作业

Homework1

庄镇华 502022370071

A Neural Networks Homework Assignment

NANJING UNIVERSITY

2023 年 3 月 23 日

题目一

怎样用 MP 神经元实现与、或、非逻辑运算？（分析：假设将逻辑运算表示为 $x_1\&x_2 == y$，也就是将 (x_1,x_2) 视为输入，y 视为输出。我们希望 MP 神经元获得从 (x_1,x_2) 到 y 的映射关系）。

解答： 实现与逻辑运算：$y = f(x_1 + x_2 - 1.5)$

实现或逻辑运算：$y = f(x_1 + x_2 - 0.5)$

实现非逻辑运算：$!x1$：$y = f(-x_1 + 0.5)$，

$!x2$：$y = f(-x_2 + 0.5)$，

$!x1 \,|\, !x_2$：$y = f(-x_1 - x_2 + 1.5)$，

$!x1 \,\&\, !x_2$：$y = f(-x_1 - x_2 + 0.5)$

其中激活函数 $f(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0. \end{cases}$

题目二

为什么要向神经元中引入激活函数，请再列举至少三种课程中未介绍的激活函数，并给出其表达式。

解答： 激活函数可以将神经元的输出限制在一个合理的范围内，同时可以增加神经网络模型的非线性，使得神经网络可以逼近其他的任何非线性函数。如果没有激活函数，神经网络的表达能力等同于一层线性层。

除课程中介绍的 sigmoid、tanh、ReLU 激活函数外，还有 LeakyReLU、swish、ELU 等激活函数。

$$\mathrm{LeakyReLU}(x) = \begin{cases} x, x \geq 0 \\ \gamma x, x < 0. \end{cases}, \text{其中 } x \text{ 是较小的正数}$$

$$\mathrm{swish}(x) = x\,\mathrm{sigmoid}(\beta x) = \frac{x}{1 + e^{-\beta x}}, \text{其中 } \beta \text{ 为超参数}$$

$$\mathrm{ELU}(x) = \begin{cases} x & , x \geq 0 \\ \alpha(e^x - 1) & , x < 0. \end{cases}, \text{其中 } \alpha \text{ 是超参数}$$

题目三

异或问题是否可以通过单个感知机神经元实现？为什么？

解答： 异或问题是不可以通过单个感知机神经元实现，因为异或问题属于线性不可分问题，而单个感知机神经元无法解决线性不可分问题。

具体证明如下：假设异或问题可以用单个感知机神经元解决，根据真值映射表，则 w_1，w_2 和 θ 必须满足如下方程组：

$$\begin{cases} w_1 + w_2 - \theta < 0 \\ w_1 + 0 - \theta \geq 0 \\ 0 + w_2 - \theta \geq 0 \\ 0 + 0 - \theta < 0 \end{cases}$$

该方程组无解，说明异或问题无法使用单个感知机神经元解决。

题目四

除异或问题外，还有哪些问题直观上非常简单但使用单个感知机神经元无法解决，请给出一个实例并说明无法解决的原因。

解答： 所有线性不可分问题单个感知机神经元都无法解决，例如 R 为二维平面上以原点为圆心，半径为 2 的圆，所有在圆内以及圆边上的点标签为 1，其余点标签为 0，求解点是否在圆内或圆上的问题。

我们用反证法证明，取以下四点：$(1,-1)$，$(-1,1)$，$(2,2)$，$(-2,-2)$，易知前两个点标签为 1，后两个点标签为 0，假设可以用单个感知机神经元解决，则 w_1，w_2 和 θ 必须满足如下方程组：

$$\begin{cases} w_1 - w_2 - \theta \geq 0 \\ -w_1 + w_2 - \theta \geq 0 \\ 2w_1 + 2w_2 - \theta < 0 \\ -2w_1 - 2w_2 - \theta < 0 \end{cases}$$

