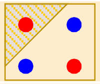
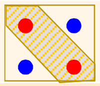
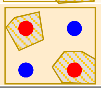
神经网络之——BP网络

主讲：刘丽珏

主要内容

引言

- 单层感知器只能进行线性可分的运算
- 多层感知器可解决线性不可分问题

结构	决策面类型	异或问题
无隐层	单一超平面	
单隐层	凸区域	
双隐层	任意形状	

引言

- 双隐层感知器就足以解决任何复杂的分类问题
- 但是
  - 隐层的权值怎么训练
  - 隐层节点不存在期望输出
  - 感知器学习
    - $y_i l_i<0$，$W_{t+1}=W_t+\eta l_i X_i$，其中y_i是计算输出，l_i是期望输出
  - 无法通过感知器的学习规则来训练多层感知器

引言

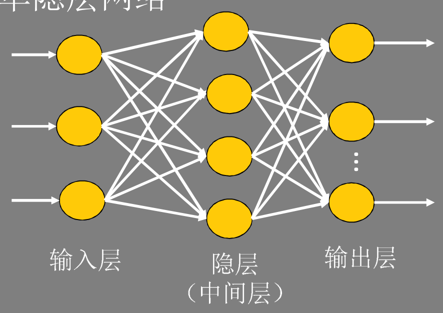
- 1966年，Minisky和Papert在他们的《感知器》一书中指出
  - 理论上还不能证明将感知器模型扩展到多层网络是有意义的
- 直接导致ANN的研究陷入低谷
- 上世纪八十年代，对ANN的研究开始复兴
  - 1982年美国加州理工学院的物理学家John J.Hopfield博士的Hopfield网络
  - David E.Rumelhart以及James L.McCelland发表的《并行分布式处理》
    - 提出了BP网络

01 认识BP网络

(BP network overview)

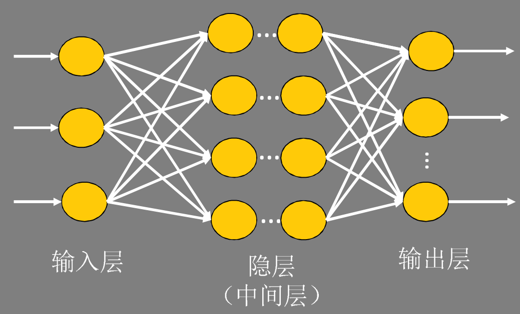
BP网络结构

- 多层前馈网络
- 单隐层网络



BP网络结构

- 多隐层网络
  - 层与层之间为全互连结构



BP网络结构

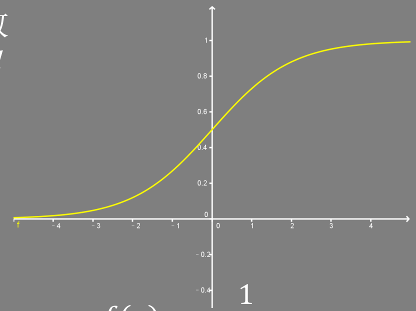
- BP网络的输入层和输出层节点个数确定
  - 输入个数根据特征的维度来定
  - 输出个数根据要进行分类的类别数来定
- 如何确定隐层节点的个数？
  - 经验公式

$$h = \sqrt{m+n} + a$$

其中h为隐层节点的个数，m为输入个数，n为输出个数，a为1~10之间的调节常数

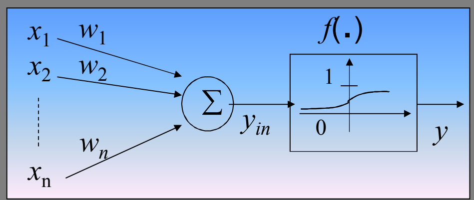
*Sigmoid*单元

- BP网络的神经元不再是线性阈值单元
- 激活函数
  - *Sigmoid*



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

*Sigmoid*单元

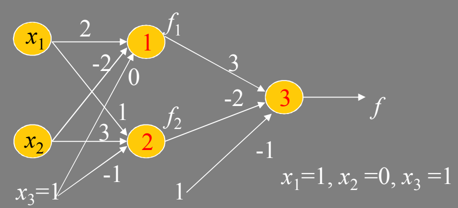


$$y_{in} = \sum_{i=1}^{n} w_i x_i = WX^T$$

$$y = f(y_{in}) = \frac{1}{1+e^{-y_{in}}}$$

正向传播过程

- 输入数据从输入层传播到输出层的过程为正向传播过程



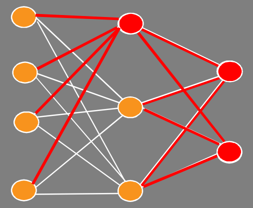
$f_1=s(1\times2+0\times(-2)+1\times0)=s(2)=1/(1+e^{-2})=0.881$

