

---

# 第七章 神经网络



# 神经网络发展史

---

## 第一阶段

- 1943年, McCulloch和Pitts 提出第一个神经元数学模型, 即M-P模型, 并从原理上证明了人工神经网络能够计算任何算数和逻辑函数
- 1949年, Hebb 发表《The Organization of Behavior》一书, 提出生物神经元学习的机理, 即Hebb学习规则
- 1958年, Rosenblatt 提出感知机网络 (Perceptron) 模型和其学习规则
- 1960年, Widrow和Hoff提出自适应线性神经元 (Adaline) 模型和最小均方学习算法
- 1969年, Minsky和Papert 发表《Perceptrons》一书, 指出单层神经网络不能解决非线性问题, 多层网络的训练算法尚无希望. 这个论断导致神经网络进入低谷

# 神经网络发展史

---

## 第二阶段

- 1982年, 物理学家Hopfield提出了一种具有联想记忆、优化计算能力的递归网络模型, 即Hopfield 网络
- 1986年, Rumelhart 等编辑的著作《Parallel Distributed Proceesing: Explorations in the Microstructures of Cognition》报告了反向传播算法
- 1987年, IEEE 在美国加州圣地亚哥召开第一届神经网络国际会议 (ICNN)
- 90年代初, 伴随统计学习理论和SVM的兴起, 神经网络由于理论不够清楚, 试错性强, 难以训练, 再次进入低谷

# 神经网络发展史

## 第三阶段

- 2006年, Hinton提出了深度信念网络(DBN), 通过“预训练+微调”使得深度模型的最优化变得相对容易
- 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
- 伴随云计算、大数据时代的到来, 计算能力的大幅提升, 使得深度学习模型在计算机视觉、自然语言处理、语音识别等众多领域都取得了较大的成功

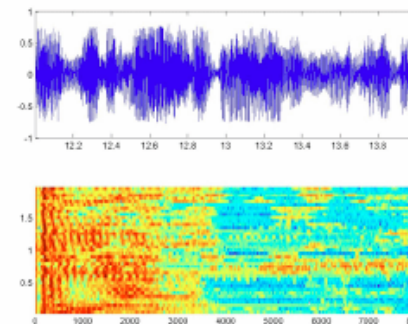
Images & Video



Text & Language

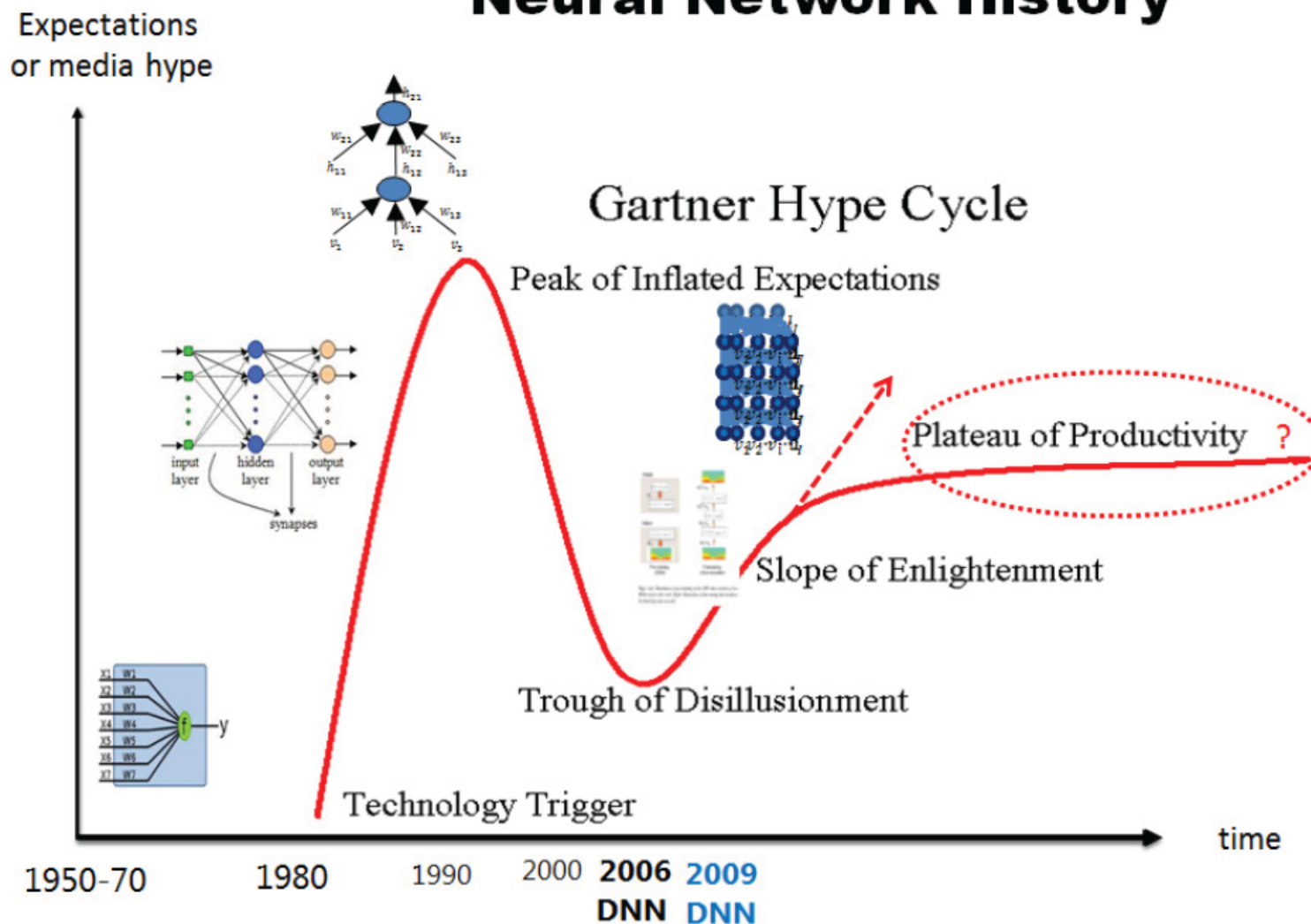


Speech & Audio



# 神经网络发展史

## Neural Network History



# 第七章 神经网络

---

## 主要内容

- 7.1 神经元模型
- 7.2 感知机与多层网络
- 7.3 误差逆传播算法
- 7.4 全局最小与局部最小
- 7.5 其他常见神经网络
- 7.6 深度学习

# 第七章 神经网络

---

## 主要内容

- 7.1 神经元模型
- 7.2 感知机与多层网络
- 7.3 误差逆传播算法
- 7.4 全局最小与局部最小
- 7.5 其他常见神经网络
- 7.6 深度学习

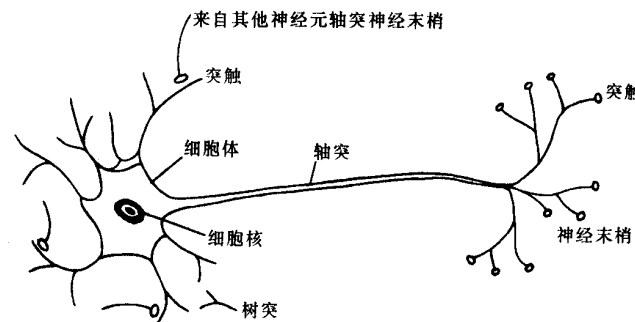
# 7.1 神经元模型

## □ 神经网络的定义

“神经网络是由具有适应性的简单单元组成的广泛并行互联的网络，它的组织能够模拟生物神经系统对真实世界物体所作出的反应”

[Kohonen, 1988, Neural Networks创刊号]

- 机器学习中的神经网络通常是指“神经网络学习”（“连接主义学习”）或者机器学习与神经网络两个学科的交叉部分
- 神经元模型即上述定义中的“简单单元”是神经网络的基本成分
- 生物神经网络：每个神经元与其他神经元相连，当它“兴奋”时，就会向相连的神经元发送化学物质，从而改变这些神经元内的电位；如果某神经元的电位超过一个“阈值”，那么它就会被激活，即“兴奋”起来，向其它神经元发送化学物质

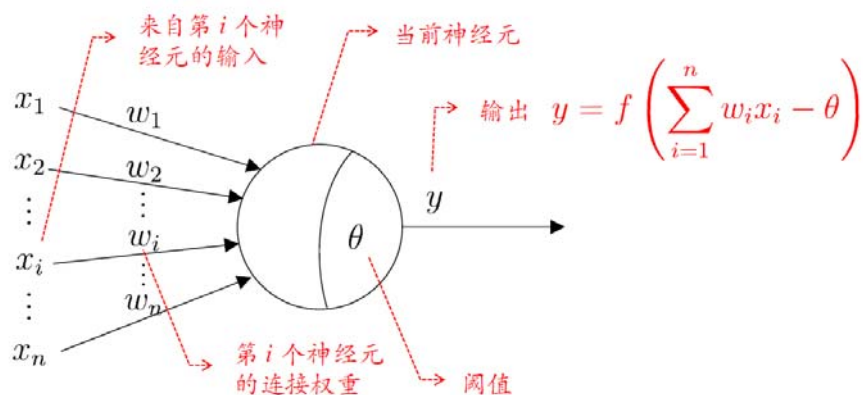




# 7.1 神经元模型

## M-P 神经元模型 [McCulloch and Pitts, 1943]

- 输入：来自其他 $n$ 个神经元传递过来的输入信号
- 处理：输入信号通过带权重的连接进行传递，神经元接受到总输入值将与神经元的阈值进行比较
- 输出：通过激活函数的处理以得到输出

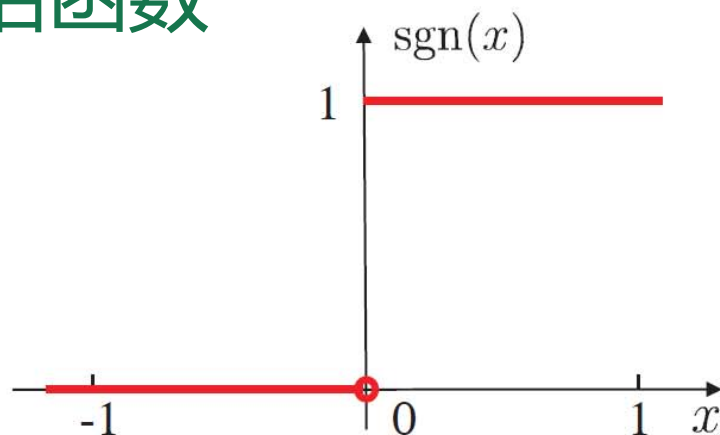


M-P 神经元模型

神经网络学得的知识蕴含在？

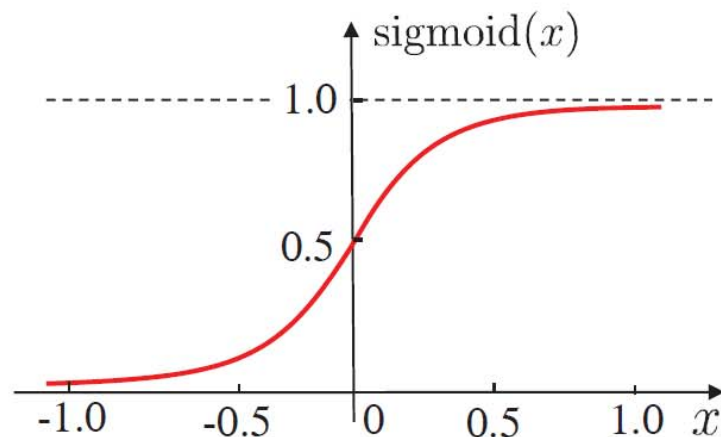
# 7.1 神经元模型

## 激活函数



$$\text{sgn}(x) = \begin{cases} 1, & \text{if } x \geq 0; \\ 0, & \text{if } x < 0. \end{cases}$$

