Neural Networks

By Trần Minh Dương - Learning Support

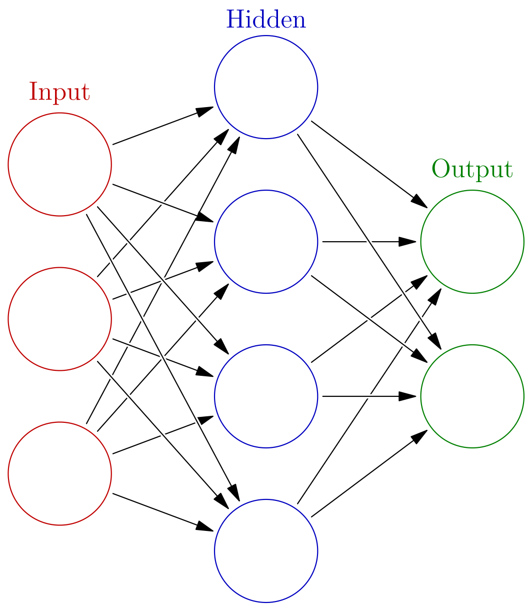
Overview

Neural networks are powerful models inspired by the human brain, capable of learning complex patterns from data. They consist of interconnected layers of neurons that process information and transform inputs into meaningful outputs. From image recognition to natural language processing, this document aims to uncover how neural networks are trained.

1. Architecture

A typical neural network is composed of the following:

- **Input Layer**: Receives raw data (features) and feeds it into the network.
- **Hidden Layers**: Performs computations with weights, biases, and activation functions to uncover patterns in the data.
- **Output Layer**: Produces the final result, such as a class label or prediction value.

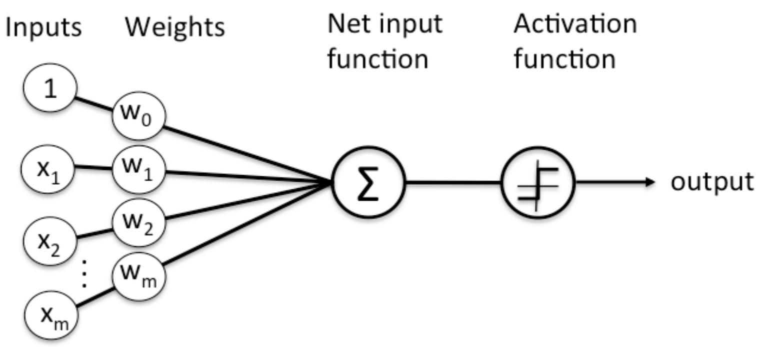


2. Learning Process

The network trains through these steps:

1. **Forward Pass**: Let data flow through the network, producing predictions.
2. **Loss Calculation**: A loss function measures prediction errors.
3. **Back Propagation**: Errors are propagated backward to adjust weights and biases.
4. **Optimization**: Algorithms like Gradient Descent minimize loss, improving predictions.

3. Forward Pass



To calculate the next node in a neural network, we do the following:

- Calculate the weighted sum of the previous nodes x_i:

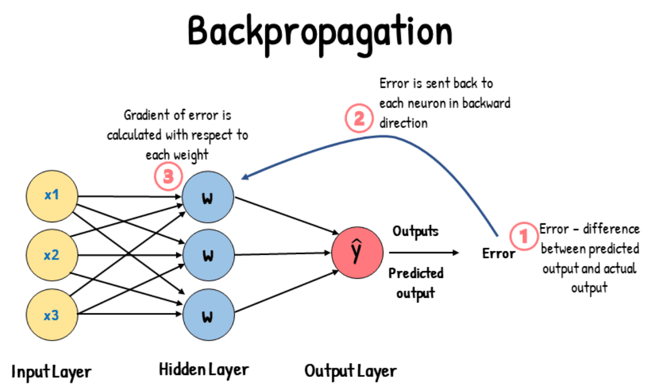
$$x_j = \sum_i x_i \cdot w_{ij} = x_1 \cdot w_{1j} + x_2 \cdot w_{2j} + \ldots$$