$f_2=s(1\times1+0\times3+1\times(-1))=s(0)=1/(1+e^{0})=0.5$

$f=s(0.881\times3+0.5\times(-2)+1\times(-1))=s(0.643)=0.655$

反向传播过程

- 从输出节点开始，反向地向第一隐含层传播由总误差引起的权值修正
  - 首先计算输出层单元的误差，并用该误差调整输出层的权值
  - 根据输出层的误差计算隐层单元的误差



02 BP学习算法

(Back Propagation)

基本思想

- 多层感知器在如何获取隐层的权值的问题上遇到了瓶颈，无法直接得到隐层的权值
- 能否先通过输出层得到输出结果和期望输出的误差，间接调整隐层的权值
- BP学习过程
  - 正向传播时，输入样本从输入层经各隐层逐层处理后,传向输出层。若输出层的实际输出与期望的输出不符,则转入误差的反向传播阶段
  - 反向传播时，将输出以某种形式通过隐层向输入层逐层反传,并将误差分摊给各层的所有单元,获得各层单元的误差信号作为修正各单元权值的依据

误差与损失函数

- 误差定义

$$E(w)=\frac{1}{2}\sum_{d\in D}(t_d-o_d)^2$$

其中，D为训练样例集合；t_d为训练样例d的期望输出；o_d为训练样例d的实际输出

- 以误差作为损失函数的经验风险，在不考虑正则项的情况下，就以误差函数作为损失函数

误差与损失函数

- 学习目的
  - 最小化误差

$$\min E(w)=\frac{1}{2}\sum_{d\in D}(t_d-o_d)^2$$

- 学习方法
  - 梯度下降

$$\Delta w_{ij}=-\eta\frac{\partial E_d}{\partial w_{ij}}$$

梯度

其中E_d为训练样本d的误差，
w_{ij}与单元j的第i个输入相关连的权值

$$w_{ij}(t+1)=w_{ij}(t)+\Delta w_{ij}$$

练习

- 对于在“会录取我吗？”例子中的7个样本数据，每个样本有两个特征

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

- 请建立一个3个输入（包括阈值单元），1个输出，1个隐层，隐层包括3个节点的BP网络，编程实现：
  - 随机初始化权值
  - 计算误差并输出

梯度下降

- 求导的链式法则
  - 设x是实数，f和g是从实数映射到实数的函数，$y=g(x)$并且$z=f(g(x))=f(y)$，则

$$\frac{dz}{dx}=\frac{dz}{dy}\frac{dy}{dx}$$

  - 将该法则扩展到向量

若$x\in R^m$，即$x=(x_1,x_2,\ldots x_m)$

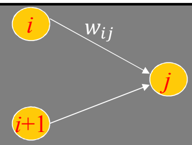
$y\in R^n$，即$y=(y_1,y_2,\ldots y_n)$

$g:R^m\to R^n, f:R^n\to R$

如果$\mathbf{y}=g(\mathbf{x})$并且$z=f(\mathbf{y})$，那么

$$\frac{\partial z}{\partial x_i}=\sum_{j=1}^{n}\frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_i}$$

梯度下降



- 计算梯度

$$\frac{\partial E_d}{\partial w_{ij}}=\frac{\partial E_d}{\partial net_j}\cdot\frac{\partial net_j}{\partial w_{ij}}$$

其中$net_j=\sum_i w_{ij}x_{ij}$，为$j$单元的输入

$$\frac{\partial net_j}{\partial w_{ij}}=\frac{\partial\sum_i w_{ij}x_{ij}}{\partial w_{ij}}=x_{ij}$$

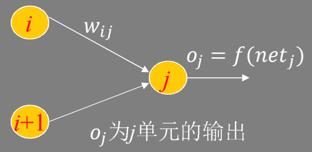
$$\frac{\partial E_d}{\partial w_{ij}}=\frac{\partial E_d}{\partial net_j}\cdot x_{ij}=\frac{\partial E_d}{\partial o_j}\cdot\frac{\partial o_j}{\partial net_j}\cdot x_{ij}$$

其中$o_j=f(net_j)$

$$\frac{\partial f(net_j)}{\partial net_j}=\frac{\partial}{\partial net_j}\left(\frac{1}{1+e^{-net_j}}\right)=o_j(1-o_j)$$

梯度下降

ZIXING AI 自兴人工智能



- 计算$\frac{\partial E_d}{\partial o_j}$
  - 情况1：j为输出单元

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j}\frac{1}{2}\sum_{k\in outputs}(t_k - o_k)^2$$

