MINOR I

Naj Kahsyap
PHS7200

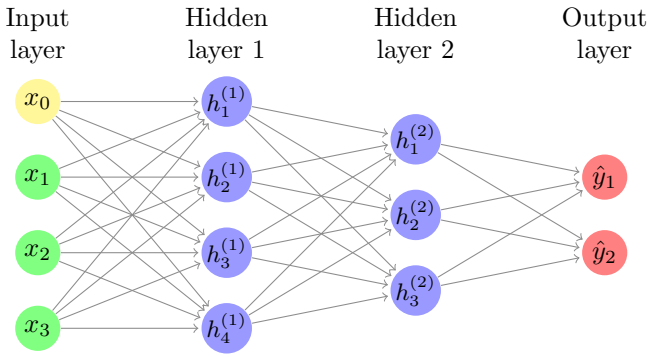
February 14, 2023

1 Backpropagation with Cross Entropy

1.1 Overview

This article introduces backpropagation for a common neural network, or a multi-class classifier. Specifically, the network has L layers, containing Rectified Linear Unit (ReLU) activations in hidden layers and Softmax in the output layer. Cross Entropy is used as the objective function to measure training loss.

1.2 Notation and Definitions



The notations used are :

- L indicates the last layer.
- l indicates a specific layer. It could be equal to L,that is, $l-1 = L-1$ but not always the case.
- The subscript k usually denotes neuron indices in the ouptut layer (layer l).
- The subscript j usually denotes neuron indices in layer $l-1$
- The subscript i usually denotes neuron indices in layer $l-2$

- $z_k^l$is the weighted sum of activations from the previous layer. That is,

$$z_k^l = b_k^l + \sum_j W_{kj}^l a_j^{l-1} \tag{1}$$

- a_k^l refers to neuron activations, $a_k l = f(z_k^l)$, where $f(.)$ is the activation function. Here, we assume that the last layer uses a softmax

$$a_k^L = \text{softmax}\left(z_k^L\right) = \frac{e^{z_k^L}}{\sum_c e^{z_c^L}} \tag{2}$$

,and the hidden layers use ReLU

$$a_k^l = \text{relu}\left(z_k^l\right) = \max\left(0, z_k^l\right) \tag{3}$$

- $t_k$is the gold probability for the k^{th} neuron in the output layer. It is one-hot encoded.

- $E$is the output error measured on one input example. We use Cross Entropy, which is defined as below

$$E = -\sum_d t_d \log a_k^L = -\sum_d t_d \left(z_d^L - \log \sum_c e^{z_c^L}\right) \tag{4}$$

1.3 Gradient Descent

Variants of gradient descent algorithms can be applied to update weight matrices W_{kj}^l.Here we use the most common update rule, which is

$$\Delta W \propto -\frac{\partial E}{\partial W}$$

. In the simpliest case, a learning rate ϵ is used to control the step size and we can calculate the derivatives using Chain Rule, so the update rule can be written as

$$\Delta W_{kj}^l = -\epsilon \frac{\partial E}{\partial a_k^l} = -\epsilon \frac{\partial E}{\partial a_k^l} \frac{\partial a_k^l}{\partial z_k^l} \frac{\partial z_k^l}{\partial W_{kj}^l} \tag{5}$$

1.4 Backpropagation

Backprpagation provide us an elegant way to calculate $\frac{\partial E}{\partial a_k^l}$ for each layer using a recursive definition of δ_k^l and δ_k^{l-1} for the adjacent layers l and $l-1$, so that the updates can be calculated, or propagated, in backward order. We will see how to derive δ_k^l and δ_k^{l-1} from deriving the updates for layer $l = L$ and $l-1$

1.5 Update for the Last Layer

Instead of calculating $\frac{\partial E}{\partial a_k^L}\frac{\partial a_k^L}{\partial z_k^L}\frac{\partial z_k^L}{\partial W_{kj}^L}$, we calculate $\frac{\partial E}{\partial z_k^L}\frac{\partial z_k^L}{\partial W_{kj}^L}$ because it simplifies the derivation a lot for Softmax and Cross Entropy.

$$\frac{\partial E}{\partial W_{kj}^L} = \frac{\partial E}{\partial z_k^L}\frac{\partial z_k^L}{\partial W_{kj}^L} \tag{6}$$

Equation (2) has the definition of the error E, and we can calculate its derivative with respect to z_k^l as follows:

$$\frac{\partial E}{\partial z_k^L} = -\sum_d t_d \left(1_{d=k} - \frac{1}{\sum_c e^{z_c^L}} e^{z_k^L}\right) \tag{7}$$

$$= -\sum_d t_d \left(1_{d=k} - a_k^L\right)$$
$$= \sum_d t_d a_k^L - \sum_d t_d 1_{d=k}$$
$$= a_k^L \sum_d t_d - t_k$$
$$= a_k^L - t_k$$

