Artificial Intelligence

9.神经网络

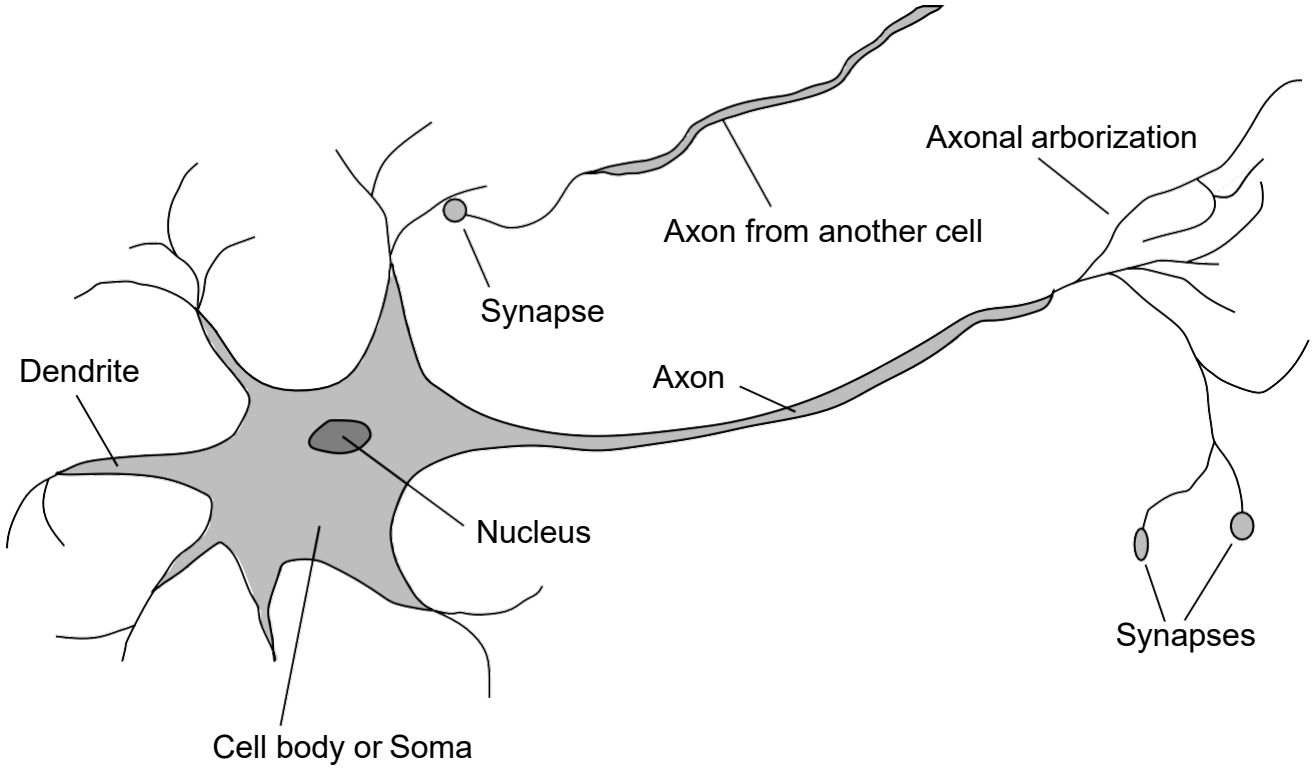
对应课本20章

本章内容

- 大脑
- 神经网络
- 感知器
- 多层感知器
- 神经网络的应用

大脑

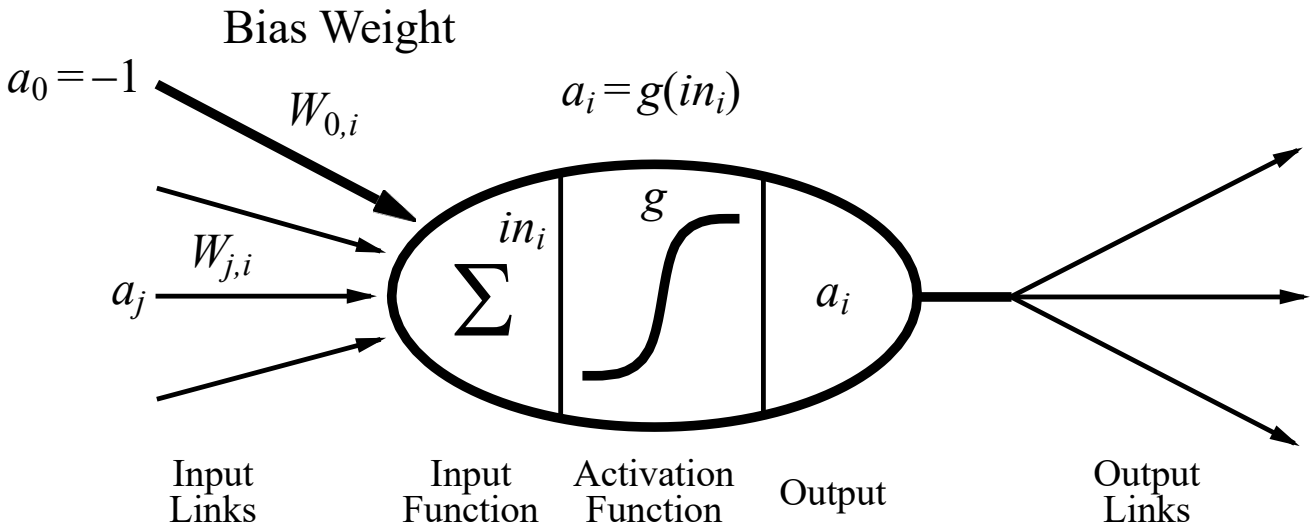
- 大脑有10^{11}个神经元，包括20种类型和10^{14}个突触
- 每个神经元的响应时间在1ms-10ms
- 信号是电势的嘈杂的“尖峰列车”



