Machine Learning with Single and Multi-Neuron Models: Gradient Descent and Activation Functions

SIW1 Aibek Zhazykbek

October 3, 2025

Abstract

This paper explores the training of simple machine learning models using gradient descent. We begin with a single-neuron linear model, then extend to a two-neuron network with different activation functions (ReLU, Sigmoid, and Tanh). The training behavior, mathematical background, and experimental results are illustrated with comparative graphs between the first and best iterations. All experiments and code are available at: `https://github.com/Adambol03/CI-SIW1-NN-ML/blob/main/SIW1%20AIBEK%20ZHAZYKBEK.ipynb`.

1 Introduction

Artificial Neural Networks (ANNs) form the backbone of modern machine learning. Even the simplest neuron model can approximate linear relationships using gradient descent. Adding more neurons with non-linear activation functions allows us to capture more complex patterns. This work demonstrates these concepts by comparing single-neuron and multi-neuron models.

2 Mathematical Background

2.1 Single Neuron Model

Prediction:

$$\hat{y} = kx + b$$

where k is the weight and b is the bias.

2.2 Two-Neuron Model

For a two-neuron hidden layer:

$$a_1 = f(w_1 x + b_1), \quad a_2 = f(w_2 x + b_2),$$

$$\hat{y} = f(w_3 a_1 + w_4 a_2 + b_3),$$

where f is the activation function.

2.3 Activation Functions

$$\text{ReLU: } f(z) = \max(0, z), \quad \text{Sigmoid: } f(z) = \frac{1}{1 + e^{-z}}, \quad \text{Tanh: } f(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

2.4 Loss Function

Mean Squared Error (MSE):

$$L = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

2.5 Gradient Descent

Parameter update:

$$\theta \leftarrow \theta - \eta \frac{\partial L}{\partial \theta}$$

where η is the learning rate.

3 Single Neuron Experiments

The single neuron was trained using gradient descent with ReLU activation.

3.1 Results

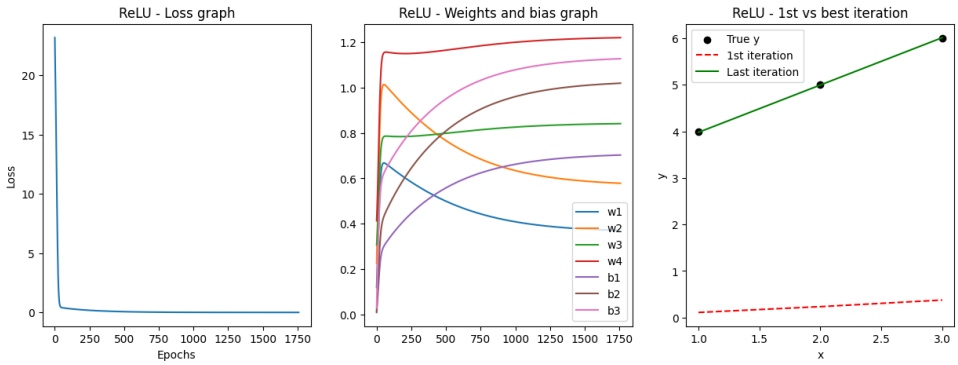

Figure 1: Single neuron training with ReLU. Best result after optimization.

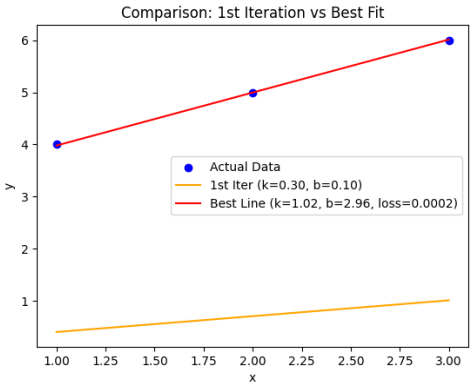

Figure 2: Comparison of first iteration and best iteration for a single neuron using ReLU.

4 Two-Neuron Model Experiments

The network was extended to two hidden neurons. This provided more flexibility and improved nonlinear approximation.

4.1 ReLU Results

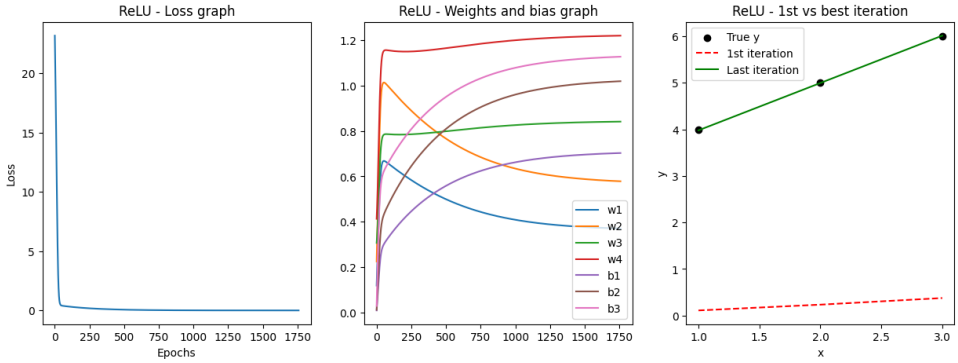

Figure 3: Best training result with two neurons using ReLU activation.