该方程组无解，说明此问题无法使用单个感知机神经元解决。

题目五

尝试通过组合多个感知机神经元来解决异或问题，请画出所设计的网络结构（包括相关联结的权重）。

解答： 要解决非线性可分问题，需考虑使用多层功能神经元，如图1中这个简单的两层感知机就能解决异或问题。

其最终的表达式为

$$y = f\left(f\left(x_1 - x_2 - 0.5\right) + f\left(x_2 - x_1 - 0.5\right) - 0.5\right),$$

其中激活函数 $f(x) = \begin{cases} 1, x \geq 0 \\ 0, x < 0. \end{cases}$，代入真值表检验符合异或要求。

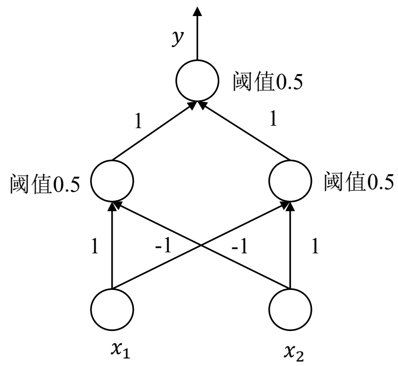


图 1: 异或问题网络结构

题目六

试使用感知机神经元对半月数据量 $N = 2000$，半月宽度 $w = 6$，x 轴偏移 $r = 10$，y 轴偏移量 $d = 2$ 的双月模型进行分类，生成双月数据集的代码可以参考如下代码。请完成以下实验

1. 生成双月数据集，并可视化数据；
2. 用感知机实现模型并对双月数据集进行训练，并可视化学习曲线和决策边界；
3. 请选择不同的学习率进行对比，可以得出什么结论

（需请提交所有的代码文件）

```python
def moon(N, w, r, d):
    ''':param w: 半月宽度 # :param r: x 轴偏移量 # :param d: y 轴偏移量
    # :param N: 半月散点数量 :return: data (2*N*3) 月亮数据集 data_dn
    (2*N*1) 标签 '''
    data = np.ones((2*N,4))
    # 半月 1 的初始化
    r1 = 10 # 半月 1 的半径,圆心
    np.random.seed(1919810)
    w1 = np.random.uniform(-w / 2, w / 2, size=N) # 半月 1 的宽度范围
    theta1 = np.random.uniform(0, np.pi, size=N) # 半月 1 的角度范围
    x1 = (r1 + w1) * np.cos(theta1) # 行向量
    y1 = (r1 + w1) * np.sin(theta1)
    label1 = [1 for i in range(1,N+1)] # label for Class 1
    # 半月 2 的初始化
    r2 = 10 # 半月 2 的半径,圆心
    w2 = np.random.uniform(-w / 2, w / 2, size=N) # 半月 2 的宽度范围
    theta2 = np.random.uniform(np.pi, 2 * np.pi, size=N) # 半月 2 的
    角度范围
    x2 = (r2 + w2) * np.cos(theta2) + r
    y2 = (r2 + w2) * np.sin(theta2) - d
    label2 = [-1 for i in range(1,N+1)] # label for Class 2
    data[:,1] = np.concatenate([x1, x2])
    data[:,2] = np.concatenate([y1, y2])
    data[:,3] = np.concatenate([label1, label2])
    return data
```