, where the $1_{d=k}$ is an identify function:

$$1_{d=k} = \begin{cases} 1 & \text{if } d = k \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Then we can define δ_k^l as

$$\delta_k^L = \frac{\partial E}{\partial z_k^L} = a_k^L - t_k \tag{9}$$

We got the first part of Equation (3) , so can move on to the second part which is $\frac{\partial z_k^L}{\partial W_{kj}^L}$. Referring to the definition of Equation (1) , this is trivial.

$$\frac{\partial z_k^L}{\partial W_{kj}^L} = a_j^{L-1} \tag{10}$$

As a result the updates for the weights in the last layer are:

$$\frac{\partial E}{\partial W_{kj}^L} = \frac{\partial E}{\partial z_k^L}\frac{\partial z_k^L}{\partial W_{kj}^L} = \delta_k^L a_j^{L-1} \tag{11}$$

We also need to do a similar derivation for the bias.

$$\frac{\partial E}{\partial b_k^L} = \frac{\partial E}{\partial z_k^L}\frac{\partial z_k^L}{\partial b_k^L} = \delta_k^L(1) = \delta_k^L \tag{12}$$

1.6 Update for the Second Last Layer

Similarly, Equation (8) - (10) derives each component of Equation (7)

$$\frac{\partial E}{\partial W_{ji}^{l-1}} = \frac{\partial E}{\partial a_j^{l-1}} \frac{\partial a_j^{l-1}}{\partial z_j^{l-1}} \frac{\partial z_j^{l-1}}{\partial W_{ji}^{l-1}} \tag{13}$$

$$\begin{aligned} \frac{\partial E}{\partial a_j^{l-1}} &= \sum_k \frac{\partial E}{\partial z_k^l} \frac{\partial z_k^l}{\partial a_j^{l-1}} \\ &= \sum_k \delta_k^l W_{kj}^l \end{aligned} \tag{14}$$

$$\begin{aligned} \frac{\partial a_j^{l-1}}{\partial z_j^{l-1}} &= f'\left(z_j^{l-1}\right) \\ \frac{\partial z_j^{l-1}}{\partial W_{ji}^{l-1}} &= a_i^{l-2} \end{aligned} \tag{15}$$

Combining all together, we get

$$\frac{\partial E}{\partial W_{ji}^{l-1}} = a_i^{l-2} f'\left(z_j^{l-1}\right) \sum_k \delta_k^l W_{kj}^l \tag{16}$$

We can define $\delta_j^{l-1} = \frac{\partial E}{\partial a_j^{l-1}} \frac{\partial a_j^{l-1}}{\partial z_j^{l-1}} = f'\left(z_j^{l-1}\right) \sum_k \delta_k^l W_{kj}^l$ and re-write Equation (11)

$$\frac{\partial E}{\partial W_{ji}^{l-1}} = \delta_j^{l-1} a_i^{l-2} \tag{17}$$

For the bias

$$\frac{\partial E}{\partial b_j^{l-1}} = \frac{\partial E}{\partial a_j^{l-1}} \frac{\partial a_j^{l-1}}{\partial z_j^{l-1}} \frac{\partial z_j^{l-1}}{\partial b_j^{l-1}} = \delta_j^{l-1}(1) = \delta_j^{l-1} \tag{18}$$

1.7 Backpropagation Summary

To this point, we got all the derivatives we need to update our specific neural network (the one with ReLU activation, softmax output, and cross-entropy error), and they can be applied to arbitrary number of layers. In fact, Backpropagation can be generalized and used with any activations and objectives. It is summarized in the following four equations:

$$\begin{aligned} \delta_k^l &= \frac{\partial E}{\partial z_k^l} \\ \delta_j^{l-1} &= f'\left(z_j^{l-1}\right) \sum_k \delta_k^l W_{kj}^l \end{aligned}$$

$$\frac{\partial E}{\partial W_{kj}^l} = \delta_k^l a_j^{l-1}$$
$$\frac{\partial E}{\partial b_k^l} = \delta_k^l$$

1.8 Numerically Stable Softmax

This is a practical implementation issue. Calculating the exponentials in Softmax is numerically unstable, since the values could be extremely large. We can do a small trick by introducing a constant C to mitigate such problem.

$$\begin{aligned} \text{softmax}\left(x_i\right) &= \frac{e^{x_i}}{\sum_j e^{x_j}} \\ &= \frac{Ce^{x_i}}{C\sum_j e^{x_j}} \\ &= \frac{e^{x_i + \log C}}{\sum_j e^{x_j + \log C}} \end{aligned} \tag{19}$$

A common choice for the constant is $\log C = -\max_j x_j$