Back-Propagation of a Forward Feed Neural Net via Multivariate Vector Calculus

Purdue University

Computer Science

Alex Aralis

June 14, 2017

1 Introduction

A Forward Feed Neural Net (FFNN) is characterized by L matrices $\mathbf{W}_l$ where L is the number of layers in the neural net and $l \in \{1, 2, ..., L\}$, an activations function $\varphi \in \mathbb{R} \to \mathbb{R}$, and an error function $E \in \mathbb{R}^M \times \mathbb{R}^M \to \mathbb{R}$ where M is the size of the output of the FFNN.

$$\boldsymbol{W}_l = \begin{bmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{m1} & \cdots & w_{mn} \end{bmatrix} \tag{1}$$

Where $w_{ij} \in \mathbb{R}$ is the weighting between the jth output of layer $l-1$ and the output of the lth layer. These weightings will be referred to using the notation $[\boldsymbol{W}_l]_{ij}$ to disambiguate the weightings of the layers.

The output of a layer l is related to the previous layer $l-1$ by linear combination and an activation function φ.

$$\boldsymbol{n}_l = \boldsymbol{W}_l \boldsymbol{\omega}_{l-1} \tag{2}$$

Where $\boldsymbol{\omega}_{l-1}$ is a vector of the outputs from the previous layer and $\boldsymbol{n}_l$ is a vector of the precursor linear combinations for $\boldsymbol{\omega}_l$. $\boldsymbol{n}_l$ is merely an intermediate result useful in later computations. Finally,

$$\boldsymbol{\omega}_l = \varphi(\boldsymbol{n}_l) \tag{3}$$

Where $\varphi(\boldsymbol{n}_l)$ is simply the application of the φ on each of the entries in $\boldsymbol{n}_l$. For all FFNN's $\boldsymbol{\omega}_0$ is the input and $\boldsymbol{\omega}_L$ is the output. Using (1-3) is enough to evaluate inputs to the FFNN.

The error function E, an output $\boldsymbol{\omega}_L$, and the desired output for an input $\boldsymbol{t}_{\boldsymbol{\omega}_0}$ determine the error of the FFNN for an input $\boldsymbol{\omega}_0$. Differentiating E with respect to the weights of each layer $[\mathbf{W}_l]_{ij}$ provides the information needed to "teach" the FFNN.

2 Problem Statement

It would be nice if,

$$\frac{\partial E}{\partial \boldsymbol{W}_l} \tag{4}$$

could be evaluated directly. Let's try with the outer most layer's weights.

$$\begin{aligned} \frac{\partial E}{\partial \boldsymbol{W}_L} &= \frac{\partial}{\partial \boldsymbol{W}_L}\left[E(\varphi(\boldsymbol{W}_L \boldsymbol{\omega}_{L-1}))\right] & (5)\\ &= \frac{\partial}{\partial \boldsymbol{W}_L}\left[E(\varphi(\boldsymbol{n}_L))\right] & (6)\\ &= \frac{\partial E}{\partial \boldsymbol{\omega}_L}\frac{\partial \boldsymbol{\omega}_L}{\partial \boldsymbol{n}_L}\frac{\partial \boldsymbol{n}_L}{\partial \boldsymbol{W}_L} & (7) \end{aligned}$$

This looks promising.

$$\frac{\partial \boldsymbol{n}_L}{\partial \boldsymbol{W}_L} = \frac{\partial}{\partial \boldsymbol{W}_L}\left[\boldsymbol{W}_L \boldsymbol{\omega}_{L-1}\right] \tag{8}$$

$$\stackrel{?}{=} \boldsymbol{\omega}_{L-1} \tag{9}$$

Hmm. However differentiation of a vector with respect to a matrix is not well defined. The other partial differentials are more easily evaluated.

$$\frac{\partial \boldsymbol{\omega}_L}{\partial \boldsymbol{n}_L} = \left[\frac{\partial \left[\boldsymbol{\omega}_L\right]_i}{\partial \left[\boldsymbol{n}_L\right]_j}\right] \tag{10}$$

$$= \begin{bmatrix} \frac{\partial[\boldsymbol{\omega}_L]_1}{\partial[\boldsymbol{n}_L]_1} & & \\ & \ddots & \\ & & \frac{\partial[\boldsymbol{\omega}_L]_N}{\partial[\boldsymbol{n}_L]_N} \end{bmatrix} \tag{11}$$

$$= \begin{bmatrix} \varphi'([\boldsymbol{n}_L]_1) & & \\ & \ddots & \\ & & \varphi'([\boldsymbol{n}_L]_N) \end{bmatrix} \tag{12}$$

