

# Basic MLP with manually-derived Backprop

Teo Asinari

October 21, 2023

## 1 Introduction

**Goal:** To design, train and use a simple 3-layer MLP for binary classification of size-2 vectors.

**Design:** of the form

$$[(layer\_size, Activation) \dots]$$

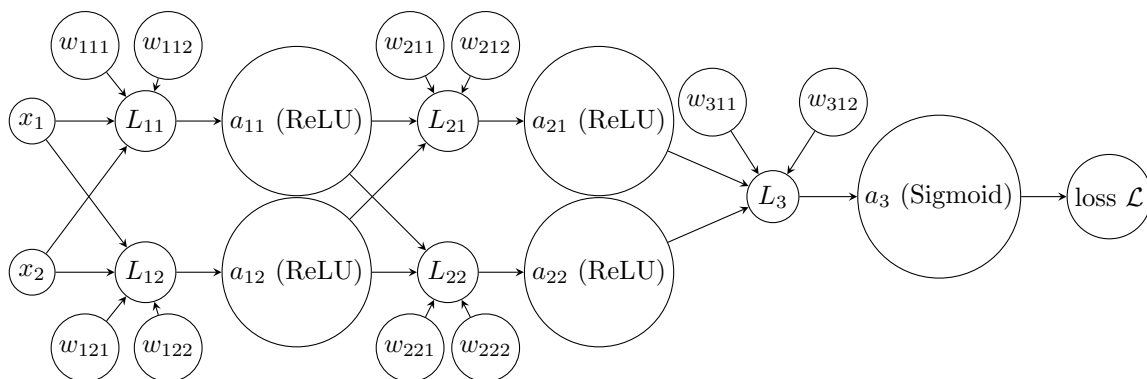
: [(2, ReLU), (2, ReLU), (1, Sigmoid)]

### 1.1 Diagrams

#### 1.1.1 Vectorized Diagram (Equiv to Roger Grosse' 'Computational Graph')

$$\begin{array}{ccccccc} & w_1 & & w_2 & & w_3 & & t \\ & \searrow & & \searrow & & \searrow & & \searrow \\ x \rightarrow L_1 \rightarrow a_1 \rightarrow L_2 \rightarrow a_2 \rightarrow L_3 \rightarrow a_3 \rightarrow \mathcal{L} \end{array}$$

#### 1.1.2 Expanded Diagram (Equiv. to Roger Grosse' 'Network Architecture')



## 1.2 Definitions

### 1.2.1 Remark on weight notation

$w_{i,j,k}$  is to say the weight at the  $i$ -th layer,  $j$ -th neuron,  $k$ -th weight. Hence  $w_{111}$  is the first weight of the first neuron in the first layer, etc.

### 1.2.2 Remark on layer notation

This is a sub-case of the weight notation. I.e.,  $L_{ij}$  is the scalar value of the  $j$ -th neuron at the  $i$ -th layer, etc.

### 1.2.3 Neuron firing calculation

This is just a straightforward dot-product. We have:

$$L_{ij} = \mathbf{w}_{ij} \mathbf{x}_i$$

Where  $\mathbf{x}_i$  in this case is referring to a more general notion of 'layer input', not necessarily just the first input to the network as in the diagrams above.

## 1.3 BackPropagation Derivation

**Notation for derivative of loss w.r.t. to a function** I will be using the following:  $\bar{f} = \frac{\partial \mathcal{L}}{\partial f}$ . This notation was introduced by Roger Grosse from the University of Toronto.

**Pa(x) and Ch(x)** these refer to the sets of parent and child vertices of a vertex in a graph.

**General Approach** Let's label the computational graph nodes as  $v_1, \dots, v_N$  with some topological ordering. Then, our general goal for backprop is to compute  $\bar{v}_i$  for  $i \in 1, \dots, N$ . With these, we can trivially calculate the weight updates. We compute a forward pass of the network, then set  $v_N = 1$ , then, for  $i = N - 1, \dots, 1$ , we have:

$$\bar{v}_i = \sum_{j \in \text{Ch}(v_i)} \bar{v}_j \frac{\partial v_j}{\partial v_i} \quad (\text{The Backprop Rule}) \quad (1)$$

