# 第六章 多层神经网络

# Multilayer Neural Networks

## 6.1 引 言

中山大學

- **神经网络定义**

　　一种在训练线性判别函数的同时学习其非线性程度的方法,这就是多层神经网络或多层感知神经网络.

　　多层神经网络的本质是，基本上执行线性判决，只是该执行过程是在输入信号的非线性映射空间进行的。它允许非线性函数的具体形式可以通过训练样本得到。

- **一种流行的训练多层网络方法**

　　反向传播算法(Back Propagation),简称为”反传算法”或”BP算法”.

- **网络的结构或拓扑在神经网络的分类中起着重要作用**

　　通过设置网络拓扑来选择模型,通过反向传播算法来估计参数。

2

- **正则化(regularization技术)**

　　就是选择或调整整个网络复杂程度的技术。

　　如果网络有太多的未知参数，则网络推广能力将变得很差（即，过拟合）。相反，如果网络参数太少，训练数据将得不到充分的学习。



## 6.2　前馈运算和分类

- 三层神经网络由一个输入层、一个隐含层、一个输出层组成。它们由可修正权值互连，这些权值由层间的连线表示。
- 两种“神经元（neuron）”：输入单元提供特征量，输出单元激发出用来分类的判别函数的值。

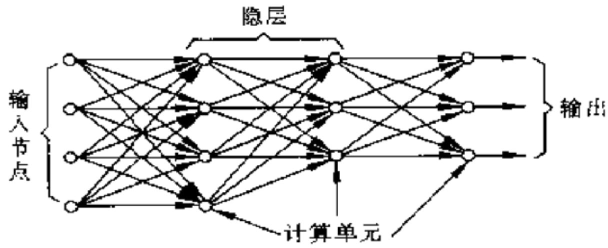
■ **前馈运算定义：**

a. 无反馈，可用一有向无环图表示。

b. 图的节点分为三类：输入节点、计算节点和输出节点。

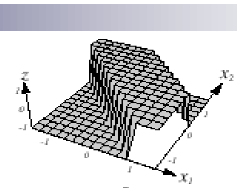
c. 每个计算单元可有任意个输入，但只有一个输出，而输出可耦合到任意多个其他节点的输入。前馈网络通常分为不同的层，第i层的输入只与第i-1层的输出相联。

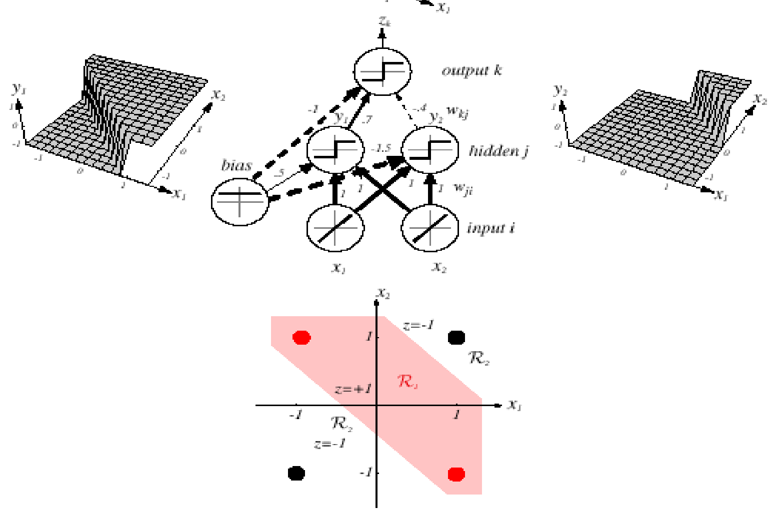
d. 输入和输出节点由于可与外界相连，直接受环境影响，称为可见层，而其他的中间层则称为隐层。如图。




5

异或问题（XOR）的例子：





- 除了连接输入单元，每个单元还连接着一个偏置。
- 净激活: $net_j = \sum_{i=1}^{d} x_i w_{ji} + w_{j0} = \sum_{i=0}^{d} x_i w_{ji} \equiv w_j^t.x,$
- 其中，下标i是输入层单元索引值，j是隐含层单元的索引值; $w_{ji}$ 表示输入层单元i到隐含层单元j的权值。（该权值也称为“突触权”。）
- 每个隐含层单元激发出一个输出分量，这个分量是它激活的非线性函数: $y_j = f(net_j)$
- 对于图6-1示例：

$$f(net) = sgn(net) \equiv \begin{cases} 1 \ \ if \ \ net \ge 0 \\ -1 \ \ if \ \ net < 0 \end{cases}$$



- 这个函数 $f(.)$ 称为激活函数（activation functions）或者“非线性”单元.
- 每个输出单元在隐含层单元信号的基础上用类似的方法算出它的净激活如下：

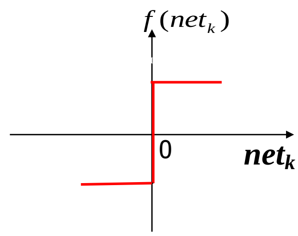
$$net_k = \sum_{j=1}^{n_H} y_j w_{kj} + w_{k0} = \sum_{j=0}^{n_H} y_j w_{kj} = w_k^t.y,$$

其中，下标k为输出层的单元索引， $n_H$ 表示隐含层单元的数目。

中山大學

- 符号函数：

$$f(net_k)=\begin{cases}1 & if\ net_k \ge 0\\ -1 & if\ net_k < 0\end{cases}$$

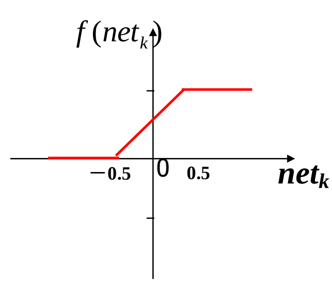





中山大學

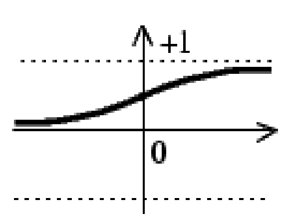
- 分段线性函数：

$$f(net_k)=\begin{cases}1, & net_k \ge \frac{1}{2}\\ net_k, & \frac{1}{2} > net_k > -\frac{1}{2}\\ 0, & net_k \le -\frac{1}{2}\end{cases}$$




- Sigmoid 函数：

$$f(net_k)=\frac{1}{1+\exp(-a\,net_k)}$$



- 如果有多个输出单元 $z_k$, 则

$$z_k = f(net_k)$$

- 当有 $c$ 输出单元 (类), 可以看作计算 $c$个判别函数：$z_k = g_k(x)$， 并且通过使判别函数$g_k(x)$ $\forall k = 1, \ldots, c$ 最大来将输入信号x分类.

对于异或问题（XOR）的例子：



- 隐含单元 $y_1$ 计算判别分界:

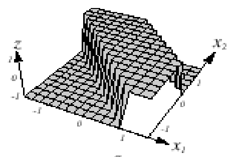
$$x_1 + x_2 + 0.5 = 0 \quad \begin{cases} \geq 0 \Rightarrow y_1 = +1 \\ < 0 \Rightarrow y_1 = -1 \end{cases}$$

- 隐含单元 $y_2$ 计算判别分界:

$$x_1 + x_2 - 1.5 = 0 \quad \begin{cases} < 0 \Rightarrow y_2 = -1 \\ \geq 0 \Rightarrow y_2 = +1 \end{cases}$$

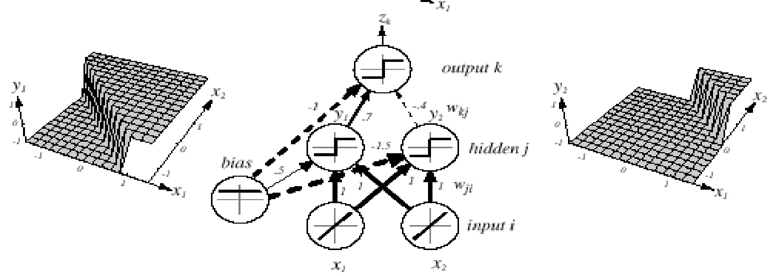
- 最终单元发出 $z_1 = +1 \Leftrightarrow y_1 = +1$ 且 $y_2 = -1$

