Derivation of the Backpropagation Algorithm

Steven Van Vaerenbergh

August 20, 2014

Abstract

The purpose of this document is to provide a full derivation of the basic backpropagation algorithm, used for training the weights of a neural network.

1 Introduction

The basic principles of feedforward artificial neural networks are assumed to be known.

1.1 Diagram

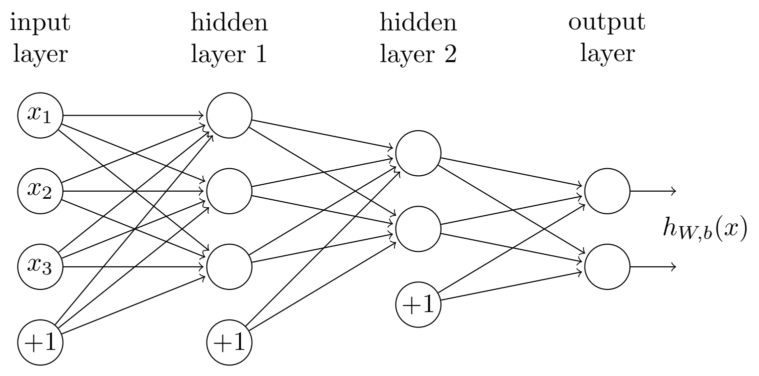

Figure 1: Diagram of a neural network with several hidden layers.

1.2 Notation

In this document we denote by $W_{ij}^{(l)}$ the weight from the j-th neuron in the l-th layer to the i-th neuron of the $l+1$-th layer. This notation is adapted from [1] and it is often used in the literature. As such, one layer of the network consists

x_i	The i-th input to the network
$f(\cdot)$	The activation function of the neurons
$a_i^{(l)}$	The output (or "activation") of unit i in layer l. Note: $x_i = a_i^{(1)}$.
$W_{ij}^{(l)}$	The weight from the j-th neuron in the l-th layer to the i-th neuron of the $l+1$-th layer
$z_i^{(l)}$	The total weighted sum of inputs to unit i in layer l
n_l	The number of layers in the neural network (including input and output layers)
s_l	The number of neurons in the l-th layer
α	The learning rate

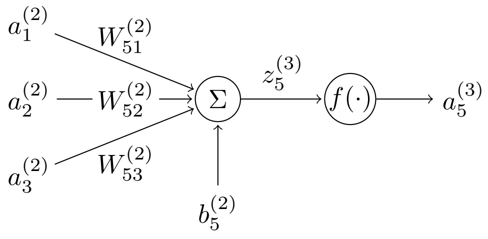

Figure 2: Detailed diagram of the used notation. As an example, this diagram focuses on the fifth neuron of the third layer; it starts at the activations of the previous layer that affect this neuron, and it ends with its own activation.

of a weighted sum of inputs $z_i^{(l)}$, an activation function, an activation term $a_i^{(l)}$, and a set of connection weights $W_{ij}^{(l)}$, in this order.

We write n_l to denote the number of layers in the neural network (including input and output layer), and s_l is the number of neurons in the l-th layer. The output of unit i in layer l is is denoted as $a_i^{(l)}$. For the input layer, this means the i-th input can be written as $x_i = a_i^{(1)}$. Finally, we denote the total weighted sum of inputs to unit i in layer l, including the bias term, as

$$z_i^{(l+1)} = \sum_{j=1}^{s_l} W_{ij}^{(l)} a_j^{(l)} + b_i^{(l)}, \tag{1}$$

so that $a_i^{(l)} = f(z_i^{(l)})$.

By adopting a matrix representation we obtain the following compact relations

$$z^{(l+1)} = W^{(l)} a^{(l)} + b^{(l)} \tag{2}$$

$$a^{(l)} = f(z^{(l)}). \tag{3}$$

The output of the last layer is denoted $h_{W,b}(x)$,

$$h_{W,b}(x) = a^{(n_l)} = f(z^{(n_l)}). \tag{4}$$

2 Derivation of the backpropagation algorithm

Suppose we have a fixed training set $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$ of m training examples. We can train our neural network using batch gradient descent. Define the cost function with respect to a single example (x, y) to be

$$J(W, b; x, y) = \frac{1}{2}\|h_{W,b}(x) - y\|^2. \tag{5}$$

Define the overall cost function to be

$$J(W, b) = \left[\frac{1}{m}\sum_{k=1}^{m} J(W, b; x^{(k)}, y^{(k)})\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{(l+1)}}\left(W_{ji}^{(l)}\right)^2. \tag{6}$$

One iteration of gradient descent updates the paramzters W, b as follows

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha\frac{\partial}{\partial W_{ij}^{(l)}}J(W, b) \tag{7}$$

$$b_i^{(l)} = b_i^{(l)} - \alpha\frac{\partial}{\partial b_i^{(l)}}J(W, b), \tag{8}$$

where α is the learning rate. The key step is computing the partial derivatives above.

