1. Feedforward neural networks

1.3a. The Backpropagation algorithm (1)

Manel Martínez-Ramón
Meenu Ajith
Aswathy Rajendra Kurup

Introduction

- So far we have seen the details of the structure of a NN, and its training criterion (maximum likelihood).
- The Backpropagation algorithm implements the criterion over the structure.
- The algorithm is based on a gradient descent strategy. We need to learn:
    - The basic algorithm
    - Its practical coding
    - The additions needed to actually make it work.

Gradient descent strategy

- The criterion consists of maximizing the cross entropy between a measured distribution $q(y|\mathbf{x})$ of the data and a model $p(y|\mathbf{x})$, implemented as

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

- We assume that $q(\cdot)$ is binary, thus the criterion is equivalent to maximize the output likelihood with respect to the data.

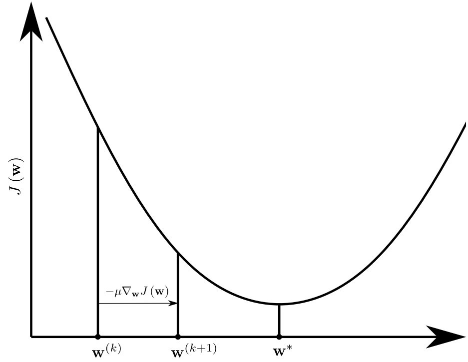
Gradient descent strategy

- The goal of the neural network training is to find the maximum of the cross entropy (or the minimum of the negative cross entropy).
- The minimum can be found by nulling the gradient of the cost function, The goal is:

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

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

Gradient descent strategy



- 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.

Gradient wrt output weigths

- Function implemented by the NN as a composition of functions

$$\begin{aligned}\mathbf{f}(\mathbf{x}) = \mathbf{o}\left(\mathbf{z}^{(L)}\right) = \mathbf{o}\left({\mathbf{W}^{(L)}}^\top \mathbf{h}^{(L-1)}\right) \\ = \mathbf{o}\left({\mathbf{W}^{(L)}}^\top \boldsymbol{\phi}\left({\mathbf{W}^{(L-1)}}^\top \mathbf{h}^{(L-2)}\right)\right) = \cdots\end{aligned} \quad (3)$$

- Derivative wrt the output weights

$$\begin{aligned}\frac{d}{dw_{i,j}^{(L)}} J_{ML}(\mathbf{y}, \mathbf{f}(\mathbf{x})) = \frac{d}{dw_{i,j}^{(L)}} J_{ML}\left(\mathbf{y}, \mathbf{o}\left(\mathbf{z}^{(L)}\right)\right) = \\ \frac{d}{dw_{i,j}^{(L)}} J_{ML}\left(\mathbf{y}, \mathbf{o}\left({\mathbf{W}^{(L)}}^\top \mathbf{h}^{(L-1)}\right)\right)\end{aligned} \quad (4)$$

- We assume here that the bias vectors are inside $\mathbf{W}^{(L)}$

Chain rule

Output weights

- 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 $w_{i,j}^{(L)}$.

$$\frac{dJ_{ML}}{dw_{i,j}^{(L)}} = \frac{dJ_{ML}}{do_j} \frac{do_j}{dz_j^{(L)}} \frac{dz_j^{(L)}}{dw_{i,j}^{(L)}} \tag{5}$$

Chain rule

Output weights

- 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

$$\frac{dJ_{ML}}{dw_{i,j}^{(L)}} = \frac{dJ_{ML}}{do_j} o_j' h_i^{(L-1)} = h_i^{(L-1)} \delta_j^{(L)} \tag{6}$$

Chain rule
Output weights

- We have used the following definition

$$\delta_j^{(L)} = \frac{dJ_{ML}}{do_j} o'_j \tag{7}$$

- This is element j of vector

$$\boldsymbol{\delta}^{(L)} = \nabla_{\mathbf{o}} J_{ML}(\mathbf{y}, \mathbf{o}) \odot \mathbf{o}' \tag{8}$$

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

Chain rule

Output weights

- 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

$$\nabla_{\mathbf{W}^{(L)}} J_{ML} = \mathbf{h}^{(L-1)} \boldsymbol{\delta}^{(L)\top} \tag{9}$$

Weight update
Output weights

By using expression (9), the update of the last layer of the NN consists in the following update operation,

$$\mathbf{W}^{(L)} \leftarrow \mathbf{W}^{(L)} - \mu \mathbf{h}^{(L-1)} \boldsymbol{\delta}^{(L)\top} \tag{10}$$

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

Weight update
Output biases

- Now, what happened with the biases? We put them inside $\mathbf{W}^{(L)}$

$$\mathbf{W}^{(L)} = \begin{pmatrix} \mathbf{w}_1^{(L)} & \mathbf{w}_2^{(L)} & \cdots & \mathbf{w}_{D_L}^{(L)} \\ \\ b_1^{(L)} & b_2^{(L)} & \cdots & b_{D_L}^{(L)} \end{pmatrix} \tag{11}$$

- so we redefined $\mathbf{h}^{(L-1)} = \begin{pmatrix} h_1^{(L-1)} & h_2^{(L-1)} & \cdots & h_{D_{L-1}}^{(L-1)} & 1 \end{pmatrix}^\top$
- therefore if

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

then

$$\frac{dJ_{ML}}{db_j^{(L)}} = \delta_j^{(L)} \tag{12}$$

This is, Eq. (10) is valid without loss of generality.