异或问题（XOR）的例子：




净激活：

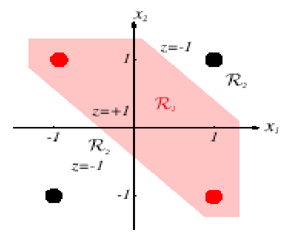
$z_1$=0.7$y_1$ -0.4 $y_2$ - 1




净激活：

$y_1$=$x_1$ + $x_2$ + 0.5

$y_2$=$x_1$ + $x_2$ -1.5




## 6.2.1 一般的前馈运算

$$g_k(x) \equiv z_k = f\left(\sum_{j=1}^{n_H} w_{kj} f\left(\sum_{i=1}^{d} w_{ji} x_i + w_{j0}\right) + w_{k0}\right) \qquad (1)$$

$(k = 1,\ldots,c)$

- 隐含单元使我们能表达更复杂的非线性函数，因此扩展了分类方法。
- 激活函数不一定是符号函数，常常要求函数是连续或可微的。
- 允许输出层的激活函数同隐含层的不一样，或者对每一个单元而言都有不同的激活函数。
- 为了分析简单，我们假设所有的激活函数都是一样的。

## 6.2.2 多层网络的表达能力

- 是否每个判决都可以用三层网络来描述？是！
- 戈尔莫戈罗夫证明了：只要选取适当的函数 $\Xi_j$ 和 $\psi_{ij}(x_i)$，任何连续函数g(x)都可以定义在单位超立方体上,即可以表示为：

$$g(\mathbf{x}) = \sum_{j=1}^{2n+1} \Xi_j \left( \sum_{i=1}^{d} \psi_{ij}(x_i) \right)$$

- 上式的含义是，2n+1个隐含单元中的每个都把d个非线性函数的和作为输入，每个特征$x_i$对应一个非线性函数。
- 可惜的是，上述构造性的描述确实显示任意期望函数都可以通过一个三层网络来执行，但它更多的价值在理论方面，而实用意义不大。

15

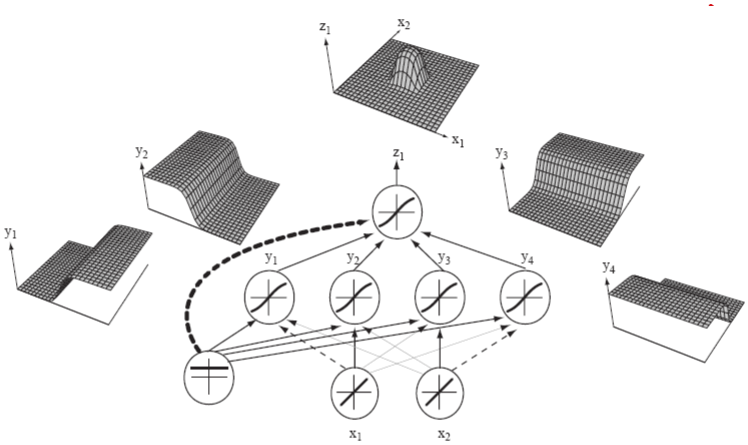



Figure 6.2: A 2-4-1 network (with bias) along with the response functions at different units; each hidden and output unit has sigmoidal transfer function $f(\cdot)$. In the case shown, the hidden unit outputs are paired in opposition thereby producing a "bump" at the output unit. Given a sufficiently large number of hidden units, any continuous function from input to output can be approximated arbitrarily well by such a network.

16

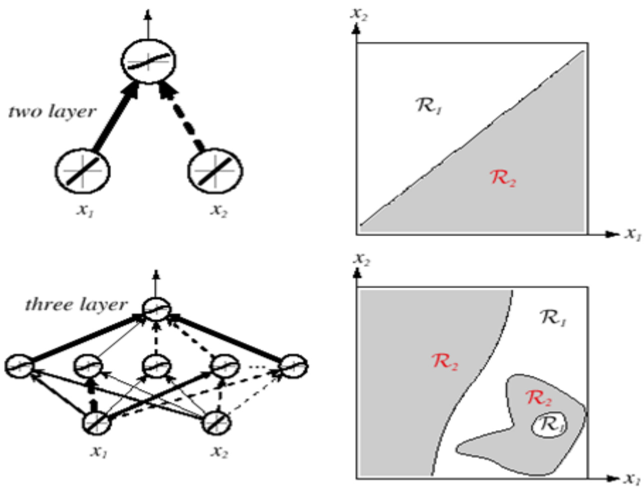



FIGURE 6.3. Whereas a two-layer network classifier can only implement a linear decision boundary, given an adequate number of hidden units, three-, four- and higher-layer networks can implement arbitrary decision boundaries. The decision regions need not be convex or simply connected. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

17

# 6.3 反向传播算法

- 任何模式分类问题都可以由一个三层网络来执行。关键问题是：如何根据训练样本和期望输出来设置合适的权。
- 反向传播算法是多层神经网络有监督训练中最简单，也是最一般的方法之一。它是LMS算法的自然延伸。

18

## 6.3 反向传播算法

- 误差反向传播学习分为四个过程:

a. 模式顺传播：一个输入向量作用于网络感知节点，它的影响经过网络一层接一层的传播。最后，产生一个输出作为网络的实际响应。在前向通过中，网络的突触权为固定的。

b. 误差逆传播：在反向通过中，突触权值全部根据误差修正规则调整。

c. 记忆训练：反复学习过程，也就是根据教师示教的希望输出与网络实际输出的误差调整连接权的过程。

d. 学习收敛：网络全局误差收敛于极小值的过程。



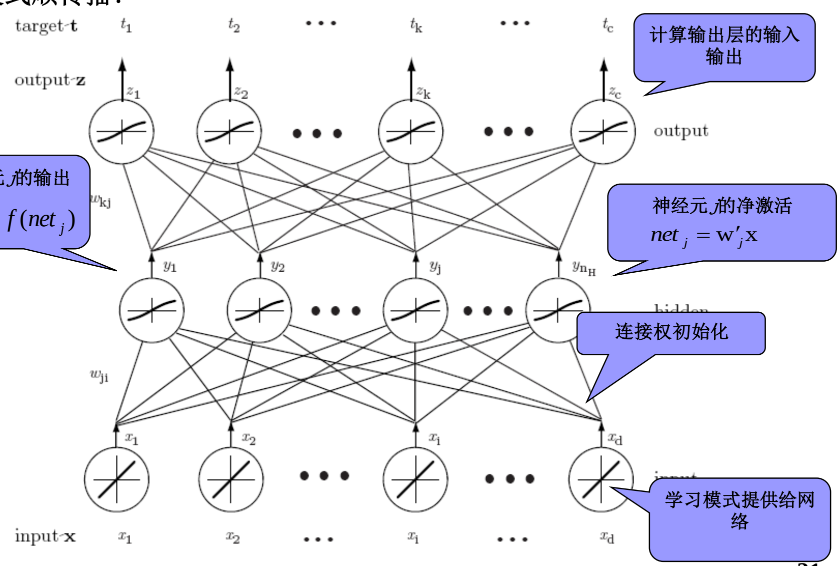
- 反向传播的威力在于允许我们对每个隐单元计算有效误差，并由此推导出一个输入层到隐含层权值的学习规则, 这就称为:

  信用分配问题（The credit assignment problem）

## 6.3.1 网络学习

中山大學

- 模式顺传播:






中山大學

- 误差反向传播阶段

（1）计算训练误差： $J(w) = \frac{1}{2}\sum_{k=1}^{c}(t_k - z_k)^2 = \frac{1}{2}\|\mathbf{t} - \mathbf{z}\|^2$

（2）按与LMS算法类似的方式对突触权值w应用一个修正值

反向传播学习规则基于梯度下降法，权值首先被初始化为随机值，然后向误差减小方向调整.

$$\Delta w = -\eta \frac{\partial J}{\partial w}$$

其中 $\eta$ 是学习率，它表示权值相对变化的尺度。
迭代算法在第m次迭代时取一个权向量并将它更新为：

$w(m+1) = w(m) + \Delta w(m)$

□ 一个隐含层到输出层的权值的误差：

$$\frac{\partial J}{\partial w_{kj}} = \frac{\partial J}{\partial net_k} \cdot \frac{\partial net_k}{\partial w_{kj}} = -\delta_k \frac{\partial net_k}{\partial w_{kj}}$$