Backpropagation is used to compute $\frac{\partial}{\partial W_{ij}^{(l)}}J(W, b; x, y)$ and $\frac{\partial}{\partial b_i^{(l)}}J(W, b; x, y)$, the partial derivatives of the cost function $J(W, b; x, y)$ defined with respect to a single example (x, y). Once we can compute these, we see that the derivative of the overall cost function $J(W, b)$ can be computed as

$$\frac{\partial}{\partial W_{ij}^{(l)}}J(W, b) = \left[\frac{1}{m}\sum_{k=1}^{m}\frac{\partial}{\partial W_{ij}^{(l)}}J(W, b; x^{(k)}, y^{(k)})\right] + \lambda W_{ij}^{(l)} \tag{9}$$

$$\frac{\partial}{\partial b_i^{(l)}}J(W, b) = \frac{1}{m}\sum_{k=1}^{m}\frac{\partial}{\partial b_i^{(l)}}J(W, b; x^{(k)}, y^{(i)}) \tag{10}$$

The derivation of the backpropagation algorithm is based on the chain rule. We wish to measure the gradient of the cost function with respect to each of the weights and biases,

$$\frac{\partial}{\partial W_{ij}^{(l)}}J(W, b; x, y) = \frac{\partial J}{\partial z_i^{(l+1)}}\frac{\partial z_i^{(l+1)}}{\partial W_{ij}^{(l)}} \tag{11}$$

$$\frac{\partial}{\partial b_i^{(l)}}J(W, b; x, y) = \frac{\partial J}{\partial z_i^{(l+1)}} \tag{12}$$

where we have abbreviated $J(W, b; x, y)$ to J. In Eq. (12) we have taken into account that $\partial z_i^{(l+1)}/\partial b_i^{(l)} = 1$, which is obtained from Eq. (1). In Eq. (11), we find that the second factor on the right can be obtained as

$$\frac{\partial z_i^{(l+1)}}{\partial W_{ij}^{(l)}} = a_i^{(l)} \tag{13}$$

The first factor in the right-hand side of Eq. (11) measures to what extent node i was "responsible" for any errors in the network's output, and it is denoted as the error term,

$$\delta_i^{(l)} = \frac{\partial J}{\partial z_i^{(l)}}. \tag{14}$$

The "delta term" is the central piece of the backpropagation algorithm. By applying the chain rule again, it is decomposed as

$$\delta_i^{(l)} = \frac{\partial J}{\partial a_i^{(l)}} \frac{\partial a_i^{(l)}}{\partial z_i^{(l)}} = \frac{\partial J}{\partial a_i^{(l)}} f'(z_i^{(l)}). \tag{15}$$

In order to analyze the first factor in the expansion of Eq. (15), we start by expanding the cost function for a single example, from Eq. (5),

$$J(W, b; x, y) = \frac{1}{2} \sum_{i=1}^{s_{n_l}} (a_i^{(l)} - y_i)^2 \tag{16}$$

For the last layer, $l = n_l$, we obtain

$$\frac{\partial J}{\partial a_i^{(n_l)}} = a_i^{(n_l)} - y_i. \tag{17}$$

For any of the previous layers, $l < n_l$, the derivation requires some additional calculation. First observe that the output of node i in this layer, $a_i^{(l)}$, affects the weighted inputs $z_j^{(l+1)}$ of each the nodes j in the following layer, by virtue of Eq. (1). Applying the chain rule to the first factor in the expansion of Eq. (15) now yields

$$\frac{\partial J}{\partial a_i^{(l)}} = \sum_{j=1}^{s_{(l+1)}} \frac{\partial J}{\partial z_j^{(l+1)}} \frac{\partial z_j^{(l+1)}}{\partial a_i^{(l)}} = \sum_{j=1}^{s_{(l+1)}} \delta_j^{(l+1)} W_{ji}^{(l)}. \tag{18}$$

In order to calculate $\partial J / \partial a_i^{(l)}$, we have used the error terms $\delta_j^{(l+1)}$ for the next layer. The backpropagation therefore first calculates the delta terms for the output layer, and then works its way back to calculate the delta terms of each previous layer based on the delta terms of its following layer.

References

[1] Andrew Ng, *UFLDL Tutorial*, `http://ufldl.stanford.edu/wiki/index.php/UFLDL_Tutorial`, 2011.