Neural network

Luis Fabián Huecas López

4 de abril de 2025

1 Main operation

1.1 Definitions and indexes

Here we specify all the matrices we will be using in the network. We denote n the total number of layers: the features, $n-2$ hidden layers and the output layer.

1. weights[i][j][k]. i, inter-layer. j, pos-node-prev-layer. k, pos-node-next-layer.
2. nodes[p][q]. p, layer. q,pos-in-the-layer.
3. b[p'][q]. p'=p-1, layer - 1. q, pos-in-the-layer.

1.2 Operation

$$nodes_{ij} = h\left(\sum_{k=0}^{k=n_{i-1}} w_{i-1,kj} \cdot nodes_{i-1,k} + b_{i-1,j}\right) \tag{1.1}$$

where n_l is the number of nodes in l-layer and $w \equiv weights$.

This equation is valid from $i = 1$ to $i = n-1$. The case for the features $nodes_{0j}$ they simply are given. In matricial and vector notation we have:

$$\vec{nodes}^{\,i} = h\left(\left(W^{i-1}\right)^T \vec{nodes}^{\,i-1} + \vec{b}^{i-1}\right) \tag{1.2}$$

2 Backpropagation

Somehow we will be able to define a function that takes into account the error with respect the correct results. We call this function the loss function, our goal is to minimize it with respect the network parameters (w_{ij},b_i). Whatever it is the loss function we need to calculate: $\frac{\partial L}{\partial w_{lij}}$ and $\frac{\partial L}{\partial b_{lj}}$. For that reason let's define the following quantities:

$$\vec{z}^{(l)} \equiv \left(W^{(l)}\right)^T \vec{nodes}^{\,(l)} + \vec{b}^{(l)} \tag{2.1}$$

this is the argument for the activation function h and:

$$\delta^{(l)} \equiv \frac{\partial L}{\partial z^{(l)}} = \frac{\partial L}{\partial h}\frac{\partial h}{\partial z^{(l)}} \tag{2.2}$$

For $i = L \equiv n-1$ (output layer) we have:

$$\vec{nodes}^{L} = h\left(\left(W^{L-1}\right)^{T}\vec{nodes}^{L-1} + \vec{b}^{L-1}\right) = h\left(\vec{z}^{(L-1)}\right) \tag{2.3}$$

In general we have the following result:

$$\delta^{(l)} = \left(\left(W^{l+1}\right)\delta^{(l+1)}\right)\frac{\partial h}{\partial z^{(l)}} \tag{2.4}$$

$$\delta_j^{(l)} = \sum_{k=0}^{k<num-nodes-next} W_{jk}^{l+1}\delta_k^{l+1}\frac{\partial h}{\partial z}(z = z_j^l) \tag{2.5}$$

$$\frac{\partial L}{\partial b_j^{(l)}} = \delta_j^{(l)} \tag{2.6}$$

$$\frac{\partial L}{\partial w_{kj}^{(l)}} = nodes_k^l\delta_j^{(l)} \tag{2.7}$$

2.1 Loss function and activation function

The cross-entropy loss function is:

$$L_{ent} = -\sum_{j=0}^{j=M-1}\hat{y}_j log(nodes_{L,j}) = -\sum_{j=0}^{j=M-1}\hat{y}_j log(h\left(z_{L-1,j}\right)) \tag{2.8}$$

where h is the activation function in the output layer and M is the number of classes. $\hat{y}$ is the real result. We have:

$$\delta^{L-1} = \frac{\partial L_{ent}}{\partial \vec{z}^{(L-1)}} = h(\vec{z}^{(L-1)}) - \hat{y} = \vec{nodes}^{L} - \hat{y} \tag{2.9}$$

For the multi classification case h is the softmax function. So for $l = L-1$ we have:

$$\frac{\partial L}{\partial b_j^{(L-1)}} = (nodes_j^L - \hat{y}_j) \tag{2.10}$$

$$\frac{\partial L}{\partial w_{kj}^{(L-1)}} = nodes_k^{L-1}(nodes_j^L - \hat{y}_j) \tag{2.11}$$

2.2 Gradients with ReLU activation function

If we work with ReLU function as our activation function the $\delta^{(l)}$ can be developed as the following expression:

$$\delta_j^{(l)} = \sum_{k=0}^{k<num-nodes-next} W_{jk}^{l+1}\delta_k^{l+1}\begin{cases}1 & \text{if } z_j^l > 0\ ,\\ 0 & \text{else}\quad .\end{cases} \tag{2.12}$$

2.3 Changing values based on gradients

$$b_j^l = b_j^l - \eta \frac{\partial L}{\partial b_j^l} \tag{2.13}$$

$$w_{kj}^l = w_{kj}^l - \eta \frac{\partial L}{\partial w_{kj}^l} \tag{2.14}$$

For $l = L - 1$ we use [2.10] and [2.11] and for $l < L - 1$ we use [2.12].