其中k单元的敏感度定义为:

$$\delta_k = -\frac{\partial J}{\partial net_k}$$

该敏感度描述总误差与单元净激活的关系。

$$\delta_k = -\frac{\partial J}{\partial net_k} = -\frac{\partial J}{\partial z_k} \cdot \frac{\partial z_k}{\partial net_k} = (t_k - z_k) f'(net_k)$$

23

因为 $net_k = w_k^t . y$，因此:

$$\frac{\partial net_k}{\partial w_{kj}} = y_j$$

结论:隐含层到输出层学习规则：

$$\Delta w_{kj} = \eta \delta_k y_j = \eta (t_k - z_k) f'(net_k) y_j$$

24

中山大學

□ 一个输入层到隐含层的权值的误差：

$$\frac{\partial J}{\partial w_{ji}}=\frac{\partial J}{\partial y_j}\cdot\frac{\partial y_j}{\partial net_j}\cdot\frac{\partial net_j}{\partial w_{ji}}$$

然而：

$$\frac{\partial J}{\partial y_j}=\frac{\partial}{\partial y_j}\left[\frac{1}{2}\sum_{k=1}^{c}(t_k-z_k)^2\right]=-\sum_{k=1}^{c}(t_k-z_k)\frac{\partial z_k}{\partial y_j}$$

$$=-\sum_{k=1}^{c}(t_k-z_k)\frac{\partial z_k}{\partial net_k}\cdot\frac{\partial net_k}{\partial y_j}=-\sum_{k=1}^{c}(t_k-z_k)f'(net_k)w_{kj}$$



中山大學

类似可定义第j隐含单元的敏感度为:

$$\delta_j\equiv f'(net_j)\sum_{k=1}^{c}w_{kj}\delta_k$$

- 它解决信用分配的核心问题：**一个隐含单元的敏感度是各输出单元敏感度的加权和，权重为 $w_{kj}$，然后与 $f'(net_j)$ 相乘**。

结论: 输入层到隐含层学习规则：

$$\Delta w_{ji}=\eta x_i\delta_j=\eta\underbrace{\left[\Sigma w_{kj}\delta_k\right]f'(net_j)}_{\delta_j}x_i$$

中山大學

■ **归纳：误差反向传播阶段**

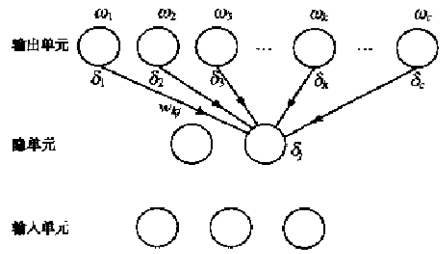
（1）计算训练误差： $J(w)=\frac{1}{2}\sum_{k=1}^{c}(t_k-z_k)^2=\frac{1}{2}\|\mathbf{t}-\mathbf{z}\|^2$

（2）按与LMS算法类似的方式对突触权值w应用一个修正值

隐含层到输出层学习规则： $\Delta w_{kj}=\eta\delta_k y_j=\eta(t_k-z_k)f'(net_k)y_j$

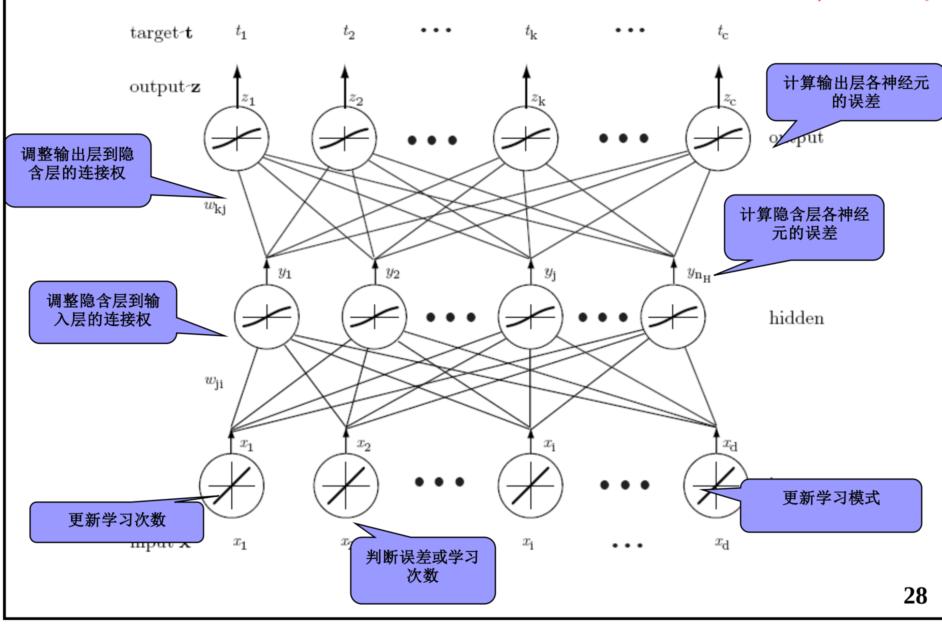
输入层到隐含层学习规则： $\Delta w_{ji}=\eta\delta_j x_i=\eta\left[\sum_{k=1}^{c}w_{kj}\delta_k\right]f'(net_j)x_i$

图 6-5 隐单元的敏感度与输出单元的敏感度的加权和成正比：$\delta_j=f'(net_j)\sum_{k=1}^{c}w_{kj}\delta_k$。这样输出单元的敏感度就反向传播“回”隐单元了






中山大學



## 6.3.2 训练协议

广义地说，有监督的训练就是给出一个类别标记已知的模式——训练集——找到网络输出，并调整权值以使实际输出更加接近于期望的目标值。三种最有用的“训练协议”是：随机训练、成批训练和在线训练。

- 随机反向传播算法伪代码:

```
Begin  initialize   n_H; w, 准则 θ, η, m ← 0
       do m ← m + 1
           x^m ← 随机地选择模式
           w_ji ←  w_ji + ηδ_j x_i; w_kj ← w_kj + ηδ_k y_j
       until ||∇J(w)|| < θ
       return w
End
```