$\forall i, j$ s.t. $i \neq j$, $\frac{\partial[\boldsymbol{\omega}_l]_i}{\partial[\boldsymbol{n}_l]_j} = 0$ because $[\boldsymbol{\omega}_l]_i = \varphi([\boldsymbol{n}_l]_i)$ does not depend on $[\boldsymbol{n}_l]_j$ making $\frac{\partial \boldsymbol{\omega}_l}{\partial \boldsymbol{n}_l}$ diagonal, also φ' must exist.

$$\frac{\partial E}{\partial \boldsymbol{\omega}_L} = \left[\frac{\partial E}{\partial[\boldsymbol{\omega}_L]_1} \frac{\partial E}{\partial[\boldsymbol{\omega}_L]_2} \cdots \frac{\partial E}{\partial[\boldsymbol{\omega}_L]_N}\right] \tag{13}$$

$$= \left[E'([\boldsymbol{\omega}_L]_1) E'([\boldsymbol{\omega}_L]_2) \cdots E'([\boldsymbol{\omega}_L]_N)\right] \tag{14}$$

Here we see E must be differentiable as well. Putting it all back together we get,

$$\begin{bmatrix} E'([\boldsymbol{\omega}_L]_1) E'([\boldsymbol{\omega}_L]_2) \cdots E'([\boldsymbol{\omega}_L]_N) \end{bmatrix} \begin{bmatrix} \varphi'([\boldsymbol{n}_L]_1) & & \\ & \ddots & \\ & & \varphi'([\boldsymbol{n}_L]_N) \end{bmatrix} \boldsymbol{\omega}_{L-1} \tag{15}$$

However this evaluates to a scalar, which cannot be correct, because we are looking for $\frac{\partial E}{\partial [\boldsymbol{W}_l]_{ij}} \forall i, j$. Perhaps there is some higher dimensional treatment of differentiation by a matrix that could yield better results, however instead I will attempt to work backwards from the end goal.

$$\boldsymbol{\Delta}_l = \left[\frac{\partial E}{\partial [\boldsymbol{W}_l]_{ij}} \right] \tag{16}$$

and to be complete the proposed extension of vector differentiation to matrix differentiation is,

$$\frac{\partial E}{\partial \boldsymbol{W}_l} = \left[\frac{\partial E}{\partial [\boldsymbol{W}_l]_{ij}} \right]^T = \boldsymbol{\Delta}_l^T \tag{17}$$

Now evaluation can proceed from the perspective of $\frac{\partial E}{\partial [\boldsymbol{W}_l]_{ij}}$ and factoring $\boldsymbol{\Delta}_l$ can be done at a later stage.

$$\frac{\partial E}{\partial [\boldsymbol{W}_l]_{ij}} = \frac{\partial E}{\partial \left[\boldsymbol{n}_l\right]_i} \frac{\partial \left[\boldsymbol{n}_l\right]_i}{\partial \left[\boldsymbol{W}_l\right]_{ij}} \tag{18}$$

$$= \frac{\partial E}{\partial \left[\boldsymbol{n}_l\right]_i} [\boldsymbol{\omega}_{l-1}]_j \tag{19}$$

Now things become slightly more interesting, because $\boldsymbol{n}_l$ depends on all the previous layers, differentiation is not simple.

$$\frac{\partial E}{\partial [\boldsymbol{n}_l]_i} = \frac{\partial E}{\partial \boldsymbol{n}_{l+1}} \frac{\partial \boldsymbol{n}_{l+1}}{\partial [\boldsymbol{\omega}_l]_i} \frac{\partial [\boldsymbol{\omega}_l]_i}{\partial [\boldsymbol{n}_l]_i} \tag{20}$$

$$= \frac{\partial E}{\partial \boldsymbol{n}_{l+1}} \frac{\partial \boldsymbol{n}_{l+1}}{\partial [\boldsymbol{\omega}_l]_i} \varphi'([\boldsymbol{n}_l]_i) \tag{21}$$

$$= \frac{\partial E}{\partial \boldsymbol{n}_{l+1}} [\boldsymbol{W}_{l+1}]_{*i} \varphi'([\boldsymbol{n}_l]_i) \tag{22}$$

$$= \begin{bmatrix} \frac{\partial E}{\partial [\boldsymbol{n}_{l+1}]_1} \cdots \frac{\partial E}{\partial [\boldsymbol{n}_{l+1}]_N} \end{bmatrix} [\boldsymbol{W}_{l+1}]_{*i} \varphi'([\boldsymbol{n}_l]_i) \tag{23}$$

$$\tag{24}$$

So $\frac{\partial E}{\partial [\boldsymbol{n}_l]_i}$ can be obtianed if $\frac{\partial E}{\partial \boldsymbol{n}_{l+1}}$ is known. What about the base case, the outermost layer L?

