

## 五、神经网络

主讲教师：周志华

---

# BP (BackPropagation:误差逆传播算法)

迄今最成功、最常用的神经网络算法，可用于多种任务（不仅限于分类）

P. Werbos在博士学位论文中正式完整描述：

P. Werbos. Beyond regression: New tools for prediction and analysis in the behavioral science. Ph.D dissertation, Harvard University, 1974

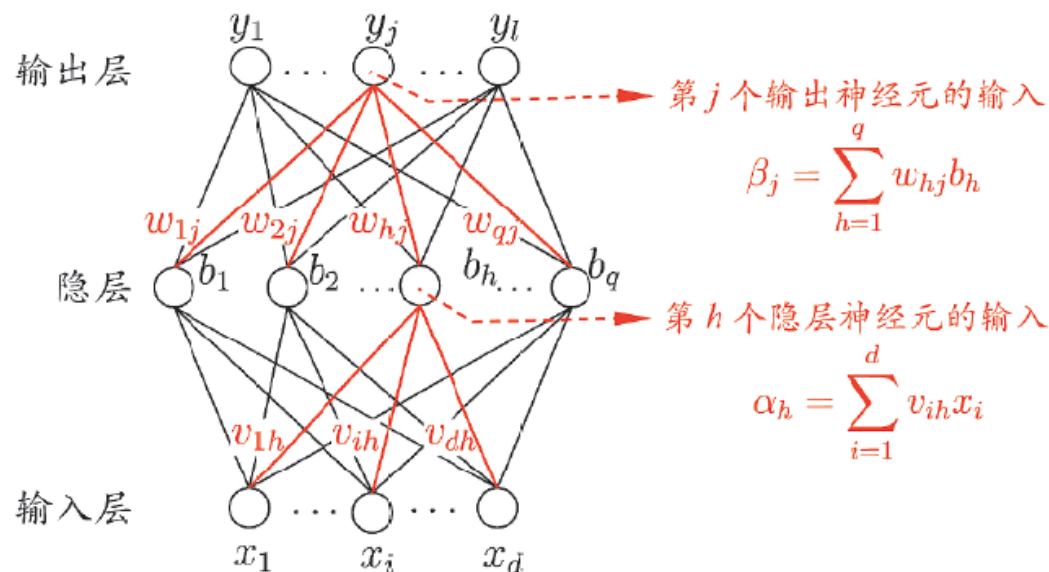
给定训练集  $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$ ,  $x_i \in \mathbb{R}^d$ ,  $y_i \in \mathbb{R}^l$

输入： $d$  维特征向量

输出： $l$  个输出值

隐层：假定使用  $q$  个  
隐层神经元

假定功能单元均使用  
Sigmoid 函数



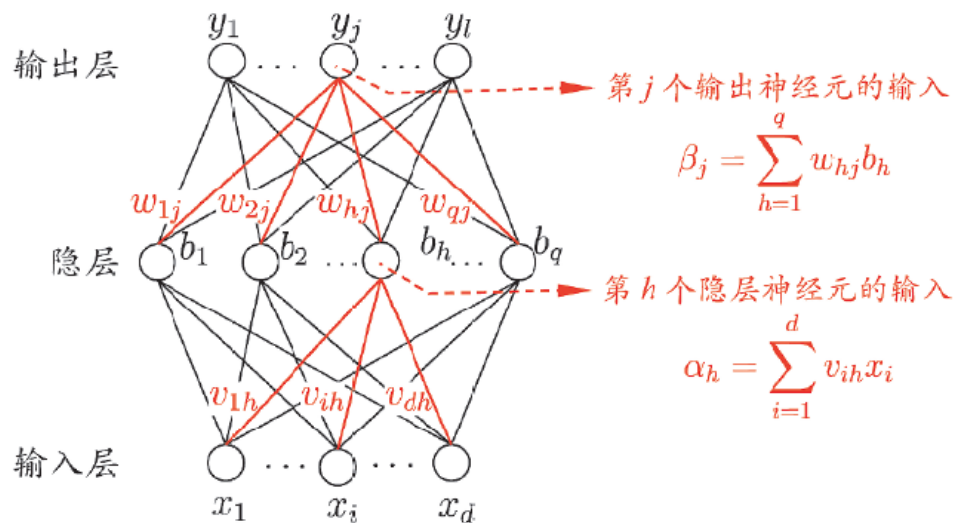
# BP 算法推导

对于训练例  $(\mathbf{x}_k, \mathbf{y}_k)$ , 假定网络的实际输出为  $\hat{\mathbf{y}}_k = (\hat{y}_1^k, \hat{y}_2^k, \dots, \hat{y}_l^k)$

