Lecture Note - 07: Neural Network

Dihui Lai

March 25, 2020

Contents

1 A Single Neuron

An artificial neuron is the basic computing unit in an artificial neural network. There are different way to define a neuron. The most common one is shown in Figure 1.

Input: A neuron receives multiple inputs x_1, x_2, ..., x_n. The signals are summed up after modulated by a set of weights w_1, w_2, ... w_n. Let us denote the weighted sum z

$$z = \sum_{i=1}^{n} w_i x_i \tag{1}$$

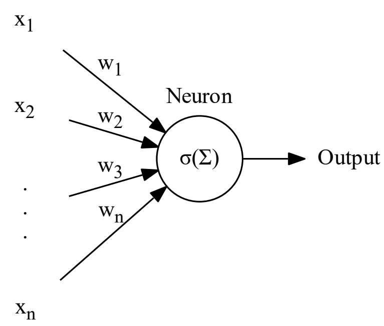

Figure 1: Schema: a single neuron with activation function σ

Usually a biased term w_0 is added to the summation and we have

$$z = \sum_{i=1}^{n} w_i x_i + w_0 \tag{2}$$

Output: The weighted sum is further transferred via an activation function σ and becomes the final output of the neuron

$$a = \sigma(z) = \sigma(\sum_{i=1}^{n} w_i x_i + w_0) \tag{3}$$

Activation: The activation function can of different types. Below is a list of common activation functions. Almost all activation functions have an S-shape except for the ReLu function.

Name	Definition
Step Function	$\sigma(z) = \begin{cases} 0 & \text{for } z < 0 \\ 1 & \text{for } z \geq 0 \end{cases}$
Logistic or sigmoid	$\sigma(z) = \frac{1}{1+e^{-z}}$
hyperbolic tangent	$\sigma(z) = \frac{(e^z - e^{-z})}{(e^z + e^{-z})}$
ReLU	$\sigma(z) = \begin{cases} 0 & \text{for } z \leq 0 \\ z & \text{for } z > 0 \end{cases}$

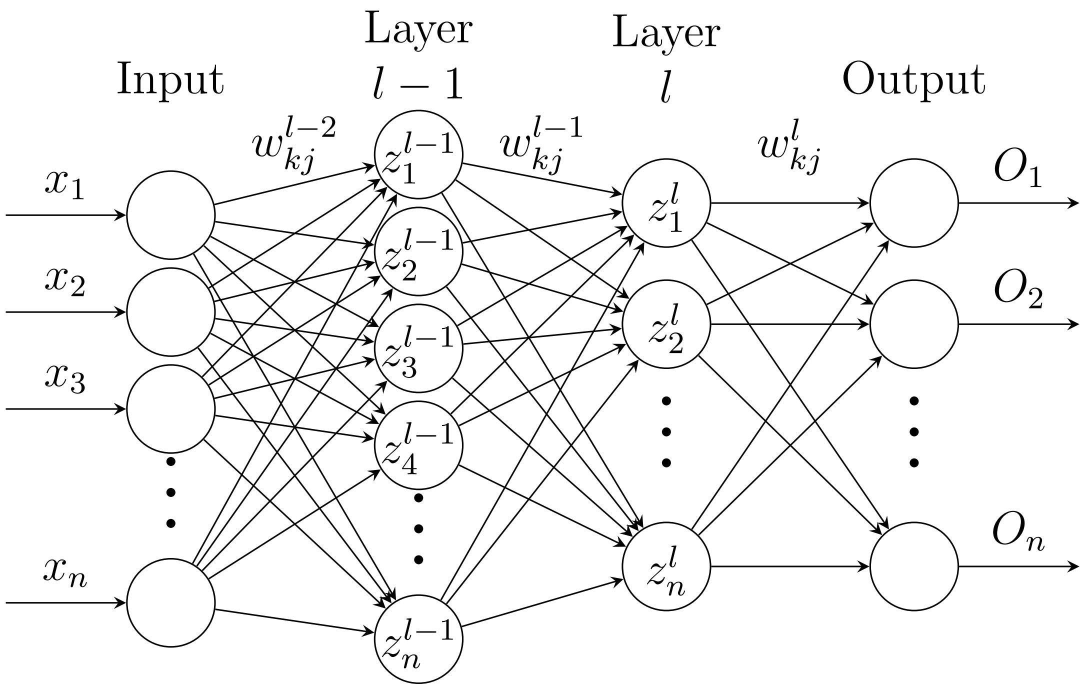

Figure 2: A neural network of multiple layer structures. The hidden layers before layer $l-1$ and after layer l are not shown.

