Week 2 Worksheet — Simple Nonlinear Node

Health Mini-Project (Single Hidden Layer, 2 Hidden Nodes)

Name: ______________________________ **Date:** ______________________

Goal: Using **two inputs only** (Height and Weight), design a **single-hidden-layer** model with two hidden nodes to classify each person as Healthy or Unhealthy. You will **define the hidden-node functions** and **combine** them linearly at the output.

Important: There is no single correct answer. Different students may create different, logical rules.

Dataset (Inputs)

Use **IDs 1–10** to create your rule ("training") and **IDs 11–12** to test it. (Same individuals as Week 1, but we will use **Height** and **Weight** only.)

ID	Height (in)	Weight (lb)	Your Score	Your Label
1	70	159		
2	65	187		
3	72	198		
4	63	121		
5	68	150		
6	67	209		
7	71	172		
8	62	181		
9	69	146		
10	73	231		
11	64	128		
12	70	154		

Your Task:

1. **Normalize** Height and Weight to the range 0–1 using the **training set** (IDs 1–10):

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}.$$

Denote the normalized inputs as $x_1 = \text{Height}_{\text{norm}}$ and $x_2 = \text{Weight}_{\text{norm}}$, so that $0 < x_1 < 1$ and $0 < x_2 < 1$.

2. **Define two hidden nodes (fixed functions):**

$$\text{Node}_{11}: \; h_1(x_1) = -x_1 + 1 \qquad \text{and} \qquad \text{Node}_{21}: \; h_2(x_2) = x_2.$$

   (Intuition: h_1 decreases as Height increases $\Rightarrow$ "denominator" effect; h_2 increases with Weight $\Rightarrow$ "numerator" effect.)

3. **Combine the hidden nodes linearly at the output:**

$$\text{Score} = a_1 \, h_1(x_1) + a_2 \, h_2(x_2),$$

   where $a_1, a_2 \geq 0$ and $a_1 + a_2 = 1$.

   - Choose a_1 and a_2 to reflect your belief about the relative importance of Height (via h_1) and Weight (via h_2).

4. **Choose a threshold T.** Example: $T = 0.50$.

$$\text{If Score} < T \Rightarrow \text{Healthy}, \quad \text{else Unhealthy.}$$

5. **Apply your rule to IDs 1–10.** Fill in "Your Score" and "Your Label" columns.

6. **Test your rule on IDs 11–12.**

Reflection:

- Which hidden node contributed more to your Score, h_1 or h_2? Why?
- How did you decide (a_1, a_2) and the threshold T?
- Compare this nonlinear design to Week 1's linear rule with four inputs. What changed in your intuition about the boundary?
- Concept check: In what sense does $h_1(x_1) = -x_1 + 1$ play a "denominator-like" role when combined with $h_2(x_2) = x_2$?

Hint: Try a few different (a_1, a_2) pairs (e.g., $[0.2, 0.8], [0.5, 0.5], [0.8, 0.2]$) and see how predictions change.

Challenge (Optional): Extend Your Nonlinear Design

You designed fixed hidden-node functions. Now explore variants and tuning:

What to try

1. **Alternative nonlinearities.** Replace h_1 or h_2 by a different shape (e.g., $h_1(x) = 1 - x^\gamma$ for some $\gamma > 0$, or $h_2(x) = \sqrt{x}$). Keep $a_1 + a_2 = 1$.
2. **Grid search.** On IDs 1–10, search (a_1, a_2) on a grid (step 0.1) and T on a grid (e.g., 0.30 to 0.70 by 0.05) to maximize training accuracy.
3. **Report.** Give the best (a_1, a_2, T), the training accuracy, and test predictions (IDs 11–12). Briefly explain the effect of your nonlinearity shape on decisions.

Concept Check: Linearity vs. Nonlinearity

Before we move on to nonlinear nodes, recall what it means for a function to be **linear.**

A function f is linear if it satisfies **both** properties:

$$\begin{aligned} &(1)\ \text{Additivity:} \quad f(x_1 + x_2) = f(x_1) + f(x_2) \\ &(2)\ \text{Homogeneity:} \quad f(cx) = c\, f(x) \end{aligned}$$

If either property fails, the function is **nonlinear.**

Examples:

Function $f(x)$	Additivity	Homogeneity	Linear?
$f(x) = 2x$	✓	✓	Yes
$f(x) = x^2$	$\times$	$\times$	No
$f(x) = x + 1$	$\times$	$\times$	No
$f(x) = \sin(x)$	$\times$	$\times$	No

Try it yourself:

1. Compute $f(3+4)$ and compare with $f(3) + f(4)$ for each function above. Which ones are equal?
2. Compute $f(2x)$ and compare with $2f(x)$. Which functions change proportionally?
3. Based on your results, explain why adding even a small constant (like $+1$) already breaks linearity.

Key idea: Adding or stacking linear functions without a nonlinear transformation still produces another linear function. To create new shapes (e.g., curves or denominators), a nonlinear operation must appear somewhere in the model.