(a) 阶跃函数



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

(b) Sigmoid 函数

典型的神经元激活函数

- 理想激活函数是阶跃函数, 0表示抑制神经元而1表示激活神经元
- 阶跃函数具有不连续、不光滑等不好的性质, 常用的是 Sigmoid 函数

# 第七章 神经网络

---

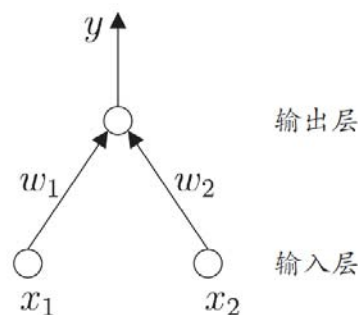
## 主要内容

- 7.1 神经元模型
- 7.2 感知机与多层网络
- 7.3 误差逆传播算法
- 7.4 全局最小与局部最小
- 7.5 其他常见神经网络
- 7.6 深度学习

## 7.2 感知机与多层网络

### 感知机

- 感知机由两层神经元组成，输入层接受外界输入信号传递给输出层，输出层是M-P神经元（阈值逻辑单元）
- 感知机能够容易地实现逻辑与、或、非运算
  - “与”  $x_1 \wedge x_2$ : 令  $w_1 = w_2 = 1, \theta = 0$ , 则
$$y = f(1 \cdot x_1 + 1 \cdot x_2 - 2),$$
 仅在  $x_1 = x_2 = 1$  时,  $y = 1$ .
  - “或”  $x_1 \vee x_2$ : 令  $w_1 = w_2 = 1, \theta = 0.5$ , 则
$$y = f(1 \cdot x_1 + 1 \cdot x_2 - 2),$$
 仅在  $x_1 = 1$  或者  $x_2 = 1$  时,  $y = 1$
  - “非”  $\neg x_1$ : 令  $w_1 = -0.6, w_2 = 0, \theta = -0.5$ , 则  
当  $x_1 = 1$  时,  $y = 0$ ; 当  $x_1 = 0$ ,  $y = 1$ .



两个输入神经元的感知机网络结构示意图

## 7.2 感知机与多层网络

---

### 感知机学习

□ 给定训练数据集, 权重  $w_i (i = 1, 2, \dots, n)$  与阈值  $\theta$  可以通过学习得到

□ 感知机学习规则

对训练样例  $(\mathbf{x}, y)$  若当前感知机的输出为  $\hat{y}$ , 则感知机权重调整规则为:

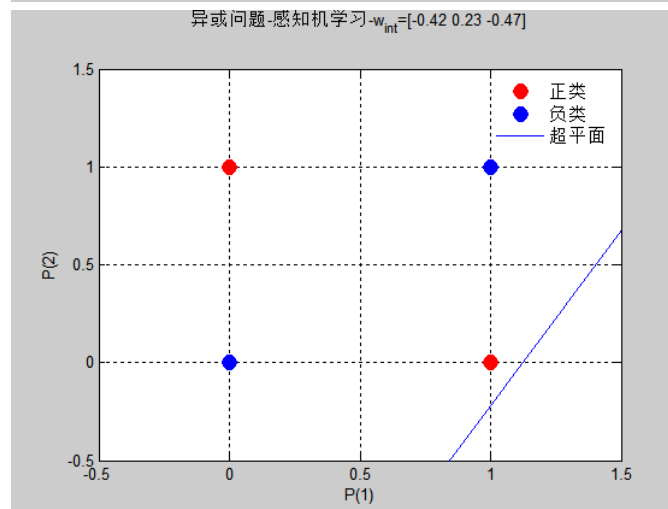
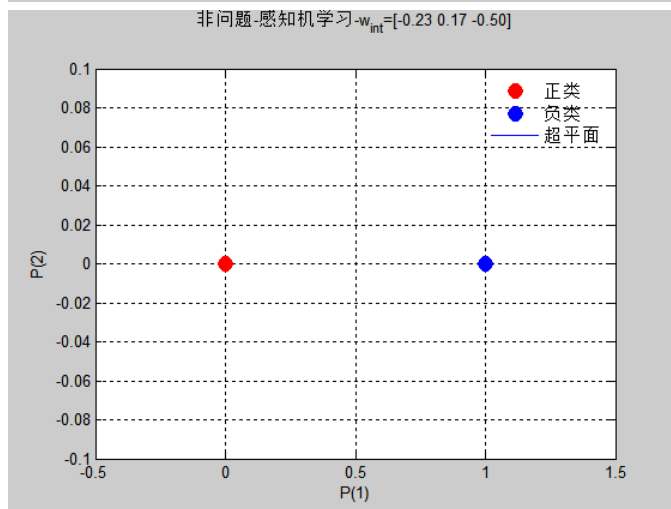
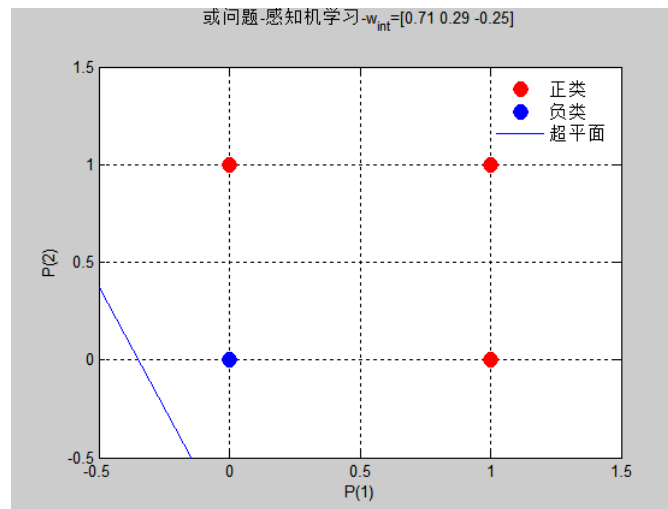
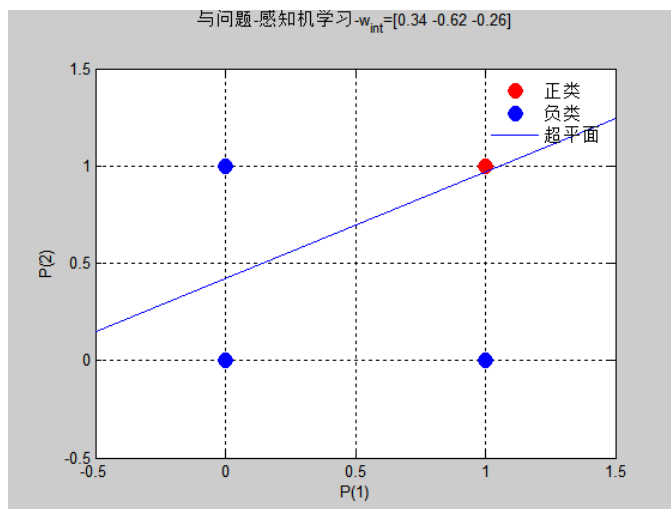
$$\begin{aligned} w_i &\leftarrow w_i + \Delta w_i \\ \Delta w_i &= \eta(y - \hat{y})x_i \end{aligned}$$

其中  $\eta \in (0, 1)$  称为学习率

若感知机对训练样例  $(\mathbf{x}, y)$  预测正确, 则感知机不发生变化; 否则根据错误程度进行权重的调整.

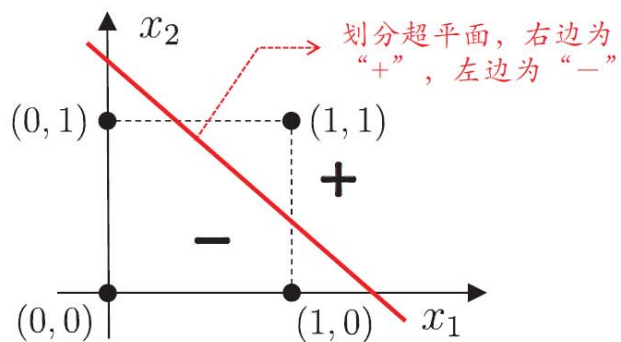
## 7.2 感知机与多层网络

### 感知机求解与、或、非问题

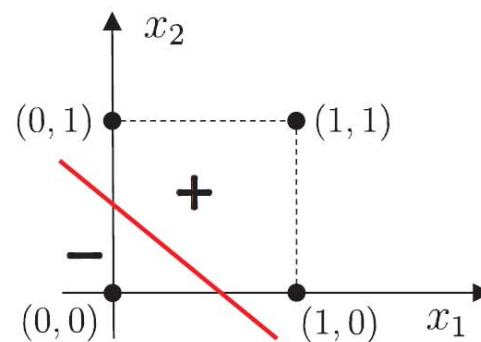


## 7.2 感知机与多层网络

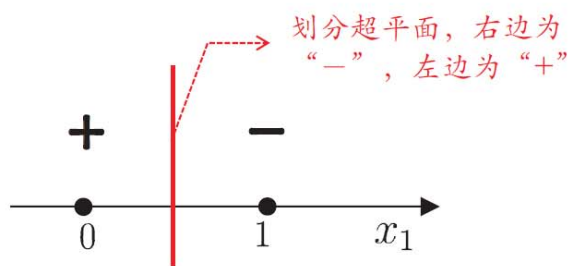
### 感知机求解与、或、非问题



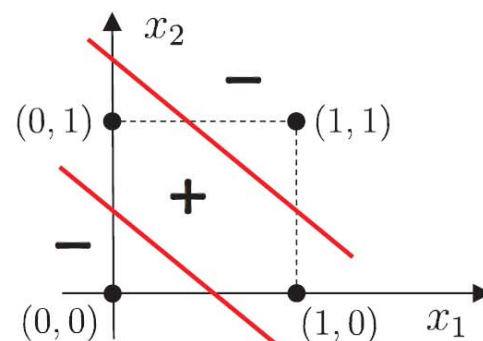
(a) “与” 问题 ( $x_1 \wedge x_2$ )



(b) “或” 问题 ( $x_1 \vee x_2$ )



(c) “非” 问题 ( $\neg x_1$ )



(d) “异或” 问题 ( $x_1 \oplus x_2$ )

线性可分的“与”“或”“非”问题与非线性可分的“异或”问题

## 7.2 感知机与多层网络

---

### 感知机学习能力

- 若两类模式线性可分，则感知机的学习过程一定会收敛；否感知机的学习过程将会发生震荡

[Minsky and Papert, 1969]

- 单层感知机的学习能力非常有限，只能解决线性可分问题
- 事实上，与、或、非问题是线性可分的，因此感知机学习过程能够求得适当的权值向量。而异或问题不是线性可分的，感知机学习不能求得合适解
- 对于非线性可分问题，如何求解？

