Mathematical theory of feedforward neural networks

Thomas Savary

Feedforward neural networks can be seen as parametric functions that take as input a feature vector $x \in \mathbb{R}^p$ and produce outputs in $\mathbb{R}^n$:

$$f_\theta : \left| \begin{array}{ccc} \mathbb{R}^p & \longrightarrow & \mathbb{R}^n \\ x & \longmapsto & f_\theta(x) \end{array} \right.$$

The dimension n of the output space depends on the task at hand (regression, binary classification, or multiclass classification).

More precisely, these models are a composition of linear functions and activation functions (generally non-linear):

$$f_\theta(x) = \sigma_m \left(\cdots \sigma_2 \left(W_2 \sigma_1 \left(W_1 x + b_1 \right) + b_2 \right) \right) = f^{(m)}_{\theta_m} \circ \cdots \circ f^{(2)}_{\theta_2} \circ f^{(1)}_{\theta_1}(x)$$

with $f^{(i)}_{\theta_i}(x) = \sigma_i \left(W_i x + b_i \right)$ a function from $\mathbb{R}^{d_{in}}$ to $\mathbb{R}^{d_{out}}$ called layer of the network and characterized by the matrix $W_i \in \mathcal{M}_{d_{out}, d_{in}}(\mathbb{R})$ and the vector $b_i \in \mathbb{R}^{d_{out}}$.

In the context of supervised learning, the goal is to optimize the network's parameters $\{W_1, b_1, \cdots, W_m, b_m\}$ by minimizing a loss function $\mathcal{L}$ on a training dataset $\mathcal{D}_n = \{(x_i, y_i)_{1 \leq i \leq n}\}$. To do this, we use a gradient descent algorithm:

$$\theta^{(k+1)} = \theta^{(k)} - \alpha \nabla_{\theta^{(k)}} \mathcal{L}$$

Hence, we need to compute the gradient of the loss function with respect to each of the network's parameters. To do this, we use the following mathematical results:

1. **Riesz representation theorem**: for any continuous linear form f on a Hilbert space $\mathcal{H}$, there exists a unique element u in $\mathcal{H}$ such that $f(x) = \langle x, u \rangle$. The differential of a function f at a point $x \in \mathcal{H}$ being a continuous linear form on $\mathcal{H}$, there exists therefore a unique element u_x such that $Df(x) = \langle \cdot, u_x \rangle$, this element is the gradient of f at x.
2. **Differential of a composition of functions**: the differential at point x of a composition of two differentiable functions $g \circ f$ is given by:

$$D(g \circ f)_x(h) = Dg_{f(x)} \circ Df_x$$

We use the second point to compute the differential of the function that takes a network parameter as input (W_i or b_i) and computes the associated loss function for a sample from the training dataset. Then, we use the first point to find the gradient of the function with respect to the parameter in question. Results and details are given in the following wikipedia article.

Remarks:

- As matrix-matrix computations are more efficient than matrix-vector ones in practice, we perform the computation for several samples of the training dataset simultaneously. This is known as a batch.
- More complex optimization algorithms are often used in practice, by adding a momemtum for example.
- In libraries such as Pytorch and Tensorflow, gradients are computed automatically by automatic differentiation.