MLP感知机

单层感知机（Perceptron）

感知机是一种最简单的神经网络模型，用于解决线性可分的二分类问题。它通过一个线性函数将输入映射为输出，并根据分类结果不断调整权重参数，实现对训练数据的正确分类。

1. 模型定义

感知机的输出形式如下：

$$o = \sigma(\langle \mathbf{w}, \mathbf{x} \rangle + b)$$

其中：

- $\mathbf{w} \in \mathbb{R}^n$：权重向量
- $\mathbf{x} \in \mathbb{R}^n$：输入特征向量
- $b \in \mathbb{R}$：偏置项
- $\langle \mathbf{w}, \mathbf{x} \rangle$：表示 $\mathbf{w}$ 与 $\mathbf{x}$ 的内积
- $\sigma(x)$：阶跃激活函数，定义如下：

$$\sigma(x) = \begin{cases} 1, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}$$

感知机的决策边界是超平面 $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$。

2. 感知机学习算法（原始形式）

感知机的训练目标是通过不断迭代调整 $\mathbf{w}$ 和 b，使所有训练样本被正确分类。
初始化参数w = 0 ，b = 0
算法流程

```
repeat
for each (x_i, y_i) in training_set:
if y_i * (⟨w, x_i⟩ + b) ≤ 0:
w = w + y_i * x_i
b = b + y_i
end if
until all samples are correctly classified
```

解释说明：

- x_i：训练样本的输入特征
- $y_i \in \{-1, +1\}$：对应的标签
- 条件 $y_i(\langle w, x_i \rangle + b) \leq 0$ 表示样本被误分类
- 更新规则的作用是使误分类样本向正确方向靠拢，从而提高模型性能

3. 损失函数

感知机使用的损失函数如下：

$$e(y, x, w) = \max(0, -y\langle w, x \rangle)$$

解释说明：

- 当 $y\langle \mathbf{w}, \mathbf{x} \rangle > 0$（分类正确）：损失为 0
- 当 $y\langle \mathbf{w}, \mathbf{x} \rangle \leq 0$（分类错误）：损失为正值，驱动参数更新
- 损失函数对误分类样本进行惩罚，而忽略已正确分类的样本

该损失函数可看作感知机的目标函数之一，用于指导梯度更新方向。

4. 收敛性说明

若训练数据是**线性可分的**，则感知机算法一定会在有限次迭代内**收敛**，即存在一组参数 $(\mathbf{w}, b)$，使得对所有训练样本都成立：

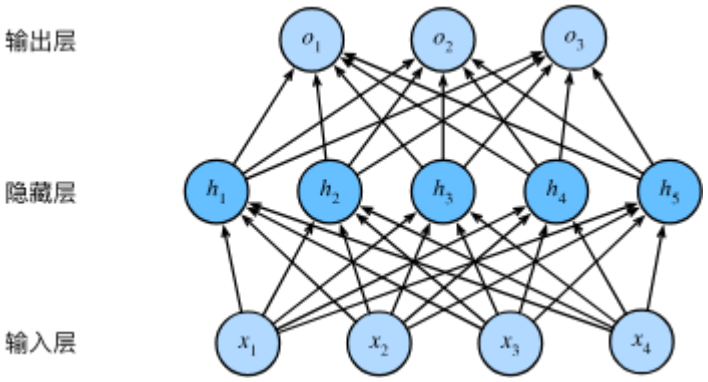
$$y_i(\langle w, x_i \rangle + b) > 0$$

解释说明：

- 感知机每次仅更新被误分类的样本
- 在每次更新后，模型朝着更好地分类方向移动
- 若样本线性可分，算法一定会停止，即不再有误分类样本

多层感知机

对于某些分类问题，不能简单地使用线性放射变换，比如图像等本质是非线性的，
多层感知机（MLP）能够通过引入多个隐藏层和非线性激活函数，



拟合复杂的非线性关系。
隐藏层(可以拓展为多层，每一层的输出是下一层的输入)
$H = \sigma(XW_1 + b_1)$
加入隐藏层进行多层仿射变换，如果不加入非线性激活函数那么整个模型还是线性函数，那么就和softmax回归相似

输出层$O = HW_2 + b_2$

常见的激活函数

ReLU激活函数

$$\mathrm{ReLU}(x) = \max(x, 0).$$

sigmod函数

$$\mathrm{sigmoid}(x) = \frac{1}{1 + \exp(-x)}.$$

tanh函数

$$\tanh(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)}.$$

多层感知机的实现

1. 使用`nn.Sequential()`来实现对神经网络多层的依次连接
   `nn.Sequential`是 PyTorch 中`torch.nn`模块提供的一个**有序容器（Ordered Container）**，用于将多个神经网络层（如`Linear`，`ReLU`，`Conv2d`，`Dropout`等）组合成一个整体模型。当你调用这个组合模型时，**数据会按顺序依次通过这些层**。
2. 初始化参数
   权重使用正态分布初始化，偏置使用0初始化
3. 定义损失函数
   从这个标量损失出发，自动计算**每个参数对损失的梯度（偏导数）**，表示“如果改动这个参数，损失会如何变化”。
4. 定义优化器
   利用计算得到的梯度，用优化器（如 SGD、Adam）调整参数，朝着减少损失的方向前进
5. 训练迭代
   重复上述轮次通过迭代一定轮次模型性能不断提升