4.2 Sigmoid Results

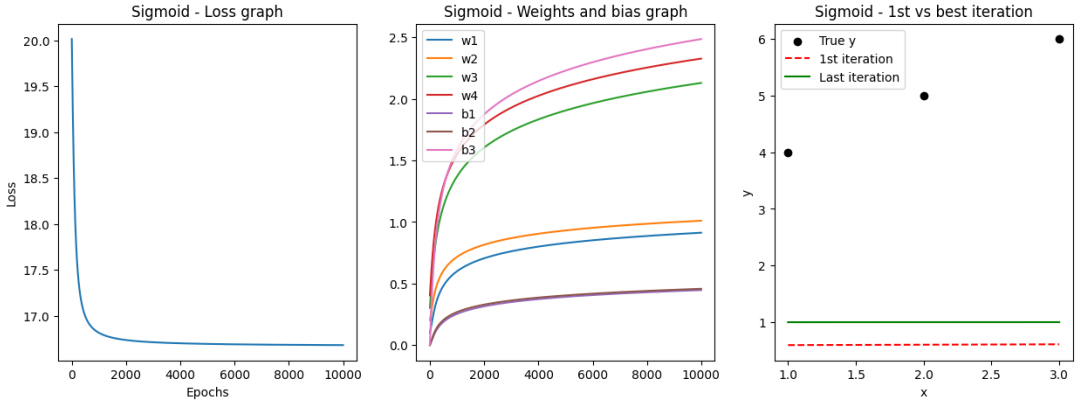

Figure 4: Best training result with two neurons using Sigmoid activation.

4.3 Tanh Results

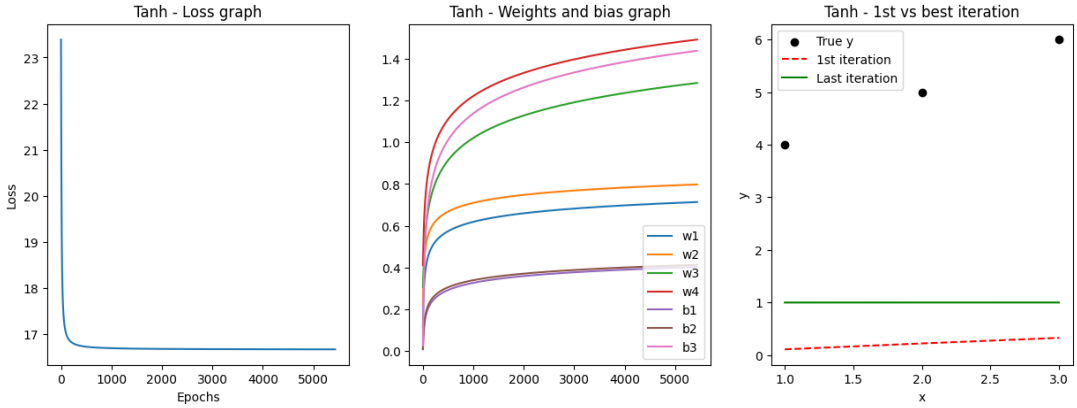

Figure 5: Best training result with two neurons using Tanh activation.

4.4 First vs Best Iteration

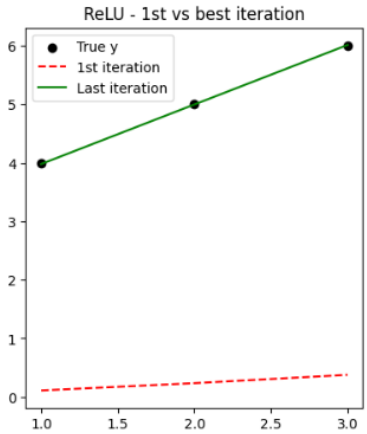

Figure 6: Comparison of first and best iterations for two neurons with ReLU activation.

5 Discussion

- A single neuron is sufficient for linear patterns.
- Adding a second neuron improves flexibility and captures nonlinear relationships.
- Sigmoid converges smoothly but can saturate.
- ReLU trains faster but may stall in some cases.
- Tanh performs better than Sigmoid due to wider gradient range.

6 Conclusion

This paper presented experiments with single and two-neuron models trained using gradient descent. Results showed that while a single neuron can approximate linear functions, adding more neurons and nonlinear activations significantly improves learning capability. Future work may involve deeper architectures and advanced optimization algorithms.

References

1. Lecture 2-3
2. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
3. Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.