第3章

感　知　机

CHAPTER 3

生物神经元对信息的传递与处理是通过各个神经元之间突触的兴奋或抑制作用实现的。根据这一事实，美国学者 Rosenblatt 于 1957 年提出了一种具有单层计算单元的神经网络，称为感知机(perceptron)。该模型是神经网络与支持向量机(Support Vector Machine，SVM)的基础，不仅如此，深度学习的很多模型也是以感知机模型为基础。感知机是神经网络与深度学习的重要模型之一，具有结构简单、运算和收敛速度较快、实用性强等特点。

3.1　感知机原理

感知机是基于层内无反馈无互联的层次结构所构建的线性二分类模型，属于有监督的学习算法，根据隐藏层的情况可将其分为单层感知机和多层感知机。

3.1.1　单层感知机

单层感知机(Single-Layer Perceptron，SLP)仅有输入层和输出层，结构如图 3-1 所示。最早的单层感知机模型只有一个输出结点，它相当于一个单独的神经元，其功能是对输入的数据进行正确的分类，即通过输入和输出正确的模式对样本进行学习后，使模型可以对输入数据进行 0、1 分类。首先，将训练数据输入到单层感知机中，然后单层感知机利用已有的模型调整参数并计算出结果，最后输出结果。单层感知机是层内无反馈无互连的层次结构，即输入层结点只与输出层结点进行全连接。单层感知机通过评估输出结果与期望值之间的误差，对输入层与输出层的连接权值进行调整，以获得一个最佳模型。

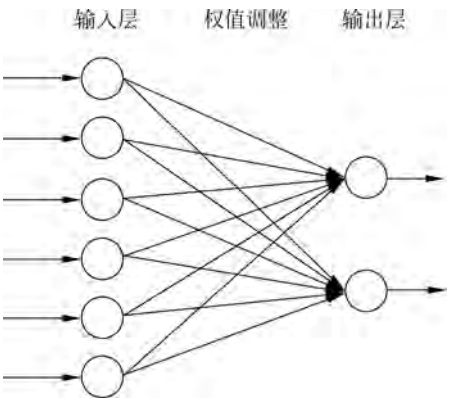

图 3-1　单层感知机结构

由于单层感知机只有一层功能神经元，其学习能力极其有限，所以单层感知机模型只能解决线性可分问题，而无法解决更加复杂的线性不可分问题，具有局限性。为了解决复杂的线性不可分问题，学者们又提出了多层感知机模型，随着感知机模型发展日益成熟，这一模型在很多领域都有广泛的应用。

3.1.2　多层感知机

多层感知机(Multi-Layer Perceptron，MLP)由单层感知机推广而来，其模型至少有三层。第一层为输入层，最后一层为输出层，中间层为隐藏层，隐藏层的数量根据实际需要进行调整。

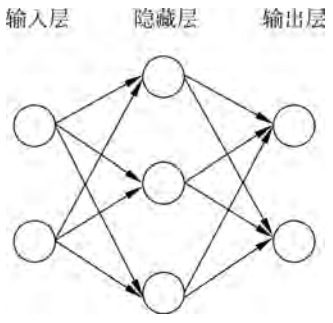

图 3-2　多层感知机结构

如图 3-2 所示，这是一个多层感知机结构的经典模型，由输入层、输出层和一层隐藏层构成。输入层的各神经元没有信息处理能力，只是将输入数据进行输出。输入层和隐藏层之间全连接，隐藏层将经过激活函数变换的结果作为输出。假设输入信息用向量 $\boldsymbol{X}$ 表示，则隐藏层的输出就是 $f(\boldsymbol{W}_1\boldsymbol{X}+b_1)$，其中，$\boldsymbol{W}_1$ 为连接权值向量(或连接系数向量)，b_1 是偏置量(即前面章节所提到的阈值，在神经网络里常称为偏置量)，函数 f 可以是 Sigmoid 函数、Tanh 函数以及其他常用的激活函数。

由此可见,一个多层感知机中,其重点在于各层之间的连接权值以及偏置量。连接权值 $\boldsymbol{W}_1$ 和偏置量 b_1 的取值将直接影响整个感知机模型的分类效果。通常,在训练模型之前,需要初始化所有参数,然后进行循环迭代地训练,通过不断地调整连接权值和偏置量,直到满足某个条件为止(比如误差足够小、迭代次数足够多等)。