$$\hat{y}_j^k = f(\beta_j - \theta_j)$$

则网络在  $(\mathbf{x}_k, \mathbf{y}_k)$  上的均方误差为:

$$E_k = \frac{1}{2} \sum_{j=1}^l (\hat{y}_j^k - y_j^k)^2$$



需通过学习确定的参数数目:  $(d + l + 1)q + l$

BP 是一个迭代学习算法, 在迭代的每一轮中采用广义感知机学习规则

$$v \leftarrow v + \boxed{\Delta v}.$$

## BP 算法推导 (续)

BP 算法基于**梯度下降**策略，以目标的负梯度方向对参数进行调整

以  $w_{hj}$  为例

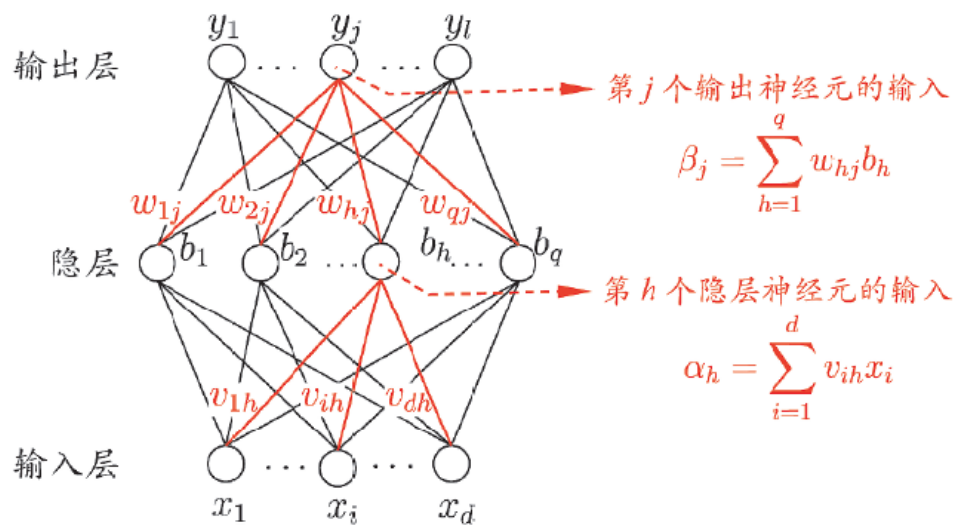
对误差  $E_k$ ，给定学习率  $\eta$ ，有：

$$\Delta w_{hj} = -\eta \frac{\partial E_k}{\partial w_{hj}}$$

注意到  $w_{hj}$  先影响到  $\beta_j$ ，

再影响到  $\hat{y}_j^k$ ，然后才影响到  $E_k$ ，有：

$$\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}}$$



← “链式法则”

# BP 算法推导 (续)

$$\frac{\partial E_k}{\partial w_{hj}} = \boxed{\frac{\partial E_k}{\partial \hat{y}_j^k}} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \boxed{\frac{\partial \beta_j}{\partial w_{hj}}}$$

$(\hat{y}_j^k - y_j^k)$        $= b_h$

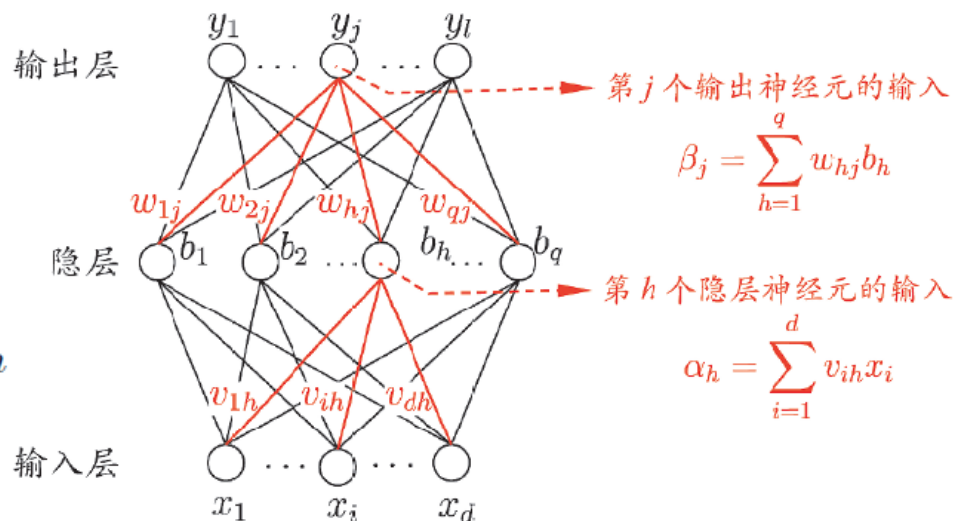
注意到  
 $\hat{y}_j^k = f(\beta_j - \theta_j)$

