Non-Linear Activation Functions in Neural Networks

Complete Notes with Diagrams – 2025 Edition

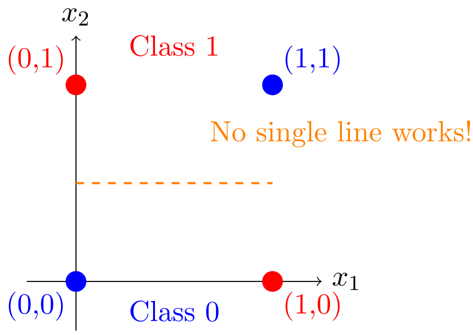
From Your Handwritten Notes

1 Main Problem: Linear Decision Boundaries Are Insufficient

Linear models (single perceptron, logistic regression) can only create straight-line decision boundaries.

XOR Truth Table – Not Linearly Separable

x_1	x_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

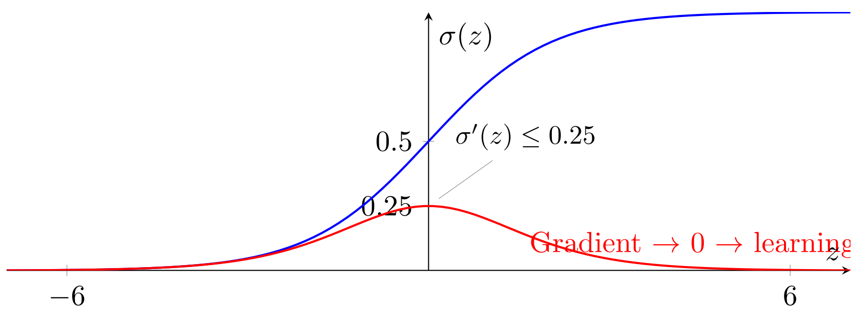


Conclusion: We need **non-linear decision boundaries**. **Solution:** Multi-layer neural networks (MLP) + **non-linear activation functions**.

2 Why Sigmoid Activation Is Problematic

$$\sigma(z) = \frac{1}{1 + e^{-z}} \qquad \sigma'(z) = \sigma(z)(1 - \sigma(z)) \leq 0.25$$

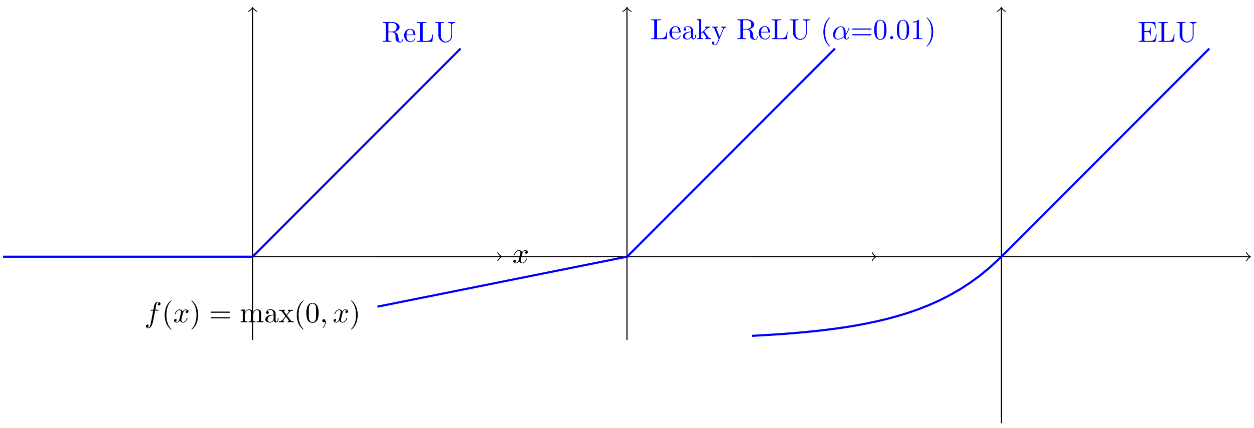
Sigmoid and its Derivative (Vanishing Gradient Problem)



2.1 Four Major Problems of Sigmoid

1. **Vanishing Gradient** → deep networks barely learn
2. **Not zero-centered** ($\sigma(z) \in (0, 1)$) → gradients always positive → zigzag updates
3. **Expensive** exponential computation
4. **Saturation** → when $|z|$ large, $\sigma'(z) \approx 0$ → neurons die

3 Better Alternatives → ReLU and Its Variants



1. **ReLU** $\quad f(x) = \max(0, x)$
    - Extremely fast (just threshold)
    - No saturation for $x > 0$ → no vanishing gradient
    - Accelerates SGD convergence
    - Induces sparsity
    - **Disadvantage:** Dying ReLU problem
2. **Leaky ReLU** $\quad f(x) = \begin{cases} x & x > 0 \\ \alpha x & x \leq 0 \end{cases} \quad (\alpha \approx 0.01)$
    - Solves dying ReLU (small negative slope)
3. **Parametric ReLU (PReLU)** → α is learned
4. **ELU** $\quad f(x) = \begin{cases} x & x > 0 \\ \alpha(e^x - 1) & x \leq 0 \end{cases}$
    - Zero-centered → faster convergence
    - Smooth everywhere
    - Slightly more expensive
5. **Swish / SiLU** $\quad f(x) = x \cdot \sigma(\beta x) \quad (\beta = 1)$
    - Smooth, non-monotonic
    - State-of-the-art in many deep networks

4 Key Insight from Backpropagation Chain Rule

For two consecutive layers:

$$\frac{\partial L}{\partial w_1} \propto \underbrace{\frac{\partial L}{\partial y} \frac{\partial y}{\partial z_2}}_{\text{layer 2}} \underbrace{\frac{\partial z_2}{\partial h_1} \frac{\partial h_1}{\partial z_1}}_{\text{activation derivative}} \underbrace{\frac{\partial z_1}{\partial w_1}}_{\text{layer 1}}$$

⇒ If any activation derivative ≈ 0 → **entire gradient vanishes!**

5 Practical Recommendations – 2025 Best Practices

What You Should Actually Use in 2025

- **Default choice (Transformers, LLMs, modern CNNs): GELU** or **Swish/SiLU**
- **Simple/shallow networks:** ReLU is still perfectly fine
- **Dying ReLU observed?** $\rightarrow$ switch to Leaky ReLU or PReLU
- **Avoid sigmoid and tanh in hidden layers** (except legacy RNNs or binary output)

6 Initialization & Training Tips

- Use **He/Kaiming initialization** with ReLU family
- Use **Xavier/Glorot initialization** only with sigmoid/tanh
- **BatchNorm / LayerNorm** dramatically helps activation distributions
- Modern optimizers (Adam, AdamW, Lion) + learning rate scheduling work excellently with ReLU-family activations

"Just don't use sigmoid in hidden layers in 2025!"