多层感知机的基本特征:①网络中每个神经元包含一个可微的非线性激活函数;②网络的输入层和输出层之间包含一个或多个隐藏层;③网络展示出高度的连接性,连接强度是由突触权值决定的。

3.2 感知机模型

假设输入空间(特征空间)是 $\boldsymbol{X}\subseteq\boldsymbol{R}^n$,输出空间是 $Y\subseteq\{+1,-1\}$。输入 $\boldsymbol{x}\subseteq\boldsymbol{X}$ 表示实例的特征向量,对应输入空间的点;输出 $\boldsymbol{y}\subseteq\boldsymbol{Y}$ 表示实例的类别。输入空间到输出空间的函数如式(3-1)所示。

$$f(x)=\text{sign}(\boldsymbol{w}\cdot\boldsymbol{x}+b) \tag{3-1}$$

其中,$\boldsymbol{w}$ 和 b 为感知机的模型参数;$\boldsymbol{w}$ 为连接权值或权值向量;b 为偏置量;$\boldsymbol{w}\cdot\boldsymbol{x}$ 表示 $\boldsymbol{w}$ 和 $\boldsymbol{x}$ 的内积;sign()是符号函数,如式(3-2)所示。

$$\text{sign}(x)=\begin{cases}-1, & x<0\\ 0, & x=0\\ 1, & x>0\end{cases} \tag{3-2}$$

也可以采用 Sigmoid()等其他转移函数。

感知机是一种线性分类模型,属于判别模型。感知机模型的假设空间是定义在特征空间中的所有线性分类模型或者线性分类器,即函数集合,如式(3-3)所示。

$$\{f\mid f(x)=\boldsymbol{w}\cdot\boldsymbol{x}+b\} \tag{3-3}$$

3.3 感知机算法

在单层感知机模型中可以调整输入层和输出层之间的连接权值,在多层感知机模型中输入层至隐藏层之间的连接权值固定不变,只能调整隐藏层和输出层之间的连接权值。无论哪种模型,连接权值的调整都是按照感知机学习算法进行的。感知机学习算法是基于随机梯度下降法来对损失函数进行最优化的算法,有原始形式和对偶形式。

3.3.1 随机梯度下降法

随机梯度下降法(Stochastic Gradient Descent,SGD)是优化神经网络的基础迭代算法之一,其思想是:在随机、小批量的子集上计算出的梯度,近似于在整个数据集上计算出的真实梯度。SGD 每一步用小批量样本迭代更新权重,如式(3-4)和式(3-5)所示。

$$w_{t+1}=w_t+\Delta w_t \tag{3-4}$$

$$\Delta w_t = -\eta \nabla_w E(w_t) \tag{3-5}$$

其中，η 是算法的学习率；$E(w_t)$是第 t 次迭代权重 w_t 的损失函数；$\nabla_w E(w_t)$为权重 w 在 t 时刻关于损失函数的一阶梯度，简记为 g_t；w_{t+1} 为 $t+1$ 时刻的权重值；w_t 为 t 时刻的权重值；Δw 为梯度算子，即每次迭代权重的更新部分。

随机梯度下降法的训练速度较快，每次迭代的计算量较小，但是训练速度较快会牺牲一部分准确度，并且最终的计算结果并不一定是最优解，整个过程的实现相对复杂，迭代次数也较多。

3.3.2　感知机学习算法

感知机学习算法是对以下最优化问题的算法。给定一个训练数据集，如式(3-6)所示。

$$T = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \cdots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\} \tag{3-6}$$

其中，$\boldsymbol{x}_i \in X \subseteq \mathbf{R}^N$，$\boldsymbol{y}_i \in Y \subseteq \{-1, +1\}$，$i = 1, 2, \cdots, N$，求参数 w、b，使其为式(3-7)中损失函数极小化问题的解：

$$\min_{w,b} L(w, b) = -\sum_{x_i \in M} y_i (w \cdot x_i + b) \tag{3-7}$$

其中，M 为误分类点的集合。

感知机学习算法是采用随机梯度下降法进行误分类驱动。首先，任意选取一个超平面 $w_0 x + b_0 = 0$，然后用梯度下降法不断地极小化损失函数，见式(3-7)。极小化过程中不是一次性使 M 中所有的误分类点的梯度下降，而是每次随机选取一个误分类点使其梯度下降。