除了k=j外，其他$\frac{\partial}{\partial o_j}(t_k - o_k)^2 = 0$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j}\frac{1}{2}\left(t_j - o_j\right)^2$$
$$= \frac{1}{2}\times 2(t_j - o_j)\frac{\partial}{\partial o_j}(t_j - o_j)$$
$$= -(t_j - o_j)$$

梯度下降

ZIXING AI 自兴人工智能

- 情况1：j为输出单元时

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j}\cdot\frac{\partial o_j}{\partial net_j} = -(t_j - o_j)o_j(1 - o_j)$$

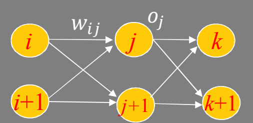
令 $\frac{\partial E_d}{\partial net_j} = -\delta_j$

则：$\frac{\partial E_d}{\partial w_{ij}} = \frac{\partial E_d}{\partial net_j}\cdot x_{ij} = -\delta_j x_{ij}$

$$\Delta w_{ij} = -\eta\frac{\partial E_d}{\partial w_{ij}} = \eta\delta_j x_{ij}$$

梯度下降

ZIXING AI 自兴人工智能



- 计算$\frac{\partial E_d}{\partial o_j}$
  - 情况2：j为隐层单元

$$\frac{\partial E_d}{\partial o_j} = \sum_k \frac{\partial E_d}{\partial net_k}\cdot\frac{\partial net_k}{\partial o_j}$$
$$= \sum_k \frac{\partial E_d}{\partial net_k}\cdot\frac{\partial(w_{1k}o_1}{}$$
$$= \sum_k \frac{\partial E_d}{\partial net_k}\cdot w_{jk}$$

链式法则：
若$x\in R^m$，$y\in R^n$，
$g: R^m\to R^n, f: R^n\to R$
若$\boldsymbol{y} = g(\boldsymbol{x})$，且$z = f(\boldsymbol{y})$，则

$$\frac{\partial z}{\partial x_i} = \sum_{j=1}^{n}\frac{\partial z}{\partial y_j}\frac{\partial y_j}{\partial x_i}$$

梯度下降

ZIXING AI 自兴人工智能

- 情况2：j为隐层单元

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j}\cdot\frac{\partial o_j}{\partial net_j} = \sum_k \frac{\partial E_d}{\partial net_k} w_{jk} o_j(1 - o_j)$$

前面定义了 $\frac{\partial E_d}{\partial net_j} = -\delta_j$

则$\frac{\partial E_d}{\partial net_j} = \sum_k -\delta_k\, w_{jk} o_j(1 - o_j)$

k为所有与j连接的下游神经元

$$\frac{\partial E_d}{\partial w_{ij}} = -\delta_j\cdot x_{ij} = \sum_k -\delta_k\, w_{jk} o_j(1 - o_j)x_{ij}$$

$$\Delta w_{ij} = -\eta\frac{\partial E_d}{\partial w_{ij}} = \eta\delta_j x_{ij}$$

梯度下降

- 总结一下，BP算法中权值的修正公式

$$\Delta w_{ij} = \eta \delta_j x_{ij}$$

  - 情况1：j为输出单元时

$$\delta_j = (t_j - o_j) o_j (1 - o_j)$$

  - 情况2：j为隐层单元

$$\delta_j = o_j (1 - o_j) \sum_k \delta_k w_{jk}$$

其中η为学习常数，t_j是j单元的期望输出，o_j是j单元的计算输出，δ_j是j单元的误差

例

- 首先计算输出层单元的误差，并用该误差调整输出层的权值

Current output: o_j=0.2

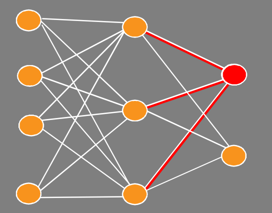
Correct output: t_j=1.0

Error $\delta_j = o_j(1-o_j)(t_j-o_j)$

0.2(1–0.2)(1–0.2)=0.128

Update weights into j

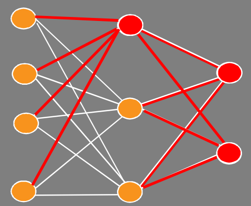
$$\Delta w_{ij} = \eta \delta_j x_{ij}$$



input hidden output

例

- 接着根据输出层的误差计算隐层单元的误差



input hidden output

$$\delta_j = o_j (1 - o_j) \sum_k \delta_k w_{jk}$$

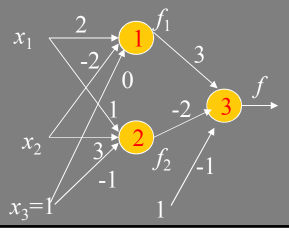
反向传播算法

- BP算法
  1. 初始化权值及阈值为小的随机数
  2. 读入样本集x_0，$x_1 \ldots x_{n-1}$及期望输出t_0，t_1，t_{n-1}
  3. 对训练集中每一样本
     ① 前向计算隐层、输出层各神经元的输出
     ② 计算期望输出与网络输出的误差
     ③ 反向计算修正网络权值和阈值
  4. 若满足精度要求或其他退出条件，则结束训练，否则转步骤3继续