多层感知机

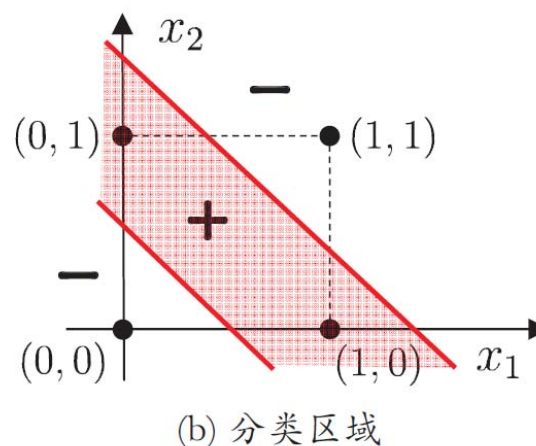
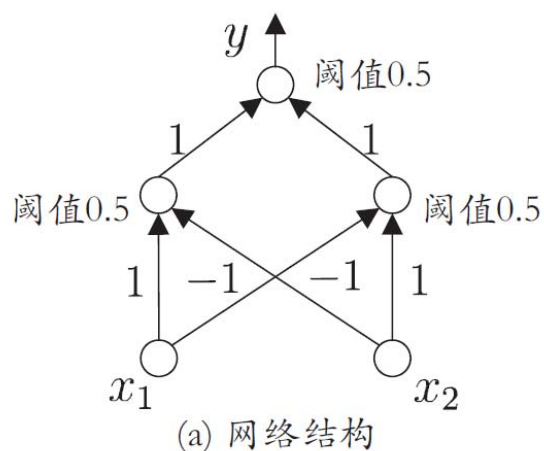




## 7.2 感知机与多层网络

### 多层感知机

#### □ 解决异或问题的两层感知机



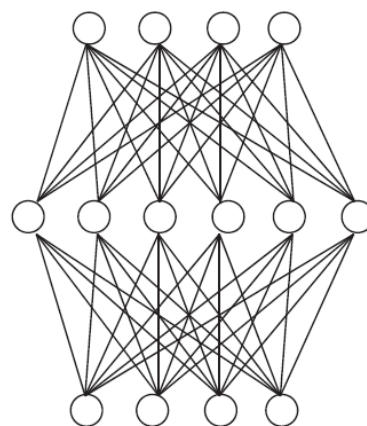
能解决异或问题的两层感知机

- 输出层与输入层之间的一层神经元，被称之为**隐层或隐含层**，隐含层和输出层神经元都是具有激活函数的功能神经元

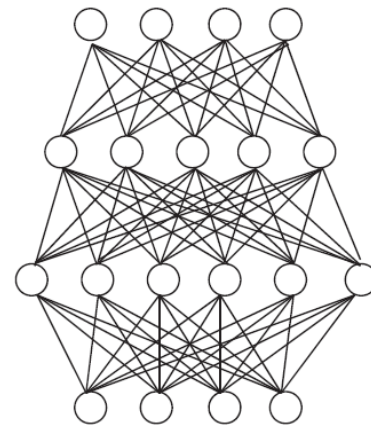
## 7.2 感知机与多层网络

### 多层前馈神经网络

- **定义**：每层神经元与下一层神经元全互联，神经元之间不存在同层连接也不存在跨层连接
- **前馈**：输入层接受外界输入，隐含层与输出层神经元对信号进行加工，最终结果由输出层神经元输出
- **学习**：根据训练数据来调整神经元之间的“连接权”以及每个功能神经元的“阈值”
- **多层网络**：包含隐层的网络



(a) 单隐层前馈网络



(b) 双隐层前馈网络

# 第七章 神经网络

---

## 主要内容

- 7.1 神经元模型
- 7.2 感知机与多层网络
- 7.3 误差逆传播算法
- 7.4 全局最小与局部最小
- 7.5 其他常见神经网络
- 7.6 深度学习

## 7.3 误差逆传播算法

**误差逆传播算法** (Error BackPropagation, 简称BP) 是最成功的训练多层前馈神经网络的学习算法

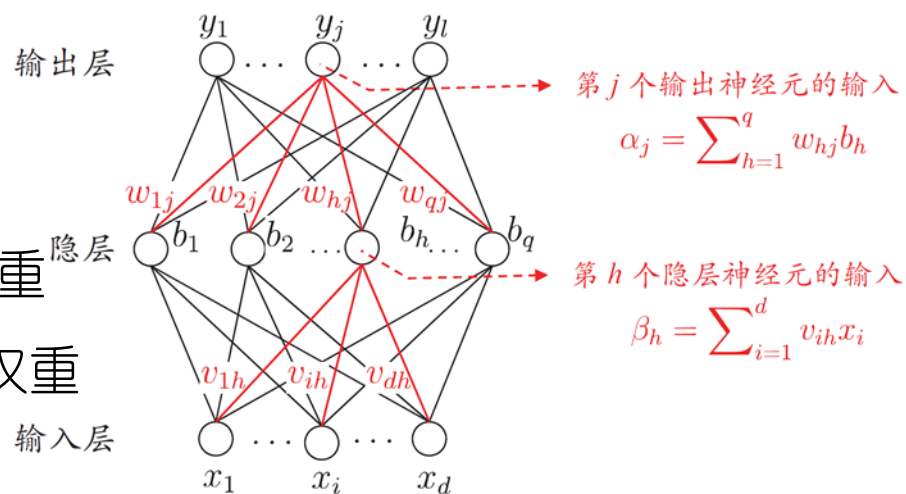
- 给定训练集  $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}$ ,  $\mathbf{x}_i \in R^d, \mathbf{y}_i \in R^l, (i = 1, 2, \dots, m)$ , 即输入示例由  $d$  个属性描述, 输出  $l$  维实值向量
- 为方便讨论, 给定一个拥有  $d$  个输入神经元,  $l$  个输出神经元  $q$  个隐层神经元的多层前向前馈网络结构
- 记号:

$\theta_j$ : 输出层第  $j$  个神经元阈值

$\gamma_h$ : 隐含层第  $h$  个神经元阈值

$v_{ih}$ : 输入层与隐层神经元之间的连接权重

$w_{hj}$ : 隐层与输出层神经元之间的连接权重



## 7.3 误差逆传播算法

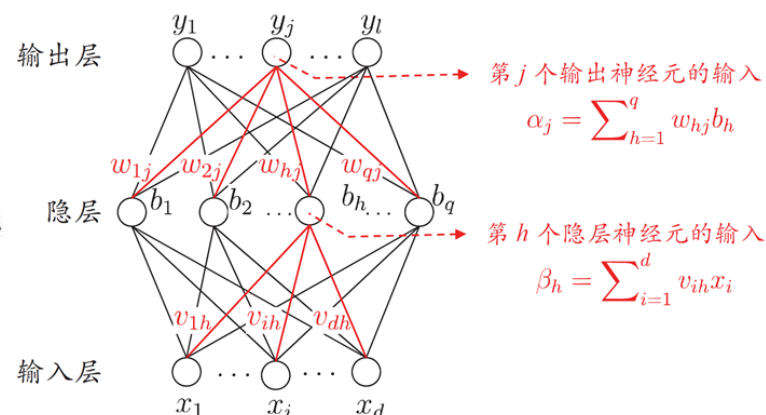
对于样本  $(\mathbf{x}_k, \mathbf{y}_k)$  假设网络的实际输出为  $\hat{\mathbf{y}}_k$

### 前向计算

step1:  $b_h = f(\beta_h - \gamma_h), \beta_h = \sum_{i=1}^d v_{ih}x_i$

step2:  $\hat{y}_j^k = f(\alpha_j - \theta_j), \alpha_h = \sum_{i=1}^q w_{hj}b_h$

step3:  $E_k = \frac{1}{2} \sum_{j=1}^l (\hat{y}_j^k - y_j^k)^2$



### 参数数目

权重:  $v_{ih}, w_{hj}$  阈值:  $\theta_j, \gamma_h$  ( $i = 1, \dots, d, h = 1, \dots, q, j = 1, \dots, l$ )

因此网络中需要  $(d + l + 1)q + l$  个参数需要优化

### 参数优化

BP是一个迭代学习算法, 在迭代的每一轮中采用广义的感知机学习规则对参数进行更新估计, 任意的参数  $v$  的更新估计式为

$$v \leftarrow v + \Delta v.$$

## 7.3 误差逆传播算法

### BP 学习算法

- 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。对误差  $E_k$ ，给定学习率  $\eta$

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

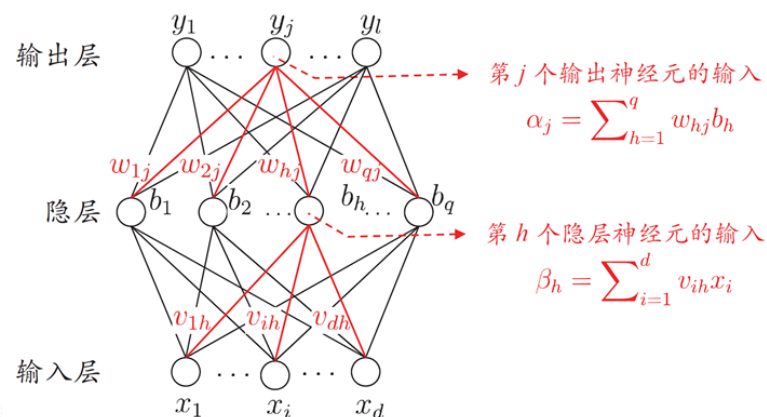
$$\frac{\partial E_k}{\partial w_{hj}} = \frac{\partial E_k}{\partial \hat{y}_j^k} \cdot \frac{\partial \hat{y}_j^k}{\partial \beta_j} \cdot \frac{\partial \beta_j}{\partial w_{hj}}$$

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

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

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

$$\Delta w_{hj} = \eta g_j b_h$$



## 7.3 误差逆传播算法

### BP 学习算法

- 类似的可以推导出：

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

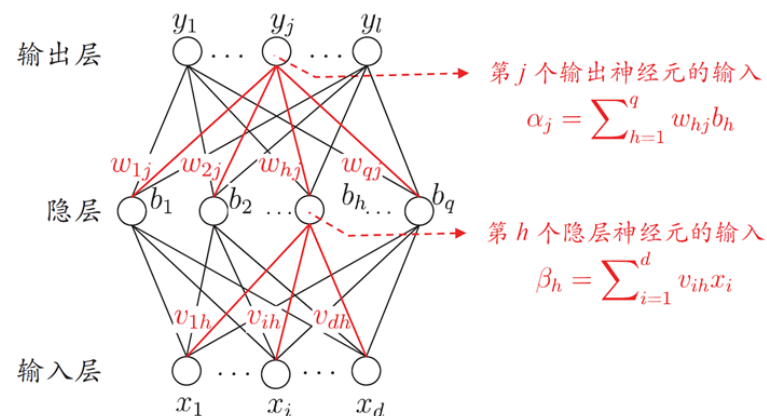
$$\Delta v_{ih} = \eta e_h x_i,$$

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

其中

$$e_h = -\frac{\partial E_k}{\partial b_h} \cdot \frac{\partial b_h}{\partial \alpha_h}$$

$$= -\sum_{j=1}^l \frac{\partial E_k}{\partial \beta_j} \frac{\partial b_h}{\partial \beta_j} f'(\alpha_h - \gamma_h)$$



$$= \sum_{j=1}^h w_{hj} g_j f'(\alpha_h - \gamma_h)$$
$$= b_h(1 - b_h) \sum_{j=1}^h w_{hj} g_j.$$