$$f'(\beta_j - \theta_j)$$

$$\hat{y}_j^k(1 - \hat{y}_j^k)$$

$$\text{令 } g_j = -\frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j}$$

$$= \hat{y}_j^k(1 - \hat{y}_j^k)(y_j^k - \hat{y}_j^k)$$



对  $\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$ , 有

$$f'(x) = f(x)(1 - f(x))$$

于是,  $\Delta w_{hj} = -\eta \frac{\partial E_k}{\partial w_{hj}} = \eta g_j b_h$

## BP 算法推导 (续)

类似地，有：

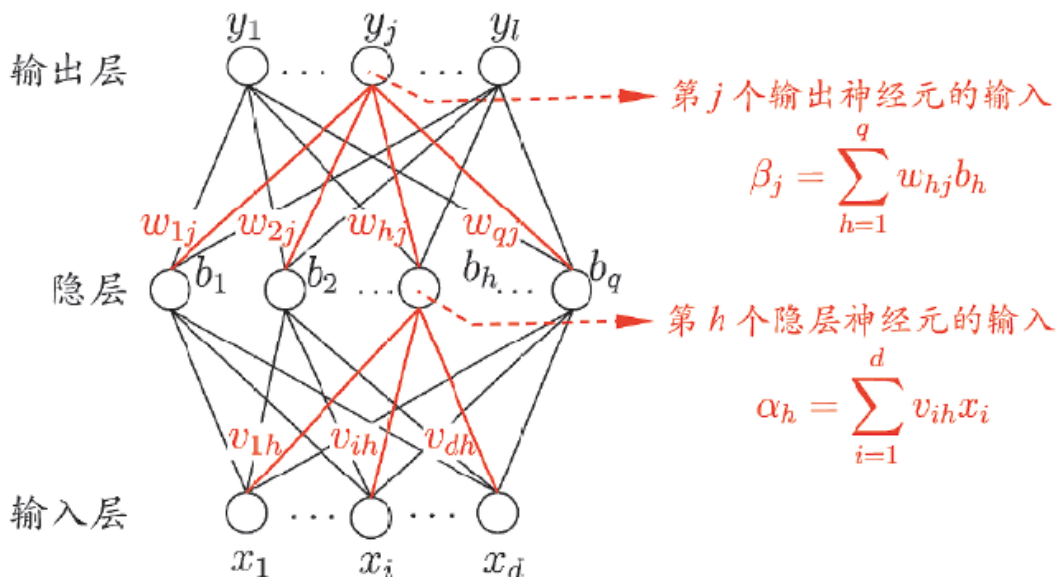
$$\Delta\theta_j = -\eta g_j$$

$$\Delta v_{ih} = \eta e_h x_i$$

$$\Delta\gamma_h = -\eta e_h$$

其中：

$$\begin{aligned} e_h &= -\frac{\partial E_k}{\partial b_h} \cdot \frac{\partial b_h}{\partial \alpha_h} \\ &= -\sum_{j=1}^l \frac{\partial E_k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial b_h} f'(\alpha_h - \gamma_h) = \sum_{j=1}^l w_{hj} g_j f'(\alpha_h - \gamma_h) \\ &= b_h(1 - b_h) \sum_{j=1}^l w_{hj} g_j \end{aligned}$$



学习率  $\eta \in (0, 1)$  不能太大、不能太小

# BP 算法

---

输入：训练集  $D = \{(\mathbf{x}_k, \mathbf{y}_k)\}_{k=1}^m$ ;  
学习率  $\eta$ .

