We consider a neural net with L dense layers, with $n^{[l]}$ neurons in layer $l = 1, \dots L$, which takes the input from $\mathbf{R}^{n^{[0]}}$. For notation simplicity we assume that the same activation function $\varphi : \mathbf{R} \to \mathbf{R}$ is used in all layers.

Let $W^{[l]}$ be the matrix of weights in layer l, i.e.,

$$W^{[l]} = [w^{[l]}_{k,j}]_{k=1,\dots,n^{[l]};j=0,\dots,n^{[l-1]}}, \quad l = 1, 2, \dots, L.$$

Let us suppose that the following vector is the input to the network,

$$a^{[0]} = \begin{bmatrix} a^{[0]}_1 \\ a^{[0]}_2 \\ \dots \\ a^{[0]}_{n^{[0]}} \end{bmatrix} \in \mathbf{R}^{n^{[0]}}.$$

Forward propagation (calculating the output), inductive step. Let us suppose that $l \in \{0, 1, \dots, L-1\}$ and that we are given a vector

$$a^{[l]} = \begin{bmatrix} a^{[l]}_1 \\ a^{[l]}_2 \\ \dots \\ a^{[l]}_{n^{[l]}} \end{bmatrix}.$$

from $\mathbf{R}^{n^{[l]}}$. We put $x^{[l]}_0 = 0$ and $x^{[l]}_j = a^{[l]}_j$ for $j \geq 1$, i.e.,

$$x^{[l]} = \begin{bmatrix} x^{[l]}_0 \\ x^{[l]}_1 \\ x^{[l]}_2 \\ \dots \\ x^{[l]}_{n^{[l]}} \end{bmatrix} = \begin{bmatrix} 1 \\ a^{[l]}_1 \\ a^{[l]}_2 \\ \dots \\ a^{[l]}_{n^{[l]}} \end{bmatrix}.$$

Vector $x^{[l]}$ is just $a^{[l]}$ with prepended element equal to one. We calculate

$$net^{[l+1]} = W^{[l+1]} x^{[l]},$$

and

$$a^{[l+1]} = \varphi(net^{[l+1]}) := \begin{bmatrix} \varphi(\sum_{j=0}^{n^{[l]}} w^{[l+1]}_{k,j} x^{[l]}_j) \end{bmatrix}_{k=1,\dots,n^{[l+1]}}.$$

Forward propagation, output of the network. Given vector $a^{[0]}$, we may apply L-times the above inductive step, to find $a^{[1]}, a^{[2]}, \dots, a^{[L]}$. The output of the network is the last vector $a^{[L]}$. In other words, considered neural network is a function of the following form

$$\mathbf{R}^{n^{[0]}} \ni a^{[0]} \mapsto a^{[L]} \in \mathbf{R}^{n^{[L]}}.$$

Loss function. Let us say that for $x^{[0]}$ we have found $a^{[L]}$ as above, but we expected to obtain another vector, $y \in \mathbf{R}^{n^{[L]}}$. We are going to modify the weights using *gradient descent*, by calculating the gradient of the loss function with respect to the weights.

Let us suppose that our loss function is of the form

$$\mathbf{L}(y, a^{[L]}) = \frac{1}{2} \sum_{k=1}^{n^{[L]}} (y_k - a^{[L]}_k)^2.$$

First we calculate

$$\frac{\partial \mathbf{L}}{\partial a^{[L]}_k} = a^{[L]}_k - y_k, \qquad k = 1, \dots, n^{[L]}.$$

The above equalities may be written in the following abbreviated form,

$$\frac{\partial \mathbf{L}}{\partial a^{[L]}} = \left[a_k^{[L]} - y_k\right]_{k=1,\dots,n^{[L]}}, \tag{0.1}$$

where on both sides we have a column vector.

