Backpropagation with Vectors in Neural Network

7 min read · Sep 3, 2024

Liyuan Chen Follow

Chain rule

Let's begin with the chain rule in scalars. Here is the article that goes through the chain rule from functions of one variable to multiple variables. I found the idea of building up a **tree diagram** quite helpful in visualizing the chain rule.

Feedforward neuron network

In feedforward neuron networks, the information flows from the input node, through the hidden node, and to the output nodes (depicted in the black right arrow), while the backpropagation works in the opposite direction as the red left arrow in the below figure.

We only consider one layer in the neuron network. In Fig 1, we have the vector input [x1, x2] and vector output [y1, y2, y3] for a single layer, and scalar loss L at the end of the network.

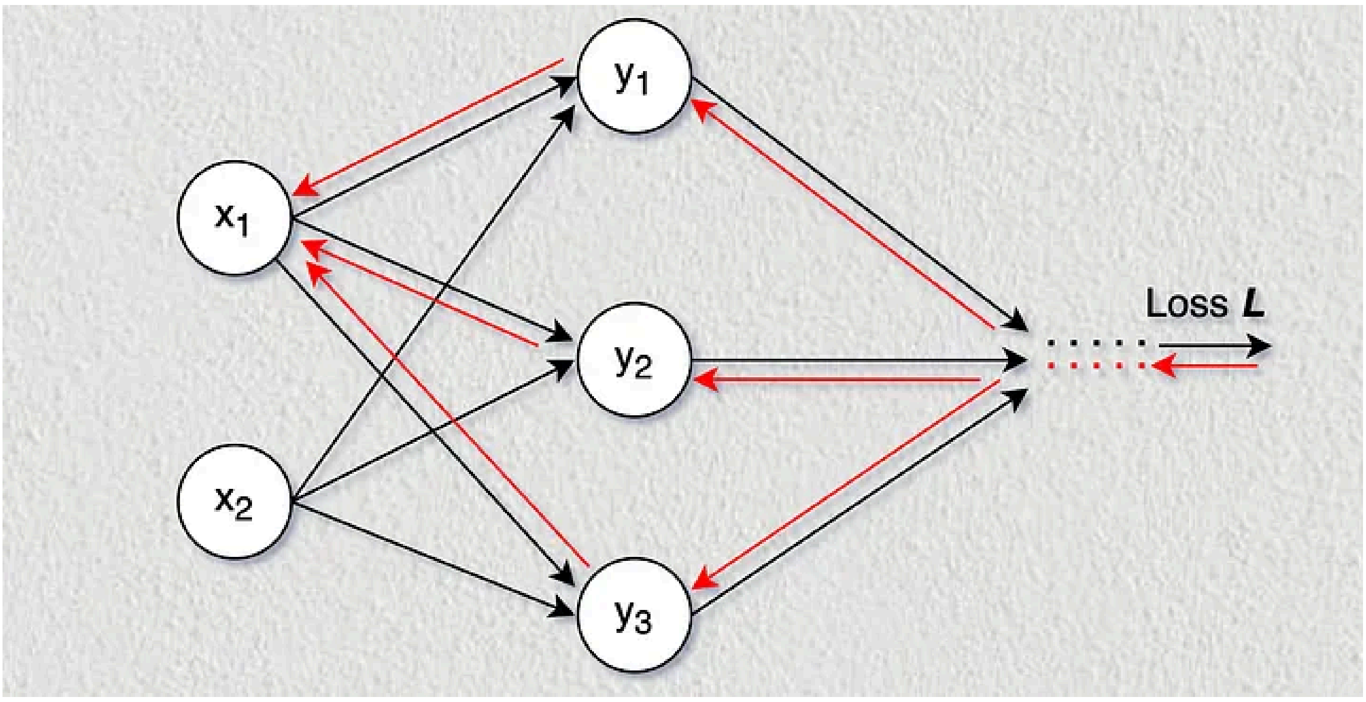


Fig 1. A layer in the feedforward neural network

Vector derivatives

Here is the definition of derivatives with vectors. Specifically, the derivatives of a vector function y, with respect to an input vector x is written as:

$$\mathbf{x} = [x_1, x_2, \ldots, x_m]^T,\ \mathbf{y} = [y_1, y_2, \ldots, y_n]^T,\ f\colon \mathbb{R}^m \to \mathbb{R}^n$$

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_m} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_m} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_n}{\partial x_1} & \frac{\partial y_n}{\partial x_2} & \cdots & \frac{\partial y_n}{\partial x_m} \end{bmatrix}$$

It is also known as the Jacobian matrix.