- 学习率  $\eta \in (0, 1)$  控制着算法每一轮迭代中的更新步长，若太长则容易震荡，太小则收敛速度又会过慢。

## 7.3 误差逆传播算法

### BP 学习算法

---

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

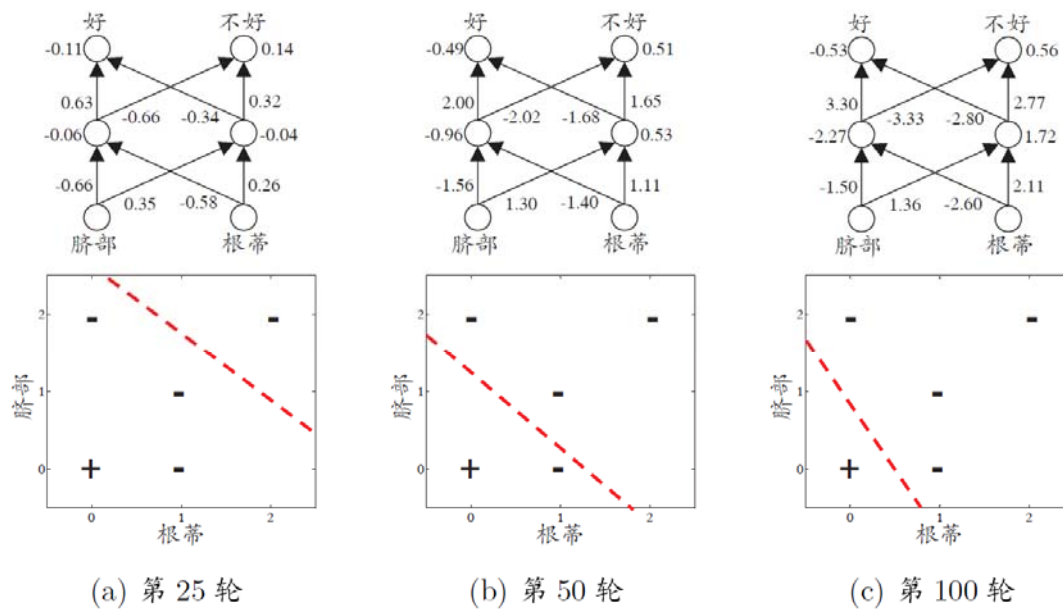
---

误差逆传播算法



## 7.3 误差逆传播算法

### BP 算法实验



在 2 个属性、5 个样本的水瓜数据上, BP 网络参数更新和分类边界的变化情况

## 7.3 误差逆传播算法

---

### □ 标准 BP 算法

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

### □ 累计 BP 算法

- 其优化的目标是最小化整个训练集上的累计误差

$$E = \frac{1}{m} \sum_{k=1}^m E_k$$

- 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

### □ 实际应用

但在很多任务中, 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

## 7.3 误差逆传播算法

---

### □ 多层前馈网络表示能力

只需要一个包含足够多神经元的隐层，多层前馈神经网络就能以任意精度逼近任意复杂度的连续函数（“万有逼近性”，强大的表示能力）

[Hornik et al. , 1989]

### □ 多层前馈网络局限

- 神经网络由于强大的表示能力，经常遭遇过拟合。表现为：训练误差持续降低，但测试误差却可能上升
- 如何设置隐层神经元的个数仍然是个未决问题（Open Problem）。实际应用中通常使用“试错法”调整

### □ 缓解过拟合的策略

- **早停**：在训练过程中，若训练误差降低，但验证误差升高，则停止训练
- **正则化**：在误差目标函数中增加一项描述网络复杂程度的部分，例如连接权值与阈值的平方和

# 第七章 神经网络

---

## 主要内容

- 7.1 神经元模型
- 7.2 感知机与多层网络
- 7.3 误差逆传播算法
- 7.4 全局最小与局部最小
- 7.5 其他常见神经网络
- 7.6 深度学习

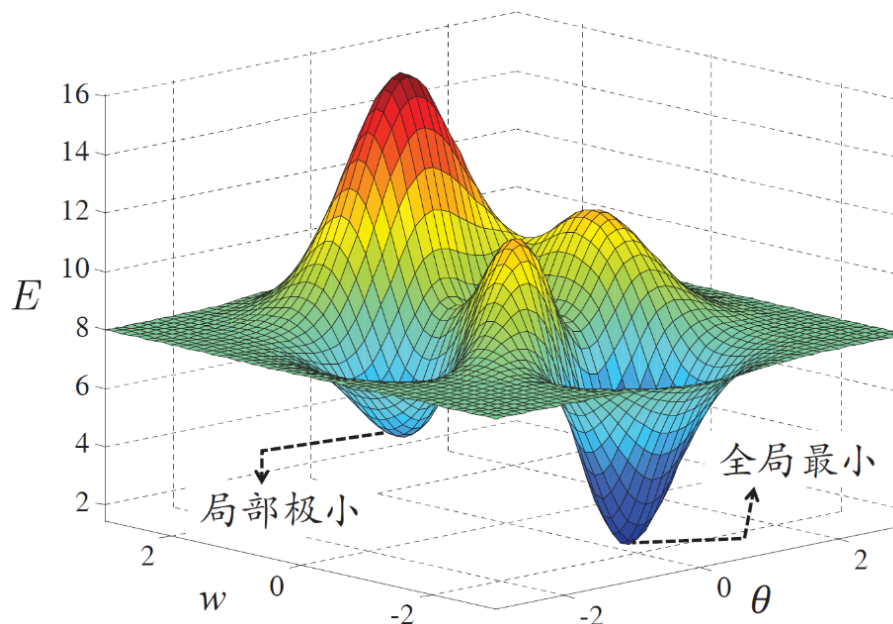
## 7.4 全局最小与局部极小

□ 对  $\mathbf{w}^*$  和  $\theta^*$ , 若存在  $\epsilon > 0$  使得

$$\forall (\mathbf{w}; \theta) \in \{\|(\mathbf{w}; \theta) - (\mathbf{w}^*; \theta^*)\| \leq \epsilon\},$$

都有  $E(\mathbf{w}; \theta) \geq E(\mathbf{w}^*; \theta^*)$  成立, 则  $(\mathbf{w}^*; \theta^*)$  为局部极小解; 若对参数空间中任意的  $(\mathbf{w}; \theta)$ , 都有  $E(\mathbf{w}; \theta) \geq E(\mathbf{w}^*; \theta^*)$ , 则  $(\mathbf{w}^*; \theta^*)$  为全局最小解. 两者对应的  $E(\mathbf{w}^*; \theta^*)$  分别称为误差函数的局部最小解和全局最小值

- 显然参数空间梯度为零的点, 只要其误差函数值小于邻点的误差函数值, 就是局部极小点
- 可能存在多个局部极小值, 但却只会有一个全局极最小值



## 7.4 全局最小与局部极小

---

### □ “跳出”局部最小的策略

基于梯度的搜索是使用最为广泛的参数寻优方法。如果误差函数仅有一个局部极小，那么此时找到的局部极小就是全局最小；然而，如果误差函数具有多个局部极小，则不能保证找到的解是全局最小。在现实任务中，通常采用一些策略“跳出”局部极小，从而进一步达到全局最小

- 多组不同的初始参数优化神经网络，选取误差最小的解作为最终参数
- 模拟退火技术 [Aarts and Korst, 1989]。每一步都以一定的概率接受比当前解更差的结果，从而有助于跳出局部极小
- 随机梯度下降。与标准梯度下降法精确计算梯度不同，随机梯度下降法在计算梯度时加入了随机因素
- 遗传算法 [Goldberg, 1989]。遗传算法也常用来训练神经网络以更好地逼近全局极小

# 第七章 神经网络

---

## 主要内容

- 7.1 神经元模型
- 7.2 感知机与多层网络
- 7.3 误差逆传播算法
- 7.4 全局最小与局部最小
- 7.5 其他常见神经网络
- 7.6 深度学习

## 7.5 其他常见神经网络

---

### RBF 网络 [Broomhead and Lowe, 1988]

□ RBF 网络是一种单隐层前馈神经网络，它使用径向基函数作为隐层神经元激活函数，而输出层则是隐层神经元输出的线性组合

#### □ RBF网络模型

假定输入为  $d$  维的向量  $\mathbf{x}$ ，输出为实值，则RBF网络可以表示为

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

其中  $q$  为隐层神经元的个数， $\mathbf{c}_i$  和  $w_i$  分别是第  $i$  神经元对应的中心和权重， $\rho(\mathbf{x}, \mathbf{c}_i)$  是径向基函数

常用的高斯径向基函数形如

$$\rho(\mathbf{x}, \mathbf{c}_i) = e^{-\beta_i \|\mathbf{x} - \mathbf{c}_i\|^2}$$



## 7.5 其他常见神经网络

---

### RBF 网络

#### □ RBF网络性质

具有足够多隐层神经元RBF 神经网络能以任意精度逼近任意连续函数

[Park and Sandberg, 1991]

#### □ RBF网络训练

- 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

## 7.5 其他常见神经网络

---

### ART 网络

#### □ 竞争学习

竞争学习是神经网络中一种常用的无监督学习策略，在使用该策略时，网络的输出神经元相互竞争，每一时刻仅有一个神经元被激活，其他神经元的状态被抑制

#### □ ART 网络 [Carpenter and Grossberg, 1987]

- ART 网络是竞争学习的重要代表
- ART 网络由比较层、识别层、识别阈值和重置模块构成
- 比较层负责接收输入样本，并将其传送给识别层神经元
- 识别层每个神经元对应一个模式类，神经元的数目可在训练过程中动态增长以增加新的模式类

## 7.5 其他常见神经网络

---

### ART 网络

#### □ ART 网络性能依赖于识别阈值

- 识别阈值高时，输入样本将会分成比较多、较精细的模式类
- 识别阈值低时，输入样本将会分成比较少、较粗略的模式类

#### □ ART 网络的优势

- ART较好的解决了竞争学习中的“可塑性-稳定性窘境”，可塑性是指神经网络要有学习新知识的能力；稳定性是指神经网络在学习新知识时要保持对旧知识的记忆
- ART网络可以增量学习或在线学习

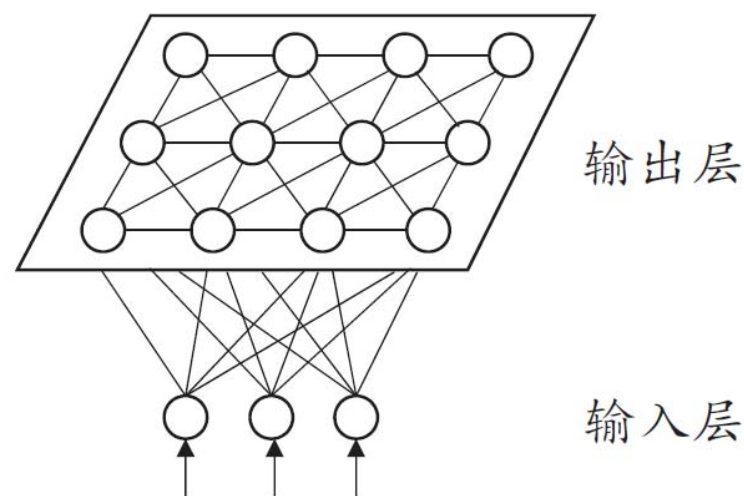
#### □ ART 网络的发展

ART2 网络、FuzzyART 网络、ARTMAP 网络

## 7.5 其他常见神经网络

### SOM 网络 [Kohonen, 1982]

- SOM 网络是一种竞争型的无监督神经网络，它能够将高维数据映射到低维空间（通常为二维），同时保持输入数据在高维空间的拓扑结构，即将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
- 如图，SOM 网络中的输出层神经元以矩阵方式排列在二维空间中，每个神经元都拥有一个权值向量，网络在接收输入向量后，将会确定输出层获胜神经元，它决定了该输入向量在低维空间中的位置