### 1.3.1 Applying the backprop rule

**Loss and final activation** Then, going backwards through the 'computational graph', starting at the end:

$$\begin{aligned} \bar{\mathcal{L}} &= 1 \\ \bar{a}_3 &= \bar{\mathcal{L}} \frac{\partial \mathcal{L}}{\partial a_3} \end{aligned} \quad (2)$$

$$\begin{aligned}
\overline{a_3} &= (1) \frac{\partial \mathcal{L}}{\partial a_3} \\
\overline{a_3} &= \frac{\partial \mathcal{L}}{\partial a_3} \\
\overline{a_3} &= \frac{\partial}{\partial a_3} \frac{1}{2} (a_3 - t)^2 \\
\overline{a_3} &= (a_3 - t) \frac{\partial}{\partial a_3} (a_3 - t) \\
\overline{a_3} &= (a_3 - t) (1) \\
\overline{a_3} &= (a_3 - t)
\end{aligned} \tag{3}$$

**Final layer** *N.B.* I use  $\sigma$  to denote the sigmoid function here, not an activation function.

$$\begin{aligned}
\overline{L_3} &= \overline{a_3} \frac{\partial a_3}{\partial L_3} \\
\overline{L_3} &= \overline{a_3} \frac{\partial}{\partial L_3} \sigma(L_3) \\
\overline{L_3} &= \overline{a_3} \sigma(L_3) (1 - \sigma(L_3))
\end{aligned} \tag{4}$$

**Final layer weights**

$$\begin{aligned}
\overline{w_{31i}} &= \overline{L_3} \frac{\partial}{\partial w_{31i}} L_3 \\
\overline{w_{31i}} &= \overline{L_3} \frac{\partial}{\partial w_{31i}} \sum_j w_{31j} a_{2j} \\
\overline{w_{31i}} &= \overline{L_3} a_{2i}
\end{aligned} \tag{5}$$

**Second layer activation**

$$\begin{aligned}
\overline{a_{2i}} &= \overline{L_3} \frac{\partial}{\partial a_{2i}} L_3 \\
\overline{a_{2i}} &= \overline{L_3} \frac{\partial}{\partial a_{2i}} \sum_j w_{31j} a_{2j} \\
\overline{a_{2i}} &= \overline{L_3} w_{31i}
\end{aligned} \tag{6}$$

### Second layer

$$\begin{aligned}\overline{L_{2i}} &= \overline{a_{2i}} \frac{\partial}{\partial L_{2i}} a_{2i} \\ \overline{L_{2i}} &= \overline{a_{2i}} \frac{\partial}{\partial L_{2i}} \text{ReLU}(L_{2i})\end{aligned}$$

Note that  $d/dx(\text{ReLU}(x))$  is the heaviside step function  $\theta(x)$ :

$$\begin{aligned}\begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \\ \overline{L_{2i}} = \overline{a_{2i}} \quad \theta(L_{2i}) \cdot (1) \\ \overline{L_{2i}} = \overline{a_{2i}} \quad \theta(L_{2i})\end{aligned}\tag{7}$$

### Second layer weights

$$\begin{aligned}\overline{w_{2ij}} &= \overline{L_{2i}} \frac{\partial}{\partial w_{2ij}} L_{2i} \\ \overline{w_{2ij}} &= \overline{L_{2i}} \frac{\partial}{\partial w_{2ij}} \sum_k w_{2ik} a_{1k} \\ \overline{w_{2i}} &= \overline{L_{2i}} a_{1j}\end{aligned}\tag{8}$$

### Input layer activation

$$\begin{aligned}\overline{a_{1i}} &= \overline{L_{21}} \frac{\partial}{\partial a_{1i}} L_{21} + \overline{L_{22}} \frac{\partial}{\partial a_{1i}} L_{22} \\ \overline{a_{1i}} &= \overline{L_{21}} \frac{\partial}{\partial a_{1i}} \sum_j w_{21j} a_{1j} + \overline{L_{22}} \frac{\partial}{\partial a_{1i}} \sum_j w_{22j} a_{1j} \\ \overline{a_{1i}} &= \overline{L_{21}} w_{21i} + \overline{L_{22}} w_{22i}\end{aligned}\tag{9}$$