McCulloch–Pitts 对于神经元的定义“unit”

- 输出是输入的“压扁”线性函数：

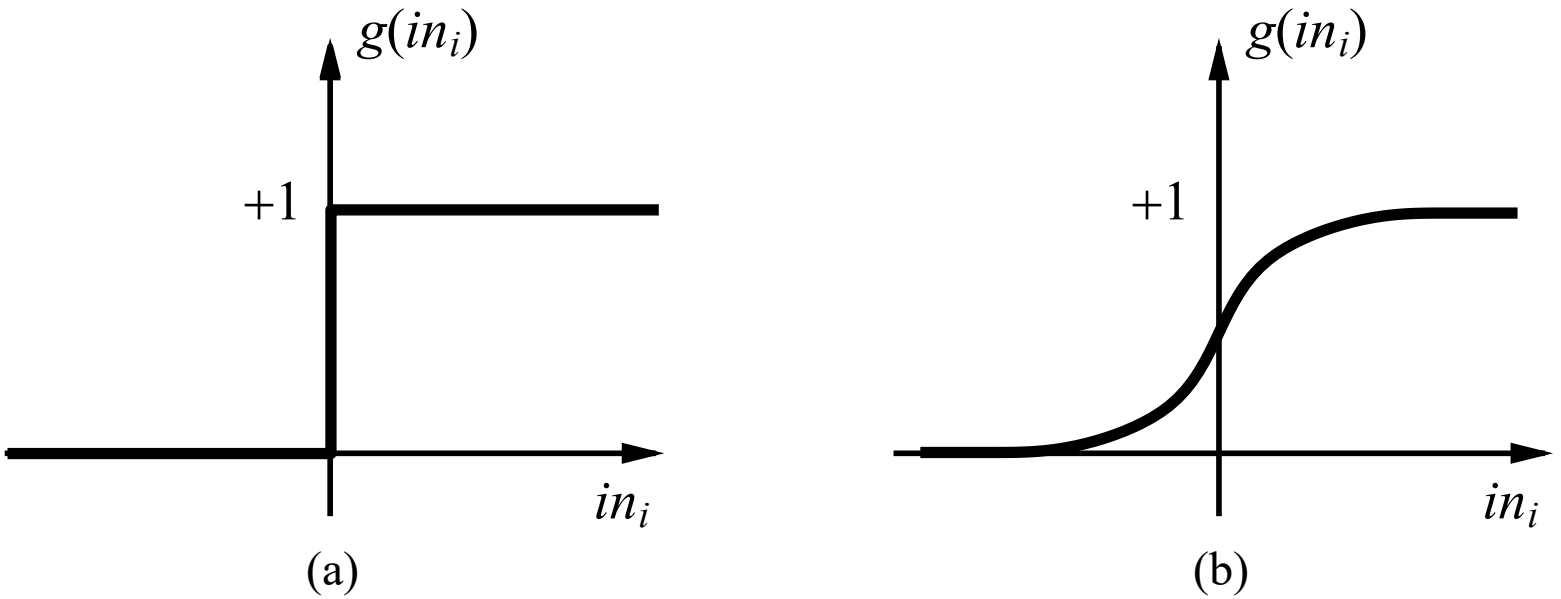
$$a_i \leftarrow g(in_i) = g(\sum_j W_{j,i} a_j)$$



