Neural Network with Layers

Let $n \in \mathbb{N}$ be the count of layers (do not count the input layer) in a given fully-connected feed-forward neural network. Then one can describe the network through an activation function $\sigma \colon \mathbb{R} \to [0,1]$, weight matrices $(W_k)_{k=1}^n$ and bias vectors $(B_k)_{k=1}^n$ with the following properties for all $k \in \mathbb{N}$ with $k \leq n$.

$$W_k \in \mathbb{R}^{p_k \times q_k} \qquad B_k \in \mathbb{R}^{p_k} \qquad p_k, q_k \in \mathbb{N}$$

p_k is the number of neurons and q_k the number of inputs in the k. layer. The dimensions have to fulfill a certain consistency condition such that the matrix-vector-product can be defined between each layer. For all $k \in \mathbb{N}$ with $k < n$ it holds

$$p_k = q_{k+1}$$

Then the output of the k. layer with respect to its input is described by the following function. We define $\bar{\sigma}$ to be the element-wise application of σ to a vector.

$$f\left[W_k, B_k\right] : [0,1]^{q_k} \to [0,1]^{p_k} \qquad f\left[W_k, B_k\right](x) \coloneqq \bar{\sigma}(W_k x + B_k)$$

The overall output f of the complete network is then described by composition of these functions.

$$f\left[(W_k, B_k)_{k=1}^n\right] : [0,1]^{q_1} \to [0,1]^{p_n}$$

$$f\left[(W_k, B_k)_{k=1}^n\right] \coloneqq f\left[W_n, B_n\right] \circ \ldots \circ f\left[W_1, B_1\right] = \bigcirc_{k=1}^n f\left[W_k, B_k\right]$$

Error Function

Given $m \in \mathbb{N}$ labeled training samples $(x_k, y_k)_{k=1}^m$ with $x_k \in [0,1]^{q_1}$ and $y_k \in [0,1]^{p_n}$ for all $k \in \mathbb{N}$ with $k \leq m$ we can define the total error of the given neural network by the mean-squared-error $\varepsilon\left[(x_i, y_i)_{i=1}^m\right]$.

$$\varepsilon\left[(x_i, y_i)_{i=1}^m\right] : \bigtimes_{k=1}^n \left(\mathbb{R}^{p_k \times q_k} \times \mathbb{R}^{p_k}\right) \to \mathbb{R}$$

$$\varepsilon\left[(x_i, y_i)_{i=1}^m\right]\left((W_k, B_k)_{k=1}^n\right) \coloneqq \frac{1}{m} \sum_{i=1}^m \left\| y_i - f\left[(W_k, B_k)_{k=1}^n\right](x_i) \right\|^2$$

Gradients of the Error Function

The gradients of the error function can be computed by using the abstract index notation. This process is error-prone and cumbersome and will only be shown for the example gradient $\nabla_{W_n} \varepsilon\left[(x_i, y_i)_{i=1}^m\right]$. First, we start by reformulating the error function with the abstract index notation.

$$\varepsilon\left[(x_i, y_i)_{i=1}^m\right]\left((W_k, B_k)_{k=1}^n\right)$$

$$
\begin{aligned}
&= \frac{1}{m} \sum_{i=1}^{m} \left\| y_i - f\left[(W_k, B_k)_{k=1}^{n}\right](x_i) \right\|^2 \\
&= \frac{1}{m} \sum_{i=1}^{m} \left(y_i - f\left[(W_k, B_k)_{k=1}^{n}\right](x_i) \right)^{\mathrm{T}} \left(y_i - f\left[(W_k, B_k)_{k=1}^{n}\right](x_i) \right) \\
&\equiv \frac{1}{m} \sum_{i=1}^{m} \left(y_{i\alpha} - f_\alpha\left[(W_k, B_k)_{k=1}^{n}\right](x_i) \right) \left(y_i{}^{\alpha} - f^{\alpha}\left[(W_k, B_k)_{k=1}^{n}\right](x_i) \right)
\end{aligned}
$$

Now we compute the partial derivate of a certain component of the gradient matrix.

$$
\begin{aligned}
&\left[\nabla_{W_n} \varepsilon\left[(x_i, y_i)_{i=1}^{m}\right]\left((W_k, B_k)_{k=1}^{n}\right)\right]^{\mathrm{T}} \\
&\equiv \partial_{W_n{}^{\mu}{}_{\nu}} \frac{1}{m} \sum_{i=1}^{m} \left(y_{i\alpha} - f_\alpha\left[(W_k, B_k)_{k=1}^{n}\right](x_i) \right) \left(y_i{}^{\alpha} - f^{\alpha}\left[(W_k, B_k)_{k=1}^{n}\right](x_i) \right) \\
&= -\frac{2}{m} \sum_{i=1}^{m} \left(y_{i\alpha} - f_\alpha\left[(W_k, B_k)_{k=1}^{n}\right](x_i) \right) \partial_{W_n{}^{\mu}{}_{\nu}} f^{\alpha}\left[(W_k, B_k)_{k=1}^{n}\right](x_i)
\end{aligned}
$$

$$
\begin{aligned}
&\partial_{W_n{}^{\mu}{}_{\nu}} f^{\alpha}\left[(W_k, B_k)_{k=1}^{n}\right](x_i) \\
&= \partial_{W_n{}^{\mu}{}_{\nu}} \sigma\left(W_n{}^{\alpha}{}_{\beta} f^{\beta}\left[(W_k, B_k)_{k=1}^{n-1}\right](x_i) + B_n{}^{\alpha} \right) \\
&= \sigma'\left(W_n{}^{\alpha}{}_{\beta} f^{\beta}\left[(W_k, B_k)_{k=1}^{n-1}\right](x_i) + B_n{}^{\alpha} \right) \delta_{\mu}^{\alpha} \delta_{\beta}^{\nu} f^{\beta}\left[(W_k, B_k)_{k=1}^{n-1}\right](x_i) \\
&= \sigma'\left(W_n{}^{\alpha}{}_{\beta} f^{\beta}\left[(W_k, B_k)_{k=1}^{n-1}\right](x_i) + B_n{}^{\alpha} \right) \delta_{\mu}^{\alpha} f^{\nu}\left[(W_k, B_k)_{k=1}^{n-1}\right](x_i)
\end{aligned}
$$