SOM 网络结构

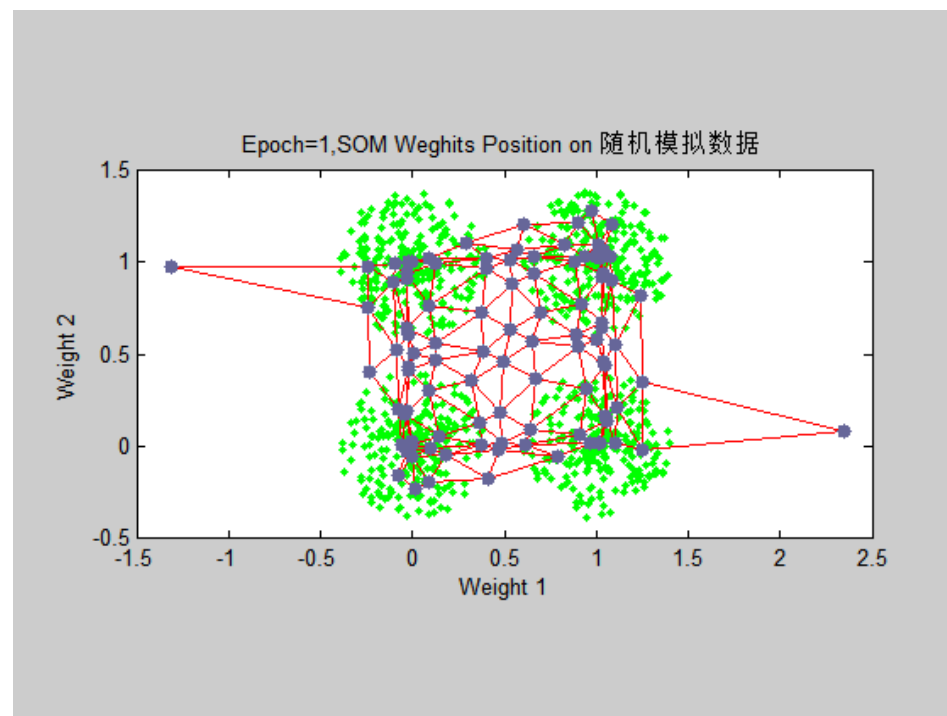
## 7.5 其他常见神经网络

### SOM 网络 [Kohonen, 1982]

#### □ SOM 网络训练

Step1: 接受到一个训练样本后, 每个输出层神经元计算该样本与自身携带的权向量之间的距离, 距离最近的神经元成为竞争获胜者

Step2: 最佳匹配单元及其近邻神经元的权值将被调整, 使得这些权向量与当前输入样本的距离缩小



## 7.5 其他常见神经网络

### 级联相关网络 [Fahlman and Lebiere 1990]

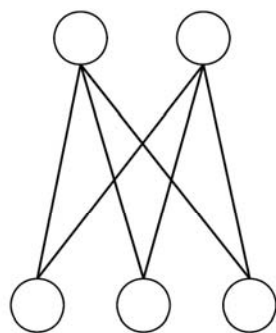
级联相关网络不仅利用训练样本优化连接权值, 阈值参数, 将网络的结构也当做学习的目标之一, 希望在训练过程中找到适合数据的网络结构

#### □ 级联与相关

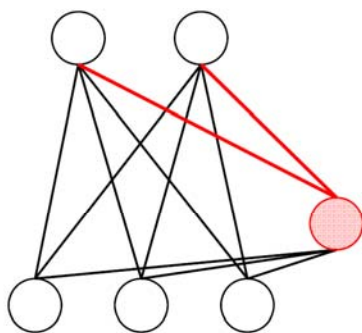
**级联:** 建立层次连接的层级结构

**相关:** 最大化新神经元的输出与网络误差之间的相关性来训练相关参数

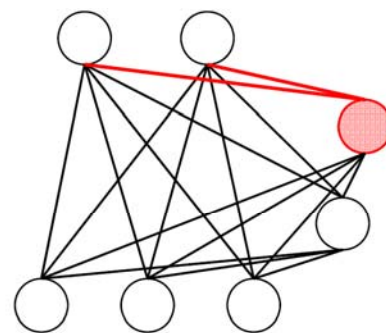
#### □ 网络优化演示



(a) 初始状态



(b) 增加一个隐层结点



(c) 增加第二个隐层结点

## 7.5 其他常见神经网络

### Elman 网络 [Elman 1990]

#### □ 循环神经网络

- 允许网络中出现环形结构，使得神经元的输出反馈回来作为输入信号
- $t$  时刻网络的输出状态：由  $t$  时刻的输入状态和  $t-1$  时刻的网络状态决定

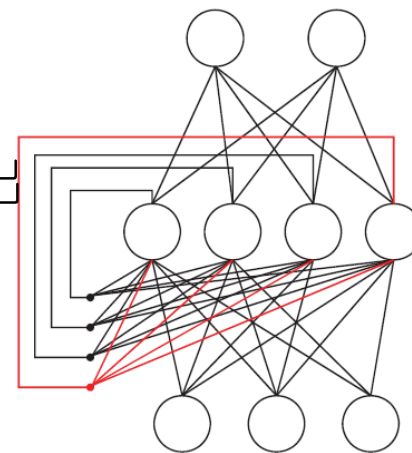
#### □ Elman 网络

Elamn 网络结构如图所示，

这种结构与前馈神经网络很相似，但是隐层神经元的输出被反馈回来，与下一时刻输入层神经元提供的信号一起，作为隐层神经元在下一时刻的输入

#### □ 训练算法

推广的BP算法 [Pineda, 1987]



Elman 网络结构

## 7.5 其他常见神经网络

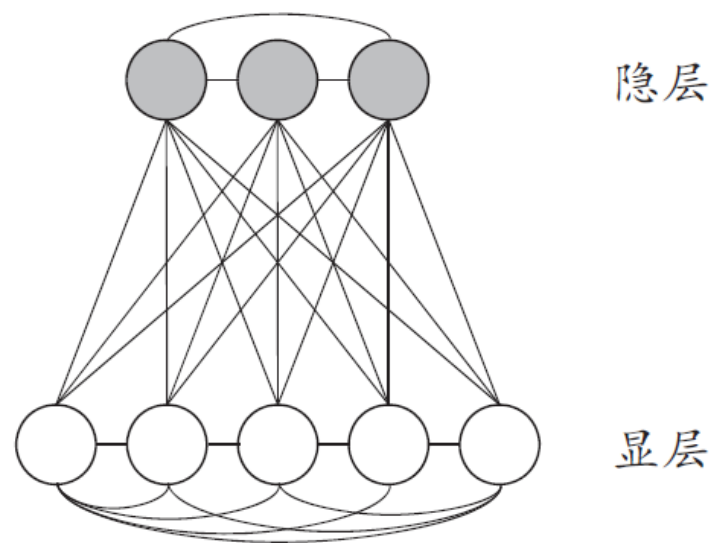
### Boltzmann 机 [Ackley et al. , 1985]

#### □ 能量模型

神经网络中有一类模型为网络定义一个“能量”，能量最小化时网络达到理想状态，而网络的训练就是在最小化这个能量函数

#### □ Boltzmann 机

- Boltzmann 机就是一种基于能量的模型
- 结构：显层与隐层
  - 显层：数据的输入输出
  - 隐层：数据的内在表达
- 神经元
  - 布尔型，即只能取0和1 两种状态，其中1 表示激活，0表示抑制



(a) Boltzmann机



## 7.5 其他常见神经网络

---

### Boltzmann 机

#### □ Boltzmann 机能量

令状态向量  $\mathbf{s} \in \{0, 1\}^n$  则其对应的Boltzmann 机能量定义为

$$E(\mathbf{s}) = - \sum_{i=1}^{n-1} \sum_{j=i+1}^n w_{ij} s_i s_j - \sum_{i=1}^n \theta_i s_i$$

其中  $w_{ij}$  表示两个神经元之间的连接权值， $\theta_i$  表示神经元的阈值

#### □ Boltzmann 分布

网络中的神经元以任意不依赖与输入值得顺序进行更新，则网络最终将达到 Boltzmann 分布，此时状态向量出现的概率将仅由其能量与所有可能状态向量的能量确定：

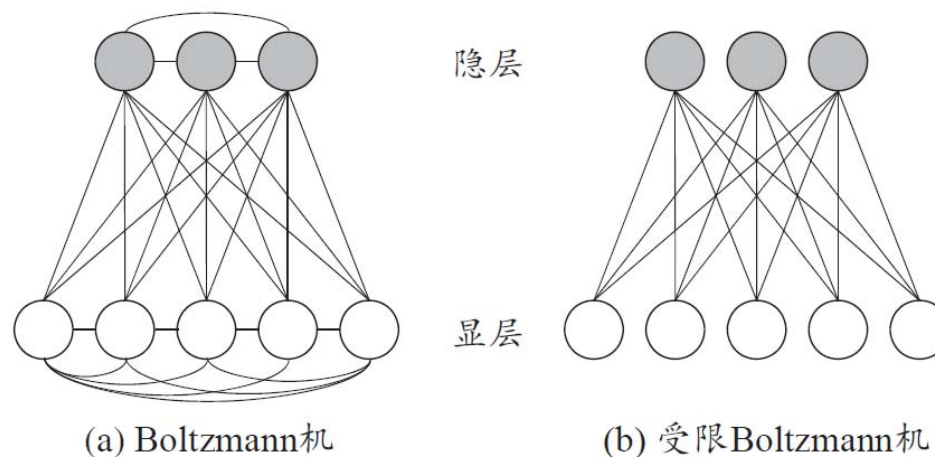
$$P(\mathbf{s}) = \frac{e^{-E(\mathbf{s})}}{\sum_t e^{-E(t)}}$$

## 7.5 其他常见神经网络

### Boltzmann 机 [Ackley et al. , 1985]

#### □ Boltzmann 机训练

- 将每个训练样本视为一个状态向量，使其出现的概率尽可能大
- 标准的 Boltzmann 机是一个全连接图，训练网络的复杂度很高，这使其难以用于解决现实任务
- 现实中常用受限 Boltzmann 机，简称RBM。RBM仅保留显层与隐层之间的连接，从而将 Boltzmann机结构有完全图简化为二部图



Boltzmann 机与受限 Boltzmann 机

## 7.5 其他常见神经网络

### 受限 Boltzmann 机 [Ackley et al. , 1985]

□ 受限 Boltzmann 机常用“对比散度”（简称:CD）算法 [Hinton, 2010] 来进行训练

- 假定网络中有  $d$  个显层神经元  $q$  个隐层神经元，令  $\mathbf{v}$  和  $\mathbf{h}$  分别是显层与隐层的状态向量，由于同一层内不存在连接，有

$$P(\mathbf{v}|\mathbf{h}) = - \prod_{i=1}^d P(v_i|\mathbf{h}),$$
$$P(\mathbf{h}|\mathbf{v}) = - \prod_{i=1}^q P(h_i|\mathbf{v}),$$

- CD算法对每个训练样本  $\mathbf{v}$  先计算出隐层神经元状态的概率分布，然后根据这个概率分布采样得到  $\mathbf{h}$ ；类似的方法从  $\mathbf{h}$  中产生  $\mathbf{v}'$ ，再从  $\mathbf{v}'$  中产生  $\mathbf{h}'$ ；连接权重的更新公式为：

$$\Delta w = \eta \left( \mathbf{v} \mathbf{h}^\top - \mathbf{v}' \mathbf{h}'^\top \right)$$

