Feed-forward neuronet recognising digits

Batarin Egor, Beksultanova Aikun, Zubritsky Ivan

April 21, 2021

Abstract

This pdf is done to descript the neuronet architecture and give step-by-step description of learning algorithms.

1 Accepted notations

$x \in \mathbb{R}^p$ - input value of neuronet, $p =$ lenght $\times$ width - number of image pixels. x consists of 0 - "white color" and 1 - "black color".

$y = y(x) \in \mathbb{R}^{10}$ - desired output value. If input x responds to a digit $i \in \{0, 1, ..., 9\}$, then $y(x)$ consists of nine zeros, and one, which is in a i-th place.

L - number of the last (output) layer.

$n : \{1, \ldots, L\} \to \mathbb{N}$ - number of neurons in the l-th layer, $n(1) = p$, $n(L) = 10$.

$$\omega^l = \begin{pmatrix} w^l_{11} & w^l_{12} & \cdots & w^l_{1,n(l-1)} \\ w^l_{21} & w^l_{22} & \cdots & w^l_{2,n(l-1)} \\ \cdots & \cdots & \cdots & \cdots \\ w^l_{j1} & w^l_{j2} & \cdots & w^l_{j,n(l-1)} \\ \cdots & \cdots & \cdots & \cdots \\ w^l_{n(l),1} & w^l_{n(l),2} & \cdots & w^l_{n(l),n(l-1)} \end{pmatrix}$$

w^l is weight matrix, which consist of w^l_{jk} - weight between j-th neuron in l layer and k-th neuron in $l-1$ layer. $l = 2, \ldots, L$.

$b^l = \left(b^l_1, b^l_2, \ldots, b^l_{n(l)}\right)^T$ is a vector of biases, b^l_j - bias of j-th neuron in the l-th layer. $l = 2, \ldots, L$.

$a^l = \left(a^l_1, a^l_2, \ldots, a^l_{n(l)}\right)^T$ is a vector of activations, b^l_j - bias of j-th neuron in the l-th layer, $x = a^1 \in \mathbb{R}^p$ - input value, $a^L \in \mathbb{R}^{10}$ - output value, predicted by neuronet. $l = 1, \ldots, L$.

We define weighted input vector $z^l = \left(z^l_1, z^l_2, \ldots, z^l_{n(l)}\right)$, where $z^l_j = \sum\limits_{k=1}^{n(l-1)} \omega^l_{jk} a^{l-1}_k + b^l_j$. It can be written in vectorized form $z^l = \omega^l a^{l-1} + b^l$. $l = 2, \ldots, L$.

$C_{\text{quad}}(x) = \frac{1}{2}\left\|y(x) - a^L(x)\right\| = \frac{1}{2}\sqrt{\sum\limits_{i=1}^{10}\left(y_i(x) - a^L_i(x)\right)^2}$ is quadratic cost, refered to a single input value x.

We will also use cross-entropy cost $C_{\text{log}}(x) = -\sum\limits_{i=1}^{10} y_i(x) \ln a^L_i(x) + (1 - y_i(x)) \ln\left(1 - a^L_i(x)\right)$

$\sigma(z) = \frac{1}{1+e^{-z}}$ - sigmoid activation function, $\sigma'(z) = \sigma(z)\left(1 - \sigma(z)\right)$. This funtion links activation and weighted input: $a^l = \sigma(z^l)$, $l = 2, \ldots, L$.

For any cost function we define $\nabla_a C = \left(\frac{\partial C}{\partial z^L_1}, \cdots, \frac{\partial C}{\partial z^L_{10}}\right)^T$, $C = \frac{1}{\text{number of inputs}} \sum\limits_x C(x)$ - average cost.

For two vectors $a = (a_1, a_2, \ldots, a_n)^T$ and $b = (b_1, b_2, \ldots, b_n)^T$ the Hadamard product $a \odot b$ is defined as

$$a \odot b = (a_1 b_1, a_2 b_2, \ldots, a_n b_n)^T$$

We define an error δ^l_j of j-th neuron in l-th layes as $\delta^l_j = \frac{\partial C}{\partial z^l_j}$, $\delta^l = \left(\delta^l_1, \ldots, \delta^l_{n(l)}\right)^T$. $l = 2, \ldots, L$.

2 Neuronet architecture

The structure of neuronet is presented in the picture below:

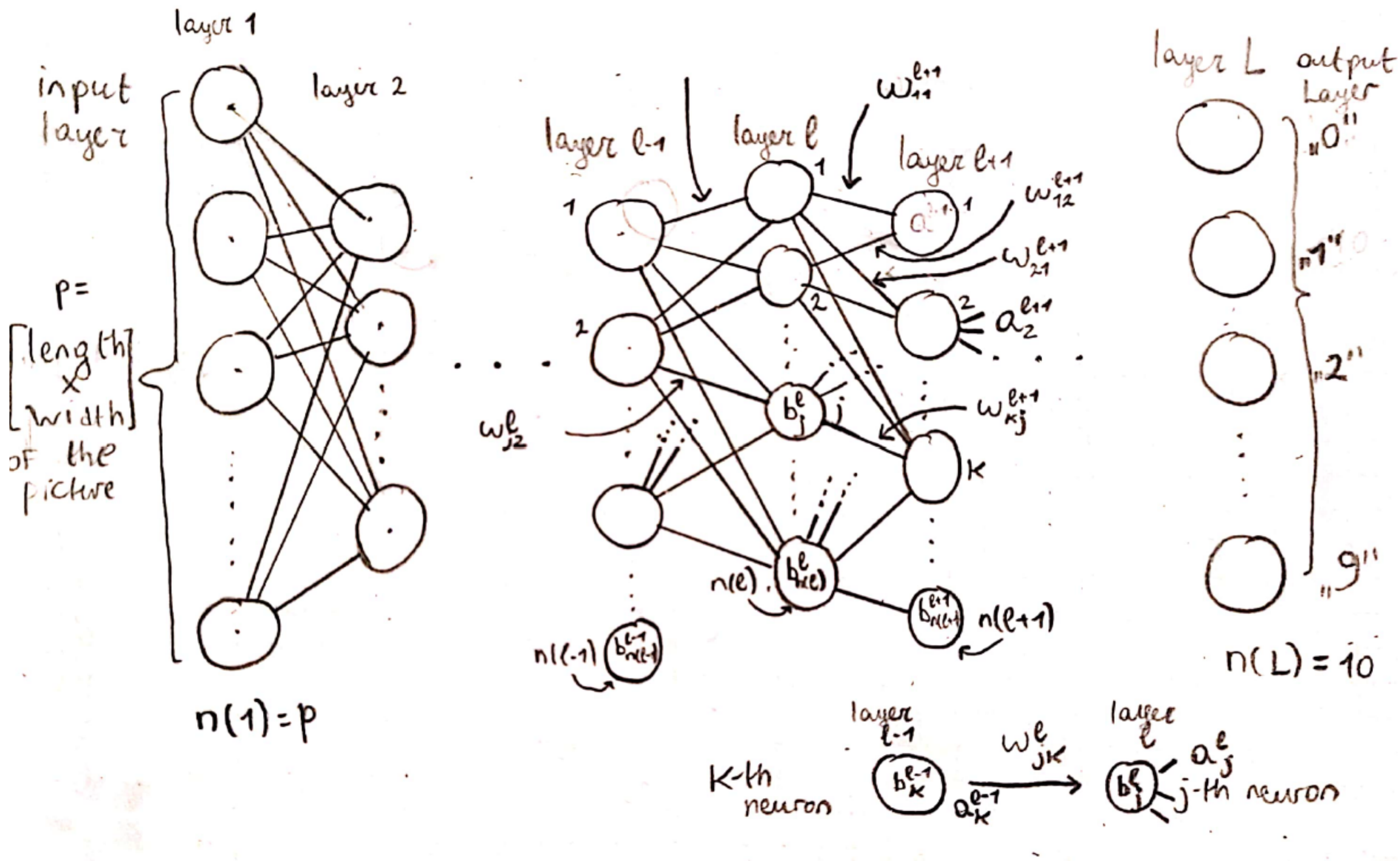

Figure 1: Neuronet structure

3 Backpropagation equations

Below four significant equations are presented, which will be used in backpropagation algorithm:

$$\delta^L = \nabla_a C \odot \sigma'(z^L) \tag{1}$$

$$\delta^l = \left(\left(\omega^{l+1}\right)^T \delta^{l+1}\right) \odot \sigma'(z^l) \tag{2}$$

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l \tag{3}$$

$$\frac{\partial C}{\partial \omega_{jk}^l} = a_k^{l-1}\delta_j^l \tag{4}$$

4 Learning algorithm

1. **Set hyper-parameters**: learning rate η, number N of epoch of training, size m of mini-batch.

2. **Repeat for each of N-th epochs of training**:

   (a) **Input a random mini-batch of m training examples x from training data**.

   (b) **For each training example x**: Set the corresponding input activation $a^{x,1} = x$, a do the following:

      i. **Feedforward**: For each $l = 2, 3, \ldots, L$ compute $z^{x,l} = \omega^l a^{x,l-1} + b^l$ and $a^{x,l} = \sigma(z^{x,l})$

      ii. **Output error $\delta_j^{x,l}$**: Compute the error vector $\delta^L = \nabla_a C \odot \sigma'(z^L)$.

      iii. **Backpropagate the error**: For each $l = L-1, L-2, \ldots, 2$ compute $\delta^l = \left(\left(\omega^{l+1} \right)^T \delta^{l+1} \right) \odot \sigma'(z^l)$.

   (c) **Gradient descent**: For each $l = L-1, L-2, \ldots, 2$ update the weights: $\omega^l \to \omega^l - \frac{\eta}{m} \sum\limits_x \delta^{x,l} \left(a^{x,l-1} \right)^T$, $b^l \to b^l - \frac{\eta}{m} \sum\limits_x \delta^{x,l}$.