- Pass it through the activation function:

$$y_j = f(x_j)$$

To further understand, see exercise below

4. Backpropagation (Mad Difficult)


Backpropagation
Error is sent back to each neuron in backward direction
2
Gradient of error is calculated with respect to each weight
3
x1
x2
x3
w
w
w
ŷ
Outputs
Predicted output
Error
1
Error - difference between predicted output and actual output
Input Layer
Hidden Layer
Output Layer

Step 1. Error Calculation

The loss function measures the difference between the predicted output and the actual output:

$$\text{Loss} = \frac{1}{2} \cdot (y_{\text{output}} - y_{\text{actual}})^2$$

(Note that this loss function varies for different problems: A regression problem may use MSE while a classification one may use log-likelihood.)

Step 2. Compute Partial Derivatives for Weight Updates

The gradient of the loss with respect to any weight w_{ij} is computed as follows:

$$\frac{\partial \text{Loss}}{\partial w_{ij}} = \frac{\partial \text{Loss}}{\partial y_{\text{output}}} \cdot \frac{\partial y_{\text{output}}}{\partial x_{\text{output}}} \cdot \frac{\partial x_{\text{output}}}{\partial y_j} \cdot \frac{\partial y_j}{\partial x_j} \cdot \frac{\partial x_j}{\partial w_{ij}}$$

Step 3. Compute Partial Derivatives for Specific Layers

For output layer weights ($w_{\text{j-output}}$):

$$\frac{\partial \text{Loss}}{\partial w_{\text{j-output}}} = (y_{\text{output}} - y_{\text{actual}}) \cdot F'(x_{\text{output}}) \cdot y_j$$

For hidden layer weights (w_{ij} with j in a hidden layer and k in the next layer):

$$\frac{\partial \text{Loss}}{\partial w_{ij}} = \sum_{k \in \text{next layer}} (y_k - y_{\text{actual}}) \cdot F'(x_k) \cdot w_{jk} \cdot F'(x_j) \cdot y_i$$

Step 4. Weight Update Rule

Weights are updated using gradient descent:

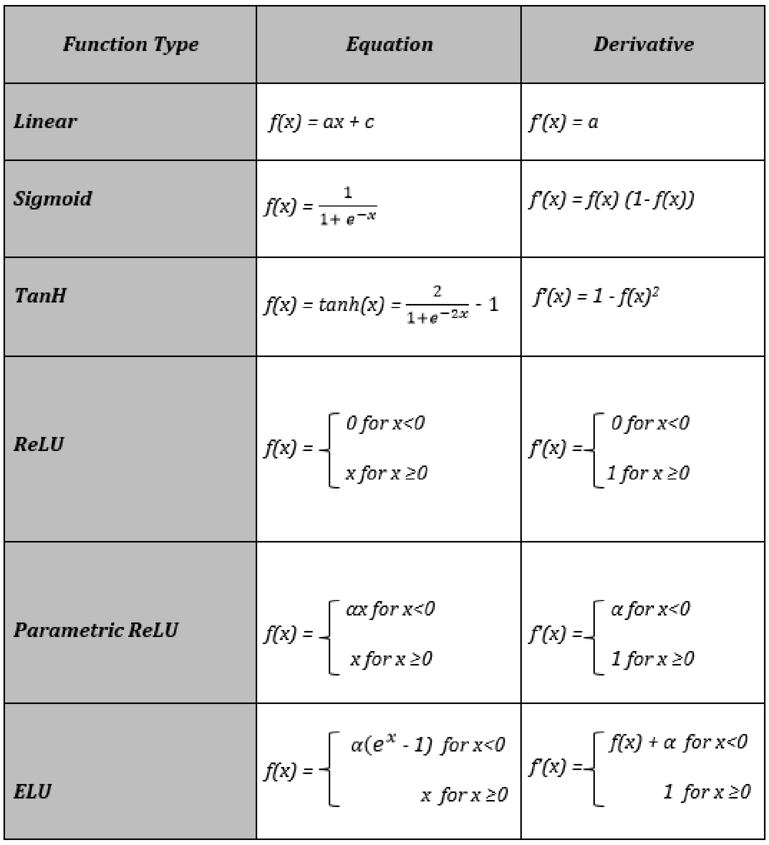
$$w_{ij}^{\text{new}} = w_{ij} - \eta \cdot \frac{\partial \text{Loss}}{\partial w_{ij}}$$