过程：

- 1: 在  $(0, 1)$  范围内随机初始化网络中所有连接权和阈值
- 2: **repeat**
- 3:     **for all**  $(\mathbf{x}_k, \mathbf{y}_k) \in D$  **do**
- 4:         根据当前参数和式(5.3) 计算当前样本的输出  $\hat{\mathbf{y}}_k$ ;
- 5:         根据式(5.10) 计算输出层神经元的梯度项  $g_j$ ;
- 6:         根据式(5.15) 计算隐层神经元的梯度项  $e_h$ ;
- 7:         根据式(5.11)-(5.14) 更新连接权  $w_{hj}$ ,  $v_{ih}$  与阈值  $\theta_j$ ,  $\gamma_h$
- 8:     **end for**
- 9: **until** 达到停止条件

输出：连接权与阈值确定的多层前馈神经网络

---

图 5.8 误差逆传播算法

# 标准 BP 算法 vs. 累积 BP 算法

---

## 标准 BP 算法

- 每次针对单个训练样例更新权值与阈值
- 参数更新频繁, 不同样例可能抵消, 需要多次迭代

## 累积 BP 算法

- 其优化目标是最小化整个训练集上的累计误差
- 读取整个训练集一遍才对参数进行更新, 参数更新频率较低

在很多任务中, 累计误差下降到一定程度后, 进一步下降会非常缓慢, 这时标准BP算法往往会获得较好的解, 尤其当训练集非常大时效果更明显.



# 缓解过拟合

主要策略：

## □ 早停(early stopping)

- 若训练误差连续  $a$  轮的变化小于  $b$ , 则停止训练
- 使用验证集：若训练误差降低、验证误差升高, 则停止训练

## □ 正则化 (regularization)

- 在误差目标函数中增加一项描述网络复杂度

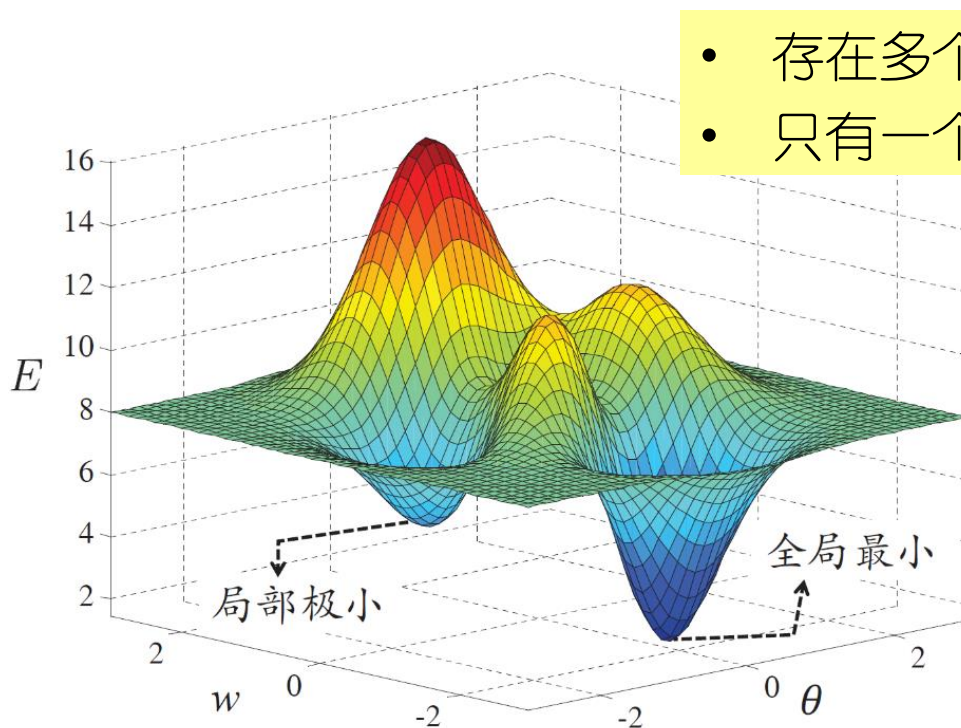
例如 
$$E = \lambda \frac{1}{m} \sum_{k=1}^m E_k + (1 - \lambda) \sum_i w_i^2$$

偏好比较小的连接权和阈值,  
使网络输出更“光滑”

# 全局最小 vs. 局部极小

神经网络的训练过程可看作一个参数寻优过程：

在参数空间中，寻找一组最优参数使得误差最小



- 存在多个“局部极小”
- 只有一个“全局最小”

“跳出”局部极小的常见策略：

- ✓ 不同的初始参数
- ✓ 模拟退火
- ✓ 随机扰动
- ✓ 演化算法
- ✓ .....

# 其他常见神经网络模型

---

- **RBF**: 分类任务中除**BP**之外最常用
- **ART**: “竞争学习”的代表
- **SOM**: 最常用的聚类方法之一
- 级联相关网络: “构造性”神经网络的代表
- **Elman**网络: 递归神经网络的代表
- **Boltzmann**机: “基于能量的模型”的代表
- .....