$$\frac{\partial E}{\partial [\boldsymbol{n}_L]_i} = \frac{\partial E}{\partial [\boldsymbol{\omega}_L]_i} \frac{\partial [\boldsymbol{\omega}_L]_i}{\partial [\boldsymbol{n}_L]_i} \tag{25}$$

$$= \frac{\partial E}{\partial [\boldsymbol{\omega}_L]_i} \varphi'([\boldsymbol{n}_L]_i) \tag{26}$$

$$= E'([\boldsymbol{\omega}_L]_i) \varphi'([\boldsymbol{n}_L]_i) \tag{27}$$

In the outermost case, E takes $\boldsymbol{\omega}_L$ directly, instead of in the context of a larger composition. This means that E' with respect to one of the elements of $\boldsymbol{\omega}_L$ is a very natural process. In order to reach the lth layer inductively using these formulas requires solving for each entry of $\frac{\partial E}{\partial \boldsymbol{n}_L}$ individually. It would be much less clumsy to solve simultaneously.

$$\frac{\partial E}{\partial \boldsymbol{n}_l} = \frac{\partial E}{\partial \boldsymbol{n}_{l+1}} \frac{\partial \boldsymbol{n}_{l+1}}{\partial \boldsymbol{\omega}_l} \frac{\partial \boldsymbol{\omega}_l}{\partial \boldsymbol{n}_l} \tag{28}$$

Our old friend $\frac{\partial \boldsymbol{\omega}_l}{\partial \boldsymbol{n}_l}$ has turned back up. Luckily, we can reuse our work from (12). Might as well give it a name while we are at it.

$$\boldsymbol{\Psi}_l = \frac{\partial \boldsymbol{\omega}_l}{\partial \boldsymbol{n}_l} \tag{29}$$

$$= \begin{bmatrix} \varphi'([\boldsymbol{n}_l]_1) & & \\ & \ddots & \\ & & \varphi'([\boldsymbol{n}_l]_N) \end{bmatrix} \tag{30}$$

Next up is,

$$\frac{\partial \boldsymbol{n}_{l+1}}{\partial \boldsymbol{\omega}_l} = \frac{\partial}{\partial \boldsymbol{\omega}_l} \left[\boldsymbol{W}_{l+1} \boldsymbol{\omega}_l \right] \tag{31}$$

$$= \boldsymbol{W}_{l+1} \tag{32}$$

Shockingly convenient.

$$\frac{\partial E}{\partial \boldsymbol{n}_l} = \frac{\partial E}{\partial \boldsymbol{n}_{l+1}} \boldsymbol{W}_{l+1} \boldsymbol{\Psi}_l \tag{33}$$

Now the entire recurrence relation is stated in one equations. Finally a few finishing touches for clarity.

$$\boldsymbol{\delta}_l = \frac{\partial E}{\partial \boldsymbol{n}_l} \tag{34}$$

$$\boldsymbol{\delta}_L = \frac{\partial E}{\partial \boldsymbol{\omega}_L} \boldsymbol{\Psi}_L \tag{35}$$

$$\boldsymbol{\delta}_{l-1} = \boldsymbol{\delta}_l \boldsymbol{W}_l \boldsymbol{\Psi}_{l-1} \tag{36}$$