Chain Rule in Backpropagation

Now let's apply the chain rule to backpropagation in feedforward neuron networks with tree diagrams in our mind.

1) backprop in add gate

In Fig 2, we have the vector input *x = [x1, x2]^T* and scalar output *y* for the single layer, and scalar loss *L* at the end of the network.

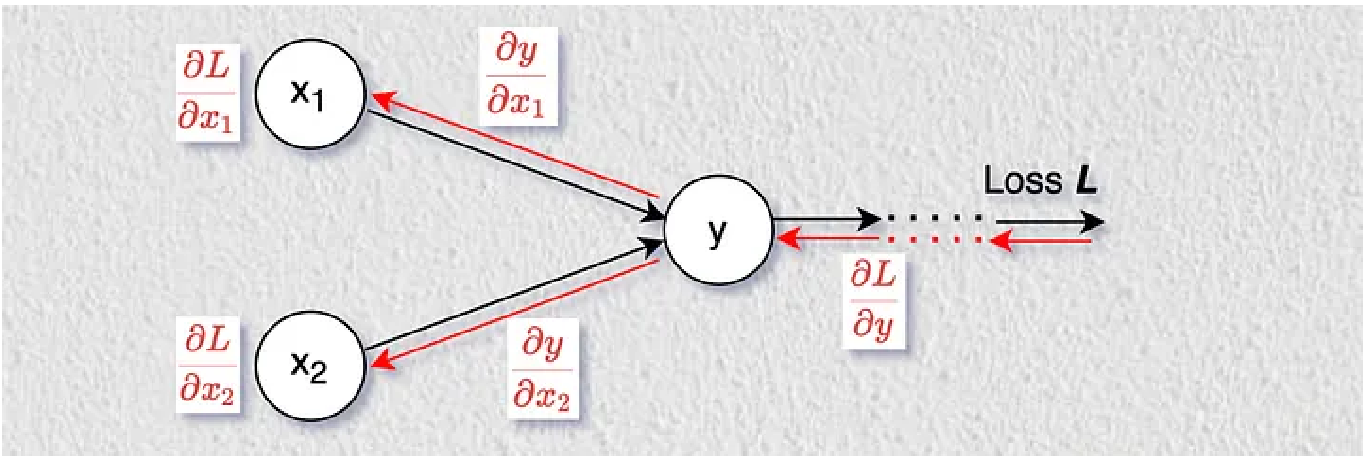

$\frac{\partial L}{\partial x_1}$
x_1
$\frac{\partial y}{\partial x_1}$
Loss *L*
y
$\frac{\partial L}{\partial y}$
$\frac{\partial L}{\partial x_2}$
x_2
$\frac{\partial y}{\partial x_2}$

Fig 2. Backprop in the add gate

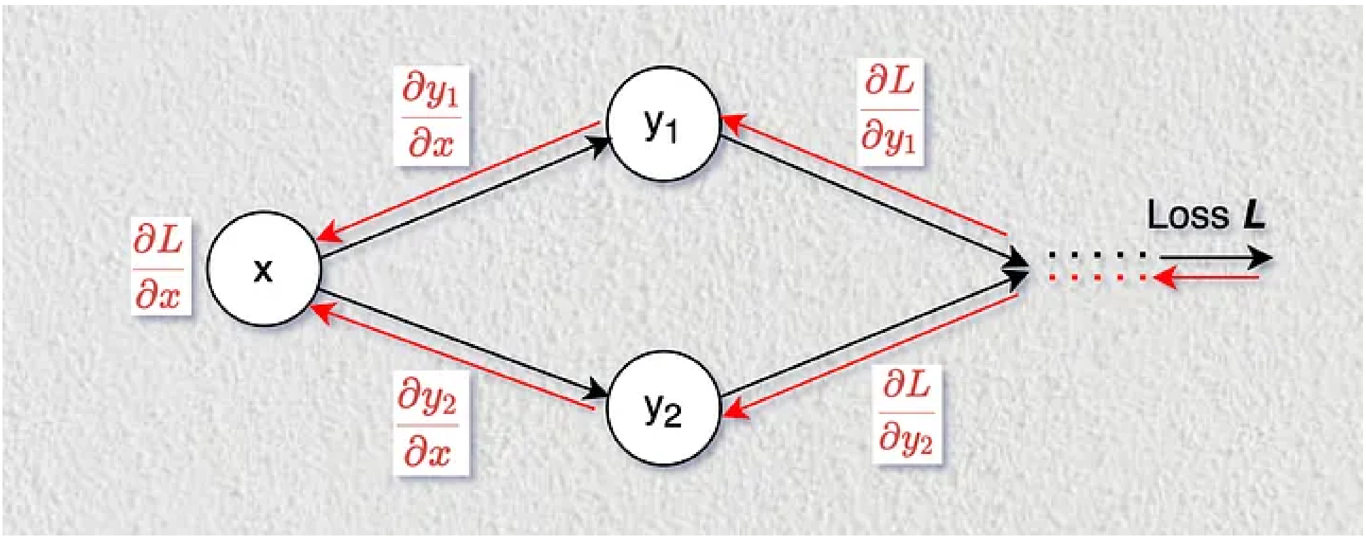
The chain rule for the add gate in backprop is listed below:

$$\frac{\partial L}{\partial x_1} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial x_1}$$

$$\frac{\partial L}{\partial x_2} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial x_2}$$

2) backprop in copy gate

In Fig 3, we have the scalar input *x* and vector output ***y*** *= [y1, y2]* for the single layer, and scalar loss *L* at the end of the network.



$\frac{\partial y_1}{\partial x}$
$\frac{\partial L}{\partial y_1}$
y_1
$\frac{\partial L}{\partial x}$
x
Loss ***L***
$\frac{\partial y_2}{\partial x}$
y_2
$\frac{\partial L}{\partial y_2}$

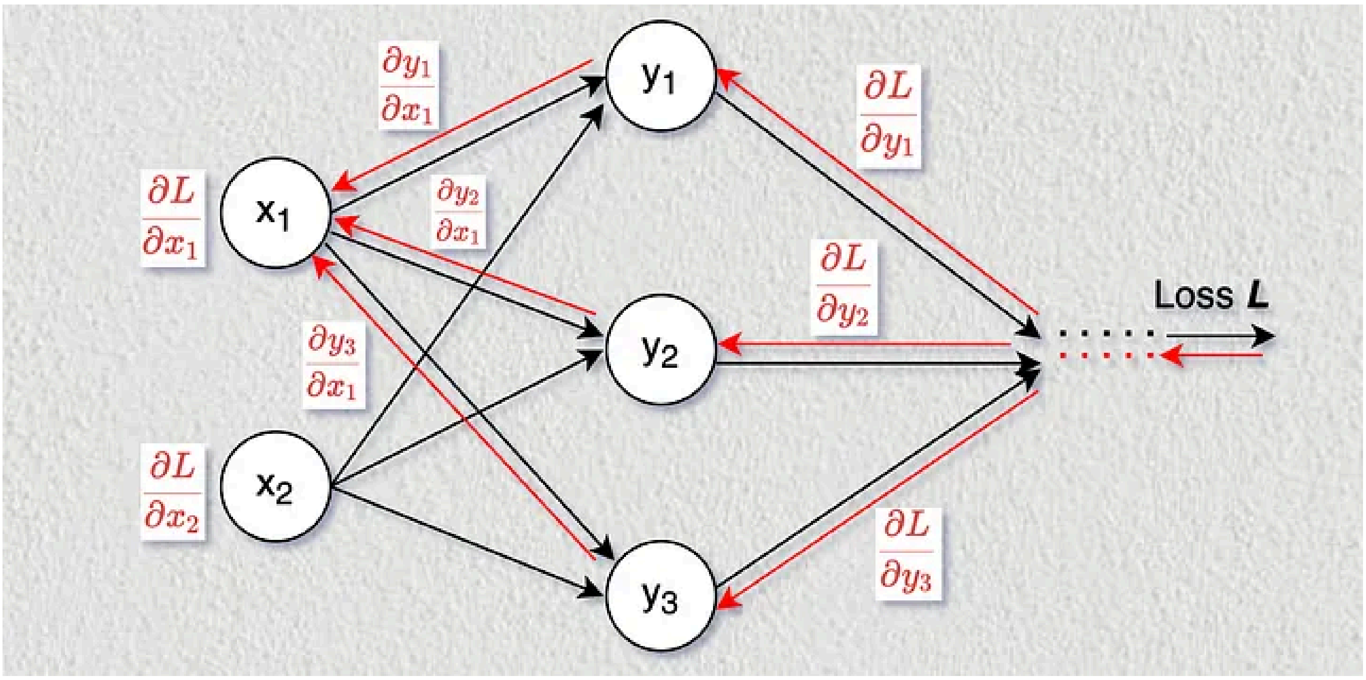
Fig 3. Backprop in the copy gate

The chain rule for the backpropagation in the copy gate is listed below:

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y_1}\frac{\partial y_1}{\partial x} + \frac{\partial L}{\partial y_2}\frac{\partial y_2}{\partial x}$$

3) backprop in a layer

In Fig 4, we have the vector input ***x*** *= [x1, x2]^T* and vector output ***y*** *= [y1, y2]^T* for the single layer, and scalar loss *L* at the end of the network.



$\frac{\partial y_1}{\partial x_1}$
$\frac{\partial y_2}{\partial x_1}$
$\frac{\partial y_3}{\partial x_1}$
$\frac{\partial L}{\partial x_1}$
$\frac{\partial L}{\partial x_2}$
x_1
x_2
y_1
y_2
y_3
$\frac{\partial L}{\partial y_1}$
$\frac{\partial L}{\partial y_2}$
$\frac{\partial L}{\partial y_3}$
Loss ***L***

Fig 4. Backprop in a layer

We can think of the layer as a combination of add gates and copy gates. The chain rule for the backpropagation in the single layer is listed below:

$$\frac{\partial L}{\partial x_1} = \frac{\partial L}{\partial y_1}\frac{\partial y_1}{\partial x_1} + \frac{\partial L}{\partial y_2}\frac{\partial y_2}{\partial x_1} + \frac{\partial L}{\partial y_3}\frac{\partial y_3}{\partial x_1}$$

$$\frac{\partial L}{\partial x_2} = \frac{\partial L}{\partial y_1}\frac{\partial y_1}{\partial x_2} + \frac{\partial L}{\partial y_2}\frac{\partial y_2}{\partial x_2} + \frac{\partial L}{\partial y_3}\frac{\partial y_3}{\partial x_2}$$

Let's put ∂*L*/∂x1 and ∂*L*/∂x2 together, and the same for ∂*L*/∂y1, ∂*L*/∂y2, and ∂*L*/∂y3, then we can get this chain rule in vector (or matrix) form:

$$\begin{bmatrix} \frac{\partial L}{\partial x_1} \\ \frac{\partial L}{\partial x_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} & \frac{\partial y_3}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} & \frac{\partial y_3}{\partial x_2} \end{bmatrix} \begin{bmatrix} \frac{\partial L}{\partial y_1} \\ \frac{\partial L}{\partial y_2} \\ \frac{\partial L}{\partial y_3} \end{bmatrix} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} \\ \frac{\partial y_3}{\partial x_1} & \frac{\partial y_3}{\partial x_2} \end{bmatrix}^T \begin{bmatrix} \frac{\partial L}{\partial y_1} \\ \frac{\partial L}{\partial y_2} \\ \frac{\partial L}{\partial y_3} \end{bmatrix}$$

Eq 1. Backprop chain rule in matrix form for a layer with **column vector** input and output

With the knowledge of derivatives with vectors, we noticed that the transposed matrix is the derivative of **output vector** ***y*** with respect to **input vector** ***x***.

Now let's write this chain rule (Eq. 1) in vector symbol and generalize the dimension of input and output vectors:

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]^T, \ \mathbf{y} = [y_1, y_2, \ldots, y_m]^T, \ f: \mathbb{R}^n \rightarrow \mathbb{R}^m$$

$$\frac{\partial L}{\partial \mathbf{x}} = \left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right)^T \frac{\partial L}{\partial \mathbf{y}}$$

Eq 2. Backprop chain rule in vector symbol for a general layer with **column vector** input and output

We can take a further look at this vector chain rule (Eq. 2): on the right side of the equation, compared to the scalar chain rule, we swapped the two items ∂**y**/∂**x** and ∂L/∂**y**, and performed a transpose operation on ∂**y**/∂**x**. It is a careful adjustment to ensure dimensional correctness in matrix multiplication when input and output vectors are both column vectors.