# 第七章 神经网络

---

## 主要内容

- 7.1 神经元模型
- 7.2 感知机与多层网络
- 7.3 误差逆传播算法
- 7.4 全局最小与局部最小
- 7.5 其他常见神经网络
- 7.6 深度学习

# 7.6 深度学习

---

## 深度学习模型

- 典型的深度学习模型就是很深层的神经网络

### □ 模型复杂度

- 增加隐层神经元的数目（模型宽度）
- 增加隐层数目（模型深度）
- 从增加模型复杂度的角度看，增加隐层的数目比增加隐层神经元的数目更有效。这是因为增加隐层数不仅增加额拥有激活函数的神经元数目，还增加了激活函数嵌套的层数

### □ 复杂模型难点

多隐层网络难以直接用经典算法（例如标准BP算法）进行训练，因为误差在多隐层内逆传播时，往往会“发散”而不能收敛到稳定状态

## 7.6 深度学习

---

### 复杂模型训练方法

#### □ 预训练+微调

- **预训练**: 无监督逐层训练是多隐层网络训练的有效手段, 每次训练一层隐层结点, 训练时将上一层隐层结点的输出作为输入, 而本层隐结点的输出作为下一层隐结点的输入, 这称为“预训练”
- **微调**: 在预训练全部完成后, 再对整个网络进行微调训练. 微调一般使用BP算法

#### □ 例子: 深度信念网络 [Hinton et al., 2006]

- 结构: 每一层都是一个受限 Boltzmann 机
- 训练方法: 无监督预训练+BP 微调

#### □ 分析

预训练+微调 的做法可以视为将大量参数分组, 对每组先找到局部看起来比较好的设置, 然后再基于这些局部较优的结果联合起来进行全局寻优

## 7.6 深度学习

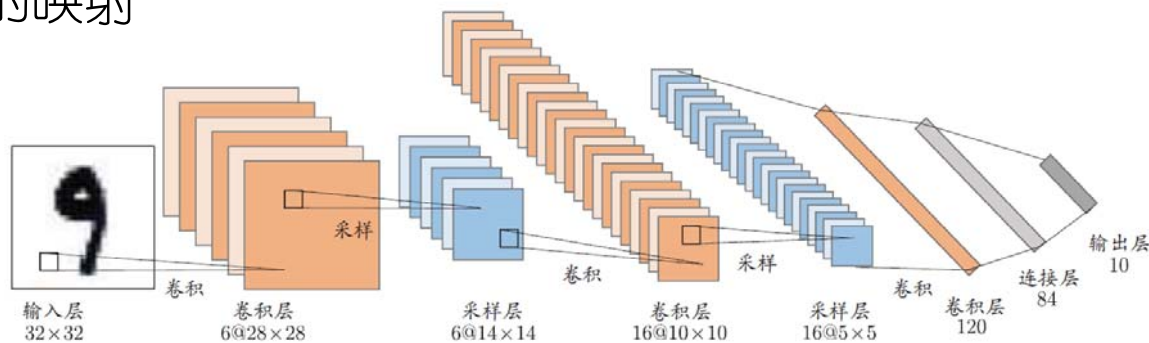
### 复杂模型训练方法

#### □ 权共享

- 一组神经元使用相同的连接权值
- 权共享策略在卷积神经网络(CNN)[LeCun and Bengio, 1995; LeCun et al., 1998]中发挥了重要作用

#### □ 卷积神经网络

结构：CNN复合多个卷积层和采样层对输入信号进行加工，然后在连接层实现与输出目标之间的映射



卷积神经网络用于手写数字识别 [LeCun et al., 1998]

## 7.6 深度学习

### □ 卷积神经网络

- **卷积层**：每个卷积层包含多个特征映射，每个特征映射是一个由多个神经元构成的“平面”，通过一种卷积滤波器提取的一种特征
- **采样层**：亦称“池化层”，其作用是基于局部相关性原理进行亚采样，从而在减少数据量的同时保留有用信息
- **连接层**：每个神经元被全连接到上一层每个神经元，本质就是传统的神经网络，其目的是通过连接层和输出层的连接完成识别任务

### □ 卷积神经网络激活函数

在CNN中通常将 sigmoid 激活函数替换为修正的线性函数 (ReLU)

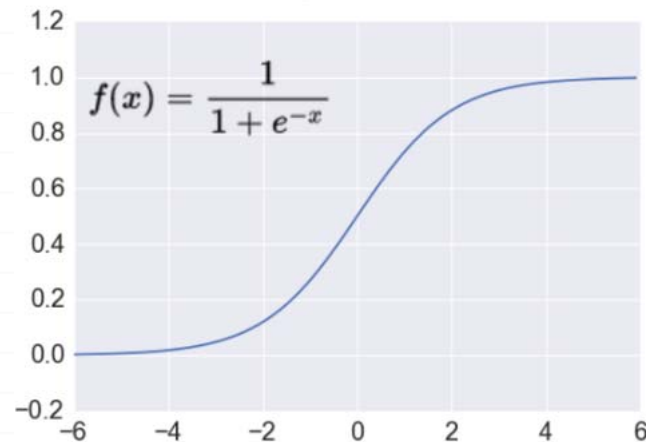
$$f(x) = \max(0, x)$$

### □ 卷积神经网络训练

CNN 可以用BP进行训练，但在训练中，无论是卷积层还是采样层，每一组神经元都是用相同的连接权，从而大幅减少了需要训练的参数数目



# Activation: Sigmoid



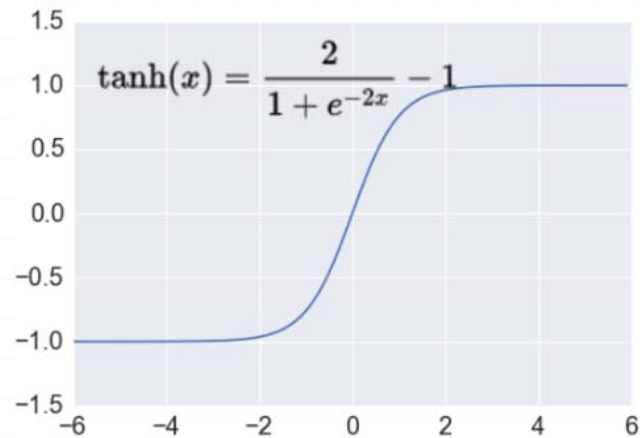
<http://adilmoujahid.com/images/activation.png>

Takes a real-valued number and "squashes" it into range between 0 and 1.

$$\mathbb{R}^n \rightarrow [0,1]$$

- + Nice interpretation as the **firing rate** of a neuron
  - 0 = not firing at all
  - 1 = fully firing
- Sigmoid neurons **saturate** and **kill gradients**, thus NN will barely learn
  - when the neuron's activation are 0 or 1 (saturate)
    - ☐ gradient at these regions almost zero
    - ☐ almost no signal will flow to its weights
    - ☐ if initial weights are too large then most neurons would saturate

# Activation: Tanh



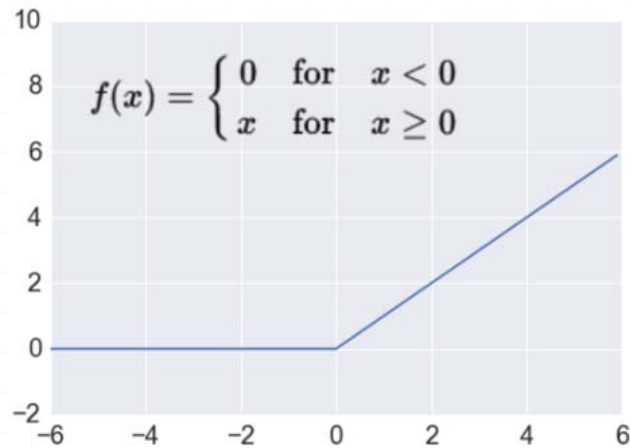
<http://adilmoujahid.com/images/activation.png>

Takes a real-valued number and "squashes" it into range between -1 and 1.

$$\mathbb{R}^n \rightarrow [-1,1]$$

- Like sigmoid, tanh neurons **saturate**
- Unlike sigmoid, output is **zero-centered**
- Tanh is a **scaled sigmoid**:  $\tanh(x) = 2\text{sigm}(2x) - 1$

# Activation: ReLU



<http://adilmoujahid.com/images/activation.png>

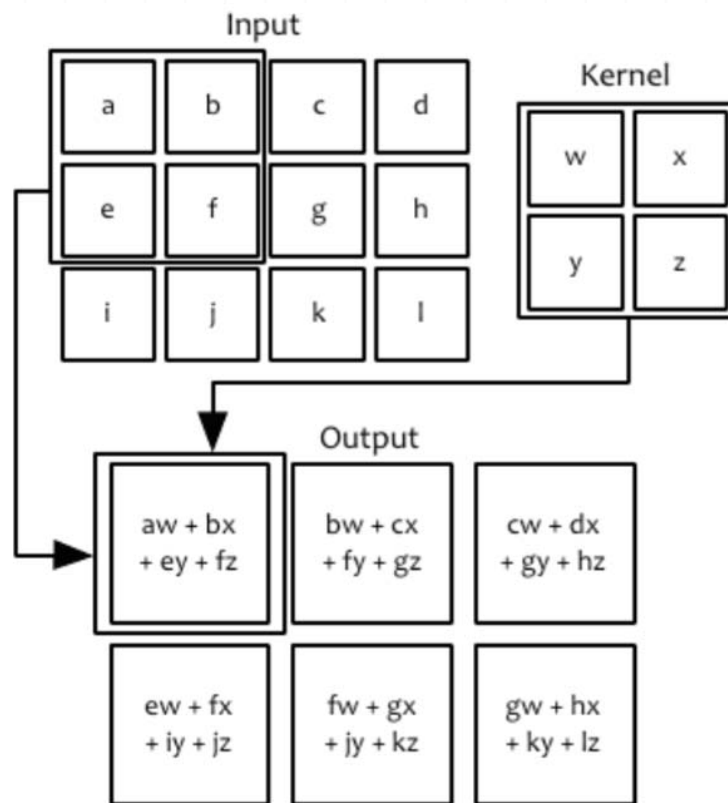
Takes a real-valued number and thresholds it at zero  $f(x) = \max(0, x)$

$$\mathbb{R}^n \rightarrow \mathbb{R}_+^n$$

Most Deep Networks use ReLU nowadays

- ☐ Trains much **faster**
  - accelerates the convergence
  - due to linear, non-saturating form
- ☐ Less expensive operations
  - compared to sigmoid/tanh (exponentials etc.)
  - implemented by simply thresholding a matrix at zero
- ☐ More **expressive**
- ☐ Prevents the **gradient vanishing problem**

# Convolution operation



1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Input matrix

1	0	1
0	1	0
1	0	1

Convolutional  
3x3 filter



1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved  
Feature

# Pooling

**Max pooling:** reports the maximum output within a rectangular neighborhood.

**Average pooling:** reports the average output of a rectangular neighborhood (possibly weighted by the distance from the central pixel).