- 这是对真实神经元的一种粗略的过度简化，但其目的是发展对简单单元网络的理解

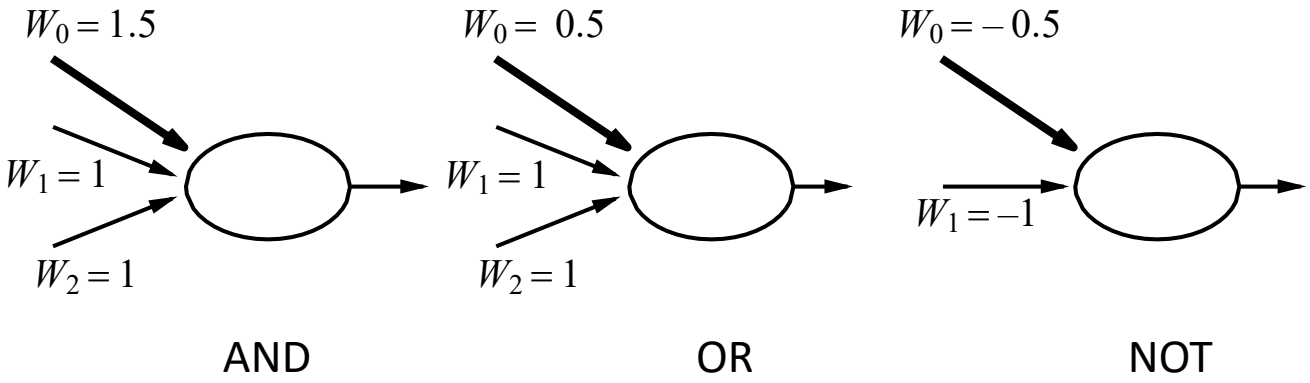
1943 McCulloch & Pitts: Boolean circuit model of brain

激活函数



- (a)是阶跃函数还是阈值函数
- (b)是一个sigmoid函数$1/(1 + e^{-x})$
- 改变偏置权值$W_{0,i}$移动阈值位置

实现逻辑功能

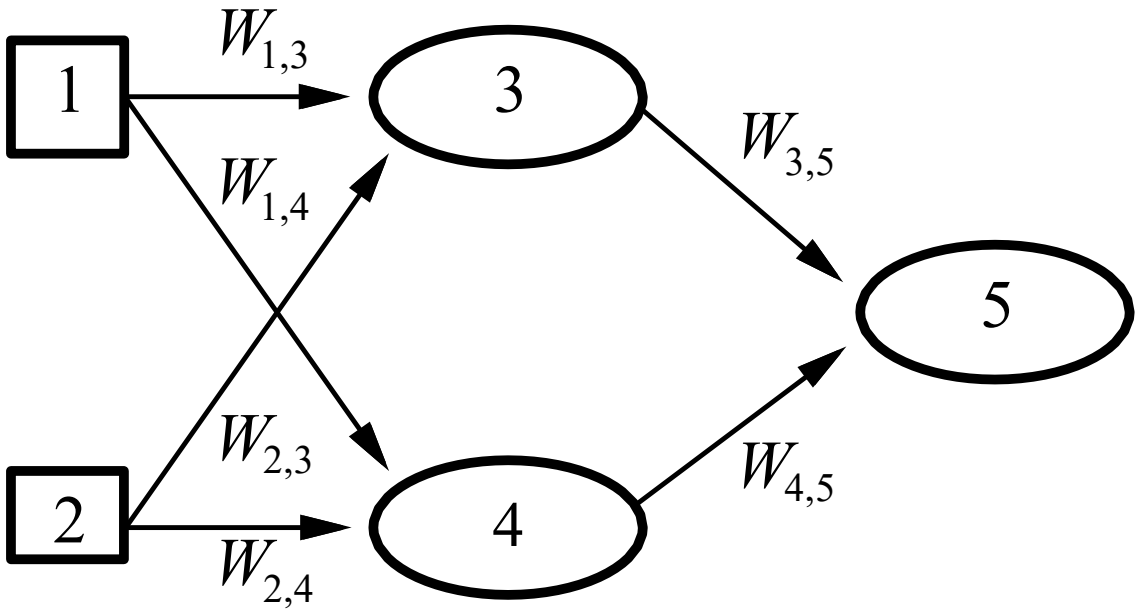


- McCulloch和Pitts假设：每个布尔函数都可以实现

网络结构

- 前馈网络：
  - 单层感知机
  - 多层感知机
- 前馈网络实现功能，内部没有状态
- 复发性网络：
  - Hopfield网络具有对称权值($W_{i,j} = W_{j,i}$)
  - $g(x) = \text{sign}(x)$, $a_i = \pm 1$， “全息联想记忆”
  - 玻尔兹曼机使用随机激活函数，
    - 这种方法与贝叶斯网络实现的MCMC类似
  - 递归神经网络具有有延迟的定向循环
    - ⇒有内部状态(如人字拖)，可以振荡等

