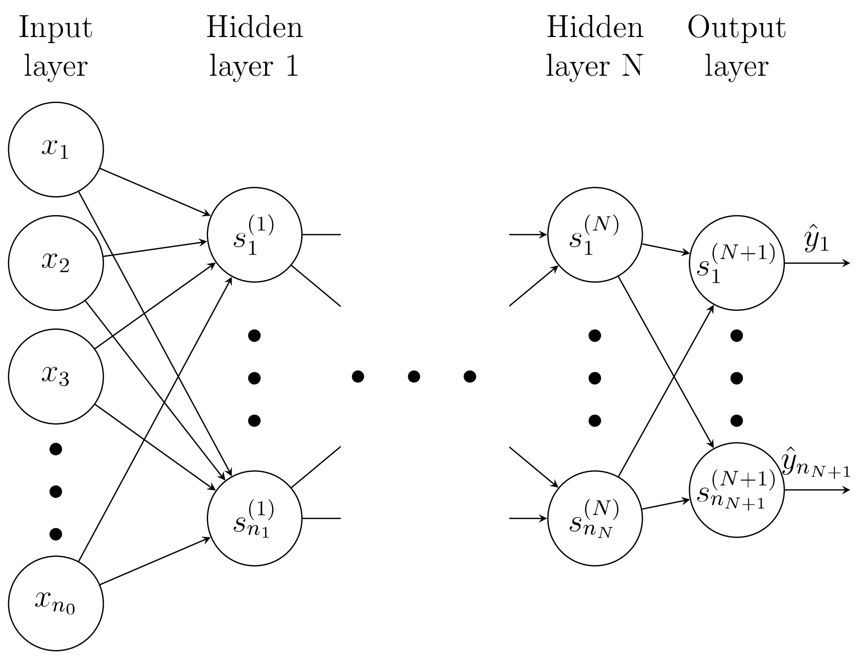

Figure 5.6.: Illustration of fully connected feedforward neural network with N hidden layers.

- Direct feedback: The own output of a neuron is used as an additional input for the neuron.
- Indirect feedback: The output of a neuron serves as an input to a neuron of a previous layer.
- Lateral feedback: The output of a neuron servers as an input to a neuron from the same layer.

Since the possibilities of how the different neurons are connected in a neural network are almost unlimited, the following section describes one of the most common forms [25]. It is a fully connected feedforward network as shown in figure (5.6). The most important properties of such a network are:

- Fully connected: All outputs from the neurons of layer i serve as inputs for all neurons in layer $i + 1$.
- Feedforward: The information flows only from one layer to the next higher layer.

For the calculation and analysis of such a neural network the following notation, which is taken from [11], is used in accordance with figure (5.5) and (5.6). Let

- $N \in \mathbb{N}$ be the number of hidden layers in the neural network.
- $N+2$ be the total number of layers in the network since the hidden layers are wrapped between an input and an output layer.
- $n_i \in \mathbb{N}, i \in \{0, 1, ..., N+1\}$, be the number of neurons in the i-th layer.
- $s_j^{(i)}$ denote the j-th neuron of the i-th layer.
- $a^{(i)} \in \mathbb{R}^{n_i}$, $a^{(i)} = (a_1^{(i)}, a_2^{(i)}, ..., a_{n_i}^{(i)})^\top$ be the vector of activation values produced by the neurons of the i-th layer.
- $b^{(i)} \in \mathbb{R}^{n_{i+1}}$, $b^{(i)} = (b_1^{(i)}, b_2^{(i)}, ..., b_{n_{i+1}}^{(i)})^\top$ be the bias vector for the linear transformation performed in all neurons of layer $i+1$.
- $W_{j.}^{(i)} \in \mathbb{R}^{1 \times n_i}$, $W_{j.}^{(i)} = (W_{j1}^{(i)}, W_{j2}^{(i)}, ..., W_{jn_i}^{(i)})$ be the weight vector for the linear transformation performed in the j-th neuron of the $i+1$-th layer. Combining all weights used in the $i+1$-th layer into a matrix $W^{(i)} \in \mathbb{R}^{n_{i+1} \times n_i}$ results in:

$$W^{(i)} = \begin{pmatrix} W_{11}^{(i)} & W_{12}^{(i)} & \cdots & W_{1n_i}^{(i)} \\ W_{21}^{(i)} & W_{22}^{(i)} & \cdots & W_{2n_i}^{(i)} \\ \vdots & \vdots & & \vdots \\ W_{n_{i+1}1}^{(i)} & W_{n_{i+1}2}^{(i)} & \cdots & W_{n_{i+1}n_i}^{(i)} \end{pmatrix} = \begin{pmatrix} W_{1.}^{(i)} \\ W_{2.}^{(i)} \\ \vdots \\ W_{n_{i+1}.}^{(i)} \end{pmatrix},$$

  with $W_{j.}^{(i)}$ being the j-th row of $W^{(i)}$.
- $z^{(i)} \in \mathbb{R}^{n_i}$, $z^{(i)} = (z_1^{(i)}, z_2^{(i)}, ..., z_{n_i}^{(i)})^\top$ be the result vector after the linear transformation performed in all neurons of layer i.
- $g : \mathbb{R} \mapsto \mathbb{R}$ be any activation function which is applied to the result of the linear transformation.
- $x \in \mathbb{R}^{n_0}$, $x = (x_1, x_2, ..., x_{n_0})^\top$ be a vector of input data to train the model.
- $y \in \mathbb{R}^{n_{N+1}}$, $y = (y_1, y_2, ..., y_{n_{N+1}})^\top$ be a vector of output data to train the model.

- $\hat{y} \in \mathbb{R}^{n_{N+1}}$, $\hat{y} = (\hat{y}_1, \hat{y}_2, ..., \hat{y}_{n_{N+1}})^\top$ be a vector of estimated output values obtained from the neural network by sending an input vector x through the network.

Remark 5.5. *Since the input layer has n_0 neurons, the input vector x must be an element of $\mathbb{R}^{n_0}$. Similarly, the estimated output $\hat{y}$ of the neural network is a vector which is an element of $\mathbb{R}^{n_{N+1}}$.*

Remark 5.6. *Since $a^{(i)}$ denotes the values that are transferred between neurons, it is both the output values of the neurons from layer i and the input values of the neurons from layer $i+1$.*

Remark 5.7. *The input values x for the neuronal network correspond to the first activation values $a^{(0)}$, e.g. $x = a^{(0)}$. Similarly, the estimated output $\hat{y}$ of the neural network corresponds to the last activation values $a^{(N+1)}$, e.g. $\hat{y} = a^{(N+1)}$.*

Remark 5.8. *For the training of a neural network many different sets of input and associated output vectors are needed. If a specific set of input or output data is to be referenced, this is done by adding a superscript, e.g. $x^{(k)}$ and $y^{(k)}$. The corresponding estimated values are also referred to as $\hat{y}^{(k)}$.*

Remark 5.9. *The weight defined as $W_{jk}^{(i)}$ connects the activation value from the k-th neuron in layer i with the j-th neuron from layer $i+1$.*

After a notation for the analysis of the neural network has been defined, the next section is devoted to the question of how data can flow through a network and how the network can learn.

5.2. Train a model

Figure (5.3) together with formula (5.1) already illustrated in the previous section how a single neuron is constructed and how it processes the inputs. Since the notation has been generalized in the previous section also formula (5.1) could be generalized.

Remark 5.10. *Let $a^{(i)} \in \mathbb{R}^{n_i}$ be the activation values from layer i, $b_j^{(i)} \in \mathbb{R}$ the bias term for the j-th neuron in layer $i+1$, $W_{j\cdot}^{(i)}$ the weights and $g : \mathbb{R} \mapsto \mathbb{R}$*