# RBF 神经网络

---

RBF: Radial Basis Function (径向基函数)

- 单隐层前馈神经网络
- 使用径向基函数作为隐层神经元激活函数

例如高斯径向基函数  $\rho(\mathbf{x}, \mathbf{c}_i) = e^{-\beta_i \|\mathbf{x} - \mathbf{c}_i\|^2}$

- 输出层是隐层神经元输出的线性组合

$$\varphi(\mathbf{x}) = \sum_{i=1}^q w_i \rho(\mathbf{x}, \mathbf{c}_i)$$

训练:

**Step1:** 确定神经元中心, 常用的方式包括随机采样、聚类等

**Step2:** 利用BP算法等确定参数

# SOM 神经网络

SOM: Self-Organizing feature Map (自组织特征映射)

- 竞争型的无监督神经网络
- 将高维数据映射到低维空间（通常为2维），高维空间中相似的样本点映射到网络输出层中邻近神经元
- 每个神经元拥有一个权向量
- 目标：为每个输出层神经元找到合适的权向量以保持拓扑结构

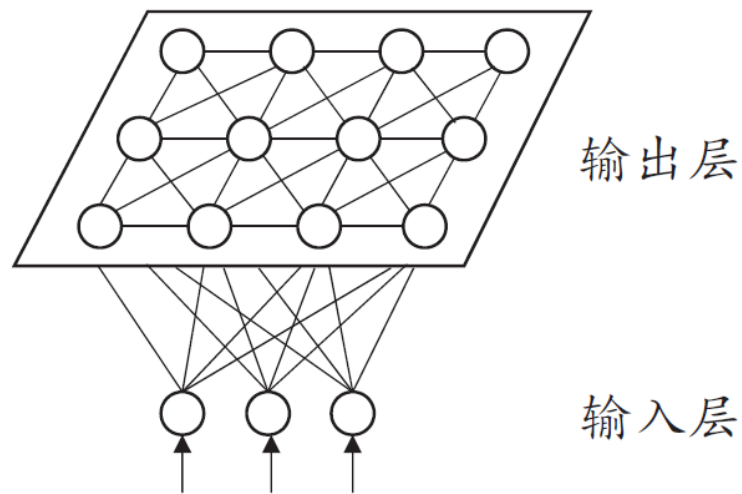


图 5.11 SOM 网络结构

训练：

- 网络接收输入样本后，将会确定输出层的“获胜”神经元（“胜者通吃”）
- 获胜神经元的权向量将向当前输入样本移动

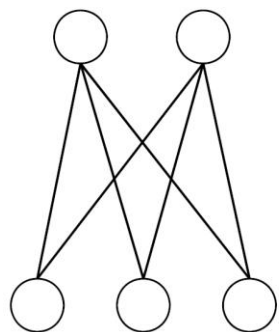
# 级联相关网络

## CC: Cascade-Correlation (级联相关)

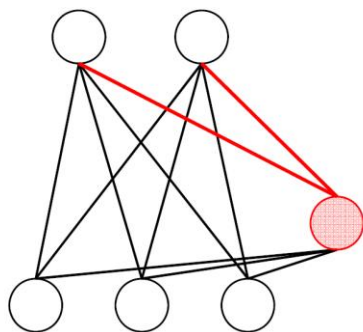
**构造性神经网络：** 将网络的结构也当做学习的目标之一， 希望在训练过程中找到适合数据的网络结构

训练：

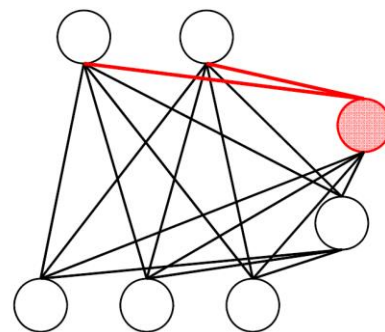
- 开始时只有输入层和输出层
- 级联 - 新的隐层结点逐渐加入，从而创建起层级结构
- 相关 - 最大化新结点的输出与网络误差之间的相关性



(a) 初始状态



(b) 增加一个隐层结点



(c) 增加第二个隐层结点

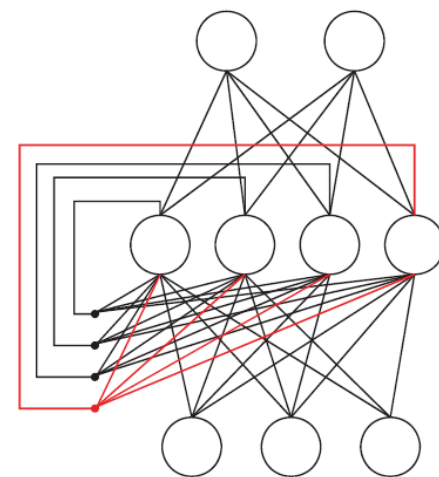
# Elman 网络

递归神经网络：Recurrent NN, 亦称 Recursive NN

- 网络中可以有环形结构, 可让使一些神经元的输出反馈回来作为输入
- $t$  时刻网络的输出状态: 由  $t$  时刻的输入状态和  $t-1$  时刻的网络状态共同决定