Note that in matrix calculus, vectors are typically treated as column vectors. So as we did for Eq. 1 ~ 2.

However, in practice, the input and output data is not only one data point, instead it's often batches of data. That's why we usually use a row vector to represent one data point and stack batches of data row-by-row. In this case, we need to do a transpose operation on the both Eq. 1 and Eq. 2. Hence, we turn Eq. 1 into:

$$\begin{bmatrix} \frac{\partial L}{\partial x_1} & \frac{\partial L}{\partial x_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial y_1} & \frac{\partial L}{\partial y_2} & \frac{\partial L}{\partial y_3} \end{bmatrix} \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} \\ \frac{\partial y_3}{\partial x_1} & \frac{\partial y_3}{\partial x_2} \end{bmatrix}$$

Eq 3. Backprop chain rule in matrix form for a layer with **row vector** input and output

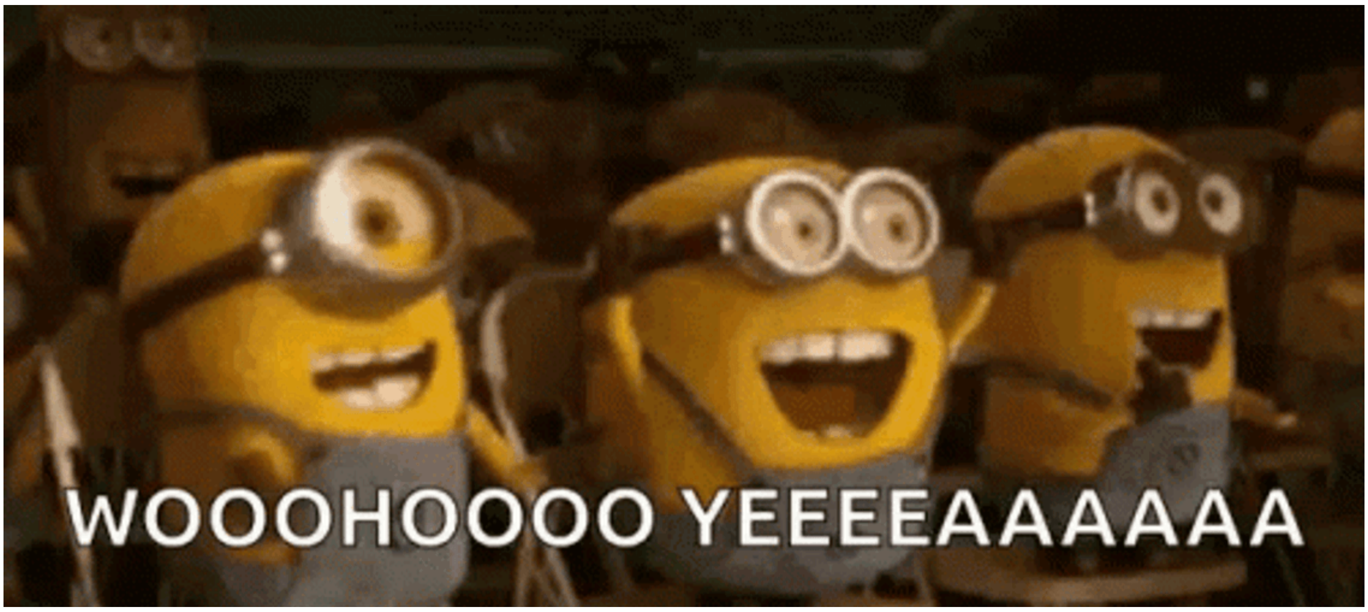
We can also write Eq. 3 in vector symbols and generalize the dimensions of input and output vectors:

$$\mathbf{x} = [x_1, x_2, \ldots, x_m],\ \mathbf{y} = [y_1, y_2, \ldots, y_n],\ f\colon \mathbb{R}^m \to \mathbb{R}^n$$

$$\frac{\partial L}{\partial \mathbf{x}} = \frac{\partial L}{\partial \mathbf{y}} \frac{\partial \mathbf{y}}{\partial \mathbf{x}}$$

Eq 4. Backprop chain rule in vector symbol for a general layer with **row vector** or **matrix** input and output

We can see that this vector chain rule (Eq. 4) shares the same format as the scalar chain rule.