29

- 目前为止，我们只考虑训练集中单个模式的误差，但实际上我们要考虑一个定义在训练集里所有模式的误差。
- 总训练误差可以写成n个单独模式误差的总和：

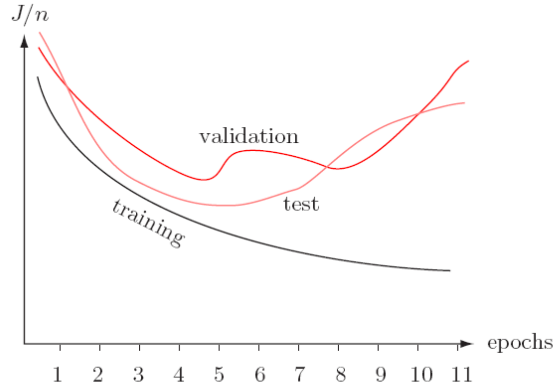
$$J=\sum_{p=1}^{n} J_p \qquad (22)$$

- 在“随机训练”中，一个权值更新有可能减少某个模式的误差，然而却增加了训练全集上的误差，不过，给出大量的单次更新，可以降低式（22）中所有给出的总误差。

30

中山大學

## 6.3.3 BP网络——学习曲线




学习曲线显示的是误差准则函数作为训练总量的一个函数。

训练总量一般用回合数表示（或者全部训练集提供的次数）。

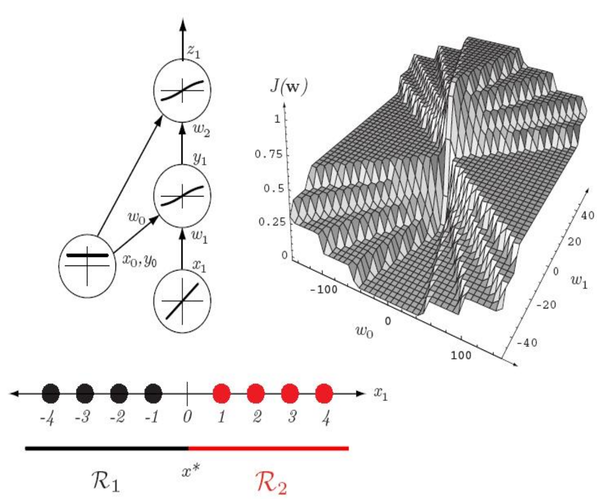


中山大學

## 6.4 误差曲面

- 小型的网络：

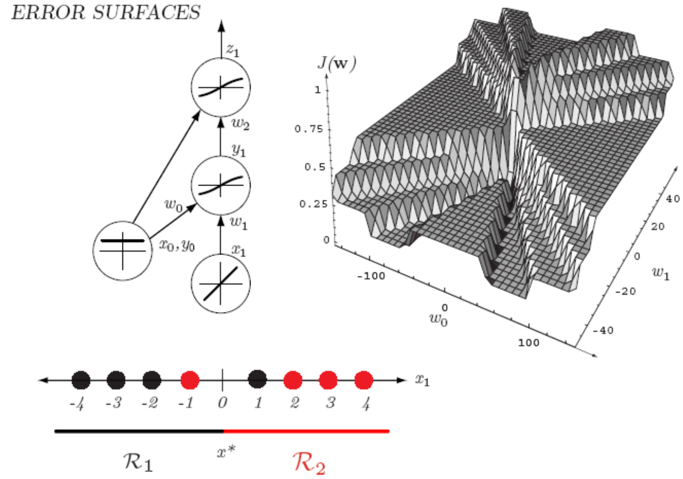
8个一维模式（每类各4个）用一个具有较陡的S型隐含单元和输出单元（含偏置）的1-1-1型网络来学习。



两种形式的极小误差解：

$-2<x^*<-1$

$1<x^*<2$








- **较大型的网络：**

  高维空间里局部极小值问题有所不同：在学习中，高维空间可以给系统提供更多的方式（维数、或自由度）以“避开”障碍或局部极小值。权值数越过剩，网络越不可能陷入局部极小值。但存在过拟合问题。

- **关于多重极小：**

  通常重新初始化权值，再训练一遍。

  局部极小问题，当误差较低时，非全局极小是可以接受的。

## Error Surface

The residual error E(w) can be visualized as a surface in the weight-space:

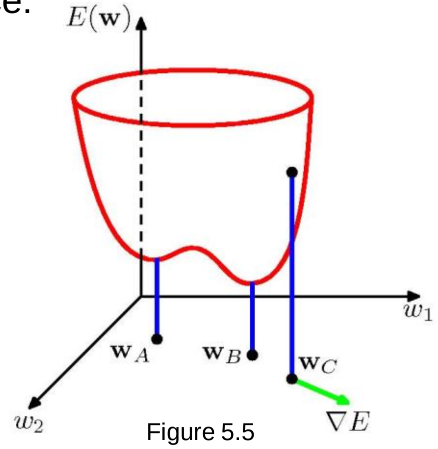
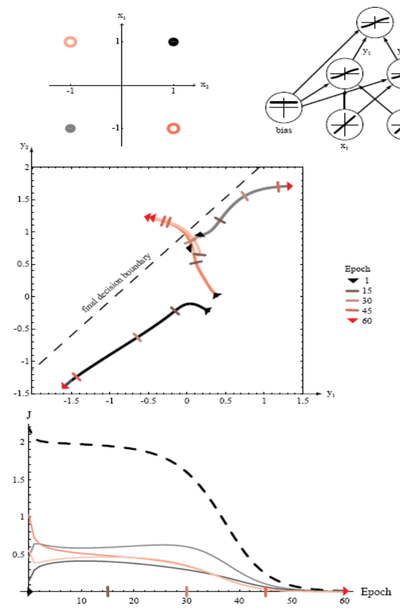



Figure 5.5



## 6.5　反向传播作为特征映射

- 隐含层到输出层是一个线性判别函数，多层神经网络所提供的新的计算能力可以归因于输入层到隐含层单元上的表示的非线性弯曲能力。
- 随着学习的进行，输入层到隐含层的权值在数量上增加，隐含层单元的非线性弯曲扭曲了从输入层到隐含层单元的空间映射。

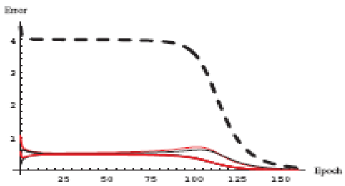


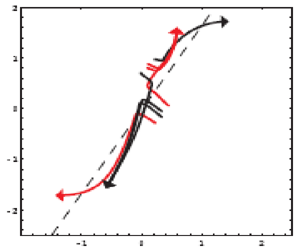

2-2-1反向传播网络以及四模式XOR问题。

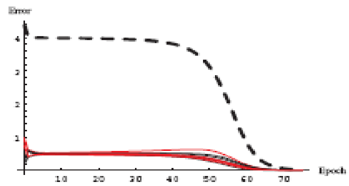
中间图：4种模式所对应的隐单元的1种输出。

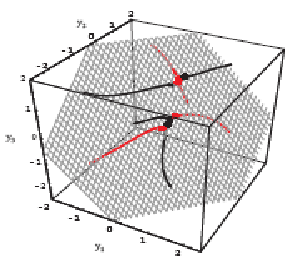
下图：学习曲线—定义在各个模式上的误差及总误差。

37





38

Figure 6.13: Seven patterns from a two-dimesional two-category nonlinearly separable classification problem are shown at the bottom.

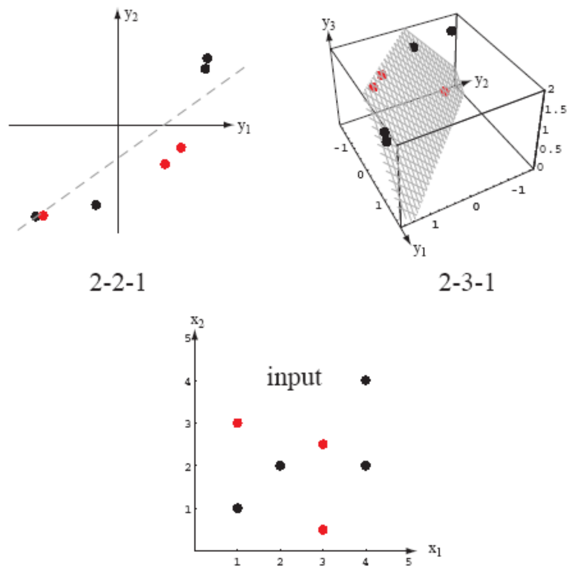
The figure at the top left shows the hidden unit representations of the patterns in a 2-2-1 sigmoidal network (with bias)

fully trained to the global error minimum;

*At the top* right is the analogous hidden unit representation for a fully trained 2-3-1 network

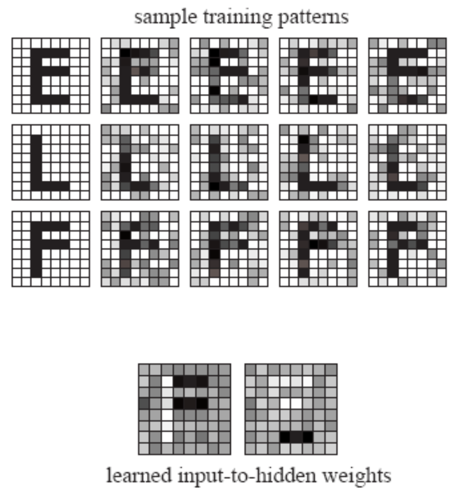
(with bias). Because of the higher dimension of the hidden layer representation, the categories are now linearly separable; indeed the learned hidden-to-output weights

implement a plane that separates the categories.