Elman 网络是最常用的递归神经网络之一

- 结构与前馈神经网络很相似, 但隐层神经元的输出被反馈回来
- 使用推广的BP算法训练



目前在自然语言处理等领域常用的 LSTM 网络, 是一种复杂得多的递归神经网络

图 5.13 Elman 网络结构

# 深度学习的兴起

- 2006年, Hinton 组发表深度学习的 Science 文章
- 2012年, Hinton 组参加ImageNet 竞赛, 使用 CNN 模型以超过第二名10个百分点的成绩夺得当年竞赛的冠军
- 在计算机视觉、语音识别、机器翻译等领域取得巨大成功

## 最著名的深度学习模型：卷积神经网络 (CNN: Convolutional NN)

[LeCun and Bengio, 1995; LeCun et al., 1998]

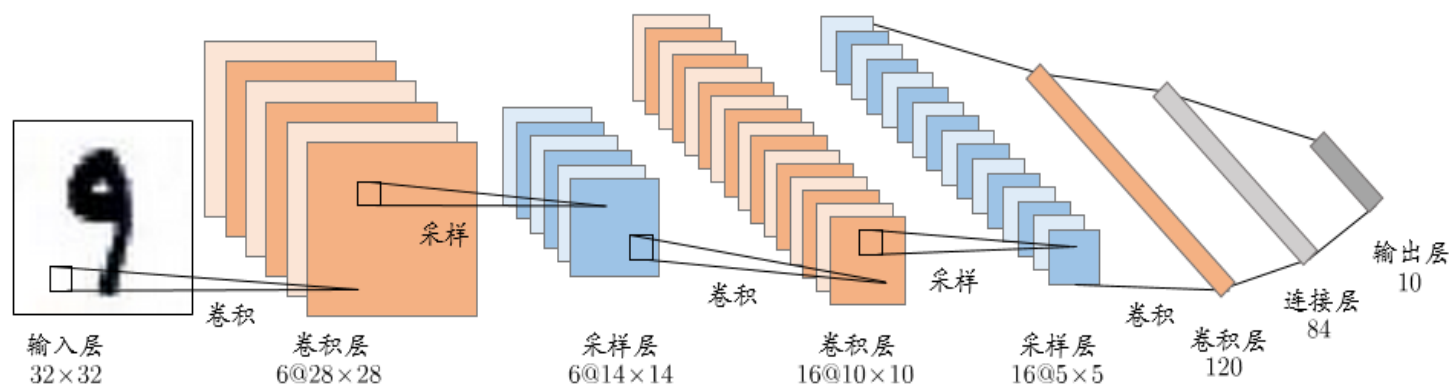


图 5.15 卷积神经网络用于手写数字识别 [LeCun et al., 1998]