前馈的例子



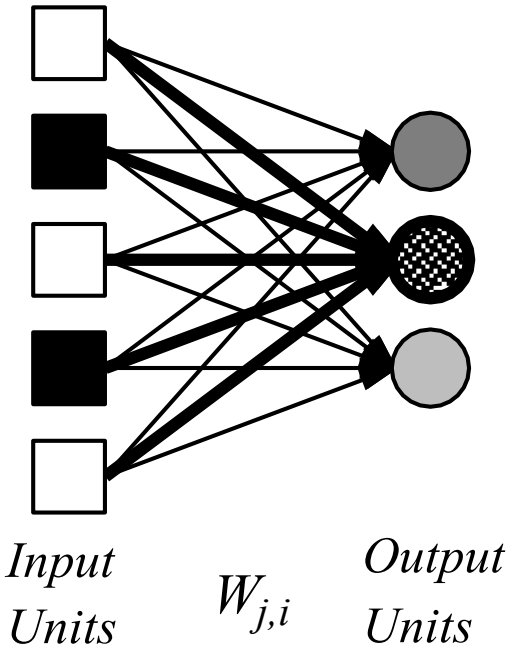
- 前馈网络=参数化的非线性函数族:

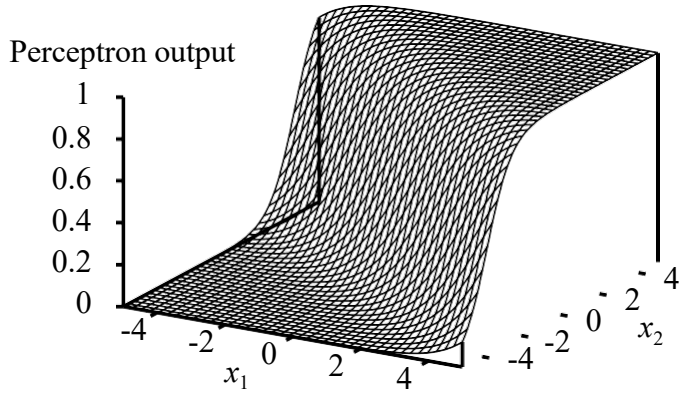
$a_5 = g(W_{3,5} \cdot a_3 + W_{4,5} \cdot a_4)$

$= g(W_{3,5} \cdot g(W_{1,3} \cdot a_1 + W_{2,3} \cdot a_2) + W_{4,5} \cdot g(W_{1,4} \cdot a_1 + W_{2,4} \cdot a_2))$