隐含层表示

对于64-2-3 的网络：学习后的权值用来分类特征的分组形式。



# Example of a neural network

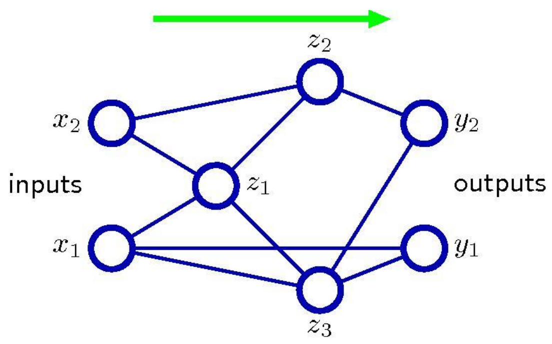



Figure 5.2

41

# Different function

Figure 5.3

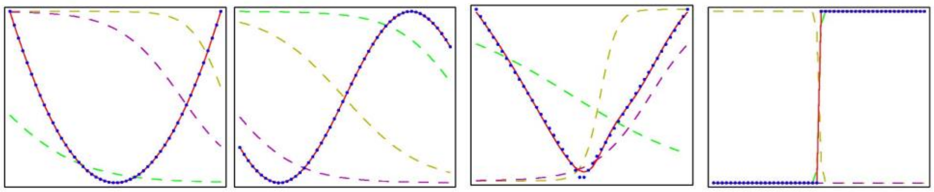


Illustration of the capability of a multilayer perceptron to approximate four different functions comprising

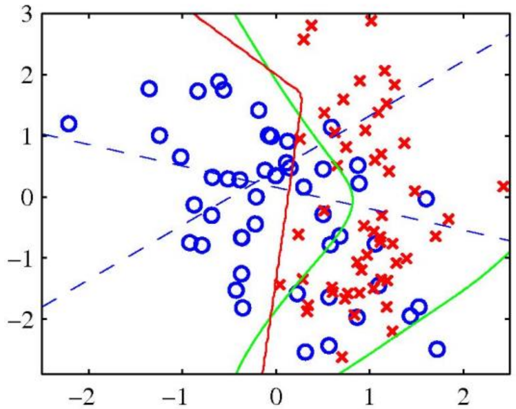
(a) $f(x) = x^2$, (b) $f(x) = \sin(x)$,

(c), $f(x) = |x|$, (d) $f(x) = H(x)$

where $H(x)$ is the Heaviside step function.

42

Example of a neural network




2-2-1网络训练

Figure 5.4



# 6.6 反向传播、贝叶斯理论及概率

## 6.6.1 贝叶斯理论与神经网络

尽管多层神经网显得有点专门化，我们可以证明，当采用均方差准则进行反向传播训练，且样本数量趋于无穷极限时，多层神经网可产生最小二乘意义下的贝叶斯判别函数。

$g_k(x;w)$是第$k$个输出单元的输出，该判别函数对应类别$\omega_k$。贝叶斯公式为：

$$P(\omega_k \mid x) = \frac{p(x \mid \omega_k)P(\omega_k)}{\sum_{i=1}^{c} p(x \mid \omega_i)P(\omega_i)} = \frac{p(x,\omega_k)}{p(x)}$$

贝叶斯判决，模式x属于具有最大判别函数$g_k(x)=P(\omega_k \mid x)$的类$\omega_k$。

假设根据

$$t_k(x) = \begin{cases} 1 & x \in \omega_k \\ 0 & \text{其他} \end{cases}$$

对于有限个训练样本$x$ 的给予单个输出单元$k$ 的准则函数的贡献是

$$
\begin{aligned}
J(w) &= \sum_{x}[g_k(x;w)-t_k(x)]^2 \\
&= \sum_{x\in\omega_k}[g_k(x;w)-1]^2 + \sum_{x\notin\omega_k}[g_k(x;w)-0]^2 \\
&= n\left\{\frac{n_k}{n}\frac{1}{n_k}\sum_{x\in\omega_k}[g_k(x;w)-1]^2 + \frac{n-n_k}{n}\frac{1}{n-n_k}\sum_{x\notin\omega_k}[g_k(x;w)-0]^2\right\}
\end{aligned}
$$

其中$n$是训练模式的总数量，$\omega_k$中有$n_k$个。



在数据取极限的情况下，上式为：

$$\lim_{x\to\infty}\frac{1}{n}J(w)=\bar{J}(w)$$

$$= P(\omega_k)\int[g_k(x;w)-1]^2 p(x|\omega_k)dx + P(\omega_{i\neq k})\int g_k(x;w)^2 p(x|\omega_{i\neq k})dx$$

$$= \int g_k(x;w)^2 p(x)dx - 2\int g_k(x;w)p(x,\omega_k)dx + \int p(x,\omega_k)dx$$

$$= \int[g_k(x;w)-p(\omega_k|x)]^2 p(x)dx + \underbrace{\int p(\omega_k|x)p(\omega_{i\neq k}|x)p(x)dx}_{\text{独立于}w}$$

反向传播规则改变权值以最小化上式的左边，从而最小化

$$\int [g_k(x;w) - p(\omega_k \mid x)]^2 p(x)dx$$

由于它对于每个类别$\omega_k$（k=1, 2, . . . , c）都成立，因此

$$\sum_{k=1}^{c} \int [g_k(x;w) - p(\omega_k \mid x)]^2 p(x)dx$$

即，在样本数量趋于无穷时，网络输出在最小二乘意义下为

$$g_k(x;w) \approx p(\omega_k \mid x) \quad (6\text{-}29)$$

**习题 E17**：完成公式(6-29)的推导步骤，证明当采用均方差准则进行BP训练时，多层神经网络判决与贝叶斯判决一致。



- **作为概率的输出**

实际生活时常不满足无限个训练数据，这时可以作概率逼近。其中一个方法是*softmax*方法，即选择指数型的输出单元非线性函数$f(net_k) \propto e^{net_k}$ 并对每种模式将输出和归一化为1.0，并用0-1目标信号进行训练：

$$z_k = \frac{e^{net_k}}{\sum_{m=1}^{c} e^{net_k}}$$

## 6.7 相关统计技术

- 投影寻踪回归：

$$z=\sum_{j=1}^{j_{max}} w_j f_j(\mathbf{v}_j^t\mathbf{x}+v_{j0})+w_0$$

- 广义叠加模型：

$$z=f\left(\sum_{i=1}^{d} f_i(x_i)+w_0\right)$$

- 多元自适应回归样条（MARS）：

$$z=\sum_{k=1}^{M} w_k \prod_{r=1}^{r_k} \phi_{kr}(x_{q(k,r)})+w_0$$



## 6.8 改进反向传播的一些实用技术

- 激活函数

  性质：

  1. 非线性，否则三层网络将不提供高于两层网络之上的任何计算能力；
  2. 饱和性，即存在最大值和最小值，这可以限定权值和激活函数的边界；
  3. 连续性和光滑性，一阶和二阶导数存在；
  4. 单调性，如果f不单调且具有多个局部极大值，将在误差曲面上引入附加的和不希望出现的极值。

中山大學

- 常用的激活函数

BP网络中每一个神经元的 $\delta$ 需要关于神经元的激活函数 $f(\bullet)$ 的导数知识。要导数存在，则需要函数$f(\bullet)$ 连续。

常用的例子为sigmoid函数，主要有两种形式：

1. logistic函数

$$f(net_j)=\frac{1}{1+\exp(-anet_j)} \quad a>0,-\infty<net_j<\infty$$

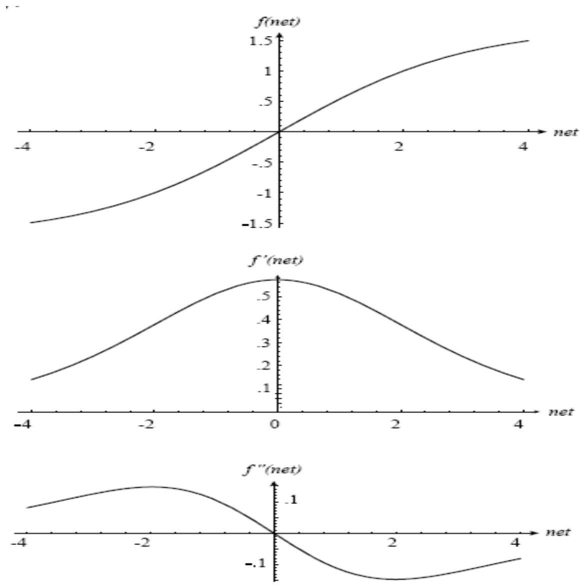
2. 双曲正切函数

$$f(net_j)=a\tanh(bnet_j) = a\frac{e^{+bnet}-e^{-bnet}}{e^{+bnet}+e^{-bnet}}, \quad (a,b)>0$$