# 深度学习是“模拟人脑”吗？

## 《IEEE 深度对话 Facebook 人工智能负责人 Yann LeCun》



**Yann LeCun**

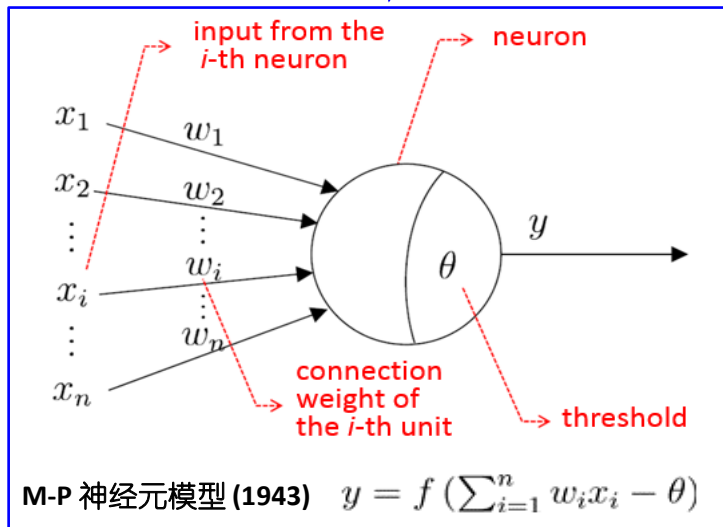
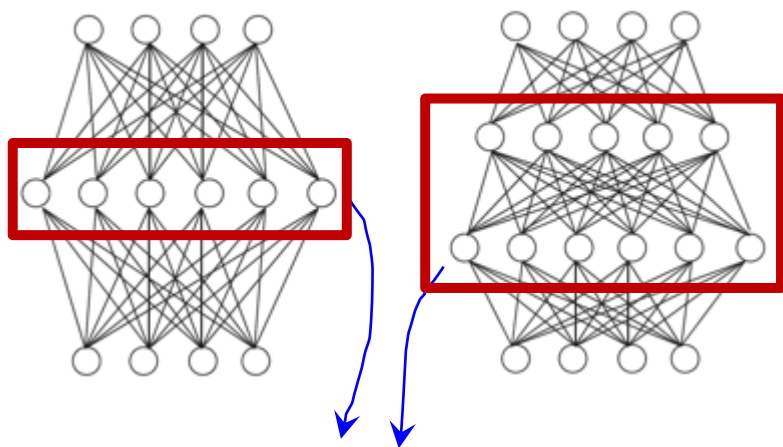
CNN的主要发明人  
深度学习“三驾马车”之一  
2019年图灵奖得主

IEEE Spectrum：这些天我们看到了许多关于深度学习的新闻 .....

**Yann LeCun：**我最不喜欢的描述是「它像大脑一样工作」，我不喜欢人们这样说的原因是，虽然深度学习从生命的生物机理中获得灵感，但它与大脑的实际工作原理差别非常非常巨大。将它与大脑进行类比给它赋予了一些神奇的光环，这种描述是危险的。

# 从“神经网络”到“深度神经网络”

人工神经网络从1950年代开始研究，  
以往一般采用单或双隐层结构



例如，ImageNet竞赛冠军：

2012: 8 层

2015: 152 层 约 6千万参数

2016: 1207 层 约 2亿参数

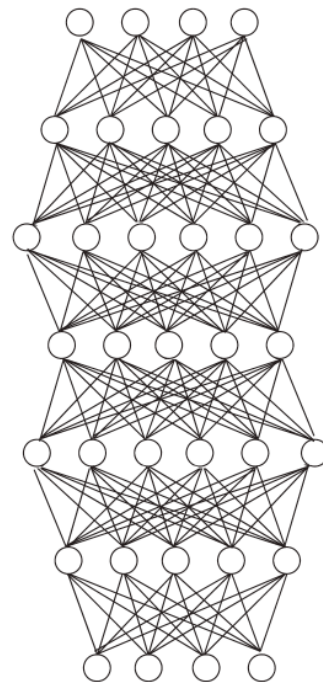
deep

“大模型”

GPT3: 1750亿参数

GPT4: 万亿参数

深度神经网络：  
很多层



神经网络实质上是多层函数  
嵌套形成的数学模型

受到了一点生物神经机制的“启发”，  
但其工作机制与脑神经网络非常不同

至今最常用的算法：BP，是完全从数学上推导出来的；  
迄今在人脑中找不到对应机制

# 重要诀窍 (tricks)

## □ 预训练+微调

- 预训练：监督逐层训练，每次训练一层隐结点
- 微调：预训练全部完成后，对全网络进行微调训练

可视为将大量参数分组，  
对每组先找到较好的局部  
配置，再全局寻优

## □ 权共享 (weight-sharing)

- 一组神经元使用相同的连接权值

减少需优化的参数

## □ Dropout

- 在每轮训练时随机选择一些参数令其不被更新(下一轮可能被更新)

降低 Rademacher 复杂度

## □ ReLU (Rectified Linear Units)

- 将 Sigmoid 激活函数修改为修正线性函数

求导容易；缓解梯度消失现象

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

## □ 交叉熵 (Cross-entropy)

- BP算法中以交叉熵  $-\frac{1}{m} \sum_{i=1}^m y_i \log \hat{y}_i$  代替均方误差  $\frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2$

