AI Lab1: 星际导航系统的神经元构建

目标: 修复“旅行者号”飞船的自动导航AI核心。

1. 背景故事

你是“旅行者号”飞船上的实习程序员。飞船在穿越小行星带时，导航AI的底层逻辑被宇宙射线损坏了。

系统现在无法判断飞船是否应该进行闪避。你的任务是重写神经网络的核心计算层。

为了适应宇宙中的不稳定能量，我们需要实现一种特殊的**“Nova 神经元”**，它使用的不是普通的数学公式，而是专门处理波动数据的算法。

2. 核心数据结构

我们已经为你定义好了神经元的结构（在 `star_math.h` 中），虽然你不需要定义它，但你需要知道它什么样：

```
// 这是一个简化版的神经元结构
typedef struct {
    double weights[5]; // 权重数组，代表5个传感器对输入的重视程度
    double bias;       // 偏置值，代表神经元的灵敏度门槛
} Neuron;
```

3. 提供的数学工具 (API)

在 `star_math.h` 中，我们预先封装好了一些工具函数，你可以直接在 `main.c` 中使用它们来辅助开发或调试。

3.1 常量定义

- `INPUT_SIZE`: 定义了传感器输入的数量，固定为 **5**。
    - *用法*: 在遍历数组时，建议使用 `i < INPUT_SIZE` 而不是 `i < 5`，以保持代码规范。

3.2 辅助函数

- `int is_close(double a, double b)`
    - *功能*: 判断两个浮点数是否足够接近（相等）。
    - *原因*: 在C语言中，直接用 `a == b` 比较小数是不准确的。
    - *返回值*: 如果相等（误差范围内）返回 1，否则返回 0。
    - *示例*: `if (is_close(result, 2.5)) { ... }`
- `void print_array(double arr[], int size)`
    - *功能*: 格式化打印一个 double 类型的数组。
    - *用途*: 如果你发现计算结果不对，可以用它把输入数组打印出来看看。
    - *示例*: `print_array(inputs, INPUT_SIZE);`

4. 你的任务

你需要完成 `main.c` 中的四个核心函数。每完成一个，系统会自动校验你的代码。

任务 A: 突触加权求和 (Synaptic Sum)

这是神经网络的基础。我们需要计算输入信号与权重的加权和。

公式:

$$S = (\sum_{i=0}^{4} input[i] \times weights[i]) + bias$$

- 输入: 一个包含5个浮点数的数组 `inputs`，以及神经元结构体。

- 逻辑: 使用 `for` 循环遍历 0 到 4，将输入与对应的权重相乘并累加，最后加上偏置值 `bias`。
- 考察点: `for` 循环，数组访问，累加器模式。

任务 B: Nova 激活函数 (Nova Activation)

普通的AI使用ReLU或Sigmoid，但在高辐射环境下，我们需要一种分段线性的“软限制”函数。

公式:

$$f(x) = \begin{cases} x & \text{if } -2 \le x \le 2 \\ 2 + 0.1 \times (x - 2) & \text{if } x > 2 \\ -2 + 0.1 \times (x + 2) & \text{if } x < -2 \end{cases}$$

- 解释: 如果信号 `x` 在 -2 到 2 之间，直接输出 `x`（线性区）。 如果信号太强（大于2），它会被极大抑制（增长率变为0.1）。 如果信号太弱（小于-2），同理被抑制。

- 考察点: 复杂的 `if-else if-else` 条件分支。

任务 C: 前向传播 (Forward Pass) 将前两步结合起来。

1. 先计算 **加权和** (调用任务A的逻辑)。
2. 将加权和的结果传入 **Nova激活函数** (调用任务B的逻辑)。
3. 返回最终结果。
4. 考察点: 函数的组合与调用，变量传递

任务 D: 能量误差评估 (Energy Loss)