Now we have the backprop vector chain rule (Eq. 4) for a general layer in the feedforward neural network. We can think of two common layers in deep learning: the linear layer and the activation layer. In practice, we call a linear function followed by an activation function a layer. However, here we separate the linear function from the activation function to better understand backpropagation in each step.

3.1) backprop in the linear layer

In the linear layer, we have a weight multiplier *w_ij* between every input node *x_i* and output node *y_j*, as shown in Fig. 5.

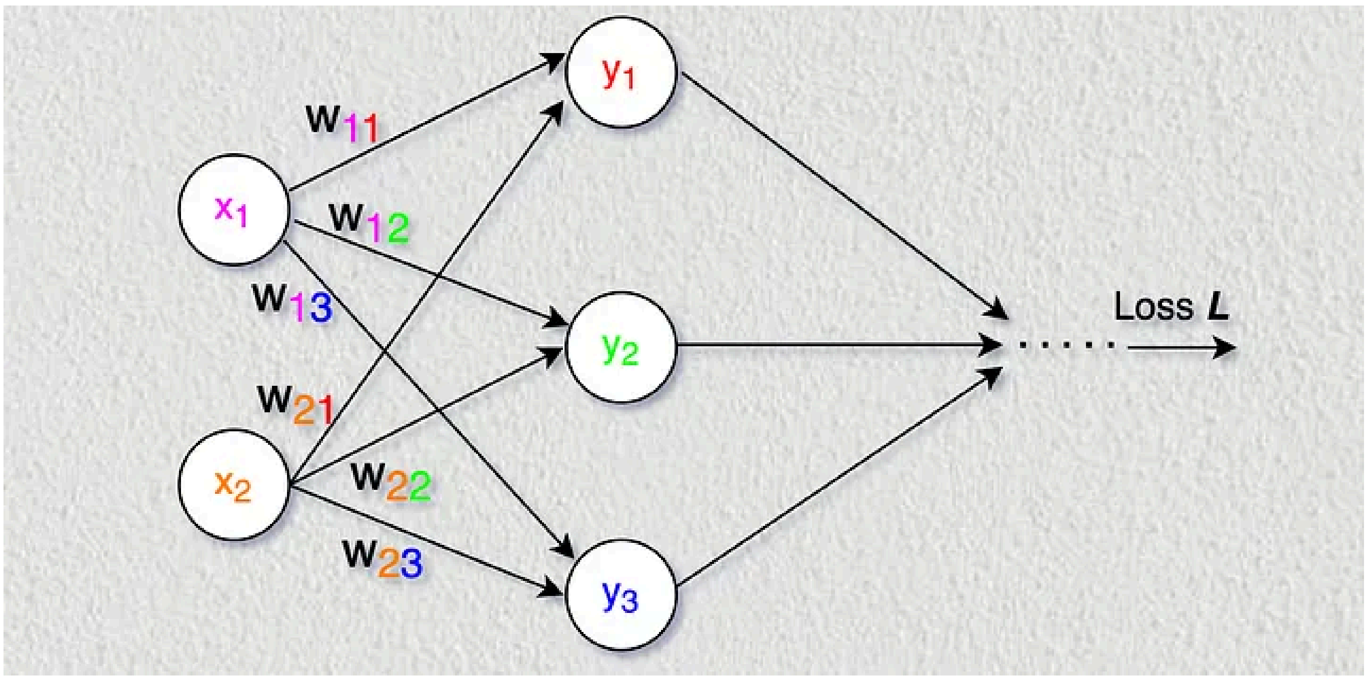


Fig 5. A linear layer

This linear layer can be written in the formulas below:

$$
\begin{aligned}
y_1 &= w_{11}x_1 + w_{21}x_2 \\
y_2 &= w_{12}x_1 + w_{22}x_2 \\
y_3 &= w_{13}x_1 + w_{23}x_2
\end{aligned}
$$

$$
y_j = \sum_i w_{ij}x_i
$$

$$
\frac{\partial y_j}{\partial x_i} = w_{ij}, \quad i = 1, 2, j = 1, 2, 3
$$

The derivative of output node *y* with respect to input node *x_i* equals *w_ij*. Let's apply this knowledge to the general backprop chain rule (Eq. 3), then we can get:

$$
\begin{aligned}
\begin{bmatrix} \frac{\partial L}{\partial x_1} & \frac{\partial L}{\partial x_2} \end{bmatrix} &= \begin{bmatrix} \frac{\partial L}{\partial y_1} & \frac{\partial L}{\partial y_2} & \frac{\partial L}{\partial y_3} \end{bmatrix} \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \\ w_{13} & w_{23} \end{bmatrix} \\
&= \begin{bmatrix} \frac{\partial L}{\partial y_1} & \frac{\partial L}{\partial y_2} & \frac{\partial L}{\partial y_3} \end{bmatrix} \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{bmatrix}^T
\end{aligned}
$$

