Backpropagation

Deep Learning notes

Matteo Lugli

November 24, 2023

1 Backpropagation

Backpropagation is used to compute the change needed in the weights for each layer to decrease the loss function. It's necessary to use a iterative algorithm because we cannot compute the derivatives analitically as we do with a simple loss function like Binary Cross Entropy. However, we can compute the change needed in each layer l by taking some information computed in the layer $l+1$, that's why we call this algorithm *backpropagation*. This way, if we start from the last layer and go backwards, we can update the weights in each single layer by going backwards.

2 Structure

Let's draw a sample DNN structure using ReLU as activation function:

$$\xrightarrow{x} \underbrace{\text{ReLU}(0, W_1 x)}_{Layer1} \xrightarrow{h1} \underbrace{\text{ReLU}(0, W_2 h_1)}_{Layer2} \xrightarrow{h2} \underbrace{\text{ReLU}(0, W_3 h_2)}_{Layer3} \xrightarrow{h3} \text{Loss} \tag{1}$$

In each layer, the algorithm computes:

1. The derivative of the loss function with respect to the **weights** of the current layer, to make the update;
2. The derivative of the loss function with respect to the **input** of the layer, to propagate it back;

2.1 Loss

The first step is to compute the partial derivative of the Loss with respect to his input, so $\frac{\partial L}{\partial h3}$. This is simple: the Loss is just a analytical function that can be derived.

2.2 Layer 3

First step is to compute the derivative of the loss with respect to the parameters. If we look at the equation, the first term is taken from the previous layer (2.1), the only term that needs to be computed is the latter. Remember that we make an intense use of the **chain rule**, which states that given $z = f(g(x))$ and $y = g(x)$, then $\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx}$

$$\begin{aligned}
\frac{\partial L}{\partial W_3} &= \frac{\partial L}{\partial h_3}\frac{\partial h_3}{\partial W_3} \\
&= \frac{\partial L}{\partial h_3}\frac{\partial \text{ReLU}(W_3 h_2)}{\partial W_3} \\
&= \frac{\partial L}{\partial h_3}\underbrace{\frac{\partial \text{ReLU}(W_3 h_2)}{\partial W_3 h_2}}_{der.of ReLU}\underbrace{\frac{\partial W_3 h_2}{\partial W_3}}_{h_2}
\end{aligned} \tag{2}$$

We successfully decomposed $\frac{\partial h_3}{\partial W_3}$ in easier terms that we are able to compute. With this value now we can perform a gradient descent step to update the weights of

the current layer. Let's now compute the value that needs to be backpropagated to the next layer ("next" in term of computation, "previous" in terms of topological structure).

$$\xleftarrow{\text{Backprop}} \frac{\partial L}{\partial h_2} = \underbrace{\frac{\partial L}{\partial h_3}}_{\text{prev. layer}} \overbrace{\frac{\partial h_3}{\partial W_3 h_2}}^{\text{der. of ReLU}} \underbrace{\frac{\partial W_3 h_2}{\partial h_2}}_{W_3} \tag{3}$$

2.3 **Layer 2**

Let's do another layer. Let's write down the math considering that we alredy have $\frac{\partial L}{\partial h_2}$ from the previous (in terms of computation, "next" in terms of topological structure) layer.

$$\begin{aligned} \frac{\partial L}{\partial W_2} &= \frac{\partial L}{\partial h_2}\frac{\partial h_2}{\partial W_2} \\ &= \frac{\partial L}{\partial h_2}\frac{\partial h_2}{\partial W_2 h_1}\frac{\partial W_2 h_1}{\partial W_2} \end{aligned} \tag{4}$$

$$\xleftarrow{\text{Backprop}} \frac{\partial L}{\partial h_1} = \frac{\partial L}{\partial h_2}\frac{\partial h2}{\partial W_2 h_1}\frac{\partial W_2 h_1}{\partial h_1} \tag{5}$$

and so on.

3 Class

$$a = ((1,1),(1,1)) \rightarrow ((4.5,4.5)(4.5,4.5)) \tag{6}$$

$$b = a + 2 \rightarrow 1 \tag{7}$$

$$c = b^2 \rightarrow 2b \tag{8}$$

$$d = 3c \rightarrow 3 \tag{9}$$

$$\text{d.mean()} \rightarrow ((\frac{1}{N},\frac{1}{N})(\frac{1}{N},\frac{1}{N})) \tag{10}$$

$$\tag{11}$$