Feature Map

6	4	8	5
5	4	5	8
3	6	7	7
7	9	7	2

max pool  
2x2 filters  
and stride 2



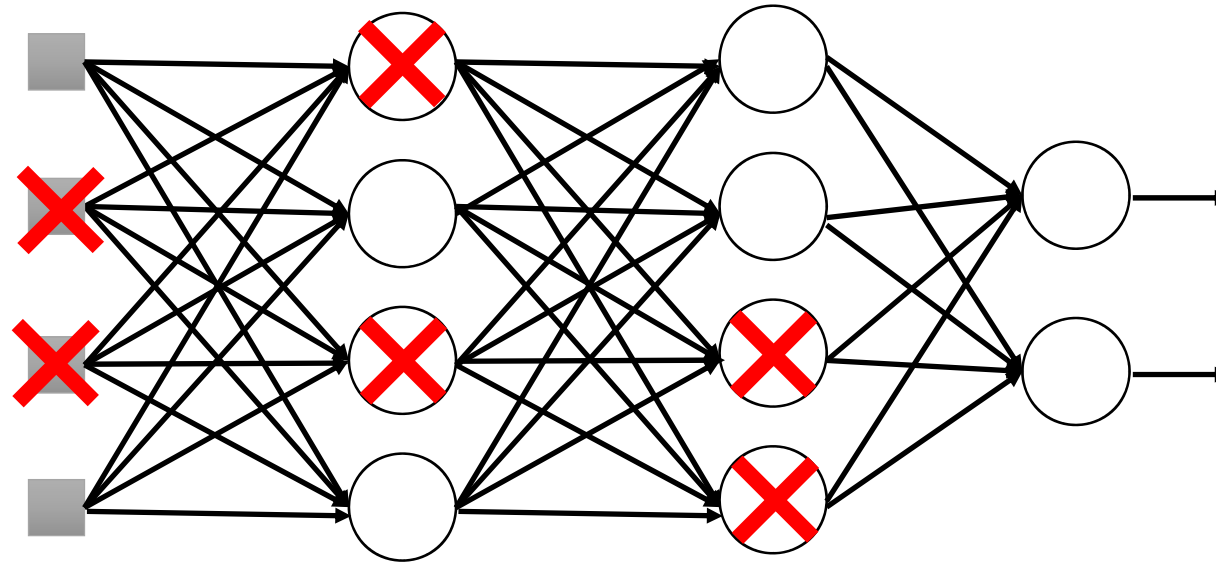
Max-Pooling


# Dropout

Pick a mini-batch

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C(\theta^{t-1})$$

Training:



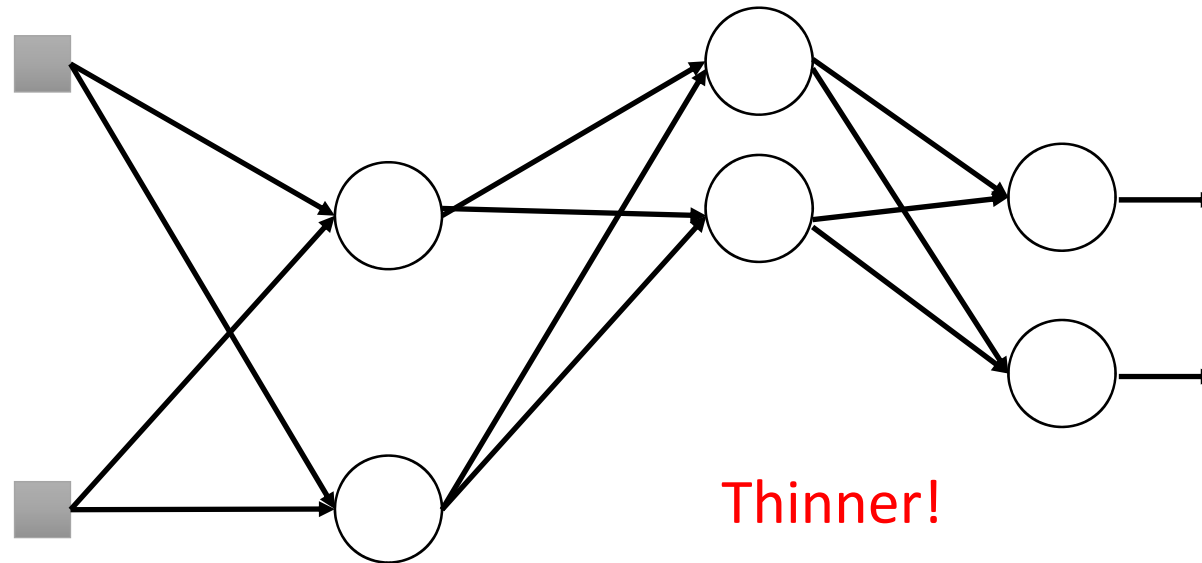
- **Each time before computing the gradients**
  - Each neuron has p% to dropout

# Dropout

Pick a mini-batch

$$\theta^t \leftarrow \theta^{t-1} - \eta \nabla C(\theta^{t-1})$$

Training:

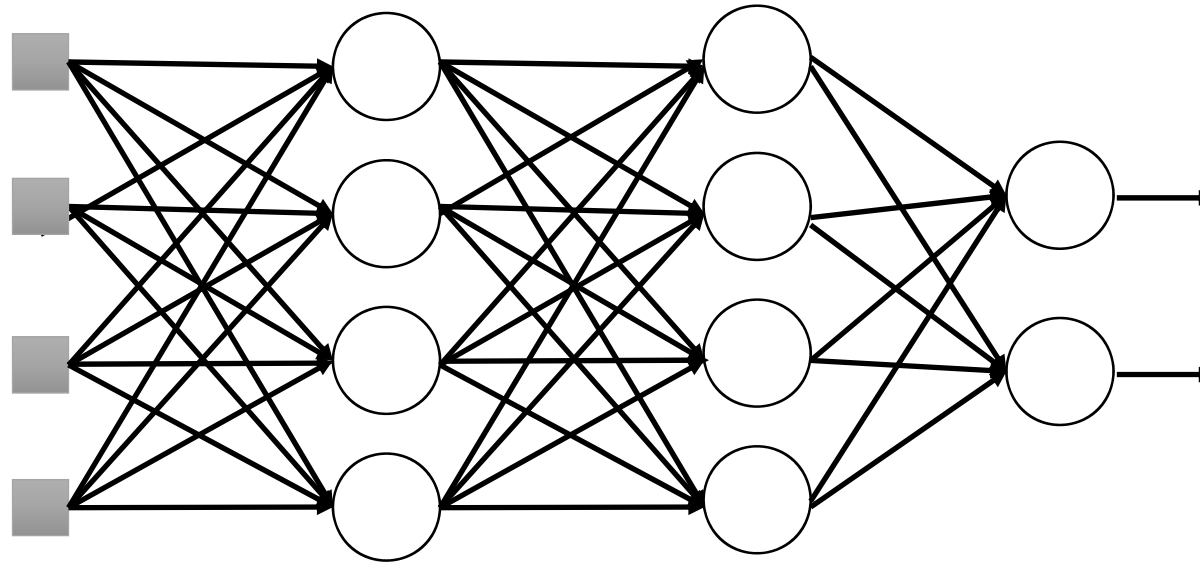


- Each time before computing the gradients
  - Each neuron has  $p\%$  to dropout
    - ➡ **The structure of the network is changed.**
  - Using the new network for training

For each mini-batch, we resample the dropout neurons

# Dropout

Testing:

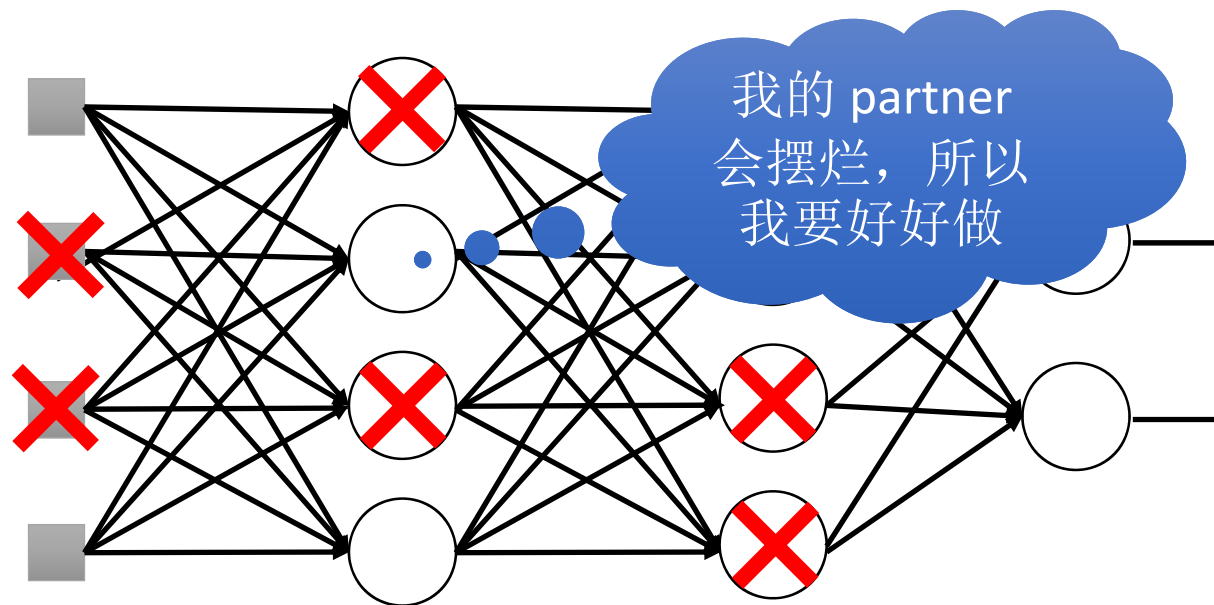


## ➤ No dropout

- If the dropout rate at training is  $p\%$ , all the weights times  $(1-p)\%$
- Assume that the dropout rate is 50%.  
If a weight  $w = 1$  by training, set  $w = 0.5$  for testing.



# Dropout - Intuitive Reason



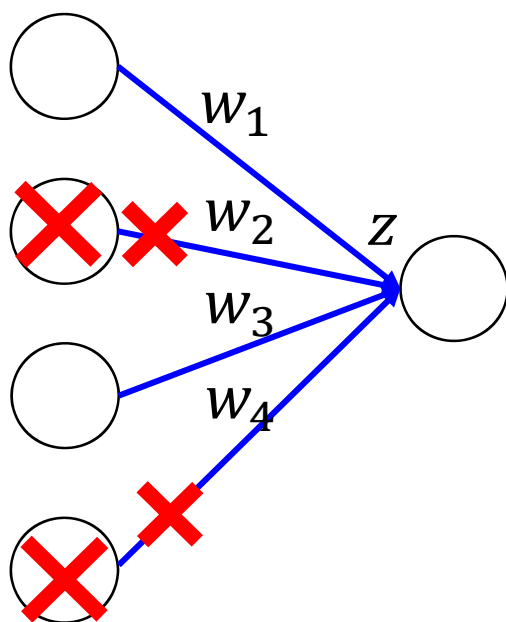
- When teams up, if everyone expect the partner will do the work, nothing will be done finally.
- However, if you know your partner will dropout, you will do better.
- When testing, no one dropout actually, so obtaining good results eventually.

# Dropout - Intuitive Reason

- Why the weights should multiply (1-p)% (dropout rate) when testing?