Step 5. Bias Update Rule (if applicable)

Biases are updated similarly to weights:

$$b_j^{\text{new}} = b_j - \eta \cdot \frac{\partial \text{Loss}}{\partial b_j}$$

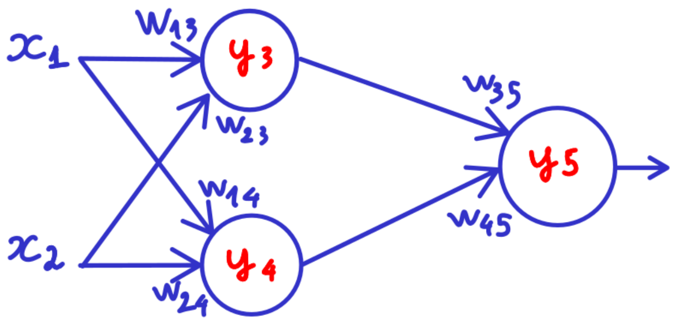
Activation Function Derivative



Function Type	*Equation*	*Derivative*
Linear	$f(x) = ax + c$	$f'(x) = a$
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)\,(1 - f(x))$
TanH	$f(x) = \tanh(x) = \frac{2}{1+e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ReLU	$f(x) = \begin{cases} 0 \text{ for } x<0 \\ x \text{ for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 \text{ for } x<0 \\ 1 \text{ for } x \geq 0 \end{cases}$
Parametric ReLU	$f(x) = \begin{cases} \alpha x \text{ for } x<0 \\ x \text{ for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha \text{ for } x<0 \\ 1 \text{ for } x \geq 0 \end{cases}$
ELU	$f(x) = \begin{cases} \alpha(e^x - 1) \text{ for } x<0 \\ x \text{ for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha \text{ for } x<0 \\ 1 \text{ for } x \geq 0 \end{cases}$

Exercise

Given the following neural network:



Weights:

- $w_{13} = 2$, $w_{23} = -3$
- $w_{14} = 1$, $w_{24} = 4$
- $w_{35} = 2$, $w_{45} = -1$

Activation Function (ReLU):

$$\text{ReLU}(v) = \begin{cases} v & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases}$$

Derivative of ReLU:

$$\text{ReLU}'(v) = \begin{cases} 1 & \text{if } v > 0 \\ 0 & \text{if } v \leq 0 \end{cases}$$

Questions:
a) Predict the output with inputs: $x_1 = 1$ and $x_2 = -1$.
b) Using back-propagation, evaluate the weights after one iteration with $\eta = 0.1$.

Solution:

a) Predict the output with inputs: $x_1 = 1$ and $x_2 = -1$

Forward Pass:

1. **At node y_3:**

$$x_3 = x_1 \cdot w_{13} + x_2 \cdot w_{23} = 1 \cdot 2 + (-1) \cdot (-3) = 5$$

$$y_3 = \text{ReLU}(x_3) = 5 \quad (\text{since } x_3 > 0)$$

2. **At node y_4:**

$$x_4 = x_1 \cdot w_{14} + x_2 \cdot w_{24} = 1 \cdot 1 + (-1) \cdot 4 = -3$$

$$y_4 = \text{ReLU}(x_4) = 0 \quad (\text{since } x_4 < 0)$$

3. **At node y_5:**

$$x_5 = y_3 \cdot w_{35} + y_4 \cdot w_{45} = 5 \cdot 2 + 0 \cdot (-1) = 10$$

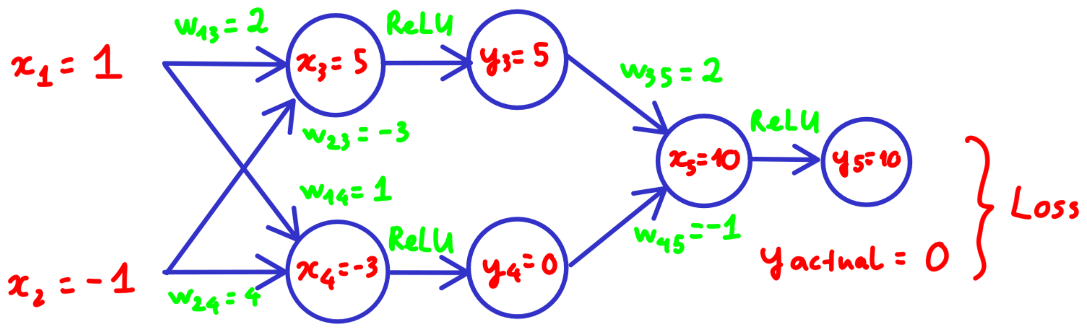
$$y_{\text{output}} = \text{ReLU}(x_5) = 10 \quad (\text{since } x_5 > 0)$$

Prediction:

$$y_{\text{output}} = 10$$

---

b) Back-propagation: Weight Updates



Loss Function:

$$\text{Loss} = \frac{1}{2} \cdot (y_{\text{output}} - y_{\text{actual}})^2 = \frac{1}{2} \cdot (10 - 0)^2 = 50$$

Gradient Formula for Any Weight w_{ij}:

$$\frac{\partial \text{Loss}}{\partial w_{ij}} = \frac{\partial \text{Loss}}{\partial y_{\text{output}}} \cdot \frac{\partial y_{\text{output}}}{\partial x_{\text{output}}} \cdot \frac{\partial x_{\text{output}}}{\partial y_j} \cdot \frac{\partial y_j}{\partial x_j} \cdot \frac{\partial x_j}{\partial w_{ij}}$$

---

1. Update Output Layer Weights (w_{35}, w_{45}):

For w_{35}:

$$\boxed{y_3 \underset{w_{35}}{\longrightarrow} x_5 \underset{\text{ReLU}}{\longrightarrow} y_5 \underset{\text{MSE}}{\longrightarrow} \text{Loss}}$$

By the **chain rule**:

$$\begin{aligned}\frac{\partial \text{Loss}}{\partial w_{35}} &= \frac{\partial \text{Loss}}{\partial y_5} \cdot \frac{\partial y_5}{\partial x_5} \cdot \frac{\partial x_5}{\partial w_{35}} \\ &= (y_5 - y_{\text{actual}}) \cdot \text{ReLU}'(x_5) \cdot y_3 \\ &= 10 \cdot 1 \cdot 5 \\ &= 50\end{aligned}$$

Update:

$$w_{35}^{\text{new}} = w_{35} - \eta \cdot \frac{\partial \text{Loss}}{\partial w_{35}} = 2 - 0.1 \cdot 50 = -3$$

For w_{45}:

$$\boxed{y_4 \underset{w_{45}}{\rightarrow} x_5 \underset{\text{ReLU}}{\rightarrow} y_5 \underset{\text{MSE}}{\rightarrow} \text{Loss}}$$

By the **chain rule**:

$$\begin{aligned}\frac{\partial \text{Loss}}{\partial w_{45}} &= \frac{\partial \text{Loss}}{\partial y_5} \cdot \frac{\partial y_5}{\partial x_5} \cdot \frac{\partial x_5}{\partial w_{45}} \\ &= (y_5 - y_{\text{actual}}) \cdot \text{ReLU}'(x_5) \cdot y_4 \\ &= 10 \cdot 1 \cdot 0 \\ &= 0\end{aligned}$$

Update:

$$w_{45}^{\text{new}} = w_{45} - \eta \cdot \frac{\partial \text{Loss}}{\partial w_{45}} = -1 - 0.1 \cdot 0 = -1$$