### Input layer

$$\begin{aligned}\overline{L_{1i}} &= \overline{a_{1i}} \frac{\partial}{\partial L_{1i}} a_1 \\ \overline{L_{1i}} &= \overline{a_{1i}} \frac{\partial}{\partial L_{1i}} \text{ReLU}(L_{1i}) \\ \overline{L_{1i}} &= \overline{a_{1i}} \quad \theta(L_{1i}) \cdot (1) \\ \overline{L_{1i}} &= \overline{a_{1i}} \quad \theta(L_{1i})\end{aligned}\tag{10}$$

### Input layer weights

$$\begin{aligned}
\overline{w_{1ij}} &= \overline{L_{1i}} \frac{\partial}{\partial w_{1ij}} L_{1i} \\
\overline{w_{1ij}} &= \overline{L_{1i}} \frac{\partial}{\partial w_{1ij}} \sum_k w_{1ik} x_{1k} \\
\overline{w_{1ij}} &= \overline{L_{1i}} x_{1j}
\end{aligned} \tag{11}$$

**Notes to self** Grosse follows a per-element approach first, then somehow transformed those results into a vectorized form involving (in some cases) rearranged multiplications and matrix transpose. I am somewhat confused/overwhelmed by this.

#### 1.3.2 Vectorized derivation:

#### 1.3.3 Forward pass vectorized:

$$\begin{aligned}
\mathbf{L}_1 &= \mathbf{w}_1 \cdot \mathbf{x} \\
\mathbf{a}_1 &= \text{ReLU}(\mathbf{L}_1) \\
\mathbf{L}_2 &= \mathbf{w}_2 \cdot \mathbf{a}_1 \\
\mathbf{a}_2 &= \text{ReLU}(\mathbf{L}_2) \\
\mathbf{L}_3 &= \mathbf{w}_3 \cdot \mathbf{a}_2 \\
a_3 &= \sigma(L_3) \\
\mathcal{L} &= \frac{1}{2}(a_3 - t)^2
\end{aligned}$$

#### 1.3.4 Backpropagation vectorized:

$$\begin{aligned}
\overline{\mathcal{L}} &= 1 \\
\overline{a_3} &= \overline{\mathcal{L}}(a_3 - t) \\
\overline{a_3} &= (a_3 - t) \\
\overline{L_3} &= \overline{a_3} \times \frac{\partial}{\partial L_3} a_3 \\
\overline{L_3} &= \overline{a_3} \times \frac{\partial}{\partial L_3} \sigma(L_3) \\
\overline{L_3} &= \overline{a_3} \times \sigma'(L_3) \\
\overline{\mathbf{w}_3} &= \overline{L_3} \frac{\partial}{\partial \mathbf{w}_3} L_3 \\
\overline{\mathbf{w}_3} &= \overline{L_3} \frac{\partial}{\partial \mathbf{w}_3} \mathbf{w}_3 \cdot \mathbf{a}_2
\end{aligned} \tag{12}$$

$$\tag{13}$$

$$\begin{aligned}
\overline{\mathbf{w}}_3 &= \overline{L}_3 \left[ \frac{\frac{\partial}{\partial w_{311}}(w_{311}a_{21} + w_{312}a_{22})}{\frac{\partial}{\partial w_{312}}(w_{311}a_{21} + w_{312}a_{22})} \right] \\
\overline{\mathbf{w}}_3 &= \overline{L}_3 \begin{bmatrix} a_{21} \\ a_{22} \end{bmatrix} \\
\overline{\mathbf{w}}_3 &= \overline{L}_3 \mathbf{a}_2 \\
\overline{\mathbf{a}}_2 &= \overline{L}_3 \frac{\partial}{\partial \mathbf{a}_2} L_3 \\
\overline{\mathbf{a}}_2 &= \overline{L}_3 \frac{\partial}{\partial \mathbf{a}_2} \mathbf{w}_3 \cdot \mathbf{a}_2 \\
\overline{\mathbf{a}}_2 &= \overline{L}_3 \left[ \frac{\frac{\partial}{\partial a_{21}}(w_{311}a_{21} + w_{312}a_{22})}{\frac{\partial}{\partial a_{22}}(w_{311}a_{21} + w_{312}a_{22})} \right] \\
\overline{\mathbf{a}}_2 &= \overline{L}_3 \begin{bmatrix} w_{311} \\ w_{312} \end{bmatrix} \\
\overline{\mathbf{a}}_2 &= \overline{L}_3 \mathbf{w}_3^T
\end{aligned} \tag{14}$$

$$\begin{aligned}
\overline{\mathbf{a}}_2 &= \overline{L}_3 \mathbf{w}_3^T
\end{aligned} \tag{15}$$

