ML HW3

Date: 09/17/2025
Week: 3
Author: Alvin B. Lin
Student ID: 112652040

Problem 1: Explanation to The Lemma

1. Reading and Explaining Lemmas:

   Your task is to read Ryck et al., On the approximation of function by tanh neural networks[1]. Focus on **Lemma 3.1** and **Lemma 3.2**.

Setup: Before stating the explanation, we define the following terms:

Definition 1: L^p Space

The **Lebesgue space** is defined as:

$$\mathrm{L}^p(\Omega) = \left\{ f : \Omega \to \overline{\mathbb{R}} : \int_\Omega |f|^p \,\mathrm{d}\mu < +\infty \right\},$$

where $\Omega \subseteq \mathbb{R}^d,\ p,\ d \in \mathbb{Z}^+,\ \mu(\cdot)$ be the **Lebesgue measure**.

$$\mathrm{L}^\infty(\Omega) = \left\{ f : \Omega \to \overline{\mathbb{R}} :\ \mu\{x \in \Omega :\ |f(x)| = \infty\} = 0 \right\},$$

where $\Omega \subseteq \mathbb{R}^d,\ d \in \mathbb{Z}^+,\ \mu(\cdot)$ be the **Lebesgue measure**.

Definition 2: L^p Norm

Given $d \in \mathbb{Z}^+$, function $f : \Omega \subseteq \mathbb{R}^d \to \overline{\mathbb{R}}$. For $p \in \mathbb{Z}^+$, the norm is defined as:

$$\|f\|_{\mathrm{L}^p} = \left(\int_\Omega |f|^p \,\mathrm{d}\mu \right)^{\frac{1}{p}},$$

for $p = +\infty$, the norm is defined as:

$$\|f\|_{\mathrm{L}^\infty} = \sup_{x \in \Omega} |f(x)|$$

Definition 3: Sobolev Space

Let $d \in \mathbb{Z}^+,\ p \in \mathbb{Z}^+ \cup \{+\infty\}$ and let $\Omega \subseteq \mathbb{R}^d$ be open, $\mathrm{L}^p(\Omega)$ be the Lebesgue space. For $k \in \mathbb{N}$, we define **Sobolev space** as:

$$W^{k,p}(\Omega) = \left\{ f \in \mathrm{L}^p(\Omega) :\ D^\alpha f \in \mathrm{L}^p(\Omega),\ \forall \alpha \in \mathbb{N}^d,\ |\alpha| \le k \right\},$$

where $\alpha = (\alpha_1, \ldots, \alpha_d),\ D^\alpha = \partial_1^{\alpha_1} \cdots \partial_d^{\alpha_d}$.

Definition 4: Seminorm on Sobolev Space

For $p \in \mathbb{Z}^+$, the seminorm of f on $W^{k,p}(\Omega)$ is

$$|f|_{W^{m,p}(\Omega)} = \left(\sum_{|\alpha|=m} \left\| D^\alpha f \right\|^p_{\mathrm{L}^p(\Omega)} \right)^{1/p}, \text{ for } m = 0, \ldots, k.$$

For $p = +\infty$, we define the seminorm:

$$|f|_{W^{k,\infty}(\Omega)} = \max_{|\alpha|=m} \left\| D^\alpha f \right\|_{\mathrm{L}^\infty(\Omega)}, \text{ for } m = 0, \ldots, k.$$

Definition 5: Norm on Sobolev Space

Based on **definition 3**, we define norm of f for $p \in \mathbb{Z}^+$:

$$\|f\|_{W^{k,p}(\Omega)} = \left(\sum_{m=0}^{k} |f|^p_{W^{m,p}(\Omega)} \right)^{1/p},$$

for $p = +\infty$, we define as

$$\|f\|_{W^{k,\infty}(\Omega)} = \max_{0 \leq m \leq k} |f|_{W^{m,\infty}(\Omega)}$$

In **Section 2.3. Neural network**, it is mentioned in this paper that: Ψ_Θ is a neural network for which activation function is $\tanh x$, where Θ is the set of parameters (weight matrices W_k, biases b_k).