假设误分类点集合 M 是固定的，那么损失函数 $L(w, b)$的梯度由式(3-8)和式(3-9)给出。

$$\nabla_w L(w, b) = -\sum_{x_i \in M} y_i x_i \tag{3-8}$$

$$\nabla_b L(w, b) = -\sum_{x_i \in M} y_i \tag{3-9}$$

随机选取一个误分类点(x_i, y_i)，对 w、b 进行更新，更新规则如式(3-10)和式(3-11)所示。

$$w \leftarrow w + \eta y_i x_i \tag{3-10}$$

$$b \leftarrow b + \eta y_i \tag{3-11}$$

其中，η 是步长$(0 < \eta \leqslant 1)$，又称为学习率(learning rate)。算法多次迭代的目标是使损失函数 $L(w, b)$不断减小，直到为 0 或小于设定值。

综上所述，得到如下算法：

算法 1(感知机学习算法的原始形式)

输入：训练数据集 $T = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \cdots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$，其中，$\boldsymbol{x}_i \in X \subseteq \mathbf{R}^N$；$\boldsymbol{y}_i \in Y \subseteq \{-1, +1\}$；$i = 1, 2, \cdots, N$；学习率 $\eta(0 < \eta \leqslant 1)$。

输出：w，b；感知机模型 $f(x) = \text{sign}(w \cdot x + b)$。

训练过程如下：

(1) 设置初值 w_0、b_0；

(2) 在训练集中随机选取样本点(x_i, y_i)；

(3) 如果 $y_i(w \cdot x_i + b) \leqslant 0$，表示为误分类，则更新参数：

$$w \leftarrow w + \eta y_i x \tag{3-12}$$

$$b \leftarrow b + \eta y_i \tag{3-13}$$

(4) 转至步骤(2)，直至训练集中没有误分类点。

从直观的角度分析：当一个实例点被误分类，即位于分离超平面 $S: w \cdot x + b$ 的错误一侧时，则调整 w, b 的值，使分离超平面向该误分类点的一侧移动，以减少该误分类点与超平面间的距离 $\frac{1}{\|w\|}|w \cdot x_i + b|$，直至误分类点被划分到正确的一侧。

算法 2(感知机学习算法的对偶形式)

对偶形式的思想：设一共有 N 个样本，对于每一个在更新过程中被使用了 n_i 次的样本(x_i, y_i)，即在所有的学习循环次数中，有 n_i 次中将该样本作为了误分类点，故用它去更新参数。

原始形式 $w \leftarrow w + \eta y_i x_i$ 和 $b \leftarrow b + \eta y_i$ 就可以写成式(3-14)和式(3-15)。

$$w = \sum_{i=1}^{N} n_i \eta y_i x_i = \sum_{i=1}^{N} \alpha_i y_i x_i \tag{3-14}$$

$$b = \sum_{i=1}^{N} n_i \eta y_i = \sum_{i=1}^{N} \alpha_i y_i \tag{3-15}$$

其中，$\alpha_i = n_i \eta$，$\alpha_i \geqslant 0$；$i = 1, 2, 3, \cdots, N$；n_i 代表对第 i 个样本的学习次数；当 $\eta = 1$ 时，表示第 i 个实例点由于误分类进行的更新次数，更新次数越多意味着该样本距离超平面越近，越难以分类。感知机模型如式(3-16)所示。

$$f(x) = \text{sign}(w \cdot x + b) = \text{sign}\left(\sum_{j=1}^{N} \alpha_j y_j x_j \cdot x + b\right) \tag{3-16}$$

输入：训练数据集 $T = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), \cdots, (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$，其中，$\boldsymbol{x}_i \in X \subseteq \mathbf{R}^N$；$\boldsymbol{y}_i \in Y \subseteq \{-1, +1\}$；$i = 1, 2, \cdots, N$；学习率 $\eta (0 < \eta \leqslant 1)$；

输出：$\boldsymbol{\alpha}$、b；感知机模型 $f(x) = \text{sign}\left(\sum_{j=1}^{N} \boldsymbol{\alpha}_j \boldsymbol{y}_j \boldsymbol{x}_j \cdot \boldsymbol{x} + b\right)$；其中 $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \cdots, \boldsymbol{\alpha}_N)^{\mathrm{T}}$。训练过程如下：

(1) 初始化 $\boldsymbol{\alpha} = 0, b = 0$；

(2) 在数据集中选取数据$(\boldsymbol{x}_i, \boldsymbol{y}_i)$；

(3) 判断是不是误分类点 $\boldsymbol{y}_i\left(\sum_{j=1}^{N} \boldsymbol{\alpha}_j \boldsymbol{y}_j \boldsymbol{x}_j \cdot \boldsymbol{x} + b\right) \leqslant 0$，如果是，即发生误判，则对 $\boldsymbol{\alpha}_i$、

$\boldsymbol{b}_i$ 进行更新，如式(3-17)和式(3-18)所示。

$$\boldsymbol{a}_i \leftarrow \boldsymbol{a}_i + \eta \tag{3-17}$$

$$\boldsymbol{b}_i \leftarrow \boldsymbol{b}_i + \eta \boldsymbol{y}_i \tag{3-18}$$

(4) 重复步骤(2)，直到所有点都被正确分类。

从对偶形式的计算式可以看到，样本之间的计算是 $\boldsymbol{x}_i \cdot \boldsymbol{x}_j$，其余计算是 N 维向量的矩阵，其中 N 是样本个数。因此对偶形式适用于样本个数比特征空间的维数小的情况。

从以上推导可以看出，感知机学习算法的原始形式和对偶形式在本质上一样，但从具体的计算过程来看，感知机学习算法的对偶形式的数据点仅以向量内积的形式出现。

3.4　感知机改进算法

感知机自身结构的限制使得感知机在处理一些问题时存在缺陷。对感知机进一步研究后，学者们提出了以下几种改进的感知机算法。

1. 口袋算法

原始的感知机算法只能对线性可分的问题进行处理，而在处理线性不可分的问题时，则会发生算法无法停止的震荡现象。在实际工作中许多学者设计出各种规则使算法终止(例如设置最大迭代次数、学习步长骤减等)，但是最终解的性质是不确定的。

为了应对这些问题，Gallant 在感知机算法的迭代过程中引入一个口袋权向量来存放正确运行次数最多的感知机权向量，其目标是找到一个最优解，并称这一感知机的改进算法为口袋算法(pocket algorithm)。在口袋算法中，需要遵循两条规则：

(1) 样本必须随机选取；

(2) 需要进行一些检查以防结果劣化。

口袋算法在处理过程中采用了贪心的近似处理，添加了一个用于存放正确运行次数最多的感知机权重与阈值的口袋权向量 $\boldsymbol{W}$，若在一次训练中得到的结果 $\boldsymbol{W}_f$ 较 $\boldsymbol{W}$ 更好，则用 $\boldsymbol{W}_f$ 取代 $\boldsymbol{W}$，否则保留 $\boldsymbol{W}$。采用口袋算法的感知机模型在处理含有噪声的数据时表现良好。

2. 核感知机算法

处理非线性分类问题的另一种方法是基于核函数的非线性感知机算法，也被称为核感知机(kernel perceptron)算法。由于当一个问题在当前维度不可分时，在更高维度的空间将有一定的概率被分开。因此，可以将模式向量映射到一个高维向量空间，在该空间中进行感知机处理，在这个运算期间仅使用两个向量的内积计算，最后用核函数代替内积计算，从而可以进行非线性处理。在核感知机算法中，基于核的非线性决策函数如式(3-19)所示。

$$f^{\varphi}(x) = \sum_{i=1}^{I} a_i y_i (x_i, x) + \beta \tag{3-19}$$

核感知机的迭代公式则如式(3-20)所示。

$$\begin{cases} a_i \Leftarrow a_i + k(x_i, x_j) y_i y_j \\ \beta \Leftarrow \beta + y_j \end{cases} \tag{3-20}$$

其中,满足 $f^{\varphi}(x)y_j \leqslant 0$ 时才进行迭代;$i=1,2,\cdots,n$;$k(x_i,x_j)$是满足 Mercer 条件的核函数。

只要满足 Mercer 条件的核函数,就可以用其来构造非线性感知机。Mercer 条件是指:对于任意的对称函数 $K(x,x')$,它是某个特征空间中的内积运算的充要条件是,对于任意的 $\varphi \neq 0$ 且 $\int \varphi^2(x)\mathrm{d}x < \infty$,则有:

$$\iint K(x,x')\varphi(x)\varphi(x')\mathrm{d}x\,\mathrm{d}x' > 0 \tag{3-21}$$

目前常用的核函数有线性核、多项式核、高斯核等。虽然核感知机对于复杂分类处理有较好的效果,但是在不可分问题处理上仍存在无法确定终止条件、终止时解的性质不确定等问题。为此国内外许多学者对于核感知机进一步改进,提出基于核函数的非线性口袋算法,即核口袋算法。核口袋算法主要利用口袋算法的思想,在核感知机算法中加入适当的检查,来改善核感知机的性能。

3. 表决感知机算法

Rosenblatt 和 Frank 在 1957 年提出的表决感知机(也称投票感知机,voted perceptron)充分利用具有大分界面的线性可分数据加快运算的速度。该方法的实现原理是假设特征向量为 $\boldsymbol{X}$,$\boldsymbol{X}$ 的标签 $\boldsymbol{y}$ 的取值范围为$\{-1,1\}$。该算法会在开始时设定一个预测向量$\boldsymbol{v}=0$,作为以后预测新的特征向量 $\boldsymbol{X}$ 的标签。即 $y'=\mathrm{sign}(\boldsymbol{v}\cdot\boldsymbol{x})$。如果预测值 y'不等于真实值 y,则更新预测向量$\boldsymbol{v}$,即$\boldsymbol{v}=\boldsymbol{v}+y\boldsymbol{x}$。如果 y 与 y'相同,则$\boldsymbol{v}$ 不变。这个过程将会反复进行。

4. 蜂群感知机算法

在感知机算法的基础上,有学者利用人工蜂群算法的特性提出了一种蜂群感知机算法(bee colony perception),其目的是要解决感知机算法在计算过程中寻找的分离超平面不唯一的问题。蜂群优化算法最终得到的分离超平面的分类效果优于传统的感知机。

感知机是从训练数据集学习分离超平面,把训练数据集分为正类和负类。但学习得到的分离超平面不唯一,原因是感知机迭代算法与初始迭代点的选取和迭代终止条件有关,使得感知机的泛化能力较差。将蜂群智能算法引入感知机的学习算法中,构建了感知机的迭代损失函数,其反映的是误分类点的个数。如式(3-22)所示。

$$\begin{cases} L(\hat{\omega}) = \sum\limits_{x_i \in M} \dfrac{1}{2}[\mathrm{sign}(-y_i(\hat{\omega}\cdot\hat{x}_i))+1] \\ -1 \leqslant \hat{\omega} \leqslant 1 \end{cases} \tag{3-22}$$

为了与蜂群算法吻合,将迭代损失函数取为蜂群算法的适应度函数,蜂群算法优化感知机分离超平面的过程就是最大化适应度函数的过程。如式(3-23)所示。

$$\max\ \mathrm{fit}(\hat{\omega}) = -\sum_{x_i \in M} \frac{1}{2}[\mathrm{sign}(-y_i(\hat{\omega}\cdot\hat{x}_i))+1] \tag{3-23}$$

蜂群感知机具体步骤流程如下:

(1) 初始化蜂群感知机模型中的控制参数，随机生成食物源的初始值。设置蜂群规模、循环次数、最大迭代步数、当前迭代步数、算法跳出循环标准等。

(2) 计算每个食物源的适应度函数值，记录最大适应度所对应的食物源、记录最优解。

(3) 采蜜蜂根据搜索更新法则(局部搜索、启发式搜索等)更新食物源，并计算相应的适应度函数值。如果新食物源的适应度函数值大于原食物源，则用新食物源代替原食物源，否则不变。

(4) 观察蜂根据采蜜蜂所提供的信息，根据轮盘赌法则更新被选中的食物源。

(5) 更新最大适应度所对应的食物源和最优解。

(6) 确定侦察蜂。若经过有限的循环次数 L 之后，某食物源没有得到更新，则放弃该食物源，同时该食物源所对应的采蜜蜂转变为侦察蜂，产生新的食物源。

(7) 若食物源更新的当前迭代步数小于跳出循环的标准，并且小于最大迭代步数，则当前迭代步数加 1，返回(2)；否则，跳出循环，保存全局最优解，停止算法运行。

此外，还有一些学者提出了平均感知机(Averaged Perceptron，AP)、信任权学习算法(confidence weighted)、被动主动算法(passive aggressive)等，对感知机的振荡现象、分类准确性等问题进行了研究与改进。

3.5　本章实践


微课视频

只有一层功能神经元的单层感知机模型学习能力有限，无法解决线性不可分的问题，为了弥补此局限，学者们又提出了多层感知机模型。本节主要利用多层感知机模型实现异或门。

异或门是数字逻辑中实现逻辑异或的逻辑门。有多个输入端、一个输出端，多输入异或门可由两输入异或门构成。若两个输入的电平相异，则输出为高电平 1；若两个输入的电平相同，则输出为低电平。异或门的逻辑表达式为 $Y=A\oplus B$(⊕为“异或”运算符)，对应的真值表如表 3-1 所示。

表 3-1　异或门的真值表

输入 A	输入 B	输出 Y
0	0	0
0	1	1
1	0	1
1	1	0

异或门可以通过或门、与非门、与门的简单组合而实现，如图 3-3 所示。

对于或门、与非门、与门这三种简单的逻辑电路均可通过结构相同的单层感知机实现，如图 3-4 所示，三个门电路之间只是权重 w_1、w_2 和偏置 b 的取值不同。

综上，异或门可通过如图 3-5 所示的二层感知机实现。

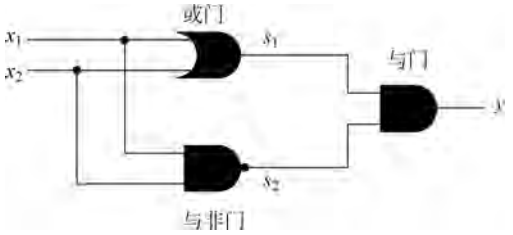

图 3-3　由或门、与非门、与门搭建异或门电路

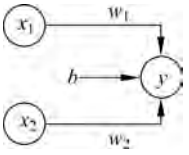

图 3-4　单层感知机实现或门、与非门、与门

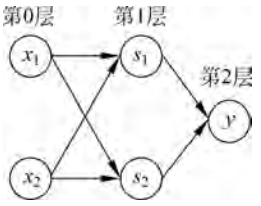

图 3-5　二层感知机实现异或门

其具体实现及主要代码如下。

1. 定义输入输出数据及指定参数

```
# 定义输入数据,有 4 种输入情况
x = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
# 定义输出数据的期望值
d = np.array([0, 1, 1, 0])
```

2. 分别定义或函数、与非函数、与函数

```
# 定义或函数,其中权重 w1 = 0.5,w2 = 0.5,偏置 b = -0.2
def OR(x1, x2):
    x = np.array([x1, x2])
    w = np.array([0.5, 0.5])
    b = -0.2
    tmp = np.sum(w * x) + b
    if tmp <= 0:
        return 0
    else:
        return 1

# 定义与非函数,其中权重 w1 = -0.5,w2 = -0.5,偏置 b = 0.7
def NAND(x1, x2):
    x = np.array([x1, x2])
    w = np.array([-0.5, -0.5])
    b = 0.7
    tmp = np.sum(w * x) + b
    if tmp <= 0:
```

```
        return 0
    else:
        return 1

# 定义与函数,其中权重 w1 = 0.5,w2 = 0.5,b = -0.7
def AND(x1, x2):
    x = np.array([x1, x2])
    w = np.array([0.5, 0.5])
    b = -0.7
    tmp = np.sum(w * x) + b
    if tmp <= 0:
        return 0
    else:
        return 1
```

3. 定义异或函数

```
# 定义异或函数
def EOR(x1, x2):
    s1 = OR(x1, x2)                    # 第一层的 s1 由或函数得到
    s2 = NAND(x1, x2)                  # 第一层的 s2 由与非函数得到
    y = AND(s1, s2)                    # 异或函数的最终输出由 s1 和 s2 的与得到
    return y
```

4. 输出结果

```
k = 0
# 输出结果
print("二层感知机实现异或门")
for index in range(len(x)):
    y = EOR(x[index][0], x[index][1])
    if y == d[index]:
        k = k + 1
    print("输入:" + str(x[index]) + "实际输出:" + str(y) + "期望输出:" + str(d[index]))
if k != 4:
    print("参数不正确,无法实现异或门,需要进一步调整")
```

最终程序运行结果如图 3-6 所示。

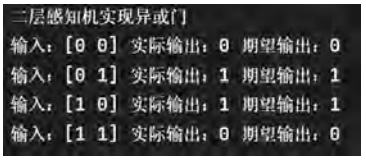

图 3-6　程序运行结果

综上,程序的流程图如图 3-7 所示。

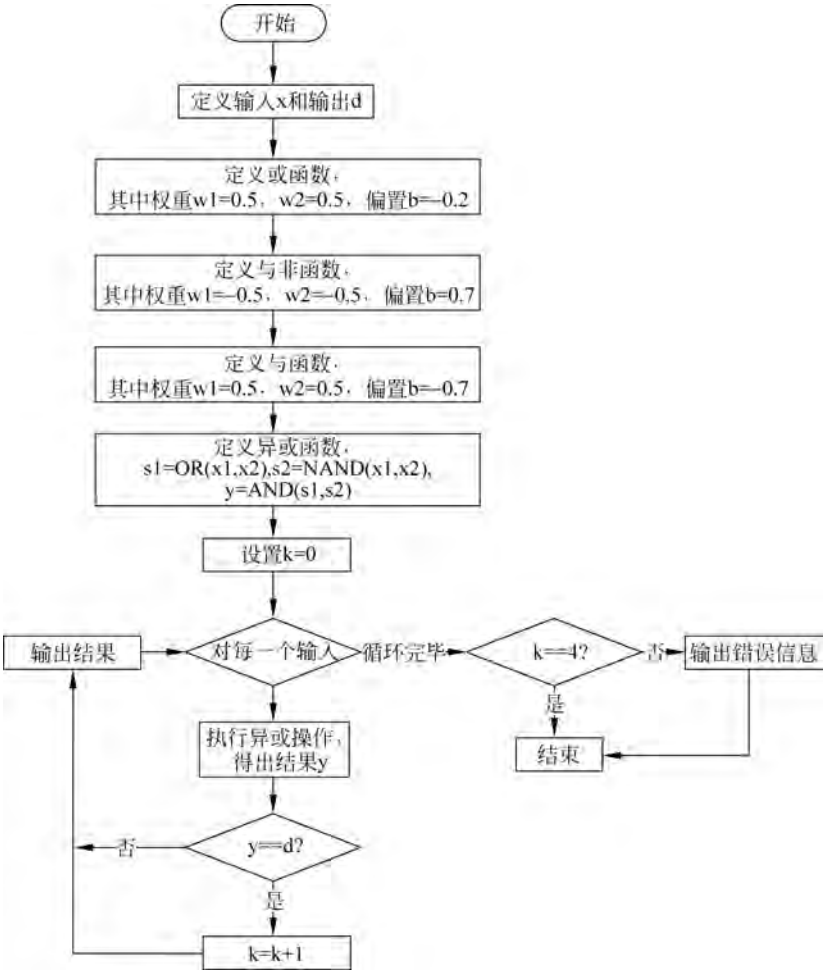

图 3-7　二层感知机实现与或门程序流程图

3.6　习题

1. 感知机模型主要用来解决什么问题？为什么会有单层感知机和多层感知机的区分呢？

2. 请简述感知机为什么不能处理异或问题。

3. 请根据感知机原始形式与对偶形式内容,说说两者之间的差异。

4. 感知机模型的局限性有哪些？应如何解决？

5. 请简述一种感知机改进算法。

6. 正样本点是 $x_1=(3,3)^{\mathrm{T}}$,$x_2=(4,3)^{\mathrm{T}}$,负样本点是 $x_3=(1,1)^{\mathrm{T}}$,试用感知机学习算法对偶形式求感知机模型,试用代码实现感知机算法的对偶形式。

7. 有 4 类 8 个输入模式,分别表示如下：

类别 1：$X^1=(1,1)^{\mathrm{T}}, X^2=(1,2)^{\mathrm{T}}$

类别 2：$X^3=(2,-1)^{\mathrm{T}}, X^4=(2,0)^{\mathrm{T}}$

类别 3：$X^5=(-1,2)^{\mathrm{T}}, X^6=(-2,1)^{\mathrm{T}}$

类别 4：$X^7=(-1,-1)^{\mathrm{T}}, X^8=(-2,-2)^{\mathrm{T}}$

设计一个感知机模型求解此问题，并设定学习训练速率 $\alpha=1$，初始连接权值和阈值分别为 $\boldsymbol{W}=\begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}$，$\theta=\begin{pmatrix}-1\\ -1\end{pmatrix}$，根据感知机学习算法训练该感知机。