---

2. Update Hidden Layer Weights (w_{13}, w_{23}):

For w_{13}:

$$\boxed{x_1 \underset{w_{13}}{\rightarrow} x_3 \underset{\text{ReLU}}{\rightarrow} y_3 \underset{w_{35}}{\rightarrow} x_5 \underset{\text{ReLU}}{\rightarrow} y_5 \underset{\text{MSE}}{\rightarrow} \text{Loss}}$$

By the **chain rule**:

$$\begin{aligned}\frac{\partial \text{Loss}}{\partial w_{13}} &= \frac{\partial \text{Loss}}{\partial y_5} \cdot \frac{\partial y_5}{\partial x_5} \cdot \frac{\partial x_5}{\partial y_3} \cdot \frac{\partial y_3}{\partial x_3} \cdot \frac{\partial x_3}{\partial w_{13}} \\ &= (y_5 - y_{\text{actual}}) \cdot \text{ReLU}'(x_5) \cdot w_{35} \cdot \text{ReLU}'(x_3) \cdot x_1 \\ &= 10 \cdot 1 \cdot 2 \cdot 1 \cdot 1 \\ &= 20\end{aligned}$$

Update:

$$w_{13}^{\text{new}} = w_{13} - \eta \cdot \frac{\partial \text{Loss}}{\partial w_{13}} = 2 - 0.1 \cdot 20 = 0$$

For w_{23}:

$$\boxed{x_2 \underset{w_{23}}{\rightarrow} x_3 \underset{\text{ReLU}}{\rightarrow} y_3 \underset{w_{35}}{\rightarrow} x_5 \underset{\text{ReLU}}{\rightarrow} y_5 \underset{\text{MSE}}{\rightarrow} \text{Loss}}$$

By the **chain rule**:

$$\begin{aligned}\frac{\partial \text{Loss}}{\partial w_{23}} &= \frac{\partial \text{Loss}}{\partial y_5} \cdot \frac{\partial y_5}{\partial x_5} \cdot \frac{\partial x_5}{\partial y_3} \cdot \frac{\partial y_3}{\partial x_3} \cdot \frac{\partial x_3}{\partial w_{23}} \\ &= (y_5 - y_{\text{actual}}) \cdot \text{ReLU}'(x_5) \cdot w_{35} \cdot \text{ReLU}'(x_3) \cdot x_2 \\ &= 10 \cdot 1 \cdot 2 \cdot 1 \cdot -1 \\ &= -20\end{aligned}$$

Update:

$$w_{23}^{\text{new}} = w_{23} - \eta \cdot \frac{\partial \text{Loss}}{\partial w_{23}} = -3 - 0.1 \cdot (-20) = -1$$

---

3. Update Hidden Layer Weights (w_{14}, w_{24}):

For w_{14}:

$$\boxed{x_1 \underset{w_{14}}{\rightarrow} x_4 \underset{\text{ReLU}}{\rightarrow} y_4 \underset{w_{45}}{\rightarrow} x_5 \underset{\text{ReLU}}{\rightarrow} y_5 \underset{\text{MSE}}{\rightarrow} \text{Loss}}$$

By the **chain rule**:

$$\begin{aligned}\frac{\partial \text{Loss}}{\partial w_{14}} &= \frac{\partial \text{Loss}}{\partial y_5} \cdot \frac{\partial y_5}{\partial x_5} \cdot \frac{\partial x_5}{\partial y_4} \cdot \frac{\partial y_4}{\partial x_4} \cdot \frac{\partial x_4}{\partial w_{14}} \\ &= (y_5 - y_{\text{actual}}) \cdot \text{ReLU}'(x_5) \cdot w_{45} \cdot \text{ReLU}'(x_4) \cdot x_1 \\ &= 10 \cdot 1 \cdot (-1) \cdot 0 \cdot 1 \\ &= 0\end{aligned}$$

Update:

$$w_{14}^{\text{new}} = w_{14} - \eta \cdot \frac{\partial \text{Loss}}{\partial w_{14}} = 1 - 0.1 \cdot 0 = 1$$