Neural networks Ψ_Θ, in this paper, can be categorised into 2 classes:

- *Shallow neural network*: For neural networks who have only 1 hidden layer.
- *Deep neural network*: For neural networks who have more than 1 hidden layers.

In this paper, for $p \in \mathbb{Z}^+,\ M \in \mathbb{R}^+$, the monomials $f_p : [-M,\ M] \to \mathbb{R}$ is defined as

$$f_p(y) := y^p; \quad \text{for instance, } f_3(y) = y^3,\ f_3(2) = 2^3 = 8.$$

Lemma 3.1

Let $k \in \mathbb{N}$ and $s \in 2\mathbb{Z}^+ - 1$. Then it holds that for all $\varepsilon \in \mathbb{R}^+$, there exists a *shallow* tanh neural network $\Psi_{s,\varepsilon} : [-M,\ M] \to \mathbb{R}^{\frac{s+1}{2}}$ of width $\frac{s+1}{2}$ such that

$$\max_{\substack{p \leq s,\\ p \text{ odd}}} \left\| f_p - (\Psi_{s,\varepsilon})_{\frac{p+1}{2}} \right\|_{W^{k,\infty}} \leq \varepsilon.$$

Moreover, the weights of $\Psi_{s,\varepsilon}$ scale as $\mathcal{O}\left(\varepsilon^{-\frac{s}{2}} (2(s+2)\sqrt{2M})^{s(s+3)} \right)$ for small ε and large s.

What to notice?

- $\Psi_{s,\varepsilon}$ is a **single-hidden-layer** neural network with $\frac{s+1}{2}$ neurons.
- The lemma only holds for closed interval $[-M,\ M]$, instead of $\mathbb{R}$.
- The norm used is **Sobolev space norm**, which is stronger than L^∞ norm.
- Taking **max** ensures for all odd $p \leq s$ the inequality holds.

What does the lemma tell us?

The lemma tells us that under a compact interval $[-M,\ M]$, $s \in 2\mathbb{Z}^+ - 1$, error $\varepsilon \in \mathbb{R}^+$, we can always find a **one-layer-$\frac{s+1}{2}$-neuron neural network** $\Psi_{s,\varepsilon}$ to approximate all the **odd** term monomial up to order s within the error ε in the sense of **Sobolev norm**, which also ensures the **derivative** $\mathrm{D}_y^n \Psi_{s,\varepsilon}$ is close to the **derivative of the object function** $\mathrm{D}_y^n y^p$ for $0 \leq n \leq k$.

Moreover, we can also predict the **growth speed of weights**, they grow as fast as $\varepsilon^{-\frac{s}{2}}(2(s+2)\sqrt{2M})^{s(s+3)}$ for large number of neurons s and small enough error ε.

Excellent, let's move on to the **lemma 3.2**, which is very similar to **lemma 3.1**.

> **Lemma 3.2**
>
> Let $k \in \mathbb{N}$ and $s \in 2\mathbb{Z}^+ - 1$. Then it holds that for all $\varepsilon \in \mathbb{R}^+$, there exists a *shallow* tanh neural network $\psi_{s,\varepsilon} : [-M,\ M] \to \mathbb{R}^s$ of width $\frac{3(s+1)}{2}$ such that
>
> $$\max_{p \leq s} \left\| f_p - (\psi_{s,\varepsilon})_p \right\|_{W^{k,\infty}} \leq \varepsilon.$$
>
> Moreover, the weights of $\psi_{s,\varepsilon}$ scale as $\mathcal{O}\left(\varepsilon^{-\frac{s}{2}}((s+2)\sqrt{M})^{\frac{3s(s+3)}{2}}\right)$ for small ε and large s.

What does the lemma tell us?

Similar to **lemma 3.1**, under a compact interval $[-M,\ M]$, $s \in 2\mathbb{Z}^+ - 1$, error $\varepsilon \in \mathbb{R}^+$, we can always find a **one-layer-$\frac{3(s+1)}{2}$ neuron-neural network** $\psi_{s,\varepsilon}$ to approximate **all** the monomial up to order s within the error ε in the sense of **Sobolev norm**, which also ensures the **derivative** $\mathrm{D}_y^n \psi_{s,\varepsilon}$ is close to the **derivative of the object function** $\mathrm{D}_y^n y^p$ for $0 \leq n \leq k$.