- 关键问题: 调整路径上的各个权重

单层感知机



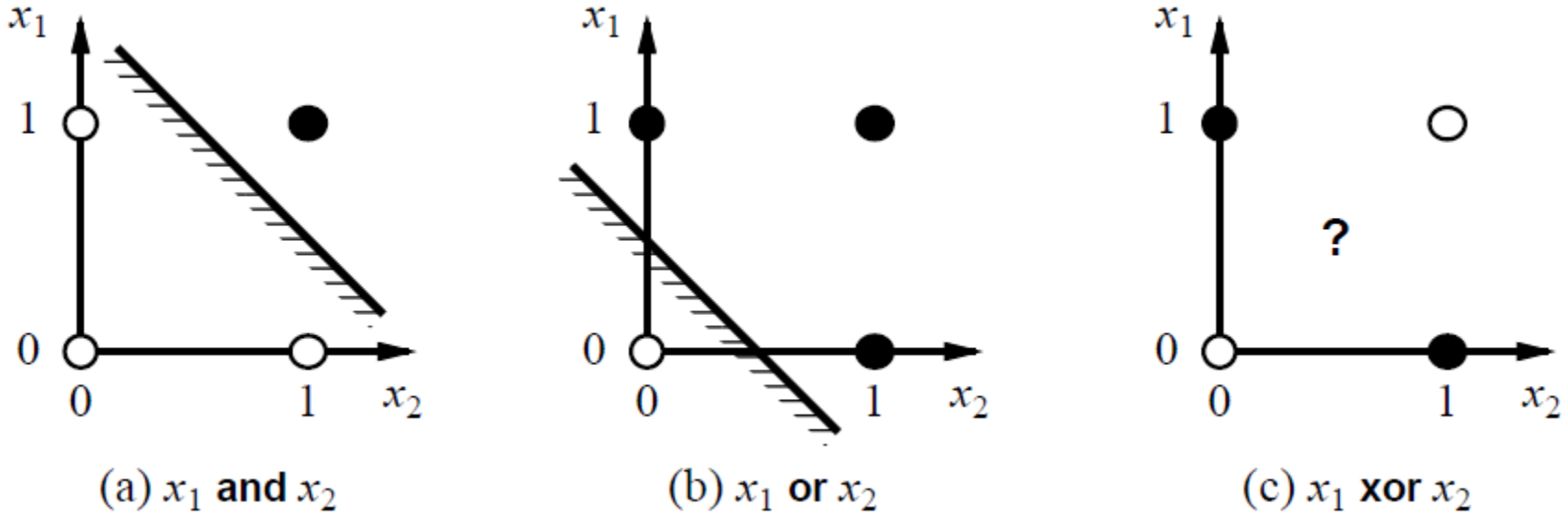


- 输出单元都独立运行，没有共享的权重
- 调整重量可以向悬崖的位置移动、方向和陡度

感知器的表达能力

- 考虑一个使用阶跃函数 g 的感知机
- 可以代表AND，OR，NOT等等，但不能表示XOR
- 输入空间中的线性分隔符可以表示为：

$$\Sigma_j W_j x_j > 0 \quad \text{or} \quad \mathrm{W} \cdot \mathrm{x} > 0$$



感知器学习

- 通过调整权值来减少训练集上的错误
- 输入x和输出y的平方误差为

$$E = \frac{1}{2}Err^2 = \frac{1}{2}(y - h_W(x))^2$$

- 采用梯度下降进行优化搜索：

$$\frac{\partial E}{\partial W_j} = Err \times \frac{\partial Err}{\partial W_j} = Err \times \frac{\partial}{\partial W_j}\left(y - g\left(\sum_{j=0}^{n} W_j x_j\right)\right)$$
$$= -Err \times g'(in) \times x_j$$

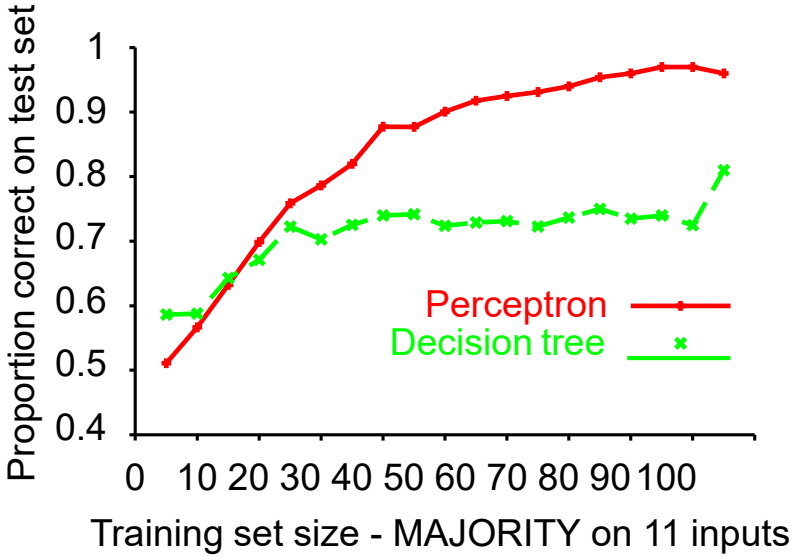
- 简单的权值更新规则：

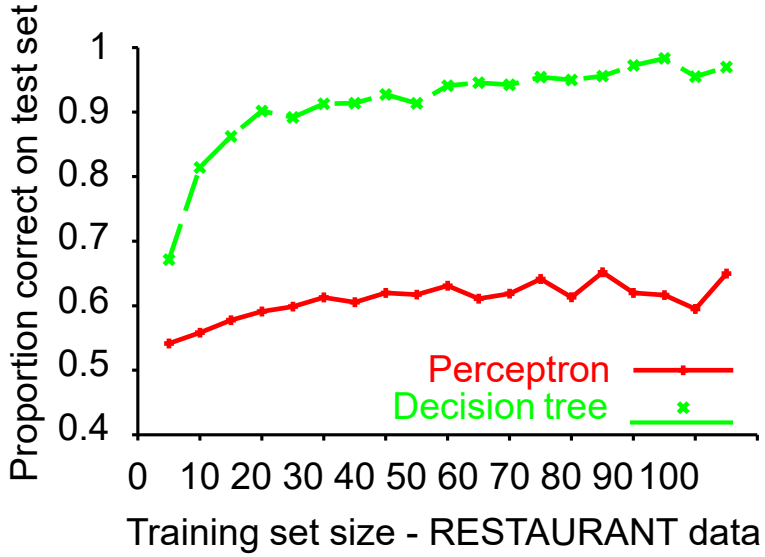
$$W_j \leftarrow W_j + \alpha \times Err \times g'(in) \times x_j$$