我们需要知道我们的预测有多准。

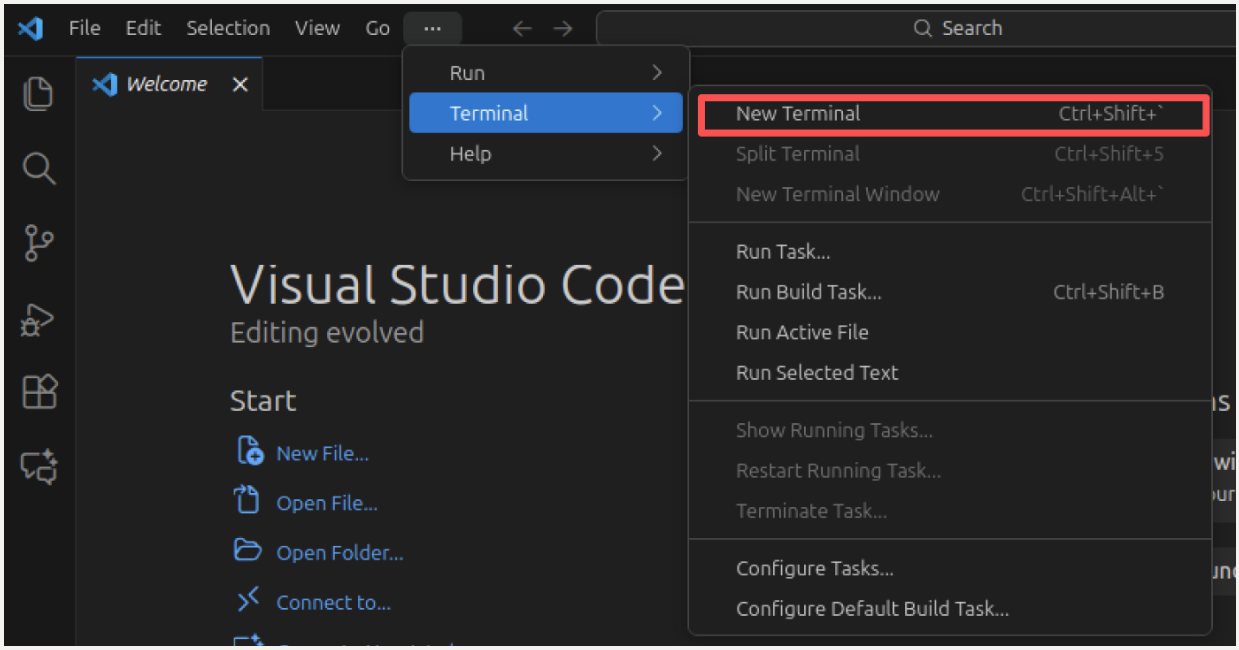
公式:

$$Loss = (Prediction - Target)^2 \times 0.5$$

- 输入: 预测值 `prediction` 和 真实值 `target`。
- 考察点: 基础数学运算。

5. 运行指南

1. 在已打开本任务目录的 **Visual Studio Code** 中新建终端



2. 在终端使用命令编译 `gcc main.c -o main`
3. 在终端使用命令运行 `./main`
4. 控制台会输出 `[TEST 1] Synaptic Sum...` 等信息。
5. 如果显示 `[PASS]`，则该阶段通过。

6. 如果显示 `[FAIL]`，它会告诉你预期值是多少，你的计算结果是多少，请根据提示修改代码。

```
==========================================
  TEST STEP 1: 突触加权求和 (Synaptic Sum)
==========================================
[FAIL] Case 1
   期待: 2.0000
   你的输出: 0.0000

==========================================
  TEST STEP 2: Nova 激活函数
==========================================
[FAIL] Linear Range. Input 1.5 should be 1.5
[FAIL] Upper Saturation. Input 3.0 should be 2.1
[FAIL] Lower Saturation. Input -10.0 should be -2.8

==========================================
  TEST STEP 3: 前向传播 (End-to-End)
==========================================
[FAIL] Forward Pass.
   期待: 6.8 (Logic: Sum is 50, Activation suppresses it)
   你的输出: 0.0000
   提示: 你在步骤 3 中调用步骤 2 的函数了吗?

==========================================
  TEST STEP 4: 能量损失 (Loss)
==========================================
[FAIL] Loss Calculation

==========================================
  Diagnostics Complete.
==========================================
```