Moreover, we can also predict the **growth speed of weights**, they grow as fast as $\varepsilon^{-\frac{s}{2}}((s+2)\sqrt{M})^{\frac{3s(s+3)}{2}}$ for large number of neurons s and small enough error ε.

What is it different from lemma 3.1?

Lemma 3.2 is an enhanced version of **lemma 3.1**, it shows that using $s+1$ more neurons can let us approximate the even monomials. Overall, to approximate an order n polynomial, we need $\frac{3(n+1)}{2}$ neurons.

How does it work in practice?

We then look at an example of how it work **theoretically**.
Example: Approximate $g(x) = 5x^5 - 3x^3 + x^2 - x$ on $[-2,\ 2]$ within error $= 10^{-4}$.

Step 1: Notice that g is a degree 5 polynomial, we choose $s = 5$, also $k = 1$.

Step 2: Our neural network will have $\frac{3(5+1)}{2} = 9$ neurons in the hidden layer.

Step 3: Since for all $p \leq 5$, we have

$$\left\| f_p - (\psi_{s,\varepsilon})_p \right\|_{W^{k,\infty}} \leq \varepsilon;$$

by the **triangular inequality**,

$$\begin{aligned}
\left\| g - \psi_{s,\varepsilon} \right\|_{\mathrm{L}^\infty} \leq \left\| g - \psi_{s,\varepsilon} \right\|_{W^{k,\infty}} &\leq 5\left\| f_5 - (\psi_{s,\epsilon})_5 \right\|_{W^{k,\infty}} + 3\left\| f_3 - (\psi_{s,\epsilon})_3 \right\|_{W^{k,\infty}} \\
&\quad + \left\| f_2 - (\psi_{s,\epsilon})_2 \right\|_{W^{k,\infty}} + \left\| f_1 - (\psi_{s,\epsilon})_1 \right\|_{W^{k,\infty}} \\
&\leq (5 + 3 + 1 + 1)\varepsilon \\
&= 10\varepsilon \\
&\leq 10^{-4}
\end{aligned}$$

we choose $\varepsilon = 10^{-5}$.

Step 4: By setting $k = 1,\ s = 5,\ \varepsilon = 10^{-5}$, **lemma 3.2** renders the theoretical **existence of neural network** $\psi_{5,10^{-5}}$ that approximate $g(x)$ on $[-2,\ 2]$ with error no more than 10^{-4}.

Moreover, since we set $k = 1$, the error between their derivatives will be less than 10^{-4}.

$$\left\| g' - (\psi_{s,\varepsilon})' \right\|_{\mathrm{L}^\infty} \leq \left\| g - (\psi_{s,\varepsilon}) \right\|_{W^{k,\infty}} \leq 10^{-4}$$

This is the reason why we say **Sobolev norm** is stronger than $\mathbf{L}^\infty$ **norm** and use **Sobolev norm** instead of $\mathbf{L}^\infty$ **norm**.

Proof Insights:

Taylor Theorem states for any given **smooth** function at $x = a$, we have:

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(a)}{k!}(x-a)^k + \frac{f^{(n+1)}(\xi(x,\ a))}{(n+1)!}(x-a)^{n+1},$$

where $n \in \mathbb{Z}^+,\ \xi(x,\ a)$ lies between x and a.

We can apply this to the **finite difference** method, we get **Stirling formula**:

$$\begin{aligned}
P_n(x) = P_{2m+1}(x) = f[x_0] &+ \frac{sh}{2}(f[x_{-1}, x_0] + f[x_0, x_1]) + s^2h^2 f[x_{-1}, x_0, x_1] \\
&+ \frac{s(s^2-1)h^3}{2}(f[x_{-2}, x_{-1}, x_0, x_1] + f[x_{-1,x_0,x_1,x_2}]) + \cdots \\
&+ s^2(s^2-1)(s^2-4)(s^2-(m-1)^2)h^{2m} f[x_{-m}, \ldots, x_m] \\
&+ \frac{s(s^2-1)\cdots(s^2-m^2)h^{2m+1}}{2}(f[x_{-m-1}, \ldots x_m] + f[x_{-m}, \ldots, x_{m+1}]),
\end{aligned}$$

for $s = 1,\ h$ the step size, for P_{2m}, we wipe out the last term.