## 1.4 Misc. Remarks

### 1.4.1 Rounding: Training vs Inference

Since we aim to train a binary classifier, the `round()` would be necessary for the correct output range. However since `round()` is not differentiable, we omit it during training, calculating fractional losses instead. We only include `round()` during inference.

### 1.4.2 2023-10-18 Remaining points of confusion

How does one go about, concretely, on an element-by-element level, determining say the matrix equivalent of derivative of a function applied to a matrix? Is the derivative applied element-wise to the existing matrix, yielding a matrix of the same dimension as the original? Then we need to have a clean notation for that without getting confusing/ambiguous. I find some of the notational conventions here unclear and confusing. The current approach is vague and imprecise, which I find bothersome.

### 1.4.3 2023-10-19 Remaining questions

- What is the meaning of  $\circ$  in the context of these vectorized equations? What is the difference between  $\circ$  and  $\cdot$ ? Answer: According to ChatGPT, it is the composition of the linear transformations represented by the matrices. Apparently ChatGPT can also be coaxed into thinking it's the same as matrix multiplication, i.e.,  $A \circ B = AB$ . OK, so if one looks at [https://math.libretexts.org/Bookshelves/Linear\\_Algebra/Interactive\\_Linear\\_Algebra\\_\(Margalit\\_and\\_Rabinoff\)/03%3A\\_Linear\\_Transformations\\_](https://math.libretexts.org/Bookshelves/Linear_Algebra/Interactive_Linear_Algebra_(Margalit_and_Rabinoff)/03%3A_Linear_Transformations_)

and\_Matrix\_Algebra/3.04%3A\_Matrix\_Multiplication#:~:text=As%20we%20will%20see%2C%20composition,of%20transformations%20and%20of%20matrices. It does seem like it really IS matrix multiplication, but then WHY bother to use this symbol? I am somewhat confused but am still feeling reassured that I am justified in assuming it's equiv. to matrix multiplication.

- Why are some of the matrices in these vectorized backprops transposed? Examples from Roger Grosse:

•

$$\overline{\mathbf{W}^{(2)}} = \overline{\mathbf{y}} \mathbf{h}^T$$

- This would be so much easier if they stated the dimensions of the various matrices/vectors in the equations
- Other questions: What does it mean to take for example:  $\frac{\partial}{\partial \mathbf{w}_3}(\mathbf{w}_3 \cdot \mathbf{a}_2)$ ? Answer: There are two observations. One, that in general, for  $\mathbf{x} \in \mathbb{R}^n, f(\mathbf{x}) : \mathbb{R}^n \mapsto \mathbb{R}$ , we have that  $\frac{\partial f}{\partial \mathbf{x}} = [\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \dots]$  Also, recall that in our MLP:  $\mathbf{w}_3 = [w_{311}, w_{312}]$  and  $\mathbf{a}_2 = [a_{21}, a_{22}]$  so  $\mathbf{w}_3 \cdot \mathbf{a}_2 = w_{311}a_{21} + w_{312}a_{22}$ , so  $\frac{\partial \mathbf{w}_3 \cdot \mathbf{a}_2}{\partial \mathbf{w}_3} = [\frac{\partial}{\partial w_{311}}(w_{311}a_{21} + w_{312}a_{22}), \frac{\partial}{\partial w_{312}}(w_{311}a_{21} + w_{312}a_{22})]$  so  $\frac{\partial \mathbf{w}_3 \cdot \mathbf{a}_2}{\partial \mathbf{w}_3} = [a_{21} + w_{312}a_{22}, w_{311}a_{21} + a_{22}] \neq \mathbf{a}_2$