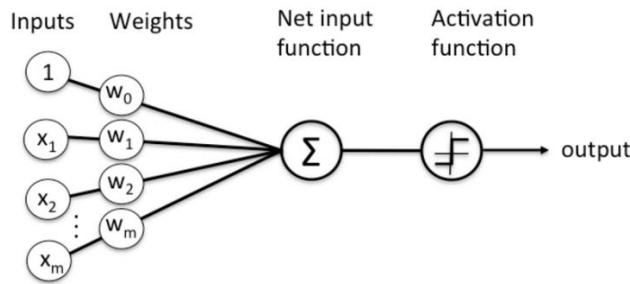


认知科学与类脑计算课程实验报告

实验题目：感知器模型实现及分类		学号：201600181073
日期：2019.5.17	班级：智能16	姓名：唐超
<p>实验目的：</p> <p>根据感知器的相关知识，使用 Python 语言实现一个简单的感知器模型，该模型能够实现简单的二分类任务（与或非运算任选一）。</p>		
<p>实验环境：</p> <p>Jupyter notebook & python3.6 & numpy</p>		
<p>主要算法及步骤：</p> <p>感知机是二分类的线性分类模型，属于监督学习算法。感知机模型如下图所示：</p> <div data-bbox="458 920 1091 1202"></div> <p>设输入向量为 $x \subseteq R^m$，输出空间为样本类别 $Y = \{1, -1\}$。由输入空间到输出空间的函数为</p> $f(x) = \text{sign}(w \cdot x + b)$ <p>其中，参数 w 为权值向量，b 称为偏置。</p> <p>如果训练集是可分的，感知机的学习目的是求得一个能将训练集正实例点和负实例点完全分开的分离超平面。为了找到这样一个平面（或超平面），即确定感知机模型参数 w 和 b，我们将所有点到超平面的距离作为损失函数（可以不考虑 $-\frac{1}{\ w\ }$）：</p> $L(w, b) = -\frac{1}{\ w\ } \sum_{x_i} y_i w \cdot x_i + b $ <p>为了极小化 $L(w, b)$，采用梯度下降的方法对参数进行更新：</p> $\frac{\partial L(w, b)}{\partial w} = -\sum_{x_i} y_i x_i$		

$$\frac{\partial L(w, b)}{\partial w} = -\sum_{x_i} y_i$$

设学习率为 η 。因此，

$$w := w + \eta y_i x_i$$

$$b := b + \eta y_i$$

程序设计：

定义 Perceptron 类，输入参数为输入数据的维度 length，及构建数据 label 的操作 operation，operation 为 “and” 或 “or”。包含构建真值表函数 truthTable，预测函数 predict，训练函数 train。

初始化权重和超参：

```
self.W = np.random.randn(length) # shape (length,)
self.b = 0
self.lr = 0.01
self.epochs = 1000
```

创建真值表时，根据输入维度，初始化 self.data = np.zeros((2**length, length))，并对每一列遍历操作，以创建 2**length 个 length 维的 0, 1 组合（具体见代码文件）。同时初始化相应的 label 数组 self.label = np.zeros((2**length, 1))，当 operation= “and” 时，self.label[0] = 1，当 operation= “or” 时，self.label[range(2**length-1)] = 1。

predict 函数都采用上一栏中的相关公式进行预测。由于在真值表中的 label 为 1 或 0，因此 train 中的参数更新公式稍作修改，如下：

$$w := w + \eta(y_i^{output} - y_i)x_i$$

$$b := b + \eta(y_i^{output} - y_i)$$

对真值表中的所有数据输入网络并更新 w, b，遍历 self.epochs 次。

调试分析：

1. 在做 $w \cdot x$ 运算时，输入真值表 data 为 numpy 的二维数组，w 为一维数组，做点积运算后，可以对其中的元素进行访问。但当对单个一维数组进行预测时，如输入为 np.array([1, 1, 1, 1]) 时， $w \cdot x + b$ 为标量，无法用

```
self.output[self.output>=0] = 1
```

```
self.output[self.output<0] = 0
```

对其中元素做 sign 函数运算，因此输入数组的 shape 不能为 (length,)，必须为 (length, 1)，例如 np.array([[1, 1, 1, 1]])。另外 shape 为 (length,) 的 numpy 数组不区分行和列，因此做点积运算时不必转置。

2. 在定义的 perceptron 类的实体下，多次改变参数调试，发现 self.W 和 self.b 都在前一次的基础上调参的基础上更新，这对调试是不利的，因此在 train 函数的前两行重置 self.W 和 self.b。
3. 在训练参数时，每次更新参数时，需要用当前参数预测出上一栏中的 y_i^{output} ，刚开始做时错把 self.predict(self.data) 写在了更新参数的循环外。

测试结果及分析：

部分测试结果如下（最后一组数据是 $w \cdot x + b$ 的结果，用以在分类错误的情况下调整 lr，epochs 重新训练）：

- length=4, operation= “or”:

测试数据:

```
[[1 1 1 1]
 [1 0 1 1]
 [0 0 0 1]
 [1 1 0 0]
 [0 0 0 0]]
```

测试结果:

```
[1. 1. 1. 1. 0.]
```

用以调整lr, epochs:

```
[ 2.09497468e+00  2.09126085e+00  3.25932196e-01  6.29708447e-01
 -3.12250226e-17]
```

- length=3, operation= “and”:

测试数据:

```
[[1 1 1]
 [0 0 0]
 [1 1 0]
 [0 0 0]]
```

测试结果:

```
[1. 0. 0. 0. 0.]
```

用以调整lr, epochs:

```
[ 0.00347956 -0.005283 -0.85 -0.02322919 -0.85 ]
```

- length=5, operation= “or”:

测试数据:

```
[[1 1 1 1 1]
```

```
[1 0 1 1 0]
```

```
[0 0 0 1 0]
```

```
[1 1 0 0 1]
```

```
[0 0 0 0 0]]
```

测试结果:

```
[1. 1. 1. 1. 0.]
```

用以调整lr, epochs:

```
[ 3.31426083  2.48033643  0.74378878  2.24953983 -0.01      ]
```

总体来说，由于任务比较简单，很容易用感知机模型实现准确分类。