For the term $f[x_l, \ldots, x_k]$, we have the recurrence formula [2]:

$$f[x_l, \ldots, x_k] = \frac{f[x_{l+1}, \ldots, x_k] - f[x_l, \ldots, x_{k-1}]}{x_k - x_l}; \quad f[x_k] = f(x_k)$$

(**Note:** This is just the insight of the centre difference method, which is not quite the same as the one used in the paper.)
This is my brief explanation of **lemma 3.1** and **lemma 3.2**, hope you find it easy to read.

Problem 2: Runge Function Derivatives

1. Use the same code from Assignment 2 - programming assignment 1 to calculate the error in approximating the derivative of the given function.

2. In this assignment, you will use a neural network to approximate both the Runge function and its derivative. Your task is to train a neural network that approximates:

   a. The function $f(x)$ itself.
   b. The derivative $f'(x)$.

   You should define a loss function consisting of two components:

   1) **Function loss**: the error between the predicted $f(x)$ and the true $f(x)$.
   2) **Derivative loss**: the error between the predicted $f'(x)$ and the true $f'(x)$.

   Write a short report (1–2 pages) explaining method, results, and discussion including:

   - Plot the true function and the neural network prediction together.
   - Show the training/validation loss curves.
   - Compute and report errors (MSE or max error).

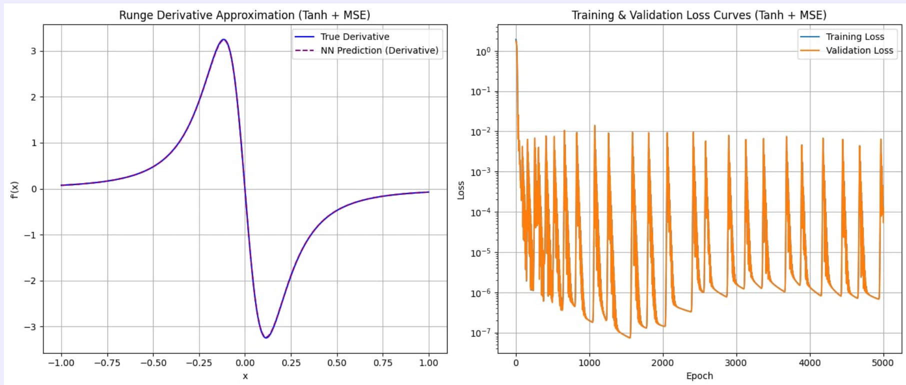
Sol: We first setup the neural network as before:

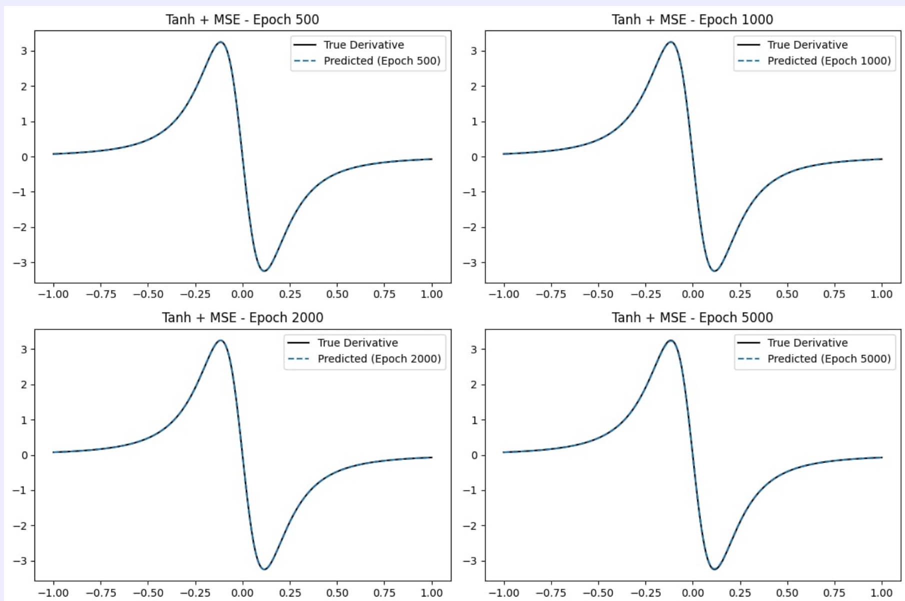
- Training set, validation set and testing set are selected uniformly on $[-1,\ 1]$ with counts $100,\ 10$ and 40, respectively.
- The neural network has 2 layers and each has 50 neurons.
- We have 3 methods to comparison:
  - $\tanh x$ as activation function & **MSE** for loss function.
  - $\tanh x$ as activation function & **sup-norm** for loss function.
  - $\cos x$ as activation function & **MSE** for loss function.
- Optimiser is chosen as **Adam** and the `epoch` is set 5000.
- Objective function is given $f(x) = \dfrac{1}{1+25x^2}$.
- Objective derivative is $f'(x) = \dfrac{-50x}{(1+25x^2)^2}$.