Eq 5. Backprop chain rule in matrix form for a linear layer with **row vector** input and output

Then we can write this chain rule (Eq. 5) in matrix symbols and generalize the dimension of input and output vectors:

$$X = [x_1, x_2, \ldots, x_m],\ Y = [y_1, y_2, \ldots, y_n],\ W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{bmatrix}$$

$$Y = XW$$

$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial Y} W^T$$

Eq 6. Backprop chain rule for a linear layer with **row vector** or **matrix** input and output

We can see that weight matrix *W* acts as a bridge between layers in backpropagation, as it is used to compute the gradient for the previous layer *∂L/∂X* based on the gradient from the subsequent layer *∂L/∂Y* during the backward pass.

In addition to calculating the gradient flow *∂L/∂X* and *∂L/∂Y* during the backward pass, we also compute the gradient *∂L/∂W* and store it locally for later optimization updates of *W* after the backward pass is completed. Here is how we get the gradient of parameters *∂L/∂W* and *∂L/∂B*.

First, we can get the **element-wise chain rule**:

$$\frac{\partial L}{\partial w_{ij}} = \sum_k \frac{\partial L}{\partial y_k} \frac{\partial y_k}{\partial w_{ij}} = \frac{\partial L}{\partial y_j} x_i$$

$$\text{only when k = j, } y_k \text{ is connected to } w_{ij}$$

> *Note: it's more efficient to calculate element-wise chain rule* ∂L/∂w_ij *leveraging the linear property, rather than using the vectorized chain rule* ∂L/∂W = (∂L/∂Y)(∂Y/∂W) *that requires computing the more complex* ∂Y/∂W *term.*

Second, according to the definition of scalar-by-matrix derivative, we can get the matrix chain rule:

$$\frac{\partial L}{\partial W} = \begin{bmatrix} \frac{\partial L}{\partial w_{11}} & \cdots & \frac{\partial L}{\partial w_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial L}{\partial w_{m1}} & \cdots & \frac{\partial L}{\partial w_{mn}} \end{bmatrix} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} \begin{bmatrix} \frac{\partial L}{\partial y_1} & \cdots & \frac{\partial L}{\partial y_n} \end{bmatrix} = X^T \frac{\partial L}{\partial Y}$$

Then we can get the gradient of parameters ∂L/∂W, and the same for ∂L/∂B:

$$X = [x_1, x_2, \dots, x_m],\ Y = [y_1, y_2, \dots, y_n],\ W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m1} & w_{m2} & \cdots & w_{mn} \end{bmatrix}$$

$$Y = XW + B$$

$$\frac{\partial L}{\partial W} = X^T \frac{\partial L}{\partial Y}$$

$$\frac{\partial L}{\partial B} = \frac{\partial L}{\partial Y}$$

Eq 7. The gradient of parameters during backprop for a linear layer with **row vector** or **matrix** input and output

Furthermore, in practice, the input and output data will be batches of row vectors, which means X and Y are matrices that stack row vectors row-by-row. Please refer to this article about backpropagation in a linear layer using minibatches. We can see that Eq. 6 generalizes well in batch processing.

3.2) backprop in the activation layer

In the activation layer, we have the vector input *x = [x1, x2],* vector output *y = [y1, y2],* and an activation function y = σ(x). Note that the input and output vectors of the activation layer share the same length.