- 也就是说，如果网络计算结果有错误，可以通过对权值增加或减少错误量来更新网络结果

感知器学习

- 感知器擅长学习规则收敛于一致的函数
  - 任何线性可分的数据集

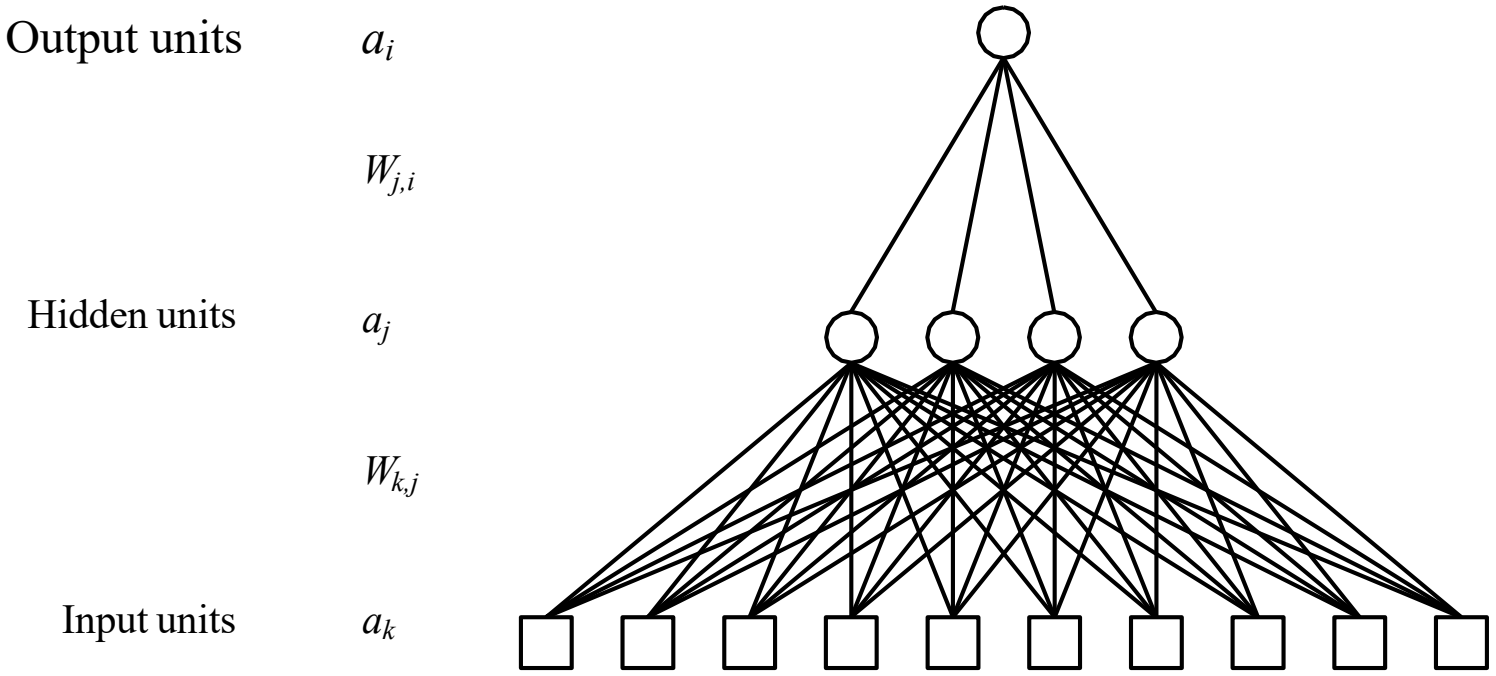




- 感知器很容易学会多数函数，DTL是无望的
- DTL很容易学习餐厅函数，感知器无法表示它

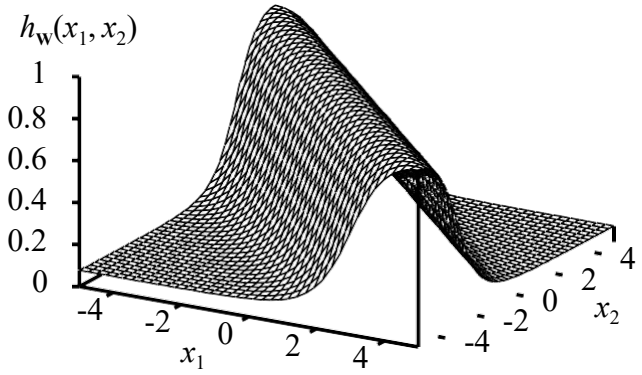
多层感知机

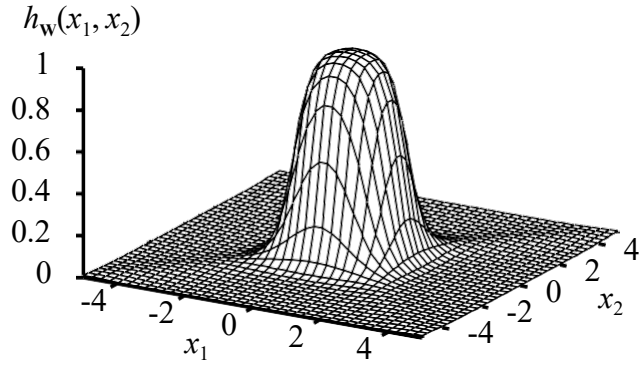
- 层和层之间通常是完全连接的
- 通常手工选择隐藏单位的数量



MLPs的表达能力

- 一般来说，连续函数需要2层权重，对大多数函数3层权重值都是足够的





- 将两个相对的阈值函数组合成一个脊
- 把两个垂直的山脊合并成隆起
- 添加各种大小和位置的凸起，以适合任何表面
- 证明需要指数级的隐藏单位

反向传播学习

- 输出层：与单层感知器相同

$$W_{j,i} \leftarrow W_{j,i} + \alpha \times a_j \times \Delta_i$$

- 这里$\Delta_i = Err_i \times g'(in_i)$
- 隐含层：从输出层反向传播错误

$$\Delta_j = g'(in_j) \sum_i W_{j,i} \Delta_i$$

- 更新隐含层权重规则：

$$W_{k,j} \leftarrow W_{k,j} + \alpha \times a_k \times \Delta_j$$

- 有趣的一点是，大多数神经科学家否认在大脑中有反向传播发生

反向传播推导

- 单个例子的平方误差定义为

$$E = \frac{1}{2}\sum_i (y_i - a_i)^2$$

- 按照这一方式，在输出层的节点上计算总和，得到：

$$\frac{\partial E}{\partial W_{j,i}} = -(y_i - a_i)\frac{\partial a_i}{\partial W_{j,i}} = -(y_i - a_i)\frac{\partial g(in_i)}{\partial W_{j,i}}$$

