Approximating the Runge Function and Its Derivative Using a Neural Network

Kai-Hung Cheng

2025/9/24

1 Introduction

The purpose of this experiment is to approximate both the Runge function and its derivative using a neural network. The Runge function is defined as:

$$f(x) = \frac{1}{1 + 25x^2}, \quad x \in [-1, 1].$$

Unlike the previous assignment, which focused only on approximating $f(x)$, this task extends the goal to learning both $f(x)$ and its derivative $f'(x)$ simultaneously through a shared neural network.

2 Method

2.1 Data Generation

A total of 300 data points were uniformly sampled from the interval $[-1, 1]$. For each x_i, we computed:

$$y_i = f(x_i), \qquad y_i' = f'(x_i) = \frac{-50x_i}{(1 + 25x_i^2)^2}.$$

The dataset was shuffled and divided equally into training, validation, and test subsets.

2.2 Neural Network Architecture

The neural network used is a fully connected multilayer perceptron (MLP) defined as:

$$x \to \text{Linear}(1, 64) \to \tanh() \to \text{Linear}(64, 64) \to \tanh() \to \text{Linear}(64, 1).$$

The tanh activation function was selected for its smooth and differentiable properties, which make it suitable for approximating continuous and differentiable target functions like the Runge function. The model consists of two hidden layers, each containing 64 neurons.

2.3 Loss Function

The total loss combines two components: the error in predicting the function itself and the error in predicting its derivative. It is defined as:

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\left[(f_\theta(x_i) - f(x_i))^2 + \left(\frac{\partial f_\theta(x_i)}{\partial x_i} - f'(x_i)\right)^2\right],$$

where $f_\theta(x)$ is the neural network output and $\frac{\partial f_\theta(x)}{\partial x}$ is obtained via automatic differentiation in PyTorch.

2.4 Training Procedure

The model was trained for 10,000 epochs using stochastic gradient descent (SGD) with a learning rate of 0.01. Both function and derivative losses were computed at each epoch for the training and validation sets. Gradients were propagated through the entire computational graph to allow the network to learn consistent derivative behavior. Loss curves were recorded to monitor convergence.

3 Results and Discussion

Figure 1 shows the neural network's performance in approximating the Runge function and its derivative. The top plot compares the true function $f(x)$ and the predicted $\hat{f}(x)$, where the two curves overlap almost perfectly, indicating high accuracy. The bottom plot presents the training and validation loss curves, which both decrease rapidly and converge smoothly to near-zero values without divergence, suggesting stable learning and no overfitting.

The test errors are summarized as follows:

$$\text{MSE}(f) = 3.0 \times 10^{-6}, \quad \text{MSE}(f') = 9.8 \times 10^{-4}.$$

The network achieved extremely low mean squared errors for both the function and its derivative, demonstrating that the combined loss formulation effectively guided the network to capture both global and local behavior of the target function.

3.1 Analysis

Including the derivative term in the loss function improved the smoothness and accuracy of the learned function. The derivative constraint ensured that the network did not only fit isolated points but also captured the slope information between them. The convergence pattern in Figure 1 shows that both losses plateaued at similar magnitudes, implying balanced learning between $f(x)$ and $f'(x)$.

4 Conclusion

This experiment demonstrates that a two-layer tanh neural network can accurately approximate both a nonlinear function and its derivative when trained with a combined loss function. The final model achieved near-perfect agreement with the Runge function, and

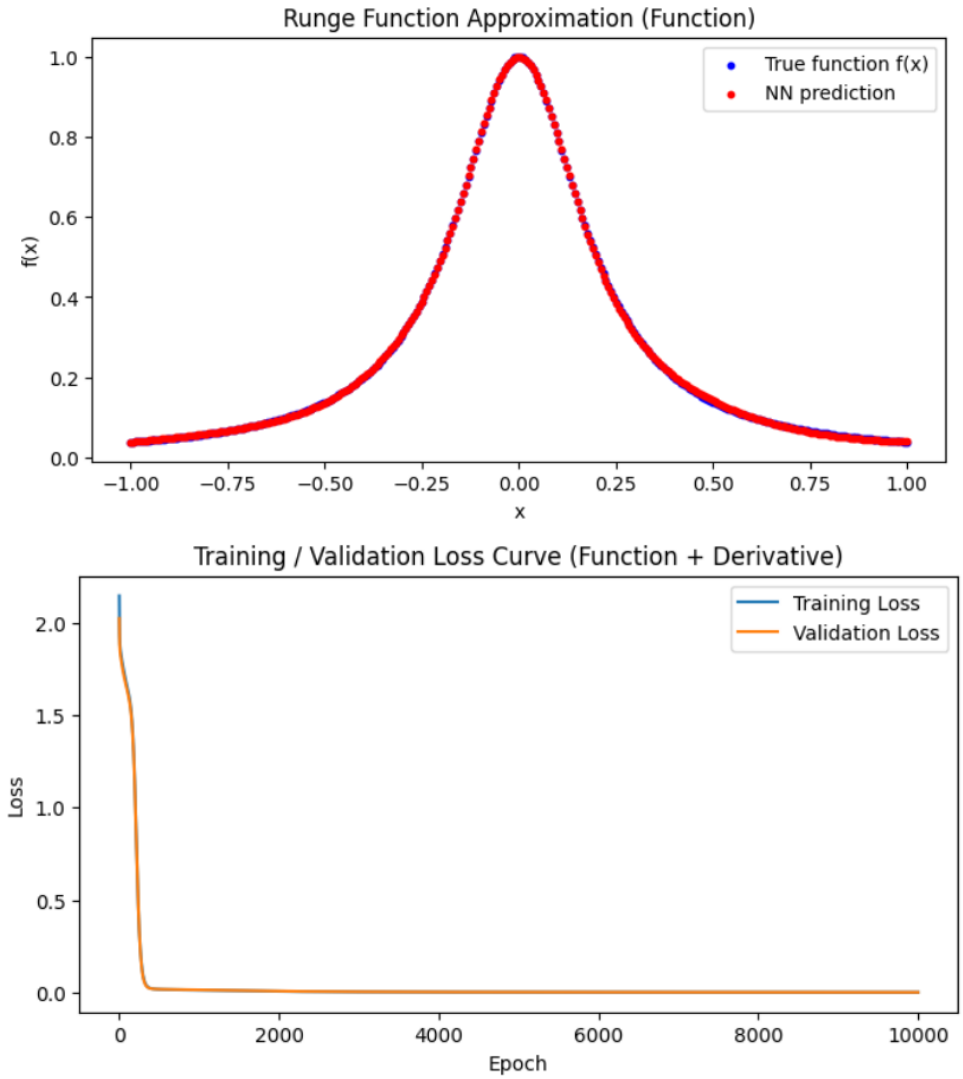

Figure 1: (Top) Neural network approximation of the Runge function. (Bottom) Training and validation loss curves combining function and derivative errors.

its derivative was learned with high precision. The results highlight the effectiveness of incorporating derivative information into the loss, which enhances smoothness, stability, and generalization.