Putting all together we get the result for the complete gradient. Please note that I have transposed the matrix to build the real gradient. We define $\bar{\sigma'}$ to be the element-wise application of σ' to a vector.

$$
\begin{aligned}
&\nabla_{W_n} \varepsilon\left[(x_i, y_i)_{i=1}^{m}\right]\left((W_k, B_k)_{k=1}^{n}\right) \\
&= -\frac{2}{m} \sum_{i=1}^{m} \left\{ \left(y_i - f\left[(W_k, B_k)_{k=1}^{n}\right](x_i) \right) \odot \bar{\sigma'}\left(W_n f\left[(W_k, B_k)_{k=1}^{n-1}\right](x_i) + B_n \right) \right\} \\
&\qquad \cdot \left\{ f\left[(W_k, B_k)_{k=1}^{n-1}\right](x_i) \right\}^{\mathrm{T}}
\end{aligned}
$$

The other gradients can be computed analogously. If we apply this process iteratively to every weight matrix and bias vector then we get recursive formulas to compute every gradient. The algorithm is given below.

Forward Feed Algorithm

For convenience we define the following for a given input $x \in [0,1]^{q_1}$.

$$f_0 \coloneqq x$$

For all $k \in \mathbb{N}$ with $k \leq n$ we compute the following recursive formulas.

$$t_k \coloneqq W_k f_{k-1} + B_k$$

$$f_k \coloneqq \bar{\sigma}(t_k)$$

t_k describes the net input vector and f_k the output vector of the k. layer. Therefore f_n is the output of the complete neural network.

Backward Feed Algorithm

Calculate a first temporary vector.

$$z_n \coloneqq (y - f_n) \odot \bar{\sigma'}(t_n)$$

For all $k \in \mathbb{N}$ with $k < n$ compute the following vectors.

$$z_k \coloneqq \left(W_{k+1}^{\mathrm{T}} z_{k+1}\right) \odot \bar{\sigma'}(t_k)$$

Get the gradients from the temporary vectors.

$$\nabla_{W_k} e = -2 z_k \cdot f_{k-1}^{\mathrm{T}}$$

$$\nabla_{B_k} e = -2 z_k$$

Use the gradients for an optimization algorithm like the steepest descent method with learning rate η.

$$W_k \quad \longleftarrow \quad W_k - \eta \nabla_{W_k} e$$

$$B_k \quad \longleftarrow \quad B_k - \eta \nabla_{B_k} e$$

Results

labeled training sample (x, y)

$$x \coloneqq \begin{pmatrix} i_1 \\ i_2 \end{pmatrix} = \begin{pmatrix} 0.05 \\ 0.10 \end{pmatrix} \qquad y \coloneqq \begin{pmatrix} o_1 \\ o_2 \end{pmatrix} = \begin{pmatrix} 0.01 \\ 0.99 \end{pmatrix}$$

initial weight matrices W_1 and W_2

$$W_1^{(0)} \coloneqq \begin{pmatrix} w_1 & w_2 \\ w_3 & w_4 \end{pmatrix} = \begin{pmatrix} 0.15 & 0.20 \\ 0.25 & 0.30 \end{pmatrix} \qquad W_2^{(0)} \coloneqq \begin{pmatrix} w_5 & w_6 \\ w_7 & w_8 \end{pmatrix} = \begin{pmatrix} 0.40 & 0.45 \\ 0.50 & 0.55 \end{pmatrix}$$

initial bias vectors B_1 and B_2

$$B_1 \coloneqq \begin{pmatrix} b_1 \\ b_1 \end{pmatrix} = \begin{pmatrix} 0.35 \\ 0.35 \end{pmatrix} \qquad B_2 \coloneqq \begin{pmatrix} b_2 \\ b_2 \end{pmatrix} = \begin{pmatrix} 0.60 \\ 0.60 \end{pmatrix}$$

net input to first layer t_1

$$t_1 = W_1 x + B_1 = \begin{pmatrix} 0.3775 \\ 0.3925 \end{pmatrix}$$

output of first layer f_1

$$f_1 = \sigma(t_1) = \begin{pmatrix} 0.593270 \\ 0.596884 \end{pmatrix}$$

net input to second layer t_2

$$t_2 = W_2 f_1 + B_2 = \begin{pmatrix} 1.10591 \\ 1.22492 \end{pmatrix}$$

output of second layer f_2

$$f_2 = \sigma(t_2) = \begin{pmatrix} 0.751365 \\ 0.772928 \end{pmatrix}$$

total error e_0

$$e_0 = \frac{1}{2} \left\| y - f_2 \right\|^2 = 0.298371$$

weight matrices after one backpropagation step $W_1^{(1)}, W_2^{(1)}$ without changing bias

$$W_1^{(0)} = \begin{pmatrix} 0.149781 & 0.199561 \\ 0.249751 & 0.299502 \end{pmatrix} \qquad W_2^{(0)} = \begin{pmatrix} 0.358916 & 0.408666 \\ 0.511301 & 0.56137 \end{pmatrix}$$

total error after one backpropagation step and a second forward feed:

$$e_1 = 0.291028$$

$$\implies e_1 < e_0$$