$$= -(y_i - a_i)g'(in_i)\frac{\partial g(in_i)}{\partial W_{j,i}}$$

$$= -(y_i - a_i)g'(in_i)\frac{\partial}{\partial W_{j,i}}\left(\sum_j W_{j,i}a_j\right) = -(y_i - a_i)g'(in_i)a_j$$

$$= -a_j\Delta_i$$

反向传播推导

$$\frac{\partial E}{\partial W_{k,i}} = -\sum_i (y_i - a_i)\frac{\partial a_i}{\partial W_{k,j}} = -\sum_i (y_i - a_i)\frac{\partial g(in_i)}{\partial W_{k,j}}$$

$$= -\sum_i (y_i - a_i) g'(in_i)\frac{\partial in_i}{\partial W_{k,j}} = -\sum_i \Delta_i \frac{\partial}{\partial W_{k,j}}\left(\sum_j w_{j,i} a_j\right)$$

$$= -\sum_i \Delta_i W_{j,i}\frac{\partial a_i}{\partial W_{k,j}} = -\sum_i \Delta_i W_{j,i}\frac{\partial g(in_i)}{\partial W_{k,j}}$$

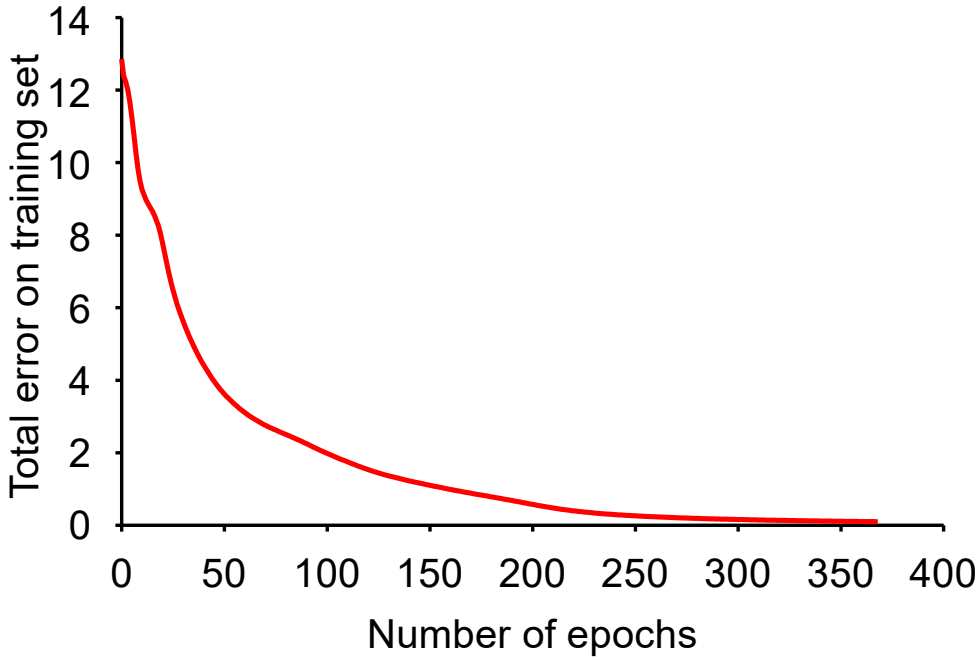
$$= -\sum_i \Delta_i W_{j,i} g'(in_j)\frac{\partial in_i}{\partial W_{k,j}}$$

$$= -\sum_i \Delta_i W_{j,i} g'(in_j)\frac{\partial}{\partial W_{k,j}}\left(\sum_k w_{k,j} a_k\right)$$

