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

STUFF TO KNOW BY HEART - EVEN WHEN DRUNK!

*1. An FFNN consists of **layers** of **transformation functions** and **weights***

*2. FFNNs' **forward pass** models **hypothesized output***

*3. FFNNs' **backward pass** computes **partial derivatives** of cost function with respect to weight layers, for use in mathematical optimization*

MODEL STRUCTURE / HYPOTHETICAL PREDICTION: FORWARD PASS

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

$$\begin{aligned}
&\text{network layer 1: } \mathbf{A}^{[1]} = \text{Input } \mathbf{X} \\
&\text{network layer 2: } \mathbf{A}^{[2]} = f^{[1]}(\mathbf{A}^{[1]}, \mathbf{W}^{[1]}) \\
&\text{network layer 3: } \mathbf{A}^{[3]} = f^{[2]}(\mathbf{A}^{[2]}, \mathbf{W}^{[2]}) \\
&\ldots \\
&\text{network layer } (L+1)\text{: } \mathbf{H} = \mathbf{A}^{[L+1]} = f^{[L]}(\mathbf{A}^{[L]}, \mathbf{W}^{[L]})
\end{aligned}$$

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 $\mathbf{Output} = f(\mathbf{Input}, \mathbf{Parameter})$ such that, given partial derivative $\frac{\partial v}{\partial \mathbf{Output}}$ 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 $b_{Parameter}$:

$$\begin{aligned}
&\frac{\partial v}{\partial \mathbf{Input}} = b_{Input}(\frac{\partial v}{\partial \mathbf{Output}}, \text{local state}) \\
&\frac{\partial v}{\partial \mathbf{Parameter}} = b_{Parameter}(\frac{\partial v}{\partial \mathbf{Output}}, \text{local state}) \\
&\text{where the term "local state" refers to current values} \\
&\text{of function } f\text{'s } \mathbf{Input}, \mathbf{Parameter} \text{ and } \mathbf{Output} \\
&\\
&\text{(for each neural network layer } l\text{, we henceforth denote} \\
&\text{its corresponding "backward functions" } b_A^{[l]} \text{ and } b_W^{[l]})
\end{aligned}$$

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 (for use 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:

$$
\begin{array}{lcl}
\dfrac{\partial c}{\partial \mathbf{A}^{[L+1]}} = \dfrac{\partial c}{\partial \mathbf{H}} = d(\mathbf{H}, \mathbf{Y}) & \Rightarrow & \dfrac{\partial c}{\partial \mathbf{W}^{[L]}} = b_W^{[L]}(\dfrac{\partial c}{\partial \mathbf{A}^{[L+1]}}, \text{local state}) \\
\Downarrow & & \\
\dfrac{\partial c}{\partial \mathbf{A}^{[L]}} = b_A^{[L]}(\dfrac{\partial c}{\partial \mathbf{A}^{[L+1]}}, \text{local state}) & \Rightarrow & \dfrac{\partial c}{\partial \mathbf{W}^{[L-1]}} = b_W^{[L-1]}(\dfrac{\partial c}{\partial \mathbf{A}^{[L]}}, \text{local state}) \\
\Downarrow & & \\
\dfrac{\partial c}{\partial \mathbf{A}^{[L-1]}} = b_A^{[L-1]}(\dfrac{\partial c}{\partial \mathbf{A}^{[L]}}, \text{local state}) & \Rightarrow & \dfrac{\partial c}{\partial \mathbf{W}^{[L-2]}} = b_W^{[L-2]}(\dfrac{\partial c}{\partial \mathbf{A}^{[L-1]}}, \text{local state}) \\
\Downarrow & & \\
\cdots & & \\
\Downarrow & & \\
\dfrac{\partial c}{\partial \mathbf{A}^{[2]}} = b_A^{[2]}(\dfrac{\partial c}{\partial \mathbf{A}^{[3]}}, \text{local state}) & \Rightarrow & \dfrac{\partial c}{\partial \mathbf{W}^{[1]}} = b_W^{[1]}(\dfrac{\partial c}{\partial \mathbf{A}^{[2]}}, \text{local state})
\end{array}
$$

ILLUSTRATION OF FORWARD & BACKWARD PASSES

$$
\mathbf{X} = \mathbf{A}^{[1]} \xrightarrow[(W^{[1]})]{f^{[1]}} \mathbf{A}^{[2]} \xrightarrow[(W^{[2]})]{f^{[2]}} \mathbf{A}^{[3]} \xrightarrow[(W^{[3]})]{f^{[3]}} \cdots \xrightarrow[(W^{[L-2]})]{f^{[L-2]}} \mathbf{A}^{[L-1]} \xrightarrow[(W^{[L-1]})]{f^{[L-1]}} \mathbf{A}^{[L]} \xrightarrow[(W^{[L]})]{f^{[L]}} \mathbf{A}^{[L+1]} = \mathbf{H} \rightarrow \text{cost: } c(\mathbf{H}, \mathbf{Y})
$$

(FORWARD PASS)

layer 1	layer 2	layer $(L-2)$	layer $(L-1)$	layer L

(BACKWARD PASS)

$$
\begin{array}{ccccccccccc}
\dfrac{\partial c}{\partial \mathbf{W}^{[1]}} & & \dfrac{\partial c}{\partial \mathbf{W}^{[2]}} & & \dfrac{\partial c}{\partial \mathbf{W}^{[3]}} & & \dfrac{\partial c}{\partial \mathbf{W}^{[L-2]}} & & \dfrac{\partial c}{\partial \mathbf{W}^{[L-1]}} & & \dfrac{\partial c}{\partial \mathbf{W}^{[L]}} \\
b_W^{[1]} \nwarrow & & b_W^{[2]} \nwarrow & & b_W^{[3]} \nwarrow & & b_W^{[L-2]} \nwarrow & & b_W^{[L-1]} \nwarrow & & b_W^{[L]} \nwarrow
\end{array}
$$

$$
\frac{\partial c}{\partial \mathbf{A}^{[2]}} \xleftarrow[b_A^{[2]}]{} \frac{\partial c}{\partial \mathbf{A}^{[3]}} \xleftarrow[b_A^{[3]}]{} \cdots \xleftarrow[b_A^{[L-2]}]{} \frac{\partial c}{\partial \mathbf{A}^{[L-1]}} \xleftarrow[b_A^{[L-1]}]{} \frac{\partial c}{\partial \mathbf{A}^{[L]}} \xleftarrow[b_A^{[L]}]{} \frac{\partial c}{\partial \mathbf{A}^{[L+1]}} = \frac{\partial c}{\partial \mathbf{H}} = d(\mathbf{H}, \mathbf{Y})
$$