中山大學

- 选择Sigmoid函数的理由
  - 非线性、单调性
  - 无限次可微
  - 当权值很大时近似阈值函数
  - 当权值很小是可近似线性函数




53

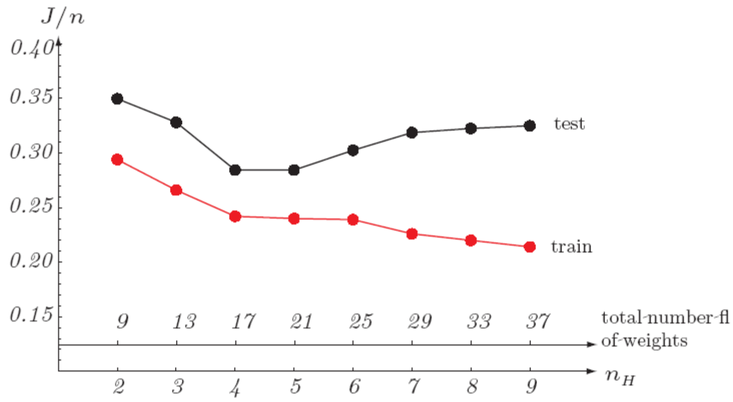
- 输入信号尺度变换
  - □进行训练数据的归一化：进行尺度变换，在全部训练集上，每维特征均值为0，并具有相同的方差--1.0
- 目标值
  - □图中输出单元在±1.716处达到饱和，但对于任意有限的netk，输出永远不可能达到饱和
- 带噪声的训练法
  - □构造虚拟或替代训练模式，如加入一个d维高斯噪声以获得真实的训练点。

54

- **隐单元数**

实验表明：增加隐含层的层数和隐含层神经元个数不一定总能够提高网络精度和表达能力。

BP网一般都选用三层网络。






- **权值初始化**

a. 初始权值的选择对于局部极小点的防止和网络收敛速度的提高均有一定程度的影响，如果初始权值范围选择不当，学习过程一开始就可能进入“假饱和”现象，甚至进入局部极小点，网络根本不收敛。

b. 在前馈多层神经网络的BP算法中，初始权、阈值一般是在一个固定范围内按均匀分布随机产生的。一般文献认为初始权值范围为-1～＋1之间，初始权、阈值的选择因具体的网络结构模式和训练样本不同而有所差别，一般应视实际情况而定。

c. 本书中考虑有$d$个输入单元，假设用相同的分布初始化权值，那么输入权值的范围为：

$$-\frac{1}{\sqrt{d}} < w_{ji} < \frac{1}{\sqrt{d}}$$

d. 隐含层输出权值：

$$-\frac{1}{\sqrt{n_H}} < w_{kj} < \frac{1}{\sqrt{n_H}}$$

- 学习率

学习率参数 $\eta$ 越小，从一次迭代到下一次迭代的网络突触权值的变化量就越小，轨迹在权值空间就越光滑。然而，这种改进是以减慢学习速度为代价的。另一方面，如果我们让 $\eta$ 的值太大以加速学习速度的话，结果有可能使网络的突触权值的变化量不稳定。

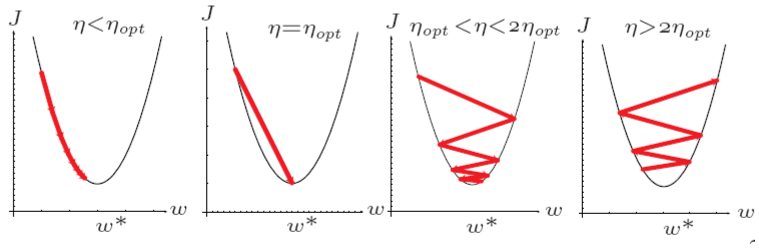



Figure 6.18: Gradient descent in a one-dimensional quadratic criterion with different learning rates. If $\eta < \eta_{opt}$, convergence is assured, but training can be needlessly slow. If $\eta = \eta_{opt}$, a single learning step suffices to find the error minimum. If $\eta_{opt} < \eta < 2\eta_{opt}$, the system will oscillate but nevertheless converge, but training is needlessly slow. If $\eta > 2\eta_{opt}$, the system diverges.

57

- Assuming the criterion function can be reasonably approximated by a quadratic which thus gives

$$\frac{\partial^2 J}{\partial w^2}\Delta w = \frac{\partial J}{\partial w}.$$

- The optimal rate is found directly to be

$$\eta_{opt} = \left(\frac{\partial^2 J}{\partial w^2}\right)^{-1}.$$

58

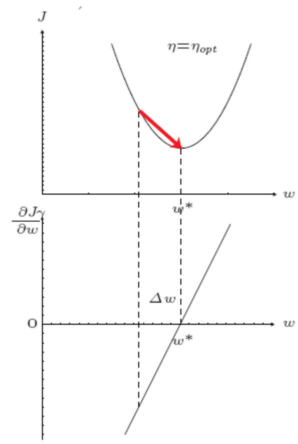



Figure 6.19: If the criterion function is quadratic (above), its derivative is linear (below). The optimal learning rate $\eta_{opt}$ insures that the weight value yielding minimum error, $w^*$ is found in a single learning step.

**59**

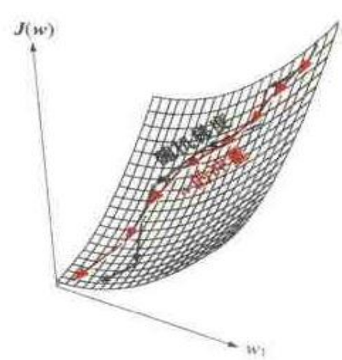
中山大學

- 冲量项(Momemtum)

  一个既要加快学习速度又要保持稳定的简单方法是修改delta法则，使它包括冲量项（惯量项）：

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta \delta_j(n) y_i(n)$$

*a*是冲量常数，通常是正数。




$$w(m+1) = w(m) + (1-\alpha)\Delta w_{bp}(m) + \alpha \Delta w(m-1)$$

图 6-18　通过式(37)将冲量嵌入随机梯度下降法中(红色箭头)，减少了总体梯度方向的偏离，从而加快了学习速度

**60**

■ 权值衰减

网络的权值大致分两类：

- 对网络具有很大影响的权值
- 对网络影响很少或者根本没有影响的权值。

后者常常造成网络推广性差。

复杂性正则化的使用鼓励多余权值取得接近0，提高泛化能力。

61

# Early stopping

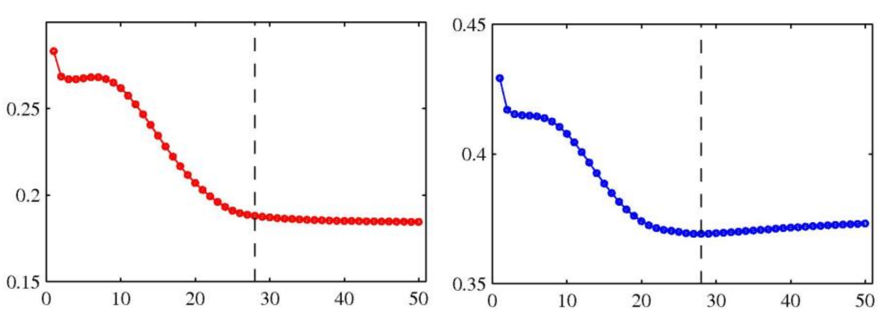


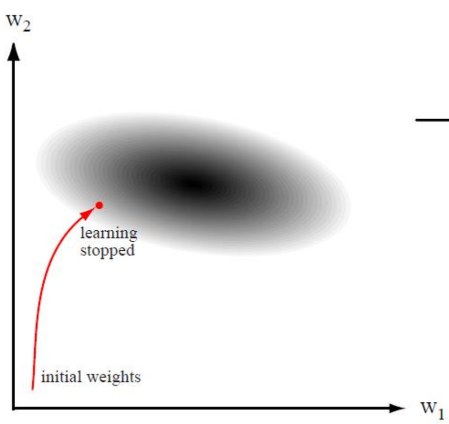

Figure 5.12

Left: training set error
Right: validation set error

The goal of achieving the best generalization performance suggests that training should be stopped at point of smallest error.