Back propagation, inductive step. Let $l \in \{1, \dots, L\}$, suppose that we are given a vector

$$\frac{\partial \mathbf{L}}{\partial a^{[l]}}.$$

Recall the formula

$$a^{[l]} = \left[\varphi\Big(\sum_{j=0}^{n^{[l-1]}} w_{k,j}^{[l]} x_j^{[l-1]}\Big)\right]_{k=1,\dots,n^{[l]}}.$$

Therefore we can calculate $\frac{\partial \mathbf{L}}{\partial w_{k_0,j_0}^{[l]}}$ using chain rule

$$\begin{aligned}
\frac{\partial \mathbf{L}}{\partial w_{k_0,j_0}^{[l]}} &= \sum_{k=1,\dots,n^{[l]}} \frac{\partial \mathbf{L}}{\partial a_k^{[l]}} \cdot \frac{\partial a_k^{[l]}}{\partial w_{k_0,j_0}^{[l]}} \\
&= \frac{\partial \mathbf{L}}{\partial a_{k_0}^{[l]}} \, \varphi' \left(\sum_{j=0}^{n^{[l-1]}} w_{k_0,j}^{[l]} x_j^{[l-1]}\right) \, x_{j_0}^{[l-1]} =: \delta_{k_0}^{[l]} x_{j_0}^{[l-1]},
\end{aligned}$$

where the column vector $\delta^{[l]}$ is given by

$$\delta^{[l]} = \left[\frac{\partial \mathbf{L}}{\partial a_{k_0}^{[l]}} \, \varphi' \left(\sum_{j=0}^{n^{[l-1]}} w_{k_0,j}^{[l]} x_j^{[l-1]}\right)\right]_{k=1,\dots,n^{[l]}}.$$

Such a notation allows us to write the above formulae in the following short form,

$$\frac{\partial \mathbf{L}}{\partial w^{[l]}} = \delta^{[l]} \cdot (x^{[l-1]})^T,$$

where on both sides we have matrices with $n^{[l]}$ rows and $(n^{[l-1]} + 1)$ columns.

Similarly we may calculate $\frac{\partial \mathbf{L}}{\partial x_{j_0}^{[l-1]}}$ using chain rule

$$\begin{aligned}
\frac{\partial \mathbf{L}}{\partial x_{j_0}^{[l-1]}} &= \sum_{k=1}^{n^{[l]}} \frac{\partial \mathbf{L}}{\partial a_k^{[l]}} \cdot \frac{\partial a_k^{[l]}}{\partial x_{j_0}^{[l-1]}} \\
&= \sum_{k=1}^{n^{[l]}} \frac{\partial \mathbf{L}}{\partial a_k^{[l]}} \varphi' \left(\sum_{j=0}^{n^{[l-1]}} w_{k,j}^{[l]} x_j^{[l-1]}\right) \, w_{k,j_0}^{[l]} = \sum_{k=1}^{n^{[l]}} w_{k,j_0}^{[l]} \delta_k^{[l]}.
\end{aligned}$$

In the matrix notation

$$\frac{\partial \mathbf{L}}{\partial x^{[l-1]}} = (W^{[l]})^T \cdot \delta^{[l]}.$$

Recall that $x_j^{[l-1]} = a_j^{[l-1]}$ for $j \geq 1$, so by omitting the first entry of the above vector we obtain

$$\frac{\partial \mathbf{L}}{\partial a^{[l-1]}}.$$

Back propagation, summary. Using formula (0.1), we may apply inductive step for $l = L$, obtaining the derivatives of the loss function with respect to the weights from the last layer, and also $\frac{\partial \mathbf{L}}{\partial a^{[L-1]}}$. This allows us to carry on applying the inductive step for $l = L-1, \ldots, 1$. In this way we will obtain the derivatives of $\mathbf{L}$ with respect to all weights, which allows us to use *gradient descent*.

Softmax function and categorical cross entropy. In categorical problems one often uses *softmax* as the activation function in the *last* layer. It has the advantage that then the output of the network has nonnegative entries with sum equal to one. Therefore this output may be interpreted as a probability distribution. The drawback is that softmax, unlike the activation function φ considered before, is the function of the whole vector $net^{[L]}$,

$$a^{[L]} = \psi(net^{[L]}) := \left[\frac{\exp(net_k^{[L]})}{\sum_{j=1}^{n^{[L]}} \exp(net_j^{[L]})} \right]_{k=1,\ldots,n^{[L]}}.$$

