1. Feedforward neural networks

1.3a. The Backpropagation algorithm (1)

Manel Martínez-Ramón

Department of Electrical and Computer Engineering
The University of New Mexico

Gradient descent strategy

- NN training criterion: maximize the cross entropy between a measured distribution $q(y|\mathbf{x})$ of the data and a model $p(y|\mathbf{x})$

$$\begin{aligned} J_{ML}(\mathbf{Y}, \mathbf{X}, \boldsymbol{\theta}) &= -\mathbb{E}\left[q\left(\mathbf{y}|\mathbf{x}\right)\log p\left(\mathbf{y}|\mathbf{x}\right)\right] \\ &\approx -\sum_j \log p\left(\mathbf{y}_j|\mathbf{x}_j\right) \end{aligned} \tag{1}$$

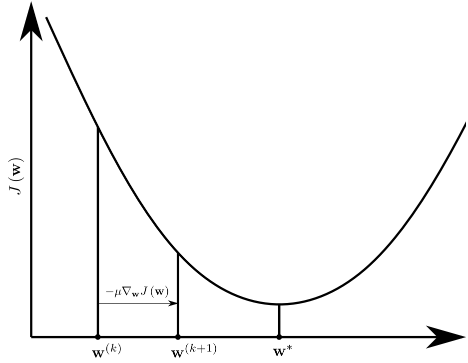
- Where $q(\cdot)$ is binary. The criterion is equivalent to maximize the output likelihood.

Gradient descent strategy

- Goal of the training: find the maximum of the cross entropy (or the minimum of the negative cross entropy).
- Optimization:

$$\frac{\partial J_{ML}(\boldsymbol{\theta})}{\partial w_{j,k}^{(l)}} = 0, \forall j, k, l \tag{2}$$

- This is an equation that cannot be solved in a single step.
- We need to proceed sequentially: gradient descent.



- Optimum value $\mathbf{w}^*$ achieved at the minimum of the cost function, where

$$\nabla_{\mathbf{w}} J_{ML}(\mathbf{w}) = 0$$

- $\mathbf{w}^k$ is modified in the direction of the gradient descent times a small constant μ.
- The operation must be repeated until the gradient is zero.

- The output activation $\mathbf{o}$ has elements o_j
- $\mathbf{z}^{(L)} = \mathbf{W}^{(L)\top}\mathbf{h}^{(L-1)}$ is a function of the previous layer, with components $h_i^{(L-1)}$.
- We apply the chain rule to these three elements and to weight $w_{i,j}^{(L)}$.

$$\frac{dJ_{ML}}{dw_{i,j}^{(L)}} = \frac{dJ_{ML}}{do_j}\frac{do_j}{dz_j^{(L)}}\frac{dz_j^{(L)}}{dw_{i,j}^{(L)}} = h_i^{(L-1)}\delta_j^{(L)} \tag{3}$$

- We have three terms:

1. $\frac{d}{do_j} J_{ML}$
2. $\frac{d}{dz_j^{(L)}} o_j\left(z_j^{(L)}\right) = o'_j$
3. $\frac{d}{dw_{i,j}^{(L)}} z_j^{(L)} = h_i^{(L-1)}$

- Therefore

$$\frac{dJ_{ML}}{dw_{i,j}^{(L)}} = \frac{dJ_{ML}}{do_j} o'_j h_i^{(L-1)} = h_i^{(L-1)} \delta_j^{(L)} \tag{4}$$

- We have used the following definition

$$\delta_j^{(L)} = \frac{dJ_{ML}}{do_j} o'_j \tag{5}$$

- This is element j of vector

$$\boldsymbol{\delta}^{(L)} = \nabla_{\mathbf{o}} J_{ML}(\mathbf{y}, \mathbf{o}) \odot \mathbf{o}' \tag{6}$$

which is the elementwise product $\odot$ of two vectors.

- Then, derivative

$$\frac{dJ_{ML}}{dw_{i,j}^{(L)}} = h_i^{(L-1)}\delta_j^{(L)}$$

is element i, j of a matrix that can be written as $\mathbf{h}^{(L-1)}\boldsymbol{\delta}^{(L)\top}$, thus

$$\nabla_{\mathbf{W}^{(L)}} J_{ML} = \mathbf{h}^{(L-1)}\boldsymbol{\delta}^{(L)\top} \tag{7}$$

By using expression (7), the update of the last layer of the NN consists in the following update operation,

$$\mathbf{W}^{(L)} \leftarrow \mathbf{W}^{(L)} - \mu \mathbf{h}^{(L-1)} \boldsymbol{\delta}^{(L)\top} \tag{8}$$

where μ is a small scalar usually called the learning rate.

- The process can be iterated down to the input layer, with the same result, and therefore the update of weight matrix $\mathbf{W}^{(l-1)}$ is

$$\mathbf{W}^{(l-1)} \leftarrow \mathbf{W}^{(l-1)} - \mu \mathbf{h}^{(l-2)} {\boldsymbol{\delta}^{(l-1)}}^{\top} \tag{9}$$

where

$$\boldsymbol{\delta}^{(l-1)} = \mathbf{W}^{(l)} \boldsymbol{\delta}^{(l)} \odot \phi' \left(\mathbf{z}^{(l-1)} \right) \tag{10}$$

where to start and end the process, we need

$$\begin{aligned} \boldsymbol{\delta}^{(L)} &= \nabla_{\mathbf{o}} J_{ML}(\mathbf{y}, \mathbf{o}) \odot \mathbf{o}' \\ \mathbf{h}^{(0)} &= \mathbf{x} \quad \text{(Input layer)} \end{aligned} \tag{11}$$