Be clear that we only measure the error between Runge function and the hypothesis function, instead of training the derivatives again with same manner done in `assignment 2`.

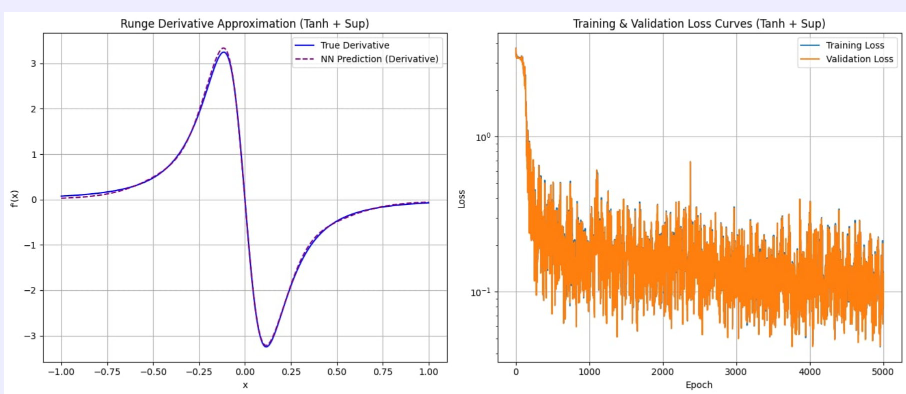
Brief results:

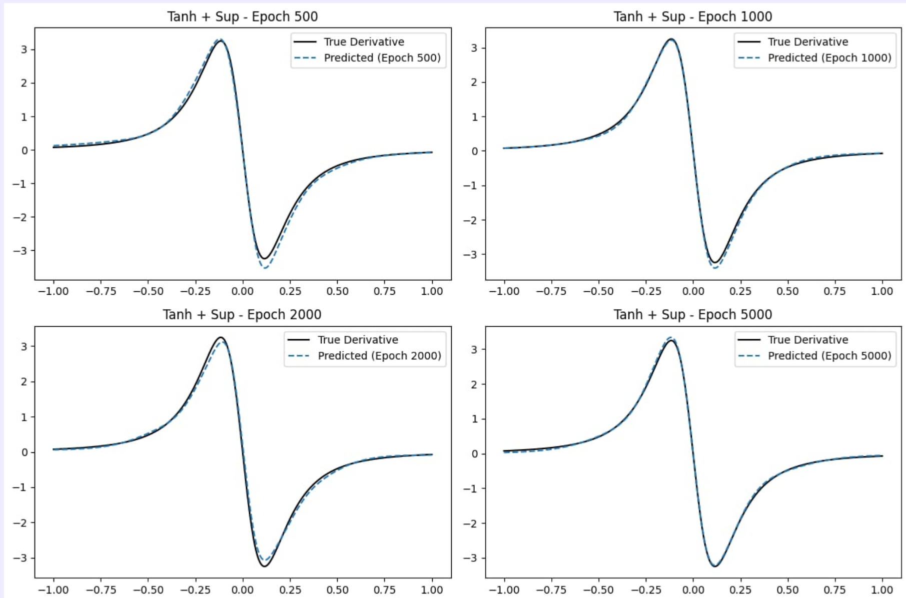
1. Result of $\tanh x$ & **MSE** method:



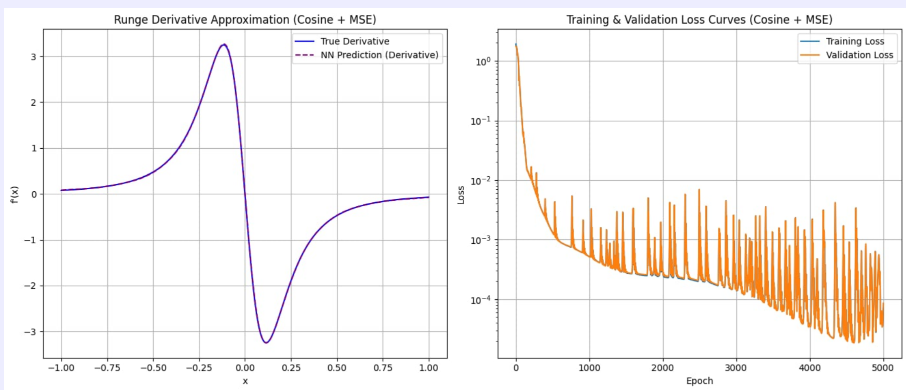


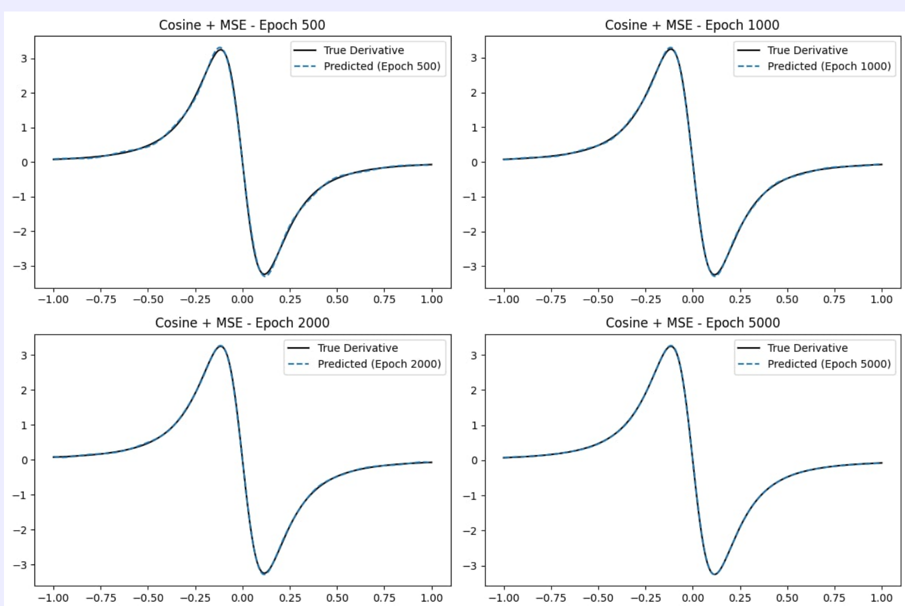
2. Result of $\tanh x$ & **sup-norm** method:





3. Result of $\cos x$ & **MSE** method:





Validation Loss Result Overview:

Function	Method	tanh + MSE	tanh + sup	cos + MSE
f(x)	Test MSE	0.000000	0.000000	0.000006
	Test Max Absolute Error	0.000489	0.000564	0.005360
f'(x)	Test MSE	0.000054	0.001418	0.000086
	Test Max Absolute Error	0.019007	0.097872	0.036319

We can see that using $\tanh x$ as activation function with loss **MSE** is the best option of the three for approximating the Runge function, it performs quite well on both original function and its derivative.

References

[1] Tim De Ryck, Samuel Lanthaler, and Siddhartha Mishra. On the approximation of functions by tanh neural networks. *Neural Networks*, 143:732–750, 2021.

[2] Richard L. Burden, J. Douglas Faires, and Annette M. Burden. *Numerical Analysis*. Cengage Learning, 10 edition, 2015.