62

# Early stopping



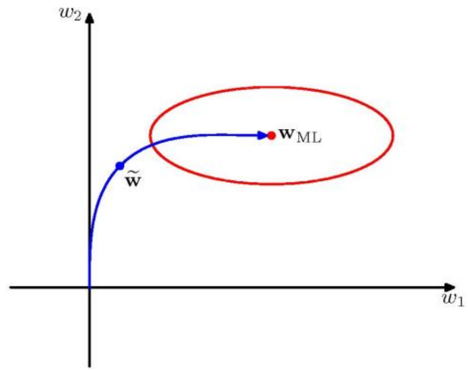




Figure 6.20



- **训练方式**

a. 随机训练：

模式是随机地从训练集中取出的，权值也根据不同的模式进行更新

b. 成批训练：

所有的模式已在训练之前全部送往网络中。

c. 在线训练：

每种模式只提供一次，不需要存储器来保存模式

- 误差准则函数

原来的平方误差准则是最常见的训练准则，然而，其他的训练准则有时候也有一些好处。下面介绍两个有用的准则函数：

互熵（cross entropy）：（可用来度量概率分布间的“距离”）

$$J(\mathbf{w})_{ce} = \sum_{m=1}^{n} \sum_{k=1}^{c} t_{mk} \ln(t_{mk}/z_{mk}),$$

基于闵可夫斯基误差：

$$J_{Mink}(\mathbf{w}) = \sum_{m=1}^{n} \sum_{k=1}^{c} |z_{mk}(\mathbf{x}) - t_{mk}(\mathbf{x})|^R,$$

可通过选择R值来调节分类器的局部性：R值越小，分类器的局部性越强。



# 6.9 二阶技术

- 牛顿法：

在梯度下降中使用牛顿法，可利用下式迭代计算w的值：

$$\begin{aligned}\mathbf{w}(m+1) &= \mathbf{w}(m) + \Delta\mathbf{w} \\ &= \mathbf{w}(m) - \mathbf{H}^{-1}(m)\left(\frac{\partial J(\mathbf{w}(m))}{\partial \mathbf{w}}\right)\end{aligned}$$

（其中H为赫森矩阵）

推导过程：

$$\begin{aligned}\Delta J(\mathbf{w}) &= J(\mathbf{w}+\Delta\mathbf{w}) - J(\mathbf{w}) \\ &\simeq \left(\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}\right)^t \Delta\mathbf{w} + \frac{1}{2}\Delta\mathbf{w}^t \mathbf{H} \Delta\mathbf{w},\end{aligned}$$

$$\left(\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}\right) + \mathbf{H}\Delta\mathbf{w} = \mathbf{0}, \qquad \Delta\mathbf{w} = -\mathbf{H}^{-1}\left(\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}\right).$$

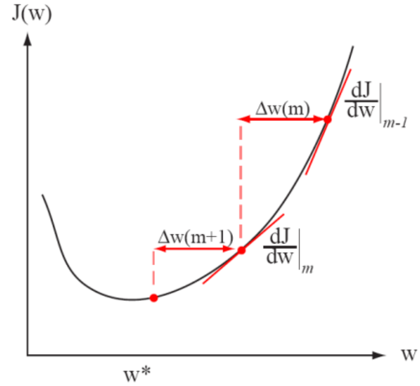
Quickprop权值更新算法

- **Quickprop(快速传播)算法:**

Quickprop算法中权值假设为独立的。可以证明，这种方法可导出如下的权值更新规则：

$$\Delta w(m+1)=\frac{\left.\frac{dJ}{dw}\right|_m}{\left.\frac{dJ}{dw}\right|_{m-1}-\left.\frac{dJ}{dw}\right|_m}\Delta w(m).$$

其中的导数是由m和m-1次迭代估计得出




- 利用了隔开一定的已知距离的两个点处的误差导数。

- **共轭梯度法**

共轭条件：

$$\Delta \mathbf{w}^t(m-1)\mathbf{H}\Delta \mathbf{w}(m)=0$$

，其中H为赫森矩阵

在第m步的下降方向是梯度方向加上一个沿着前面的下降方向的元素：

$$\Delta \mathbf{w}(m)=-\nabla J(\mathbf{w}(m))+\beta_m \Delta \mathbf{w}(m-1)$$

各项间的相互比例由 $\beta_m$ 控制。通常它可以用如下两个公式中的一个来计算：

Fletcher-Reeves:

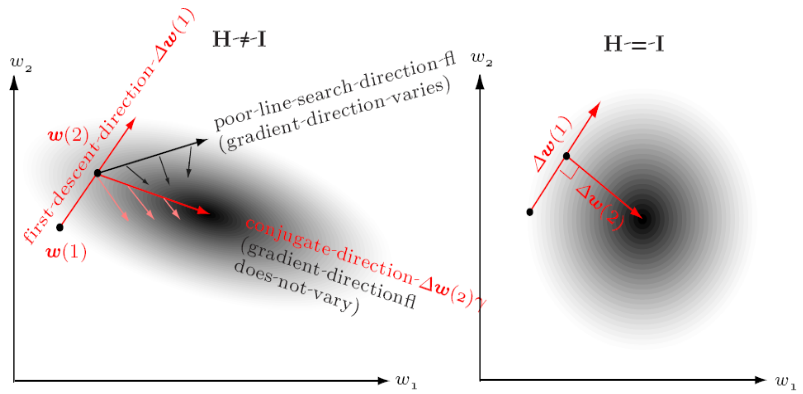
$$\beta_m=\frac{[\nabla J(\mathbf{w}(m))]^t \nabla J(\mathbf{w}(m))}{[\nabla J(\mathbf{w}(m-1))]^t \nabla J(\mathbf{w}(m-1))}$$

Polak-Ribiere:

$$\beta_m=\frac{[\nabla J(\mathbf{w}(m))]^t[\nabla J(\mathbf{w}(m))-\nabla J(\mathbf{w}(m-1))]}{[\nabla J(\mathbf{w}(m-1))]^t \nabla J(\mathbf{w}(m-1))}$$

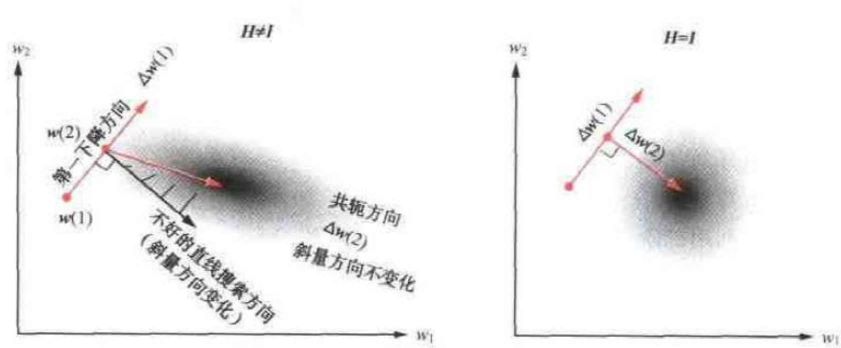
68

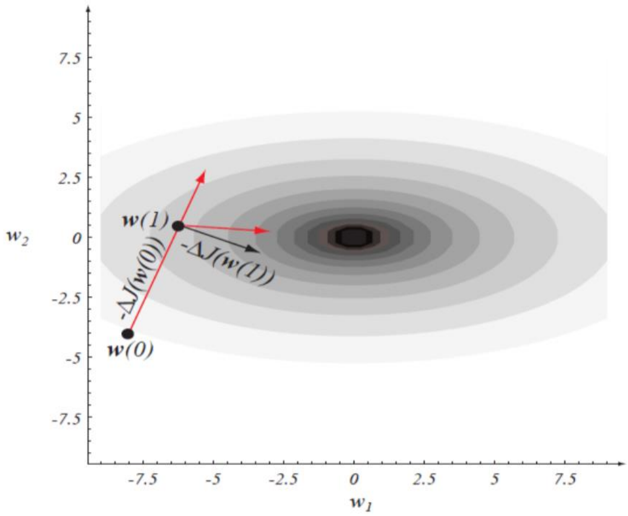
- “共轭方向”－在下降过程中梯度方向不改变，而仅仅只是幅值改变的方向。沿该方向不会破坏前面下降步骤的贡献。










$$\frac{d}{ds}\left[[\mathbf{w}(1)+s\Delta\mathbf{w}(1)]^t\,\mathbf{H}\,[\mathbf{w}(1)+s\Delta\mathbf{w}(1)]\right] =$$

$$\frac{d}{ds}\left[\left[\begin{pmatrix}-6.202\\0.496\end{pmatrix}+s\begin{pmatrix}2.788\\-.223\end{pmatrix}\right]^t\begin{pmatrix}.2&0\\0&1\end{pmatrix}\left[\begin{pmatrix}-6.202\\0.496\end{pmatrix}+s\begin{pmatrix}2.788\\-.223\end{pmatrix}\right]\right] = 0$$

71

# 6.10 其他网络和训练算法

- 径向基函数网络

径向基函数（radial basis function，RBF）网络的设计可以看作是一个高维空间中的曲线拟和（逼近）问题。

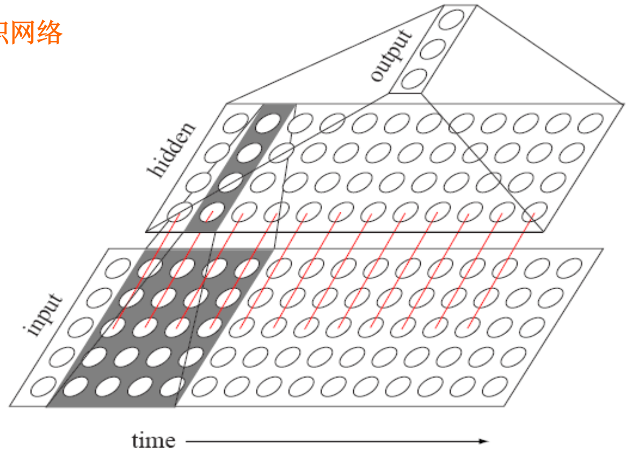
这里考虑插值函数（内核）的通用形式$f(\|x-c_i\|)$ ，该函数的变量是从中心$x$ 到输入变量 的欧氏距离，$f$称为RBF。函数 可以有多种形式，例如：

$$f(x)=\exp\left(-\frac{1}{2\sigma_i^2}\|x-c_i\|^2\right)$$

$$f(x)=\frac{\sigma^2}{\sigma^2+\|x-c_i\|^2}$$

72

■ 卷积网络




一个时延神经网络（TDNN）利用权值共享来保证了对沿一维的移动也可以将模式识别出来。

类似的平均约束也可以加在隐含层和输出层的各单元间，



# Convolutional networks

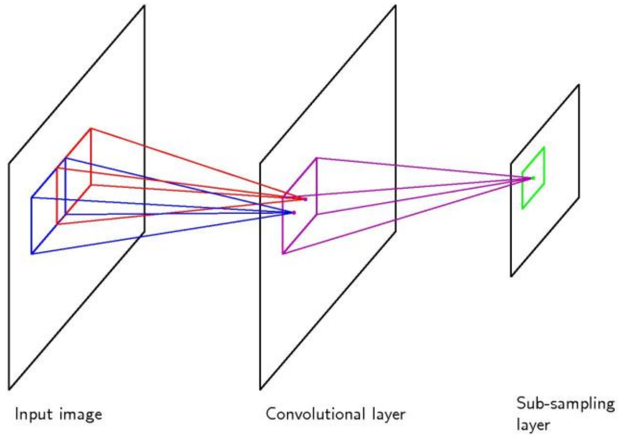
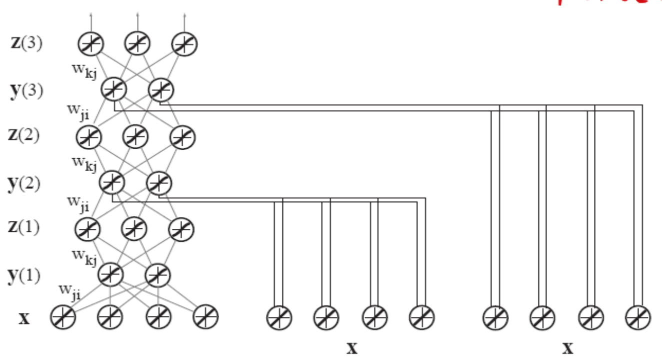


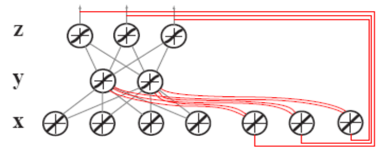

Figure 5.17

## ■ 递归网络

底图：递归网络，具有红色的递归连接；

上图：等价的静态网络，具有很多隐含层和扩展的权值共享



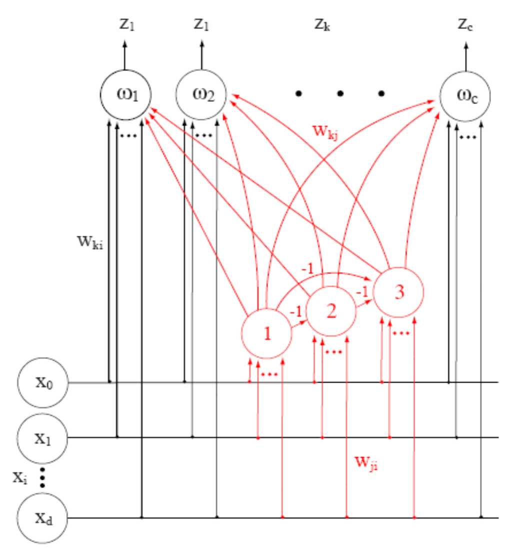








## ■ 级数相关技术

图 6-26　通过级联相关的一个多层网络训练从输入层完全同输出层(黑色)互连开始。这种权值，$w_{ki}$，利用一个 LMS 准则完全训练，同第 5 章所讨论的一样。如果所得的训练误差并不足够低，第一个隐单元(红色标记 1)被引入，与输入层和输出层完全连接。这些新的红色权值被完全训练，而先前的(黑色)权值保持固定。如果所得训练误差仍然不足够小，第二个隐单元(标记 2)被类似引入，完全互连接；它还接收先前每一个隐单元的乘上 −1 的输出。通过这种方式继续训练接下来的隐单元，直到训练误差低到可以被接受的程度



中山大學

# 6.11 正则化、复杂度调度和剪枝

- 正则化

  构造一个新的准则函数，该函数不仅取决于典型的训练误差，还决于分类器 复杂程度：

$$J = J_{pat} + \lambda J_{reg}$$

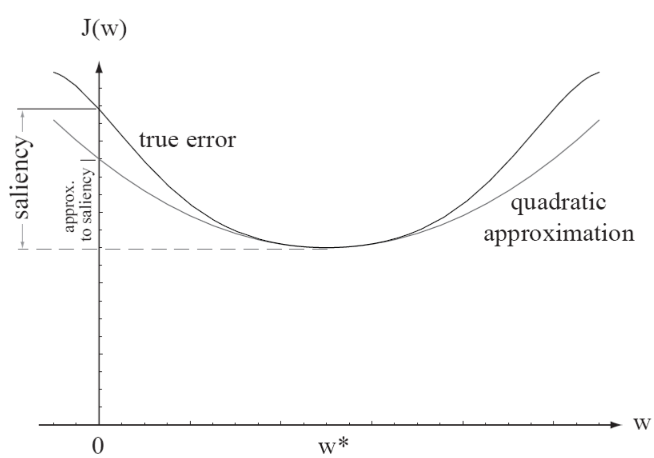
  参数 $\lambda$ 的大小决定了正则项作用的强弱程度。

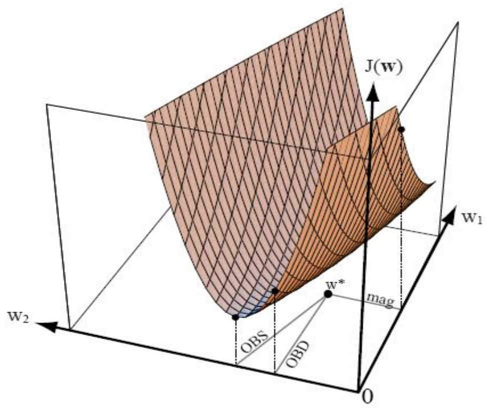
- Wald统计法

  其基本思想是：我们可以估计出模型中的某个参数的重要性，然后就可以消除最不重要的参数了。比如在网络中，这样的参数可以是某个权值。其主要方法有最佳脑损伤（OBD）和最佳脑外科（OBS）法.



中山大學






该图显示了作为权值函数的二次误差曲面以及全局极小值。

在准则函数的二次近似中，**OBD**法假设**Hessian**矩阵是对角化的，而**OBS**法采用完整的**Hessian**矩阵。

**79**

# 本章小结

- 前馈运算
- BP反传算法
- 误差曲面
- 反传算法特性
- 其它相关统计技术
- 反传算法的实用改进技术
- 二阶技术
- 其它网络

**80**