解答： 1. 生成双月数据集，并可视化数据。数据可视化如图2所示。

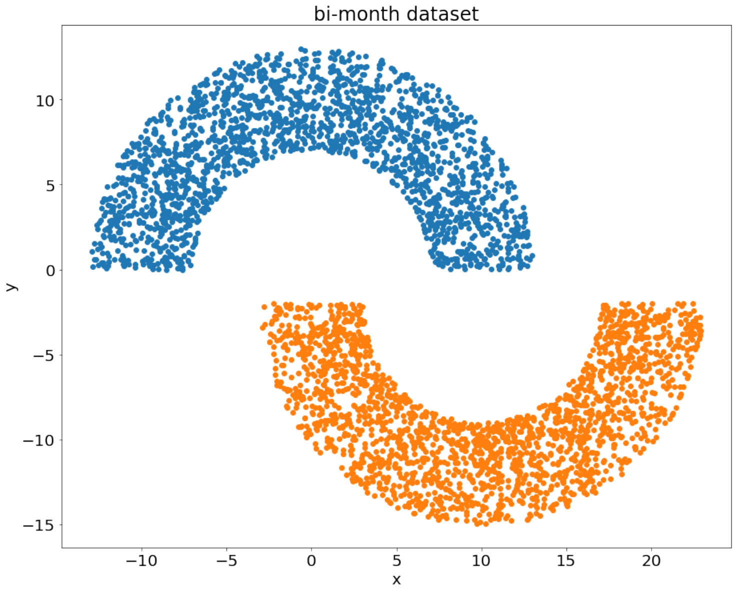


图 2: 双月数据集可视化

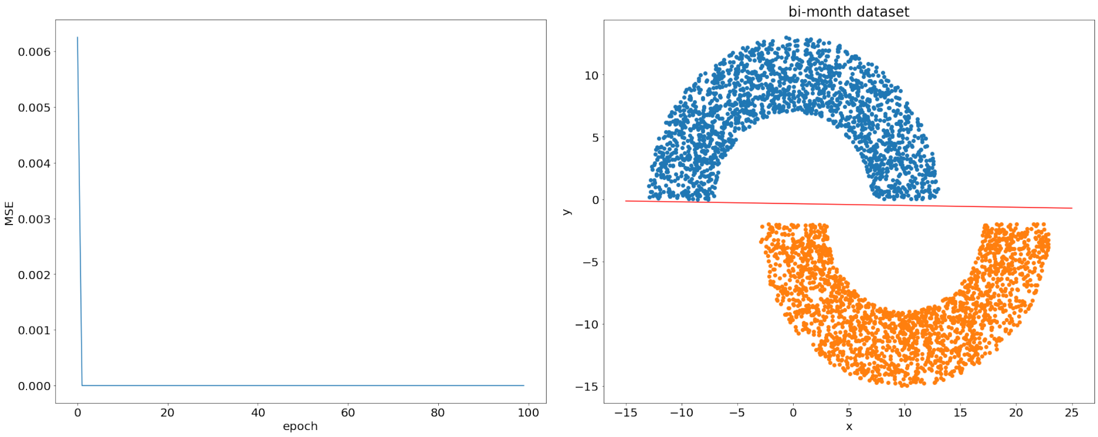
2. 感知机模型代码实现如下所示，学习曲线如图3(a) 所示，决策边界如图3(b) 所示。

```python
losses = []
class Perceptron(object):
    def __init__(self, x, y, learning_rate):
        self.x = x
        self.y = y
        self.learning_rate = learning_rate
        self.w = np.zeros(x.shape[1]) # 权重
        self.b = 0 # 偏置
        self.activate_func = np.sign # 激活函数
        self.out = None # 权重

    def calculate(self, x):
        return self.activate_func(np.dot(self.w, x.T) + self.b)

    def update(self, x, y):
        self.w += self.learning_rate * x.T * (y - self.out)
        self.b += self.learning_rate * (y - self.out)

    def train(self, epochs):
        for _ in range(epochs):
            loss = 0
            for i in range(self.x.shape[0]):
                self.out = self.calculate(self.x[i])
                loss += (self.out - self.y[i]) ** 2
                self.update(self.x[i], self.y[i])
            losses.append(loss / self.x.shape[0])
```



(a) 学习曲线 (b) 决策边界

图 3: 感知机模型学习曲线与决策边界

3. 请选择不同的学习率进行对比，可以得出什么结论。

实验尝试了学习率分别设置为 $\{1e-4, 1e-2, 1, 1e2\}$ 的场景，发现感知机模型学习曲线均在 epoch=1 时候收敛，并且最终学到的决策边界的直线斜率和截距分别稳定在-0.0144 和 0.3637 附近，这一方面可能是因为数据集较为简单，另一方面也说明了感知机模型对学习率超参数具有一定的鲁棒性。