For w_{24}:

$$\boxed{x_2 \underset{w_{24}}{\rightarrow} x_4 \underset{\text{ReLU}}{\rightarrow} y_4 \underset{w_{45}}{\rightarrow} x_5 \underset{\text{ReLU}}{\rightarrow} y_5 \underset{\text{MSE}}{\rightarrow} \text{Loss}}$$

By the **chain rule**:

$$\begin{aligned}\frac{\partial \text{Loss}}{\partial w_{24}} &= \frac{\partial \text{Loss}}{\partial y_5} \cdot \frac{\partial y_5}{\partial x_5} \cdot \frac{\partial x_5}{\partial y_4} \cdot \frac{\partial y_4}{\partial x_4} \cdot \frac{\partial x_4}{\partial w_{24}} \\ &= (y_5 - y_{\text{actual}}) \cdot \text{ReLU}'(x_5) \cdot w_{45} \cdot \text{ReLU}'(x_4) \cdot x_2 \\ &= 10 \cdot 1 \cdot (-1) \cdot 0 \cdot (-1) \\ &= 0\end{aligned}$$

Update:

$$w_{24}^{\text{new}} = w_{24} - \eta \cdot \frac{\partial \text{Loss}}{\partial w_{24}} = 4 - 0.1 \cdot 0 = 4$$

Final Updated Weights

- $w_{13} = \boxed{0}$, $w_{23} = \boxed{-1}$
- $w_{14} = \boxed{1}$, $w_{24} = \boxed{4}$
- $w_{35} = \boxed{-3}$, $w_{45} = \boxed{-1}$

In [9]:

```python
#Here's the programming approach using pytorch
import torch
import torch.nn as nn
import torch.optim as optim

class SimpleNN(nn.Module):
    def __init__(self):
        super(SimpleNN, self).__init__()
        self.w13 = torch.tensor(2.0, requires_grad=True)
        self.w23 = torch.tensor(-3.0, requires_grad=True)
        self.w14 = torch.tensor(1.0, requires_grad=True)
        self.w24 = torch.tensor(4.0, requires_grad=True)
        self.w35 = torch.tensor(2.0, requires_grad=True)
        self.w45 = torch.tensor(-1.0, requires_grad=True)
    def forward(self, x1, x2):
        # Forward pass
        x3 = x1 * self.w13 + x2 * self.w23
        y3 = torch.relu(x3)

        x4 = x1 * self.w14 + x2 * self.w24
        y4 = torch.relu(x4)

        x5 = y3 * self.w35 + y4 * self.w45
        y5 = torch.relu(x5)

        return y5

model = SimpleNN()
# Inputs and target
x1 = torch.tensor(1.0)
x2 = torch.tensor(-1.0)
y_actual = torch.tensor(0.0)  # Actual value
eta = 0.1 # Learning rate
# Optimizer (Stochastic Gradient Descent)
optimizer = optim.SGD([model.w13, model.w23, model.w14, model.w24, model.w35, model.w

y5 = model(x1, x2) # Forward pass
loss = 0.5 * (y5 - y_actual) ** 2 # Compute the loss
loss.backward() # Back Propagate to compute gradients
optimizer.step() # Update weights

print("Updated Weights:")
print(f"w13: {model.w13.item():.1f}")
print(f"w23: {model.w23.item()}")
print(f"w14: {model.w14.item()}")
print(f"w24: {model.w24.item()}")
print(f"w35: {model.w35.item()}")
print(f"w45: {model.w45.item()}")
```

```
Updated Weights:
w13: -0.0
w23: -1.0
w14: 1.0
w24: 4.0
w35: -3.0
w45: -1.0
```

This document was created in Jupyter Notebook by Trần Minh Dương (tmd).

If you have any questions or notice any errors, feel free to reach out via Discord at @tmdhoctiengphap or @ICT-Supporters on the USTH Learning Support server.

Check out my GitHub repository for more projects: GalaxyAnnihilator/MachineLearning .