Thus the back propagation for the last layer will have a different form, which we will now find. We assume that *categorical cross entropy*,

$$\mathbf{L}(y, a^{[L]}) = -\sum_{k=1}^{n^{[L]}} y_k \log(a_k^{[L]}) = -\sum_{k=1}^{n^{[L]}} y_k \left(net_k^{[L]} - \log(\sum_{j=1}^{n^{[L]}} \exp(net_j^{[L]})) \right).$$

is our loss function, which is typical when softmax is used for activation. Note that $a_k^{[L]} > 0$, hence the function $\mathbf{L}$ is well-defined. Recall that

$$net_k^{[L]} = \sum_{j=0}^{n^{[L-1]}} w_{k,j}^{[L]} x_j^{[L-1]}, \qquad k = 1, \ldots, n^{[L]}.$$

We calculate

$$\begin{aligned}
\frac{\partial \mathbf{L}}{\partial w_{k_0,j_0}^{[L]}} &= -y_{k_0} x_{j_0}^{[L-1]} + \sum_{k=1}^{n^{[L]}} y_k \frac{1}{\sum_{j=1}^{n^{[L]}} \exp(net_j^{[L]})} \cdot \left(\frac{\partial}{\partial w_{k_0,j_0}^{[L]}} \exp(net_{k_0}^{[L]}) \right) \\
&= -y_{k_0} x_{j_0}^{[L-1]} + \sum_{k=1}^{n^{[L]}} y_k \frac{\exp(net_{k_0}^{[L]})}{\sum_{j=1}^{n^{[L]}} \exp(net_j^{[L]})} x_{j_0}^{[L-1]} \\
&= \left(\left(\sum_{k=1}^{n^{[L]}} y_k \right) \cdot a_{k_0}^{[L]} - y_{k_0} \right) x_{j_0}^{[L-1]}.
\end{aligned}$$

Putting

$$\delta^{[L]} = \left[\left(\sum_{j=1}^{n^{[L]}} y_j \right) \cdot a_k^{[L]} - y_k \right]_{k=1,\ldots,n^{[l]}}, \tag{0.2}$$

we obtain the same formula as before, namely

$$\frac{\partial \mathbf{L}}{\partial w^{[L]}} = \delta^{[L]} \cdot (x^{[L-1]})^T.$$

Similarly, we will calculate

$$
\begin{aligned}
\frac{\partial \mathbf{L}}{\partial x_{j_0}^{[L-1]}} &= -\sum_{k=1}^{n^{[L]}} y_k w_{k,j_0}^{[L]} + \sum_{k=1}^{n^{[L]}} y_k \frac{1}{\sum_{j=1}^{n^{[L]}} \exp(net_j^{[L]})} \cdot \left(\frac{\partial}{\partial x_{j_0}^{[L-1]}} \sum_{j=1}^{n^{[L]}} \exp(net_j^{[L]}) \right) \\
&= -\sum_{k=1}^{n^{[L]}} y_k w_{k,j_0}^{[L]} + \sum_{k=1}^{n^{[L]}} y_k \sum_{p=1}^{n^{[L]}} \frac{\exp(net_p^{[L]}) w_{p,j_0}^{[L]}}{\sum_{j=1}^{n^{[L]}} \exp(net_j^{[L]})} \\
&= -\sum_{p=1}^{n^{[L]}} y_p w_{p,j_0}^{[L]} + \sum_{k=1}^{n^{[L]}} y_k \sum_{p=1}^{n^{[L]}} a_p^{[L]} w_{p,j_0}^{[L]} \\
&= \sum_{p=1}^{n^{[L]}} \left(\Big(\sum_{k=1}^{n^{[L]}} y_k \Big) a_p^{[L]} - y_p \right) w_{p,j_0}^{[L]} \\
&= \sum_{p=1}^{n^{[L]}} \delta_p^{[L]} w_{p,j_0}^{[L]}.
\end{aligned}
$$

We have again obtained the same formula as before, in matrix notation,

$$\frac{\partial \mathbf{L}}{\partial x^{[L-1]}} = (W^{[L]})^T \cdot \delta^{[L]}.$$

To sum up, the same formulae hold provided we modify the definition of $\delta^{[L]}$ (only for the last layer) by putting (0.2).

Finally, let us see that the formula for $\delta^{[L]}$ may be simplified, if we additionally assume that $\sum_{k=1}^{n^{[L]}} y_k = 1$ (which is typical for classification tasks). Then

$$\delta^{[L]} = \left[a_k^{[L]} - y_k \right]_{k=1,\dots,n^{[l]}}.$$