(35) can be used to propagate inward starting from the outermost layer L of the FFNN. And to solve to original problem,

$$\frac{\partial E}{\partial \left[\boldsymbol{n}_l\right]_i} = [\boldsymbol{\delta}_l]_i \tag{37}$$

and,

$$\frac{\partial E}{\partial[\boldsymbol{W}_l]_{ij}} = [\boldsymbol{\delta}_l]_i[\boldsymbol{\omega}_{l-1}]_j \tag{38}$$

therefore,

$$\begin{aligned}
\boldsymbol{\Delta}_l &= \left[\frac{\partial E}{\partial[\boldsymbol{W}_l]_{ij}}\right] && (39)\\
&= [[\boldsymbol{\delta}_l]_i[\boldsymbol{\omega}_{l-1}]_j] && (40)\\
&= \begin{bmatrix} [\boldsymbol{\delta}_l]_1[\boldsymbol{\omega}_{l-1}]_1 & \cdots & [\boldsymbol{\delta}_l]_1[\boldsymbol{\omega}_{l-1}]_N \\ \vdots & \ddots & \vdots \\ [\boldsymbol{\delta}_l]_M[\boldsymbol{\omega}_{l-1}]_1 & \cdots & [\boldsymbol{\delta}_l]_M[\boldsymbol{\omega}_{l-1}]_N \end{bmatrix} && (41)\\
&= \boldsymbol{\delta}_l^T \boldsymbol{\omega}_{l-1}^T && (42)\\
&= (\boldsymbol{\omega}_{l-1}\boldsymbol{\delta}_l)^T && (43)\\
&= \left(\frac{\partial \boldsymbol{n}_l}{\partial \boldsymbol{W}_l}\frac{\partial E}{\partial \boldsymbol{n}_l}\right)^T && (44)\\
&= \frac{\partial E}{\partial \boldsymbol{W}_l}^T && (45)
\end{aligned}$$

Which is what we wanted in (17). Strangely the chain rule ordering is exchanged.

Lastly, the entire update formula for $\boldsymbol{W}_l$ is,

$$\boldsymbol{W}_l' = \boldsymbol{W}_l - \eta\boldsymbol{\Delta}_l \tag{46}$$

where η is the learning rate.

3 Reframing

So we have the result, however the path to the solution was somehow unsatisfying. Consider this restatement,

$$\boldsymbol{\Delta}_l = \frac{\partial E}{\partial \boldsymbol{W}_l}^T = \frac{\partial E}{\partial \boldsymbol{W}_l^T} \tag{47}$$

That's interesting, but $\boldsymbol{W}_l^T$ does not appear anywhere in the FFNN model. So,

$$\boldsymbol{n}_l^T = \boldsymbol{\omega}_{l-1}^T \boldsymbol{W}_l^T \tag{48}$$

Notice $\boldsymbol{n}_l^T$ is produced with a matrix multiplication where $\boldsymbol{W}_l^T$ is on the right. This suggests a different chain rule decomposition.

$$\begin{align}
\frac{\partial E}{\partial \boldsymbol{W}_l^T} &= \frac{\partial E}{\partial \boldsymbol{n}_l^T} \frac{\partial \boldsymbol{n}_l^T}{\partial \boldsymbol{W}_l^T} \tag{49}\\
&= \frac{\partial E}{\partial \boldsymbol{n}_l^T} \boldsymbol{\omega}_{l-1}^T \tag{50}\\
&= \frac{\partial E}{\partial \boldsymbol{n}_l}^T \boldsymbol{\omega}_{l-1}^T \tag{51}\\
&= \boldsymbol{\delta}_l^T \boldsymbol{\omega}_{l-1}^T \tag{52}\\
&= \boldsymbol{\Delta}_l \tag{53}
\end{align}$$

The same result, but without the chain rule inversion and now the result of differentiation is directly equal to $\boldsymbol{\Delta}_l$, the desired result.

4 The Algorithm

Here is an iterative approach that requires having precomputed $\boldsymbol{\omega}_*$. Of course $\boldsymbol{\omega}_*$ can be recomputed directly each iteration by evaluating the FFNN out to the lth step. This introduces $O(l^2)$ unnecessary matrix multiplications to an otherwise linear algorithm so should be avoided. Also, $\boldsymbol{\omega}_0$ (the input) and $\boldsymbol{t}$ (the correct output) are needed to evaluate the FFNN and E respectively.

Result: Updated $\boldsymbol{W}_*$, $\boldsymbol{W}'_*$.

$\boldsymbol{\delta} \leftarrow \frac{\partial E}{\partial \boldsymbol{\omega}_L} \boldsymbol{\Psi}_L$

while *for* l *from* L *to* 1 *inclusive* **do**

  $\boldsymbol{W}'_l \leftarrow \boldsymbol{W}_l - \eta \boldsymbol{\delta}^T \boldsymbol{\omega}_{l-1}^T$

  $\boldsymbol{\delta} \leftarrow \boldsymbol{\delta} \boldsymbol{W}_l \boldsymbol{\Psi}_{l-1}$

end