2 Neural Network: Forward Path

Each neuron at layer l receives inputs from all neuron from the previous layer $l-1$,

$$z_k^l = \sum_j w_{kj}^{l-1} a_j^{l-1} \tag{4}$$

Here, a_j^{l-1} is the input from jth neuron from $l-1$ layer. The neurons transfer the input signal z_k^l via a transfer function σ and send as input to to the next layer

$$a_k^l = \sigma(z_k^l) \tag{5}$$

Inserting equation (4) to (5), we have

$$z_k^l = \sum_j w_{kj}^{l-1}\sigma(z_j^{l-1}) = \sum_j w_{kj}^{l-1} a_j^{l-1} \tag{6}$$

3 The Derivatives of Neural Network

In order to calculate the neural network properly we need to understand a few derivatives.

- The partial derivatives of z_k^l against z_j^{l-1}

$$\frac{\partial z_k^l}{\partial z_j^{l-1}} = w_{kj}^{l-1}\sigma'(z_j^{l-1}) \tag{7}$$

- The partial derivatives of z_k^l against a weight element w_{kj}^{l-1}

$$\frac{\partial z_k^l}{\partial w_{kj}^{l-1}} = \sigma(z_j^{l-1}) \tag{8}$$

- Given a function of $f(z_j^{l-1})$, we can explicitly express it in terms of z^ls,

$$f(z_j^{l-1}) = f(z_1^l(z_j^{l-1}), z_2^l(z_j^{l-1})...z_m^l(z_j^{l-1})...)$$

  its derivative w.r.t z_j^{l-1} is

$$\frac{\partial f}{\partial z_j^{l-1}} = \sum_m \frac{\partial f}{\partial z_m^l}\frac{\partial z_m^l}{\partial z_j^{l-1}} = \sum_m \frac{\partial f}{\partial z_m^l} w_{mj}^{l-1}\sigma'(z_j^{l-1}) \tag{9}$$

4 Backpropagation

Consider neural network that has N layers, the cost function is dependent on all the zs of neurons in all layers

$$C\left(\vec{z}^N(\vec{z}^{N-1}(...\vec{z}^l(\vec{z}^{l-1})...)...\vec{z}^1)\right) \tag{10}$$

In order to find the weights that optimiaze the cost function, we can use gradient descent. Recall the gradient descent methods updates weights of layer $l-1$ as of the following

$$w_{kj}^{l-1} \leftarrow w_{kj}^{l-1} - \eta\frac{\partial C}{\partial w_{kj}^{l-1}} \tag{11}$$

By using the chain rule, we have

$$\frac{\partial C}{\partial w_{kj}^{l-1}} = \frac{\partial C(...(z_k^l(w_{kj}^{l-1}, ...)))}{\partial w_{kj}^{l-1}} = \frac{\partial C}{\partial z_k^l}\frac{\partial z_k^l}{\partial w_{kj}^{l-1}} \tag{12}$$

There are two terms on the R.H.S. of the equation,

- Using equation (9), the first term becomes

$$\frac{\partial C}{\partial z_k^l} = \sum_m \frac{\partial C}{\partial z_m^{l+1}} w_{mk}^l \sigma'(z_k^l)$$

Use notation $\delta_k^l = \frac{\partial C}{\partial z_k^l}$ for short, we have

$$\delta_k^l = \sum_m \delta_m^{l+1} w_{mk}^l \sigma'(z_k^l) \tag{13}$$

- Using what we calculated in equation (8), the second term is simply the output from jth neuron in layer $l-1$

$$\frac{\partial z_k^l}{\partial w_{kj}^{l-1}} = \sigma(z_j^{l-1}) = a_j^{l-1} \tag{14}$$

Finally, we get our iteration methods for calculating the gradient of the cost function w.r.t the weights i.e.

$$\frac{\partial C}{\partial w_{kj}^{l-1}} = \delta_k^l a_j^{l-1} \tag{15}$$

5 Neural Network Training Algorithm

So the overall backpropagation algorithm works as the following:

- Initialize all weights w_{mk}^l in every layer of the neural network
- Feedfoward: use formula (6) to caculate the quantities z_k^l, a_k^l for each neuron in all layers. Note in the input layer $a_k = x_k$, x_k are he inputs to the $1st$ layer neurons
- Backpropagation: after calculating the feedforward path, we use the backpropagation to figure out all the δ_k^l and update the weights. In contrast to the forward path. The back propagation starts by calculating the δ_k^L of the last layer L. The δ_k^l is then iteratively calculated using equation (13).
- Calculate the gradient of the cost function w.r.t to all weights w_{kj}^{l-1}
- updates the weights using equation (11)