$$= -\sum_i \Delta_i W_{j,i} g'(in_j) a_k = -a_k \Delta_j$$

反向传播学习

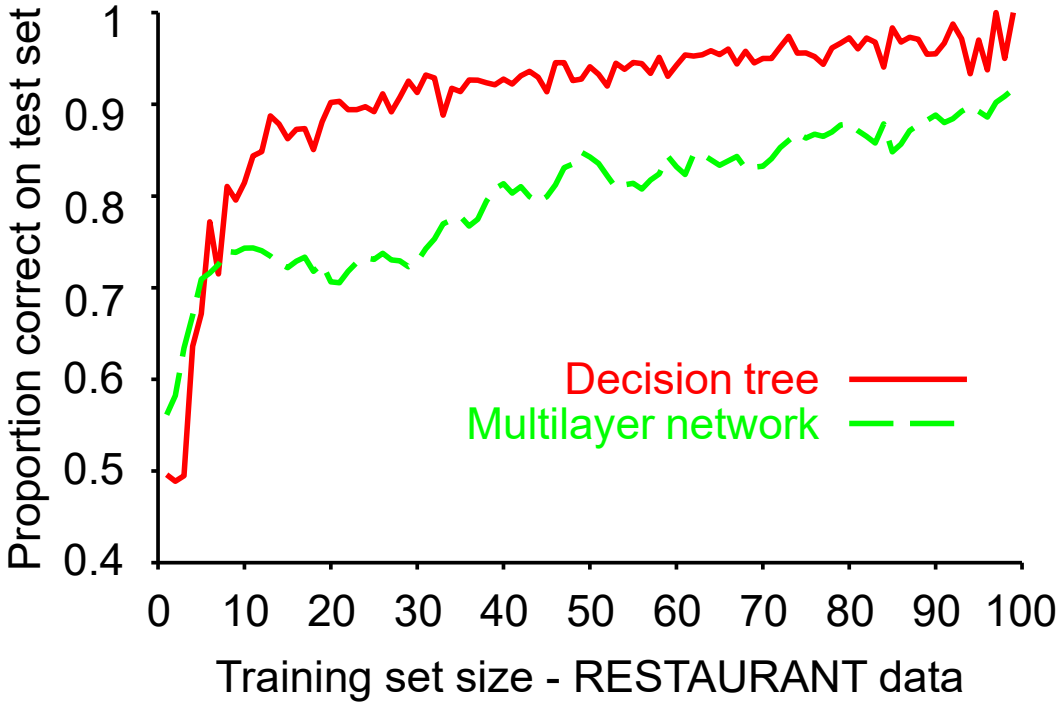
- 在每个epouch中，对在所有训练样本上合并梯度更新值
- 典型的训练曲线



- 典型问题：缓慢收敛，局部极小值

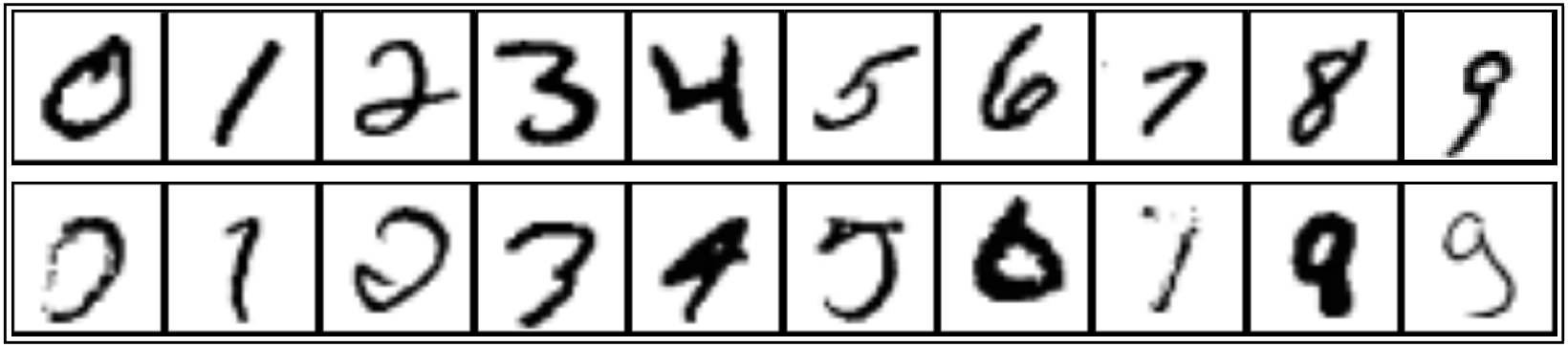
反向传播学习

- 在多层感知机上使用4个隐藏单元进行训练：



- MLPs对于复杂的模式识别任务来说是很好的，但是产生的假设并不容易被理解

手写识别



- 3-nearest-neighbor = 2.4% error
- 400–300–10 unit MLP = 1.6% error
- LeNet: 768–192–30–10 unit MLP = 0.9% error
- kernel machines, vision algorithms= 0.6% error

总结

- 大多数大脑有很多神经元，每个神经元都可近似为线性阈值单位
- 感知器(单层网络)表达能力不足
- 多层网络具有充分的表现力，可以通过梯度下降训练，即误差反向传播
- 应用广泛：语音、驾驶、书写、欺诈检测等