

Forward-Feeding Neural Networks (FFNNs): Forward & Backward Passes

STUFF TO KNOW BY HEART - EVEN WHEN DRUNK!

1. FFNNs are **supervised** learning models
2. An FFNN consists of **layers** of **transformation functions** and **weights**
3. FFNNs' **forward pass** models **hypothesized output**
4. FFNNs' **backward pass** computes **partial derivatives** of cost function with respect to weight layers, for use in mathematical optimization

Model Structure / Hypothesis: Forward Pass

In a generalized sense, an FFNN models Hypothesized Output $\mathbf{H} = h(\mathbf{X}, \mathbf{W}^{[1]}, \mathbf{W}^{[2]}, \dots, \mathbf{W}^{[L]})$ through 1 input layer and L additional layers of transformation functions and parameters (called “weights”) in the following manner:

network layer 1: $\mathbf{A}^{[1]} = \text{Input } \mathbf{X}$

network layer 2: $\mathbf{A}^{[2]} = f^{[1]}(\mathbf{A}^{[1]}, \mathbf{W}^{[1]})$

network layer 3: $\mathbf{A}^{[3]} = f^{[2]}(\mathbf{A}^{[2]}, \mathbf{W}^{[2]})$

...

network layer $(L + 1)$: $\mathbf{H} = \mathbf{A}^{[L+1]} = f^{[L]}(\mathbf{A}^{[L]}, \mathbf{W}^{[L]})$

where:

- \mathbf{A} ’s are called the layers’ “**activations**” and inter-layer parameters \mathbf{W} ’s are called “**weights**”. The way the FFNN computes \mathbf{H} from input \mathbf{X} through layers of transformation functions and weights is called the “**forward pass**”.
- Each “**forward function**” f is a structurally pre-defined transformation function $\text{Output} = f(\text{Input}, \text{Parameter})$ such that, given partial derivative $\frac{\partial v}{\partial \text{Output}}$ of a scalar variable v with respect to **Output**, the following partial derivatives with respect to **Input** and **Parameter** can be computed by certain “**backward functions**” b_{Input} and $b_{\text{Parameter}}$:

$$\frac{\partial v}{\partial \text{Input}} = b_{\text{Input}}\left(\frac{\partial v}{\partial \text{Output}}, \text{local state}\right)$$

$$\frac{\partial v}{\partial \text{Parameter}} = b_{\text{Parameter}}\left(\frac{\partial v}{\partial \text{Output}}, \text{local state}\right)$$

where the term “local state” refers to current values of function f ’s **Input**, **Parameter** and **Output**

(for each neural network layer l , we henceforth denote

its corresponding “backward functions” $b_A^{[l]}$ and $b_W^{[l]}$)

The purpose of knowing such partial derivatives will become clear later when we discuss the “**backward pass**” or “**backpropagation**” procedure.

Backward Pass / Backpropagation procedure to derive $\frac{\partial c}{\partial \mathbf{W}^{[l]}}$ for each layer l , to be used in optimization

With the structure of the transformation functions f 's fixed, in the learning/training process, our job is to adjust/update the values of weight layers $\mathbf{W}^{[1]}$, $\mathbf{W}^{[2]}$, ..., $\mathbf{W}^{[L]}$ so as to make the cost function $c(\mathbf{H}, \mathbf{Y})$ decrease. This invariably requires us to know or be able to estimate the partial derivative $\frac{\partial c}{\partial \mathbf{W}^{[l]}}$ for each layer l . We can compute such partial derivatives through the following “backpropagating” procedure:

$$\begin{aligned}
 & \frac{\partial c}{\partial \mathbf{A}^{[L+1]}} = \frac{\partial c}{\partial \mathbf{H}} = d(\mathbf{H}, \mathbf{Y}) & \Rightarrow & \frac{\partial c}{\partial \mathbf{W}^{[L]}} = b_W^{[L]} \left(\frac{\partial c}{\partial \mathbf{A}^{[L+1]}}, \text{local state} \right) \\
 & \Downarrow \\
 & \frac{\partial c}{\partial \mathbf{A}^{[L]}} = b_A^{[L]} \left(\frac{\partial c}{\partial \mathbf{A}^{[L+1]}}, \text{local state} \right) & \Rightarrow & \frac{\partial c}{\partial \mathbf{W}^{[L-1]}} = b_W^{[L-1]} \left(\frac{\partial c}{\partial \mathbf{A}^{[L]}}, \text{local state} \right) \\
 & \Downarrow \\
 & \frac{\partial c}{\partial \mathbf{A}^{[L-1]}} = b_A^{[L-1]} \left(\frac{\partial c}{\partial \mathbf{A}^{[L]}}, \text{local state} \right) & \Rightarrow & \frac{\partial c}{\partial \mathbf{W}^{[L-2]}} = b_W^{[L-2]} \left(\frac{\partial c}{\partial \mathbf{A}^{[L-1]}}, \text{local state} \right) \\
 & \Downarrow \\
 & \dots \\
 & \Downarrow \\
 & \frac{\partial c}{\partial \mathbf{A}^{[2]}} = b_A^{[2]} \left(\frac{\partial c}{\partial \mathbf{A}^{[3]}}, \text{local state} \right) & \Rightarrow & \frac{\partial c}{\partial \mathbf{W}^{[1]}} = b_W^{[1]} \left(\frac{\partial c}{\partial \mathbf{A}^{[2]}}, \text{local state} \right)
 \end{aligned}$$

Illustration of Forward and Backward Passes

The following diagram illustrates an FFNN's forward and backward passes:

