神经网络之——感知器

主讲：刘丽珏

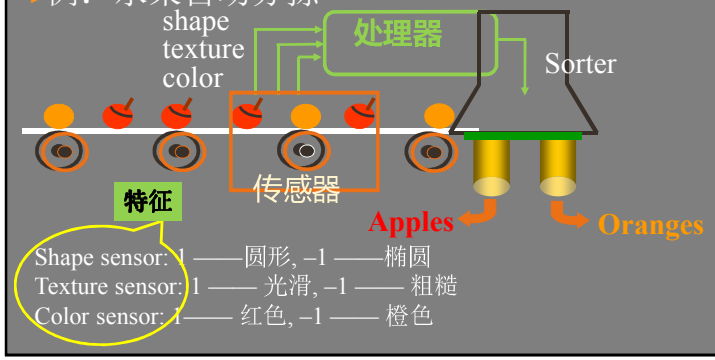
主要内容

- 引言
- 神经网络(Neural Network)概述
  - 生物神经网络简介
  - 从生物神经元到人工神经元
- 人工神经元
  - 基本结构
  - 权值与阈值
  - 逻辑运算
  - 线性可分与不可分
- 人工神经网络
- 感知器(Perceptron)

引言

传统程序怎么写？

- 例：水果自动分拣



引言

- 传统程序

  If *shape*=1 && *texture*=1 && *color*=1 then *apple*
  else *orange*

  OR

  If *shape*=1 && *texture*=1 && *color*=1 then *apple*
  elseif If *shape*=1 && *texture*=-1 && *color*=-1 *orange*

- 可能会出现什么问题？
  - 如果传送带上来了一个不圆的苹果，或者一个表面光滑的橘子会怎么样？

引言

- 换一种思路
  - 简便起见传感器信息用一个3维向量p表示，则

$$p = \begin{bmatrix} shape \\ texture \\ color \end{bmatrix} \quad p(apple) = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad p(orange) = \begin{bmatrix} 1 \\ -1 \\ -1 \end{bmatrix}$$

  - 令$w = [0 \quad 1 \quad 1]$
  - 代码改写成

if wp>=1 then *apple*
else *orange*

会发生什么？

思考题

- 若将苹果p=[1 1 1]^{T}和桔子p=[1 -1 -1]^{T}代入wp得到什么结果？
  - w=[0 1 1]　　权向量
- 假设传送带上送来了以下一些水果
  - 不圆的苹果p=[-1 1 1]^T
  - 光滑的桔子p=[1 1 -1]^T
  - 颜色偏红的桔子p=[1 -1 1]^T

会出现什么结果？

- 如果写成传统的IF-ELSE结构的代码需要写多少？
- 在苹果和桔子的分拣中起关键作用的因素是什么？

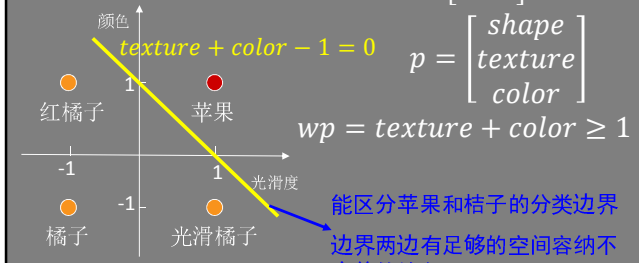
思考题解析

- 显然在分拣中起关键作用的是**w**和wp >=**1**中的**1**
- 由于形状特征的权值为0，我们仅考虑表面光滑度和颜色

w=[0 1 1]

$$p = \begin{bmatrix} shape \\ texture \\ color \end{bmatrix}$$

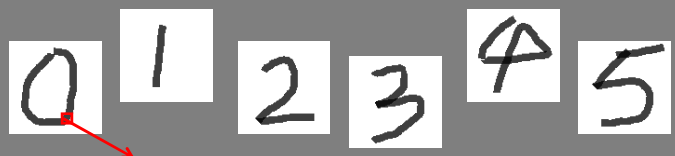
$$wp = texture + color \geq 1$$



能区分苹果和桔子的分类边界

边界两边有足够的空间容纳不完美的输入

引言——更复杂的例子



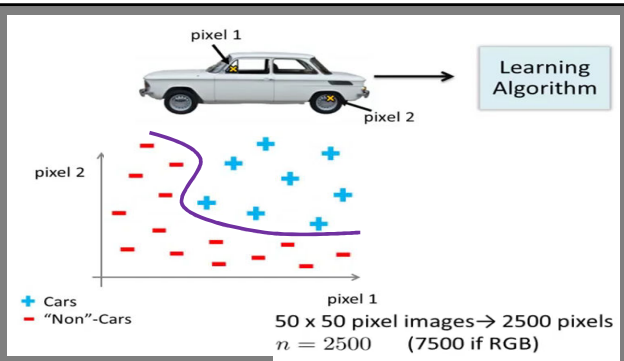
识别问题本质上就是“分类”

0.25098041	0.25098041	0.24705882	0.25098041	0.24705882
0.24705882	0.24705882	0.25098041	0.25098041	0.24705882
0.25098041	0.24705882	0.24705882	0.25098041	0.24705882
0.24705882	0.25098041	0.24705882	0.24705882	1
0.24705882	0.24705882	0.24705882	1	1
0.25098041	0.24705882	0.24705882	1	1
0.25098041	0.24705882	1	1	1
0.25098041	0.24705882	1	1	1
0.25098041	1	1	1	1
0.25098041	1	1	1	1

Computer Vision: Car detection

Testing:

What is this?



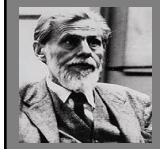
识别过程就是寻找合适的“分类边界”（决策边界）

$$x = \begin{bmatrix} \text{pixel 1 intensity} \\ \text{pixel 2 intensity} \\ \vdots \\ \text{pixel 2500 intensity} \end{bmatrix}$$

01 神经网络概述

(Neural network overview)

神经网络概述



麦克洛奇 (McCulloch)

皮茨 (Pitts)

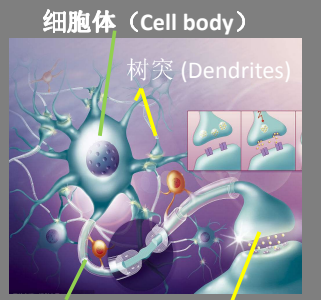
- McCulloch & Pitts (1943) 被公认为第一个人工神经网络的设计者 (MP model)
- 试图通过用计算模型模拟生物神经系统的方法来理解生物神经系统的工作模式
- 高度并行性使得其计算效率非常高
- 有助于理解神经表示的“分布式”特征
- 早期的“专家系统”是用大量“如果-就”(If - Then) 规则定义的，自上而下的思路
- 人工神经网络 (Artificial Neural Network)，标志着另外一种自下而上的思路

神经网络概述——成功案例

- 利用神经网络技术构建新型专家系统
- NETTALK vs. DECTALK
  - 两者均为英文发音软件
  - 任务是把书写的英文转化成英文发音
  - DECTALK
    - 由DEC公司开发的专家系统，其发音正确率达到95%
    - 开发用了超过20年的时间
    - 使用一个发音规则表，以及一个非常大的字典说明例外情况
  - NETTALK
    - 神经网络版本的DECTALK
    - 开发只用了一个暑假的时间
    - 经过16小时的训练后，阅读100个单词的正确率达到98%
    - 采用15,000个单词训练后，在测试集上的正确率为86%

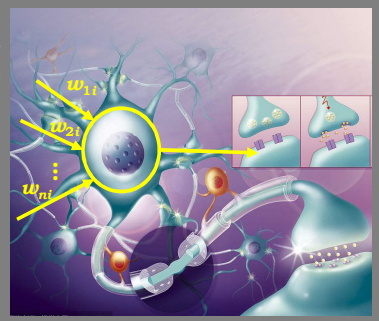
生物神经网络简介

- 生物神经元细胞结构
- 当树突将生物电信号传递到细胞体时，若引起细胞膜内和膜外的电位差超过阈值，则神经元进入兴奋状态，继续产生神经冲动传导下去
- 人脑中有10^{11}-10^{12}个神经元
- 每个神经元平均与10^{4}个其他神经元相连接形成神经网络



从生物神经元到人工神经元

- 生物神经元的突触可根据自身经验调整大小和连接强度
- 生物神经元的学习机制——Hebbian learning
  - “Neurons that fire together, wire together.”



02 人工神经元

(Artificial Neuron)

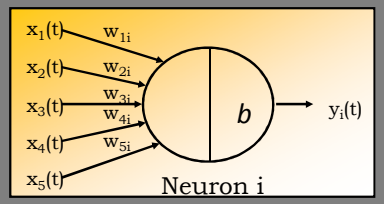
基本结构

刺激

$$u_i(t) = \sum_j w_{ij} \cdot x_j(t)$$

响应

$$y_i(t) = f(u_i(t) + b)$$



x_j =第j个输入，或称第j个特征
b = *threshold* (阈值)or *bias*(偏移)
$y_i(t)$ = output of neuron i at time t
w_{ij} = 从神经元i 到j的连接权值
f = *transfer function*(变换函数), or *activation function*(激励函数，激活函数)

思考题

- 回到苹果和桔子的分拣任务

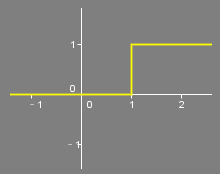
  if wp>=1 then *apple*
  else *orange*

- 请问这个和上面讲的人工神经元有什么联系呢？
  - w，p，和1分别对应着什么？
  - 在这个问题中建立的神经元采用了什么激活函数？

思考题解析

- 显然w，p，和1分别对应着人工神经元的权值、输入和阈值
- 在这个问题中建立的神经元采用的激活函数为

$$f(x) = \begin{cases} 1 & wp \geq 1 \\ 0 & wp < 1 \end{cases}$$

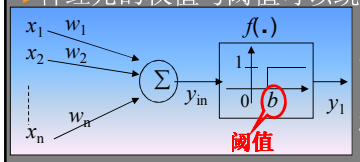


二值函数（阶跃函数）

二值函数可以做二分类——逻辑单元

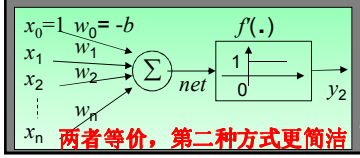
权值与阈值

- MP模型采用的激活函数为二值函数
  - 可进行逻辑运算
  - 又称阈值逻辑单元（ TLU，Threshold Logic Unit ）
  - 将不同的输入转换为0或1的输出
- 神经元的权值与阈值可以统一起来



$$y_{in} = \sum_{i=1}^{n} w_i x_i$$

$$y_1 = f(y_{in}) = \begin{cases} 1 & y_{in} \geq b \\ 0 & y_{in} < b \end{cases}$$

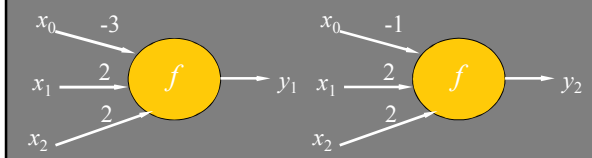


$$net = \sum_{i=0}^{n} w_i x_i$$

$$y_2 = f'(net) = \begin{cases} 1 & net \geq 0 \\ 0 & net < 0 \end{cases}$$

思考题

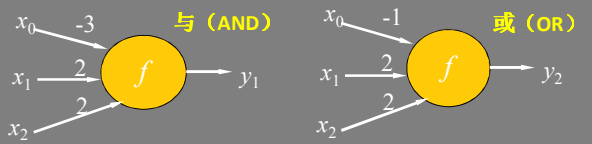
- 设输入向量分别为[1,0,1]，[1,1,0]，对于以下两个神经元，其输出分别是什么？



其中 $f(x)=\begin{cases}1 & x\geq 0\\ 0 & x<0\end{cases}$

思考题解析

- 设输入向量分别为[1,0,1]，[1,1,0]，对于以下两个神经元，其输出分别是什么？



x_0	x_1	x_2	WX^T	输出
1	0	1	$1\times(-3)+0\times2+1\times2=-1<0$	$y_1=0$
			$1\times(-1)+0\times2+1\times2=1>0$	$y_2=1$
1	1	0	$1\times(-3)+1\times2+0\times2=-1<0$	$y_1=0$
			$1\times(-1)+1\times2+0\times2=1>0$	$y_2=1$

逻辑运算

- TLU可进行逻辑运算
- 基本逻辑运算包括
  - 否定运算（非，┐，～）
  - 合取运算（与，∧）
  - 析取运算（或，∨，可兼或）

P	┐P
0(F)	1(T)
1(T)	0(F)

P	Q	P∧Q
0	0	0
0	1	0
1	0	0
1	1	1

P	Q	P∨Q
0	0	0
0	1	1
1	0	1
1	1	1

逻辑运算

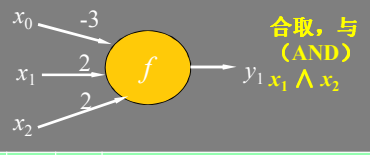
- 其他常用逻辑运算
  - 条件运算（如果…则，→）
  - 等值运算（当且仅当，↔）

P	Q	P→Q
0	0	1
0	1	1
1	0	0
1	1	1

P	Q	P↔Q
0	0	1
0	1	0
1	0	0
1	1	1

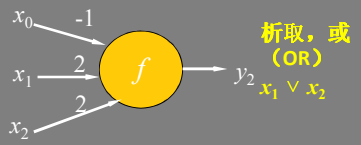
显然P↔Q ⇔（P→Q）∧（Q→P）

逻辑运算



x_0	x_1	x_2	WX^T	输出
1	0	0	1×(-3)+0×2+0×2=-3<0	0
1	0	1	1×(-3)+0×2+1×2=-1<0	0
1	1	0	1×(-3)+1×2+0×2=-1<0	0
1	1	1	1×(-3)+1×2+1×2=1>0	1

逻辑运算



x_0	x_1	x_2	WX^T	输出
1	0	0	1×(-1)+0×2+0×2=-1<0	0
1	0	1	1×(-1)+0×2+1×2=1>0	1
1	1	0	1×(-1)+1×2+0×2=1>0	1
1	1	1	1×(-1)+1×2+1×2=3>0	1

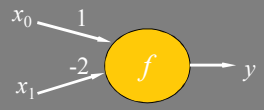
思考题

- 应该怎样用TLU实现一个逻辑否定运算呢？
  - 需要定义几个输入？
  - 权值或阈值应该有什么样的特征？
- 提示
  - 否定运算是一元运算

思考题解析

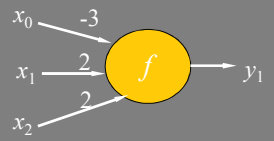
- 否定运算是一元运算，除了用来等价表示阈值的固定输入x_0=1外，只需一个输入x_1
- 分析否定运算的真值表

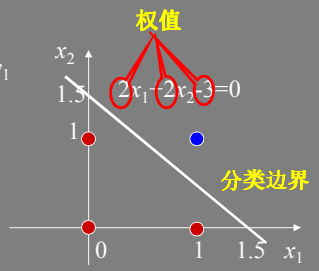
x_0	x_1	┐x_1	WX^T	条件
1	0	1	$w_0*1+w_1*0 \geq 0$	$w_0 \geq 0$
1	1	0	$w_0*1+w_1*1<0$	$w_0 < -w_1$



线性可分与不可分

- 通过实现逻辑运算，TLU到底做了什么？

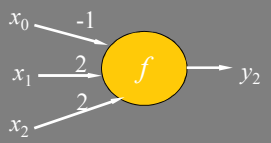


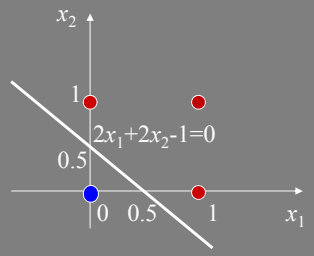


x_0	x_1	x_2	输出
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

线性可分与不可分

- 析取运算

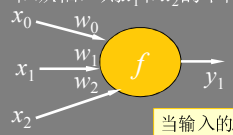


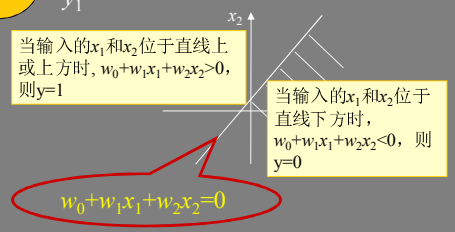


x_0	x_1	x_2	输出
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

线性可分与不可分

- TLU的基本功能为线性划分
- 当有两个输入x_1和x_2时，若将x_1和x_2分别看成平面上的横轴和纵轴，则x_1和x_2的不同值将对应该平面上的不同点





线性可分与不可分

- 对于两个输入，TLU通过权值阈值决定的直线，将平面上的点划分为两类，分别对应神经元的兴奋状态和抑制状态
- 对于三个输入，TLU则通过权值阈值决定的平面，将空间中的点划分为两类
- 对于多个输入来说，权值阈值对应的就是一个超平面，将超空间中的点进行分类
- 称能通过单个TLU解决的问题为线性可分的
- 与、或、非等简单逻辑都可通过单个TLU实现

线性可分与不可分

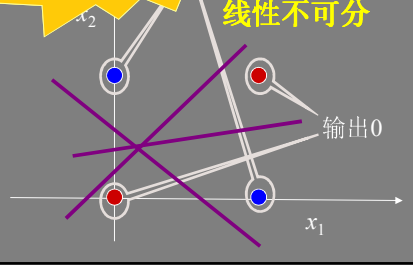
- 例：x_1表示选小王当班长，x_2表示选小李当班长
  - 怎样表示选小王或小李当班
    - $x_1 \vee x_2$?
    - NO
  - 异或问题（XOR

无论怎样，都不可能用一条直线将两类点分开

线性不可分

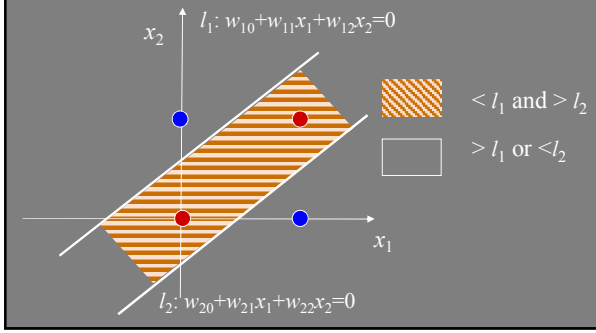
x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

异或逻辑真值表



线性可分与不可分

- 异或不是线性可分的，无法通过单个TLU实现
- 怎么办？
  - 多个神经元互连构成网络形成对空间的更复杂划分



线性可分与不可分

- 从另一个角度来看看异或
  - 选小王或小李当班长
  - 选小王但是不选小李当班长，或者不选小王但是选小李当班长

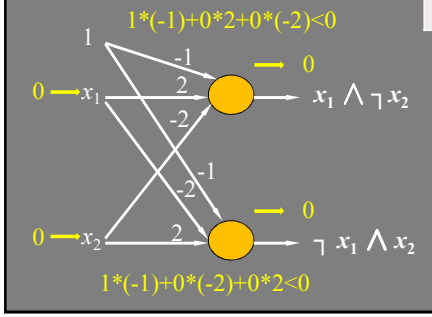
$(x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2)$

x_1	x_2	x_1 XOR x_2	$(x_1 \wedge \neg x_2) \vee (\neg x_1 \wedge x_2)$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

线性可分与不可分

- 异或问题第一层神经元设计

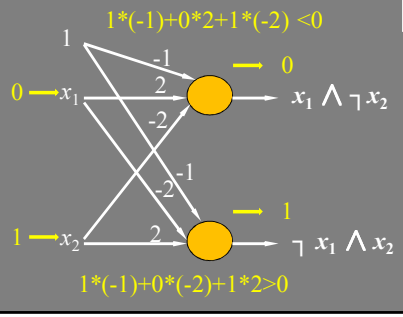
x_1	x_2	$x_1 \wedge \neg x_2$	$\neg x_1 \wedge x_2$
0	0	0	0
0	1		
1	0		
1	1		



线性可分与不可分

- 异或问题第一层神经元设计

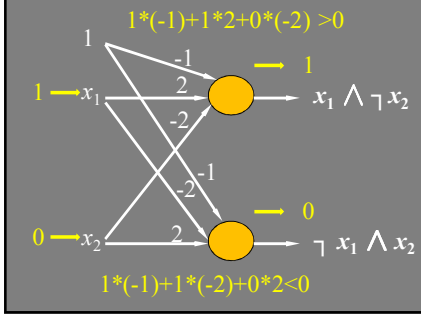
x_1	x_2	$x_1 \wedge \neg x_2$	$\neg x_1 \wedge x_2$
0	0	0	0
0	1	0	1
1	0		
1	1		



线性可分与不可分

- 异或问题第一层神经元设计

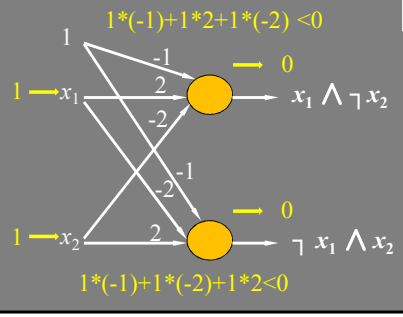
x_1	x_2	$x_1 \wedge \neg x_2$	$\neg x_1 \wedge x_2$
0	0	0	0
0	1	0	1
1	0	1	0
1	1		



线性可分与不可分

- 异或问题第一层神经元设计

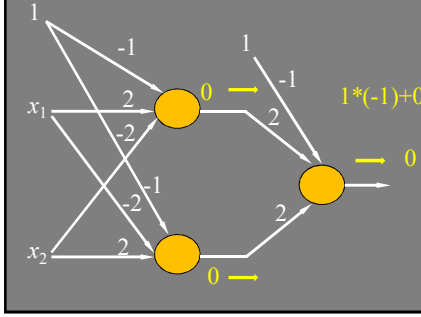
x_1	x_2	$x_1 \wedge \neg x_2$	$\neg x_1 \wedge x_2$
0	0	0	0
0	1	0	1
1	0	1	0
1	1	0	0



线性可分与不可分

- 异或问题第二层神经元设计

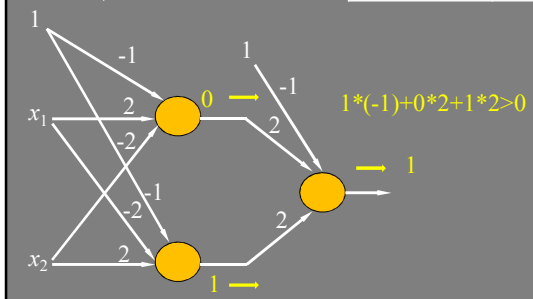
$X_1=x_1 \wedge \neg x_2$	$X_2=\neg x_1 \wedge x_2$	$X_1 \vee X_2$
0	0	0
0	1	
1	0	



线性可分与不可分

- 异或问题第二层神经元设计

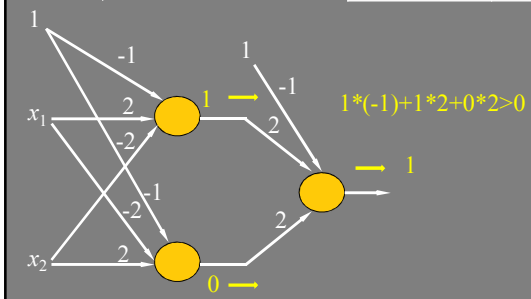
$X_1=x_1 \wedge \neg x_2$	$X_2=\neg x_1 \wedge x_2$	$X_1 \vee X_2$
0	0	0
0	1	1
1	0	



线性可分与不可分

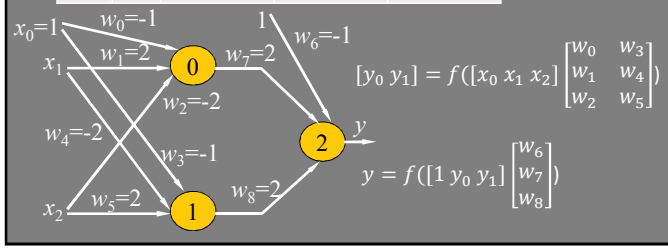
- 异或问题第二层神经元设计

$X_1=x_1 \wedge \neg x_2$	$X_2=\neg x_1 \wedge x_2$	$X_1 \vee X_2$
0	0	0
0	1	1
1	0	1



线性可分与不可分

x_1	x_2	$x_1 \wedge \neg x_2$	$\neg x_1 \wedge x_2$	x_1 XORx_2
0	0	0	0	0
0	1	0	1	1
1	0	1	0	1
1	1	0	0	0



$$[y_0\ y_1] = f([x_0\ x_1\ x_2]\begin{bmatrix} w_0 & w_3 \\ w_1 & w_4 \\ w_2 & w_5 \end{bmatrix})$$

$$y = f([1\ y_0\ y_1]\begin{bmatrix} w_6 \\ w_7 \\ w_8 \end{bmatrix})$$

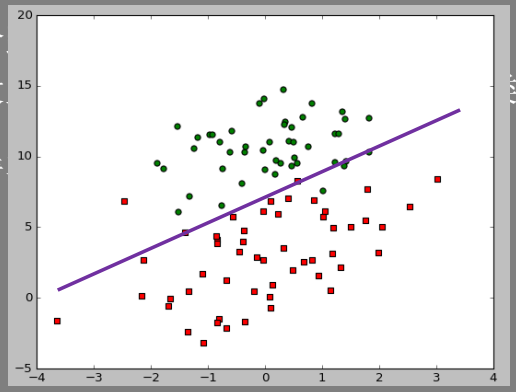
练习

- 编程实现用神经网络进行异或运算
- 要求：
  - 从键盘接收两个输入值，给出输出结果
- 提示
  - 将输入加上x_0=1一起用数组存储
  - 两层神经元的权值分别用两个权矩阵存储，第一层的为3×2的矩阵，第二层的为3×1的矩阵
  - 将输入向量与第一层的权矩阵进行矩阵相乘运算计算第一层的输出，注意对相乘后的矩阵用阶跃函数进行变换，即若结果矩阵中对应的元素大于等于0则将其变为1，否则为0
  - 将上一步的结果前面再插入一个输入1构成第二层的输入向量，再执行上述操作
- 思考
  - 试试输入[0.2 0.8]，看看你的神经网络会输出什么结果？为什么？
  - 多试试其他输入，看看会有什么结果
  - 如果用传统的IF…THEN来编写，这样的输入会导致什么结果？

思考

怎么找到这个分类边界？

- 采用阶
- 通过设
任务
- 如果情



03 人工神经网络

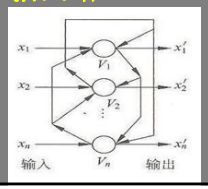
(Artificial Neural Network)

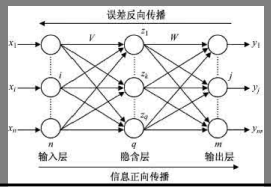
人工神经网络三要素

- 人工神经网络(Artificial Neural Network, ANN)是多个神经元以一定的拓扑结构互连组成的网络
- 神经网络三要素
  - 神经元
    - 输入、输出
    - 权值、阈值
    - 激活函数
  - 网络拓扑结构
  - 学习算法

神经网络的拓扑结构

- 递归（反馈）网络(*feedback network*，*recurrent network*)
  - 多个神经元互连组织成一个互连神经网络
- 前馈（多层）网络(*feedforward network*)
  - 具有递阶分层结构，同层神经元间不存在互连
  - 应用最广泛、影响最大的为**BP网络（反向传播网络）**

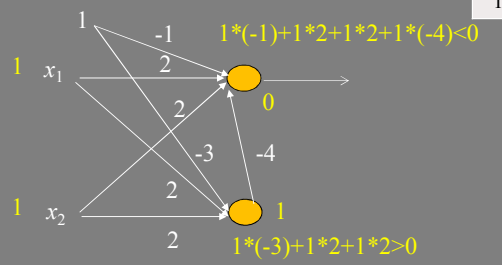




递归网络例

- 例：异或逻辑的递归网络实现

x_1	x_2	x_1 XOR x_2
0	0	0 √
0	1	1 √
1	0	1 √
1	1	0 √



人工神经网络的学习算法

- 神经学习是对人脑神经系统学习机理的一种模拟
- 人脑学习机理的两大学派
  - 化学学派
    - 认为人脑经学习所获得的信息是记录在某些生物大分子之上的，就像遗传信息记录在DNA上
  - 突触修正学派
    - 是人工神经网络学习和记忆机制研究的心理学基础
    - 人脑学习所获得的信息是分布突触连接上的
    - 学习和记忆过程是一个在训练中完成的突触连接权值的修正和稳定过程
    - 权值修正学派一直是人工神经网络研究的主流学派

人工神经网络的学习算法

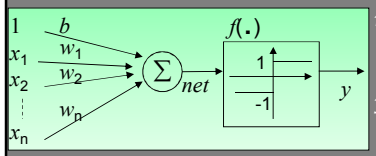
- 有监督学习（Supervised learning）
  - 能够根据期望的和实际的网络输出（对应于给定输入）间的差来调整神经元间连接的强度
- **无监督学习**（Unsupervised learning）
  - 不需要知道期望输出
- **增强学习**（Reinforcement learning）
  - 采用一个“评论员”来评 价与给定输入对应的神经网络输出的优度（质量因数）

04 感知器

(Perceptron)

感知器（Perceptron）

- 1958年Rosenblatt提出
  - 建立在MP神经元模型上，以解决线性可分的两类问题
  - 最简单的机器学习方法之一
  - 与MP稍有不同，两类记为{+1，-1}
  - 最简单的感知器



$$net = \sum_{i=1}^{n} w_i x_i + b$$

$$y = sgn(net) = \begin{cases} +1 & net \geq 0 \\ -1 & net < 0 \end{cases}$$

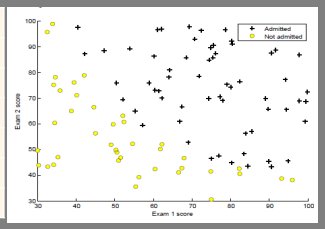
感知器的工作过程

- 数据准备阶段
  - 收集并标记数据
    - 根据处理的问题，确定需要收集哪些特征
    - 根据确定的特征收集数据
    - 对每个样本数据进行标注，即给出其正确分类
    - 将样本数据集分为训练集和测试集
- 学习训练阶段
  - 从大量已标注的样本数据中采用学习算法学习建立分类模型，确定每个神经元的权值和阈值
  - 对训练好的模型进行交叉验证
    - 用测试集中的数据对模型进行测试
- 正式投入使用

样本数据示例

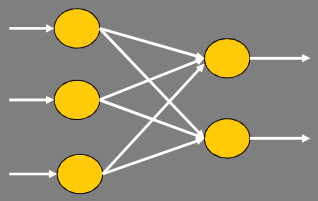
- 会录取我吗？
  - 根据两次考试的成绩预测申请的学校会不会录取我？
  - 两个特征，对应的神经元应有两个输入
    - 加上阈值，有三个输入

67.94685548	46.67857411	0
70.66150955	92.92713789	1
76.97878373	47.57596365	1
67.37202755	42.83843832	0
89.67677575	65.79936593	1
50.53478829	48.85581153	0
62.27101367	69.95445795	1



单层感知器与多层感知器

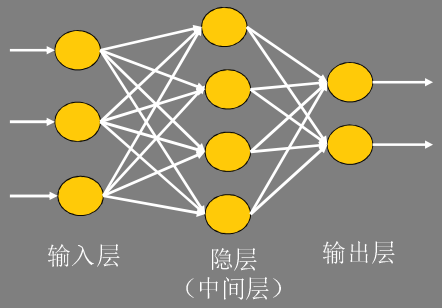
- 单层感知器



输入层　　输出层

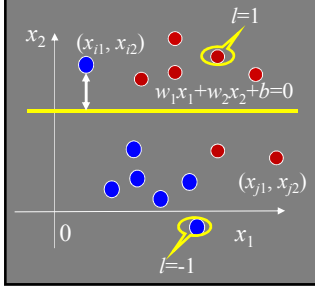
单层感知器与多层感知器

- 多层感知器



感知器学习

- 最简单的神经网络学习算法
- 有监督学习



分类错误点(x_{i1}, x_{i2})到分类边界的距离：

$$d_i = \frac{|w_1x_{i1} + w_2x_{i2} + b|}{\sqrt{{w_1}^2 + {w_2}^2}} = \frac{|WX_i^T + b|}{\|W\|}$$

其中$W = [w_1\ w_2]$，X_i=$[x_{i1}\ x_{i2}]$

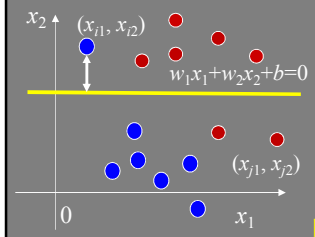
由于l_i=-1，$WX_i^T + b > 0$

所以$d_i = \frac{-l_i(WX_i^T + b)}{\|W\|}$

类似的分类错误点(x_{j1}, x_{j2})到分类边界的距离：

$$d_j = \frac{|WX_j^T + b|}{\|W\|} = \frac{-l_j(WX_j^T + b)}{\|W\|}$$

感知器学习



- 所有分类错误点到分类边界的距离之和：

$$L(W,b) = \sum_{X_i \in M} d_i = \frac{-1}{\|W\|} \sum_{X_i \in M} l_i(WX_i^T + b)$$

其中，M为分类错误点集合

目标：$L(W,b)$=0

当$\|W\| \neq 0$时，可简化为

$$L(W,\ b) = -\sum_{X_i \in M} l_i(WX_i^T + b)$$

损失函数

损失函数

- 设M是分类错误的点构成的集合，则
  - 对$X_i \in M$，$L(W,b)_i = -l_i(WX_i^T + b)$
  - 方便起见，令$b = w_0$，$X_i = [1\ X_i]$则上式简写成

$$L(W) = -\sum_{X_i \in M} l_i WX_i^T$$

其中$L(W)_i$是X_i的损失函数，$l_i \in \{-1,\ +1\}$是样本i的分类标签

损失函数

- 对$X_j \notin M$，令$L(W)_j = 0$
  - 注意到，此时$-l_j W {X_j}^T < 0$
- 综上，对样本集中任一样本X_k，

  $L(W)_k = \max(0, -l_k W {X_k}^T)$
- 对所有m个训练样本，损失函数

$$L(W) = \sum_{k=1}^{m} L(W)_k = \sum_{k=1}^{m} \max(0, -l_k W {X_k}^T)$$

损失函数与结构风险

- 一般来说，在进行有监督学习任务时，使用的每一个算法都有一个目标函数，算法便是对这个目标函数进行优化
- 目标函数一般为两部分之和
  - 损失函数
    - 经验风险函数，用来评价模型的预测值$Y'=f(X)$与真实值Y的不一致程度
    - 不同的分类算法其损失函数的定义不同
  - 正则项
    - 提高泛化能力
- 由损失项(Loss term)加上正则项(Regularization term)构成结构风险形成目标函数

感知器学习

- 目标函数只考虑经验风险
  - $L(W) = \sum_{k=1}^{m} \max(0, -l_k W {X_k}^T)$
- 学习条件和目的
  - 我们有什么？
    - 已标记的大量样本数据
  - 要做什么？
    - 求得能使目标函数最小化的W值
- 怎么做？
  - 不断的利用实际输出和期望输出之间的误差来调整权值

感知器学习

- 设l_i和y_i分别是第i个样本数据的标签和当前感知器的输出，则存在四种不同情况：
  - 第一种：l_i=+1，y_i=+1，分类正确，W_{t+1}=W_t
  - 第二种：l_i=-1，y_i=-1，分类正确，W_{t+1}=W_t
  - 第三种：l_i=+1，y_i=-1，分类错误，$W_{t+1} \neq W_t$
    - 此时，$net = W {X_i}^T < 0$，需要增大net
    - 令$W_{t+1} = W_t + \eta X_i = W_t + \eta l_i X_i$，$\eta \in (0,1]$学习率，则

      $net_{t+1} = W_{t+1} {X_i}^T = W_t {X_i}^T + \eta \|X_i\|^2 > net_t$

感知器学习

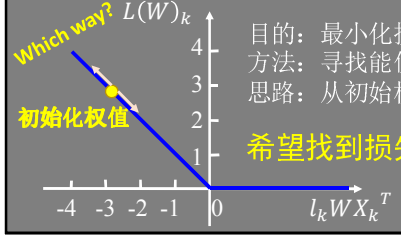
- 设l_i和y_i分别是第i个样本数据的标签和当前感知器的输出，
  - 第四种： l_i=-1， y_i=+1，分类错误，$W_{t+1} \neq W_t$
    - 此时， $net = WX_i^T > 0$，需要减小net
    - 令$W_{t+1} = W_t - \eta X_i = W_t + \eta l_i X_i$，则
      $net_{t+1} = W_{t+1}X_i^T = W_t X_i^T - \eta \|X_i\|^2 < net_t$

感知器学习

- 总结一下四种情况
  - 分类正确的情况
    $y_i l_i$>0，W不需要调整
  - 分类错误的情况
    $y_i l_i$<0， $W_{t+1} = W_t + \eta l_i X_i$
- 例
  - 设$X_i = [1\ x_1\ x_2]$, $W_t = [w_0\ w_1\ w_2]$,
    $l_i = -1$， $\eta = 1$， $y_i = 1$，则
    $W_{t+1} = [w_0\ w_1\ w_2] - [1\ x_1\ x_2]$
    $= [w_0 - 1 \quad w_1 - x_1 \quad w_2 - x_2]$

梯度与梯度下降

- Why?
  - 为什么是$l_i X_i$？跟损失函数有什么关系？
  - η是干什么的？有什么作用？
- 再看损失函数，对第k个样本数据
  - $L(W)_k = \max(0, -l_k W X_k^T)$， $l_k \in \{-1, +1\}$

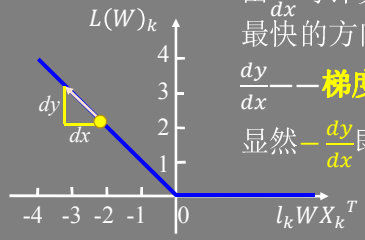


目的：最小化损失函数
方法：寻找能使损失函数取最小值的W
思路：从初始权值开始，不断迭代调整

希望找到损失函数下降最快的方向

梯度与梯度下降——Why $l_i X_i$？

- 如何获得损失函数下降最快的方向？
  - 已知损失函数方程



由$\frac{dy}{dx}$可计算斜率，得到上升最快的方向

$\frac{dy}{dx}$——**梯度，**也就是**导数**

显然$-\frac{dy}{dx}$即下降最快的方向

梯度与梯度下降——Why l_iX_i？

- 权值更新公式

$$W_{t+1} = W_t + \eta l_i X_i$$

  - 先考虑简单情况，X_i与W均只有一维，即输入特征只有一个，不考虑阈值（假设其为0），上式可简化为

$$w_{t+1} = w_t + \eta l_i x_i$$

- 已经得知应该沿梯度下降的方向更新权值
  - 损失函数

$$L(w)_k = \max(0, -l_k w x_k{}^T) = \max(0, -l_k w x_k)$$

$$-\frac{d(L(w)_k)}{dw} = \frac{d(l_k w x_k)}{dw} = l_k x_k$$

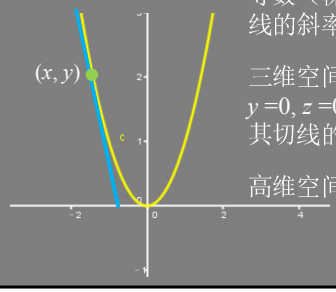
梯度下降的方向

梯度与梯度下降——Why l_iX_i？

- 若$W = [w_0 \quad w_1 \quad \ldots]$是多维向量，则求损失函数对每一维$w_i$的偏导数作为该维的梯度
- 例
  - 设第i个样本$X_i = [1\ x_1\ x_2]$, $W_t = [w_0\ w_1\ w_2]$
    - $L(W)_i = -l_i W X_i{}^T = -l_i(w_0 + w_1 x_1 + w_2 x_2)$
    - $\frac{\partial L(W)_i}{\partial w_0} = \frac{\partial(-l_i(w_0+w_1x_1+w_2x_2))}{\partial w_0} = -l_i = -l_i x_0$
    - $\frac{\partial L(W)_i}{\partial w_1} = \frac{\partial(-l_i(w_0+w_1x_1+w_2x_2))}{\partial w_1} = -l_i x_1$
    - $\frac{\partial L(W)_i}{\partial w_2} = \frac{\partial(-l_i(w_0+w_1x_1+w_2x_2))}{\partial w_2} = -l_i x_2$
  - 即
    - w_0=$w_0 + l_i x_0$，w_1=$w_1 + l_i x_1$，w_2=$w_2 + l_i x_2$(设$\eta = 1$)
    - 写成矩阵形式$W_{t+1} = W_t + l_i X_i$

梯度与梯度下降

- 更一般的情况
- 若损失函数图像如下



导数（梯度）——过(x, y)点的切线的斜率

三维空间中偏导数即函数在x =0, y =0, z =0这三个面上的投影曲线，其切线的斜率

高维空间类似

梯度与梯度下降——Why η？

- 我们已经知道了权值调整的方向，但是每次调整多少呢？
- η决定了每次调整的步长，$\eta \in (0,1]$
  - 步长太大，会导致迭代过快，甚至有可能错过最优解
  - 步长太小，迭代速度太慢，收敛速度慢
  - 取值取决于数据样本，是感知器学习中需要不断试验调整的参数

梯度与梯度下降

- 感知器学习采用随机梯度下降来最小化损失函数
  - 每次用训练样本中的一个样例的梯度来更新权值
- 感知器学习算法
  - 输入：给定正例集合P和反例集合N，对所有$x \in P$，$f(x)=1$，所有$x \in N$，$f(x)=-1$，$x \in R^n$
  - 输出：$w \in R^n$

1.Initialize weights to

$$w = \sum_{x \in P} x - \sum_{x \in N} x, \qquad \eta = random \quad (0,1)$$

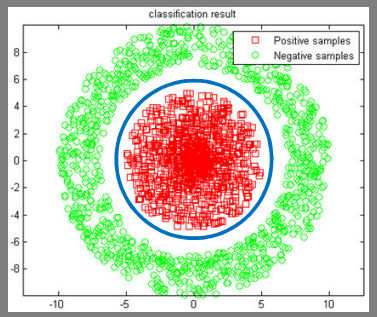
2.随机（顺序）选择 $x \in P \cup N$

3.If $lwx<0$ Update $w = w + \eta lx$（η为学习常数，l为期望输出）

4.Goto 2 until outputs of all training examples are correct

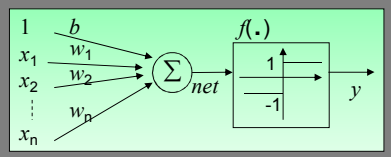
感知器的局限性

- 单层感知器只能解决线性可分问题
- 对于多层网络，感知器学习不再适用
  - 隐层节点没有所谓“期望输出”
  - 阶跃函数不可导



回到神经元

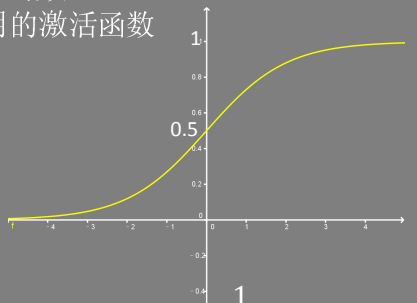
- 感知器的神经元是TLU，激活函数决定了它只能做二分类



- 要处理非线性可分数据，必须令输出能成为输入的非线性函数
- 且使用梯度下降方法，该函数必须连续可导

其他常用激活函数

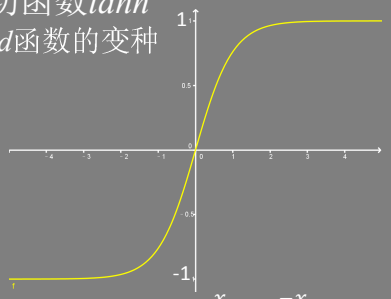
- *Sigmoid*函数
  - 最常用的激活函数



$$f(x) = \frac{1}{1+e^{-x}}$$

其他常用激活函数

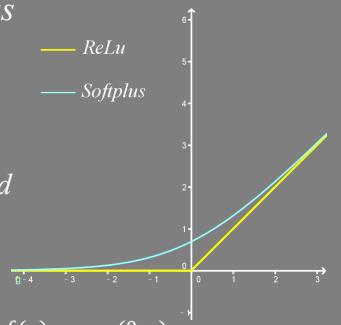
- 双曲正切函数*tanh*
  - *Sigmoid*函数的变种



$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

其他常用激活函数

- *ReLu*函数和*Softplus*函数
  - *ReLu*——自适应线性单元（Rectified linear unit）
  - *Softplus*——*Sigmoid*的原函数，其导数为*Sigmoid*函数



$ReLu$: $f(x) = \max(0, x)$

$Softplus$: $f(x) = \ln(1 + e^x)$