# *Using a convolutional neural network to classify the Street View House Number dataset*

Andrea Ferretti
andrea.ferretti1@studenti.unimi.it

## Neural networks

Neural networks are a family of predictors characterized by the combination of simple computational units, called neurons. A neuron typically performs the following computation: $g(\boldsymbol{x}) = \sigma(\boldsymbol{\omega}^T \boldsymbol{x})$, where the elements of the vector $\boldsymbol{\omega}$ are the parameters of the neuron, $\sigma$ is a non-linear function called activation function, and $\boldsymbol{x}$ is the vector $x_0, \ldots, x_n$ where $x_0 = 1$ and $x_i$ for $i \in \{i, \ldots, n\}$ the output computed by another neuron. In the supervised learning setting, neurons are combined in a graph-like structure resulting in a computational network able to learn, by adjusting the parameters $\boldsymbol{\omega}$ of each neuron, the underlying mapping between data points and their corresponding labels.

One of the most basic architecture a neural network can assume is called feedforward network. In this type of network neurons are organized into successive layers: one input layer, one or more hidden layers, and one output layer. The computation flows from the input layer towards the output layer and each neuron passes its output to all the neurons in the next layer. The $i$-th neuron in the input layer simply outputs the value of the $i$-th dimension of the data point. The neurons in the other layers compute a non-linear function of the weighted sum of the outputs of the neurons in the previous layer (plus a bias), every neuron has therefore his own vector of weights and a bias term. The funciton computed by the neurons in the hidden layers is

called the activation function and usually is some kind of sigmoid function such as the logistic function:

$$S(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

The neurons in the output layer compute a function that varies depending on the type of problem: for a regressor the identity function can be used to obtain a real valued output for each neuron, for a classifier, instead, the softmax function is used. The softmax function $softmax$ : $\mathbb{R}^m \to \mathbb{R}^m$ is defined as:

$$softmax(\boldsymbol{v})_i = \frac{e^{\boldsymbol{v}_i}}{\sum_{t=0}^{m} e^{\boldsymbol{v}_t}}, \text{ for } i = 0, \ldots, m-1 \text{ and } \boldsymbol{v} \in \mathbb{R}^m \tag{2}$$

By having a network with as many output neurons as the number of possible classes to be predicted a probability distribution over those classes is obtained using softmax. It is to note that in order for this function to be computed every output neuron needs to know the values of the other output neurons (the various $\boldsymbol{v}_t$) before the softmax is applied. In Figure 1 is given an example of a feedforward neural network.

Consider the pair $(\boldsymbol{x}, y)$ consisting of a data point and its corresponding label, let $\hat{y}$ be the label predicted by the network, in order to assest the goodness of the prediction a non-negative loss function $\ell(y, \hat{y})$ is used so that the greater is the value of the loss the worse is the prediciton. In the case of a classification problem $y \in Y$, where $Y$ is the ordered set of possible symbolic labels, and $|Y| = m$, therefore by having a network with $m$ neurons in the output layer and using the $softmax$ function, the network prediction $\hat{\boldsymbol{y}}$ represents a probability distribution over $Y$ given the data point $\boldsymbol{x}$. For conveniency of notation let $y_i^*$ indicate the first element of $Y$ for $i = 0$, the second element of $Y$ for $i = 1$ and so on for all the elements of $Y$. This means that the $i$-th element of $\hat{\boldsymbol{y}}$ gives the probability that the label for data point $\boldsymbol{x}$ will be $y_i^*$ in the network prediction. In this case as the loss function the categorical cross-entropy loss can be

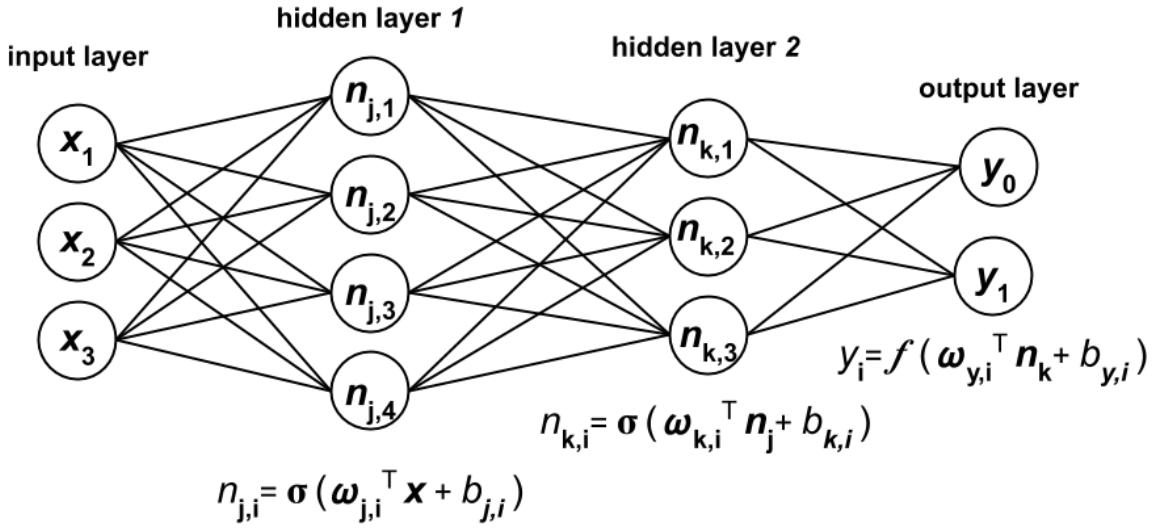


Figure 1: A feedforward network where on each neuron there is a label representing the value computed by it and passed to the nodes of the next layer through the arcs connecting them. A neuron assigns a weight $\omega$ to each incoming arch and performs the shown linear combination. The sigmoid funciton of equation (1) can be used as the activation function $\sigma$. The function computed by the neurons in the output layer could be $softmax$ (2) where $\boldsymbol{v}_i = \boldsymbol{\omega}_{y,i}^T \boldsymbol{n}_k + b_{y,i}$. Here the bias $b$ is explicited in the computation rather than adding another dimension to the various vectors $\boldsymbol{\omega}$, $\boldsymbol{n}$, and $\boldsymbol{x}$ so that $\omega_{r,i,0} = b_{r,i}$ for $r \in \{j, k, y\}$ and $x_0 = n_{p,0} = 1$ for $p \in \{j, k\}$.

used and it is defined as:

$$\ell(y, \hat{y}) = -\sum_{i=0}^{m-1} \mathbb{I}(y_i^* = y) \log(\hat{y}_i) \tag{3}$$

where $\mathbb{I}(E) \in \{0, 1\}$ is the indicator function of an event $E$, meaning that $\mathbb{I}(E) = 1$ if and only if $E$ occours and $\mathbb{I}(E) = 0$ otherwise. Equation 3 can be simplified to

$$\ell(y, \hat{y}) = -log(\hat{y}_i), \text{ where } i \text{ verifies } y_i^* = y$$

## The Street View House Number dataset

## Model training

## Results