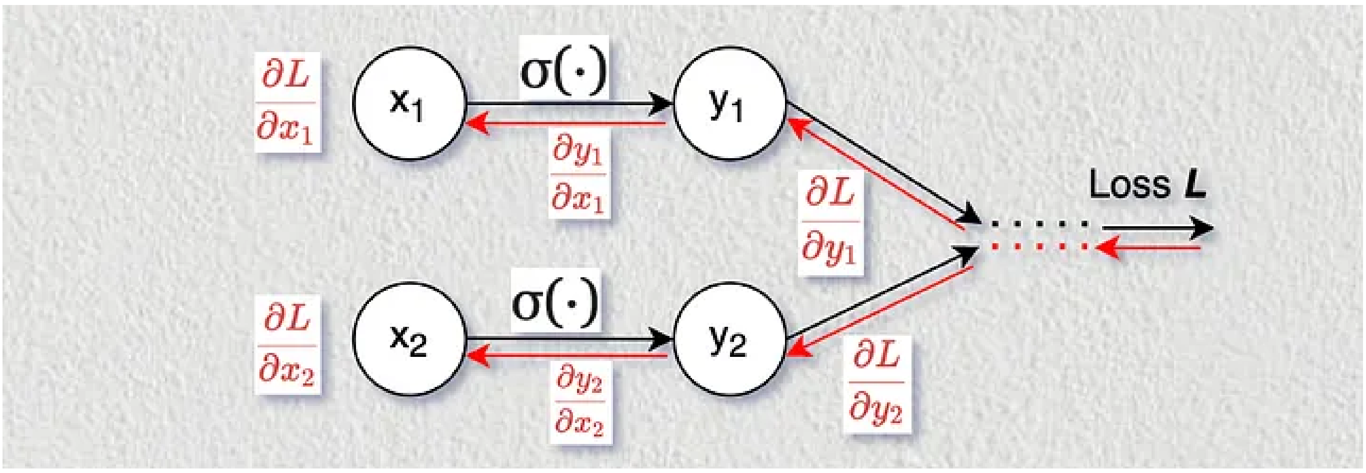

$\frac{\partial L}{\partial x_1}$
x_1
$\sigma(\cdot)$
y_1
$\frac{\partial y_1}{\partial x_1}$
$\frac{\partial L}{\partial y_1}$
Loss L
$\frac{\partial L}{\partial x_2}$
x_2
$\sigma(\cdot)$
y_2
$\frac{\partial y_2}{\partial x_2}$
$\frac{\partial L}{\partial y_2}$

Fig 6. An activation layer

This activation layer can be written in the formulas below:

$$y_i = \sigma(x_i), \quad i = 1, 2$$

$$\frac{\partial y_j}{\partial x_i} = \begin{cases} \sigma'(x_i), & \text{if } j = i, \\ 0, & \text{otherwise} \end{cases}$$

Let's apply this formula to the general backprop chain rule (Eq. 3), then we can get:

$$
\begin{aligned}
\begin{bmatrix} \frac{\partial L}{\partial x_1} & \frac{\partial L}{\partial x_2} \end{bmatrix} &= \begin{bmatrix} \frac{\partial L}{\partial y_1} & \frac{\partial L}{\partial y_2} \end{bmatrix} \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} \end{bmatrix} \\
&= \begin{bmatrix} \frac{\partial L}{\partial y_1} & \frac{\partial L}{\partial y_2} \end{bmatrix} \begin{bmatrix} \sigma'(x_1) & 0 \\ 0 & \sigma'(x_2) \end{bmatrix} \\
&= \begin{bmatrix} \frac{\partial L}{\partial y_1} & \frac{\partial L}{\partial y_2} \end{bmatrix} \odot \begin{bmatrix} \sigma'(x_1) & \sigma'(x_2) \end{bmatrix}
\end{aligned}
$$

Eq 8. Backprop chain rule in matrix form for an activation layer with **row vector** input and output

In Eq. 8, we convert matrix multiplication with a diagonal matrix into element-wise multiplication (Hadamard product), which uses minimal memory and is simpler to compute. Then we can write Eq. 8 in matrix symbol and generalize the dimension of input and output vectors:

$$
X = [x_1, x_2, \ldots, x_m],\ Y = [y_1, y_2, \ldots, y_m],\ Y = \sigma(X)
$$

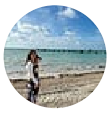
$$
\frac{\partial L}{\partial X} = \frac{\partial L}{\partial Y} \odot \sigma'(X)
$$

Eq 9. Backprop chain rule for an activation layer with **row vector** or **matrix** input and output

And that's all I want to share about backpropagation in feedforward neural network :)

NOTE: All images are by the author.

Backpropagation · Deep Learning · Neural Networks · Machine Learning



Follow

Written by Liyuan Chen

13 followers · 17 following

AI & Robotics Enthusiast

No responses yet