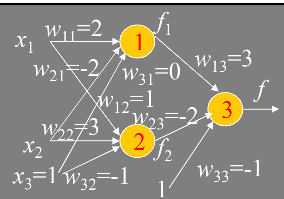
BP网络应用例

- 奇偶校验网络
  - 用下图BP网络实现奇偶校验(当前参数为随机初始化的结果)，该网络除阈值输入外有两个输入，若输入数据中有奇数个1，网络输出0，否则输出1，训练数据见表

x_1	x_2	x_3	t
1	0	1	0
0	0	1	1
0	1	1	0
1	1	1	1



BP网络应用例



- 第一次训练
  - 训练数据

$x_1=1, x_2=0, x_3=1, t=0$，$\eta=1$

$f_1=s(1\times2+0\times(-2)+1\times0)=s(2)=1/(1+e^{-2})=0.881$

$f_2=s(1\times1+0\times3+1\times(-1))=s(0)=1/(1+e^{0})=0.5$

$f=s(0.881\times3+0.5\times(-2)+1\times(-1))=s(0.643)=0.655$

$\delta=0.655\times(1-0.655)\times(0-0.655)=-0.148$

$\delta_1=0.881\times(1-0.881)\times(-0.148\times3)=-0.047$

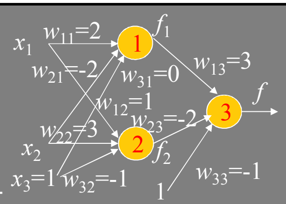
$\delta_2=0.5\times(1-0.5)\times(-0.148\times-2)=0.074$

$w_{13}=3+0.881\times(-0.148)=2.870$

$w_{23}=-2+0.5\times(-0.148)=-2.074$

$w_{33}=-1+1\times(-0.148)=-1.148$

BP网络应用例



- $x_1=1, x_2=0, x_3=1, t=0$，$\eta=1$
- $f_1=0.881$，$f_2=0.5$，$f=0.655$

$\delta=-0.148$，$\delta_1=-0.047$，$\delta_2=0.074$

$w_{11}=2+1\times(-0.047)=1.953$

$w_{21}=-2+0\times(-0.047)=-2$

$w_{31}=0+1\times(-0.047)=-0.047$

$w_{12}=1+1\times0.074=1.074$

$w_{22}=3+0\times0.074=3$

$w_{32}=-1+1\times0.074=-0.926$

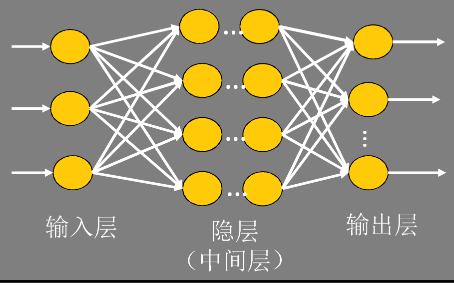
多分类问题

- 需要区分的类别超过两类时，输出节点的数量=需要区分的类别数
- 例：鸢尾花数据集（iris flower dataset）
  - 数据示例

萼片长度	萼片宽度	花瓣长度	花瓣宽度	类别
5.3cm	3.7cm	1.5cm	0.2cm	Iris-setosa (山鸢尾)
7cm	3.2cm	4.7cm	1.4cm	Iris-versicolor (杂色鸢尾)
6.3cm	3.3cm	6cm	2.5cm	Iris-virginica (维吉尼亚鸢尾)

多分类网络

- 输出层有多个神经元，其个数等于类别数
- 输出值为一个向量$(y_1, y_2, \ldots, y_n)$，其中n为类别数，y_i中只有一个为1，其余均为0，表示属于第i类



例

- 鸢尾花识别BP网络
  - 输入层
    - 4个输入
    - 分别对应萼片长度、萼片宽度、花瓣长度和花瓣宽度四个特征
  - 隐层
    - 1个隐层
    - 包含10个神经元
  - 输出层
    - 3个神经元
    - 对应3种类别的输出分别为(1,0,0)，(0,1,0)，(0,0,1)