更能体现分类任务的特性

尚有许多**trick**缺乏关于奏效原因的合理猜测

# “深度学习”就是“深度神经网络学习”吗？

以往一般就是这样认为

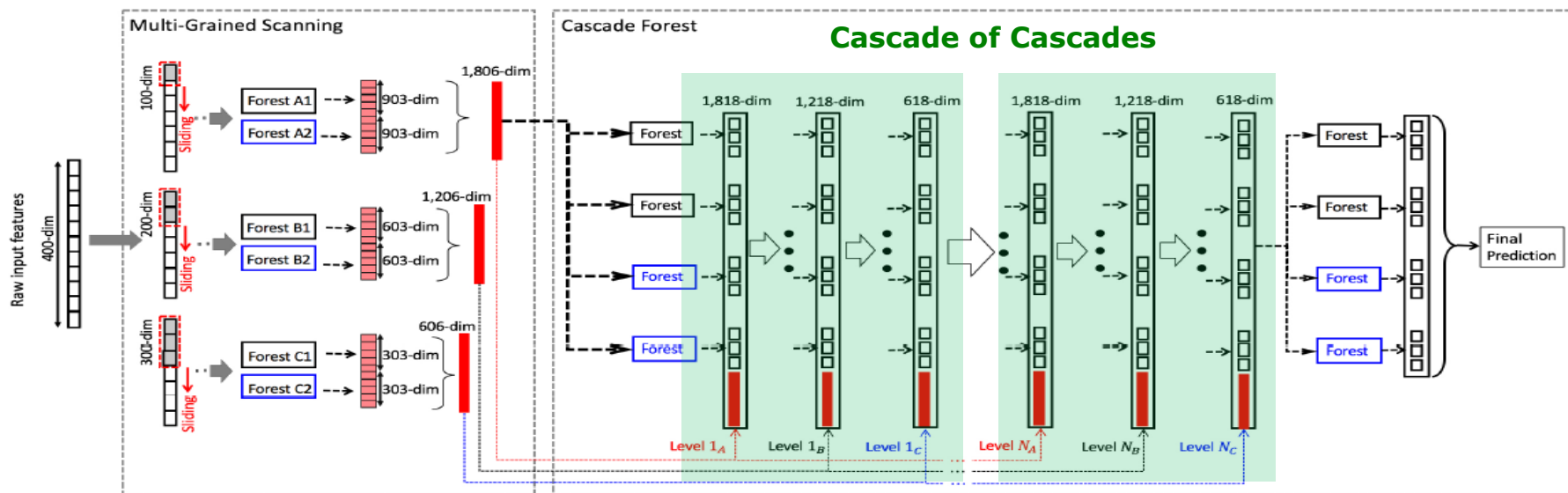
例如，**SIAM News** 的封面文章

SIAM (Society for Industrial and Applied Mathematics)



# 非神经网络的深度学习模型与方法

“深度森林” (Deep Forest), 这是第一个“非神经网络”、不基于可微构件、不使用BP算法训练、甚至不依赖梯度计算的深度学习模型



开启了基于非可微构件的深度学习模型的探索

- Z.-H. Zhou and J. Feng. Deep forest. **National Science Review**, 2019, 6(1): 74-86. (early version in IJCAI 2017) Code: <http://www.lamda.nju.edu.cn/deep-forest/>

---

深度学习并非“突然出现”的  
“颠覆性技术”，

而是经过了长期发展、  
很多研究者做出贡献，

“冷板凳”坐“热”的结果

---

---



# 例如: CNN (卷积神经网络)

引发深度学习热潮,  
被广泛应用

信号处理中的卷积 [最晚1903年已在文献中出现]

D. Hubel & T. Wiesel 关于  
猫视皮层的研究 [1962]

福岛邦彦(Fukushima)  
在神经网络中引入卷积 [1982]

G. Hinton研究组将8层CNN用于  
ImageNet竞赛获胜 [2012]

30年

Y. LeCun 引入BP算法训练  
卷积网络, CNN成型 [1989]

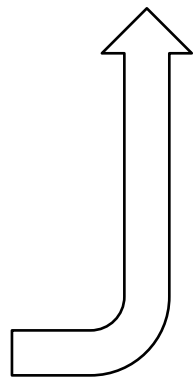
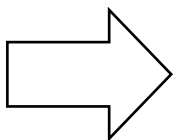
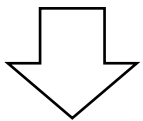
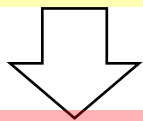
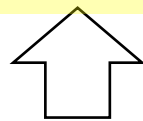
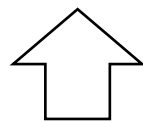
H. Lee et al. 引入无监督  
逐层训练CNN [2009]

20年

G. Hinton通过无监督逐层  
训练, 构建深层模型 [2006]

Y. LeCun and Y. Bengio,  
完整描述CNN [1995]

Y. LeCun et al., CNN 用于  
支票手写字符识别[1998]



前往第六站.....