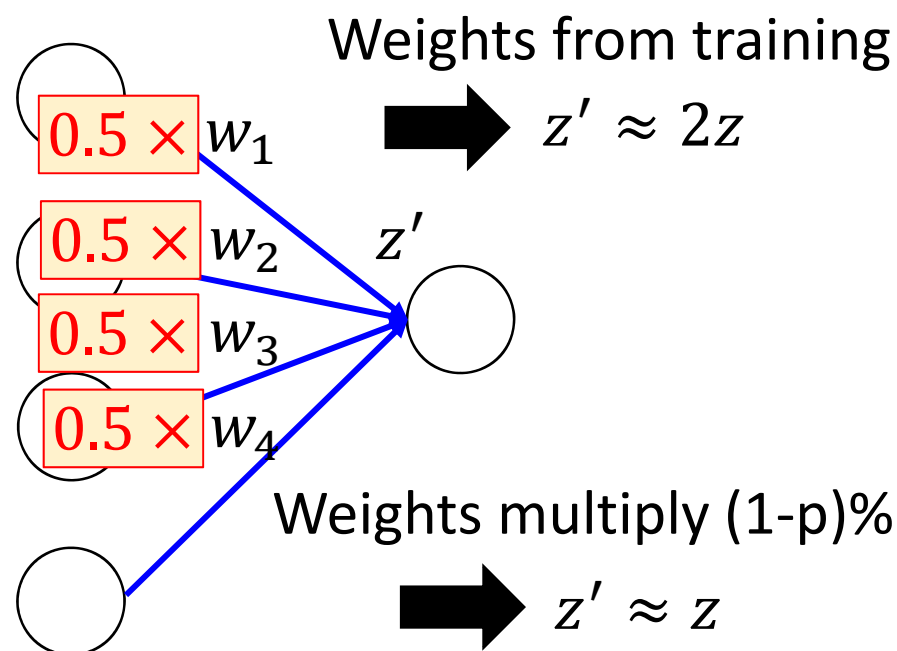
## Training of Dropout

Assume dropout rate is 50%



## Testing of Dropout

No dropout



## 7.6 深度学习

### □ 理解深度学习

- “特征工程” VS “特征学习” 或者 “表示学习”
- 特征工程由人类专家根据现实任务来设计，特征提取与识别是单独的两个阶段；



手工设计  
特征

分类识别

- 特征学习通过深度学习技术自动产生有益于分类的特征，是一个端到端的学习框架。

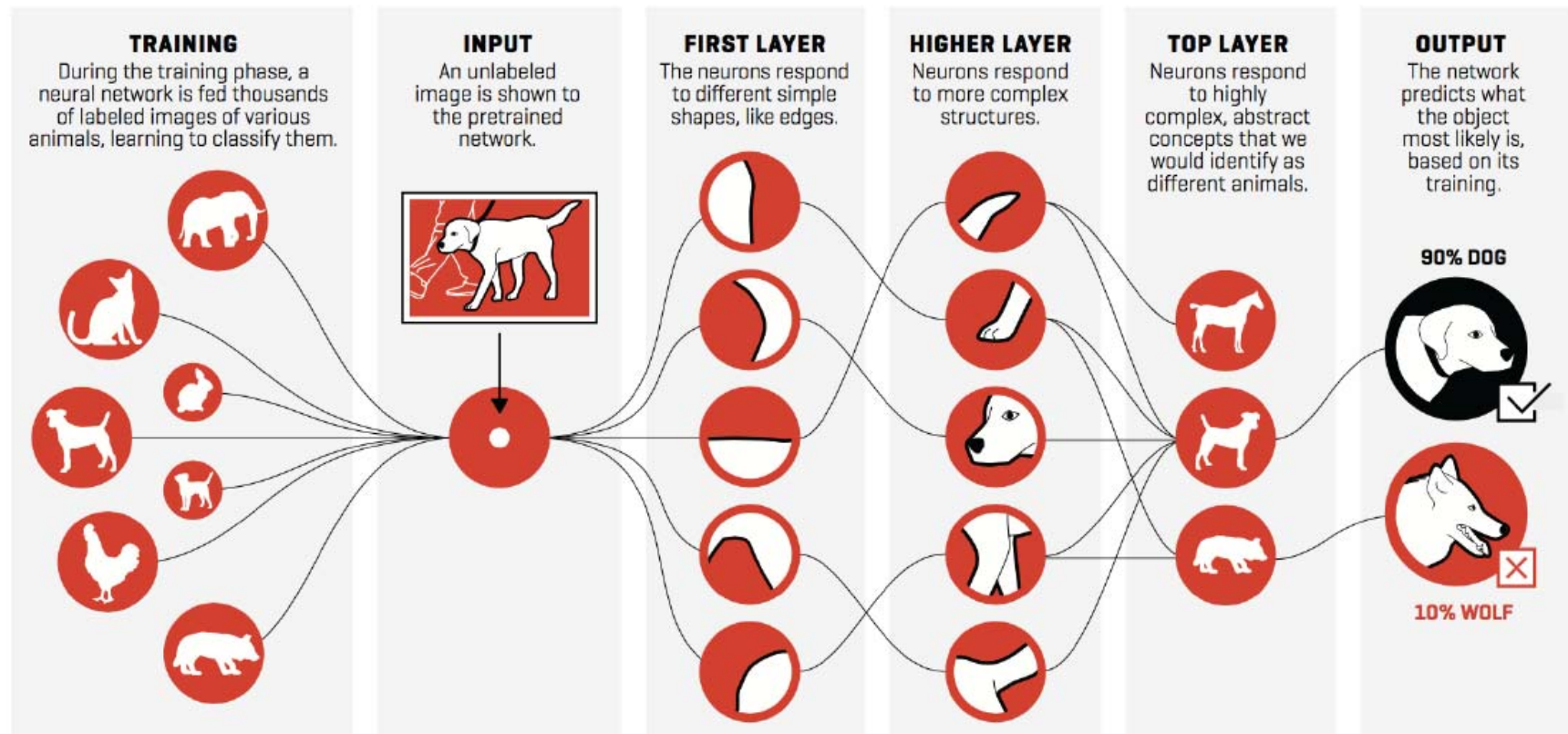


学习生成  
特征

分类识别

## 7.6 深度学习

### 理解深度学习



# Feature detectors

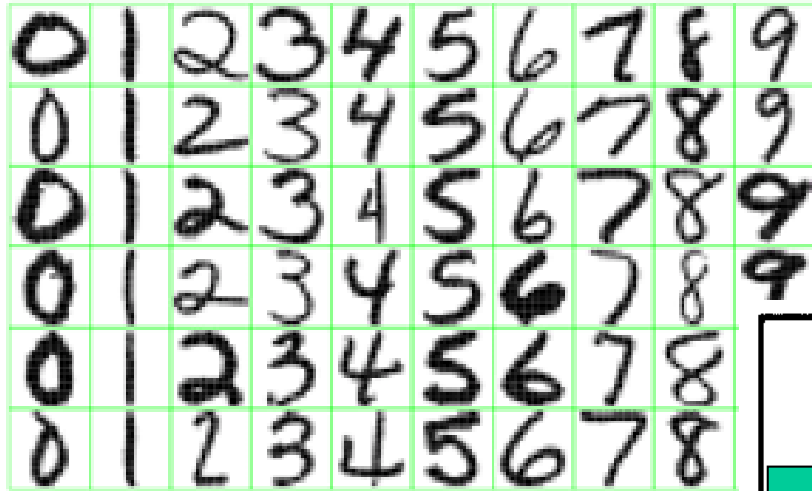
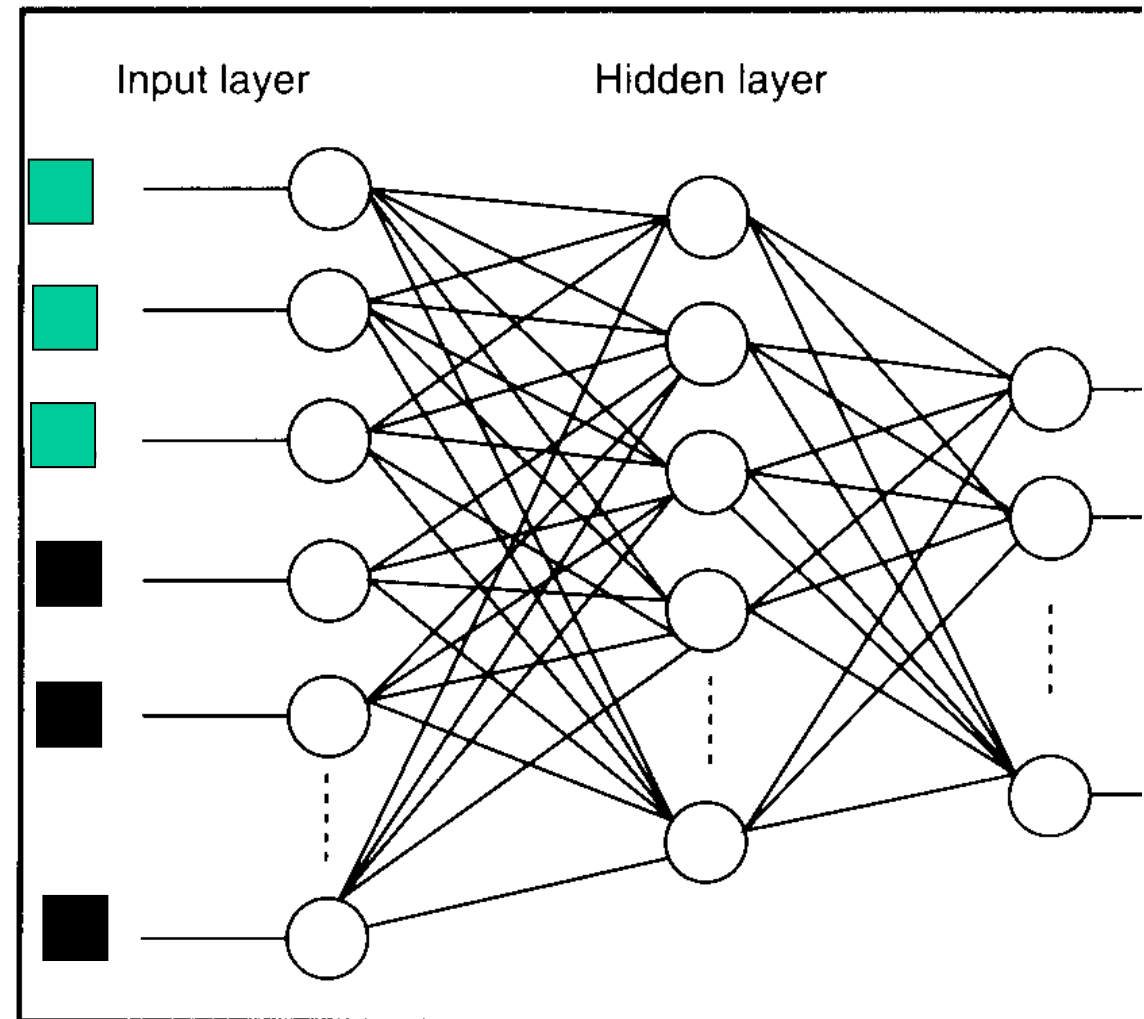
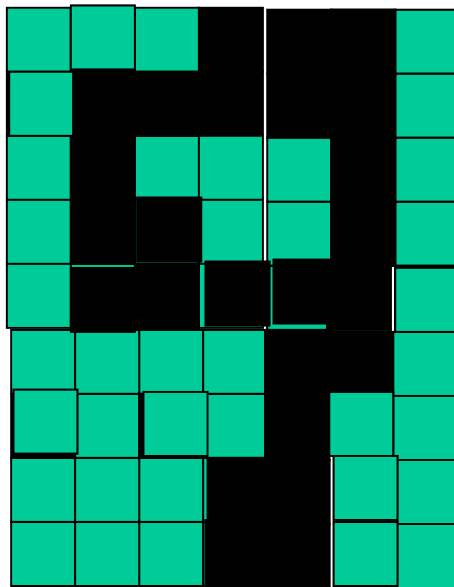
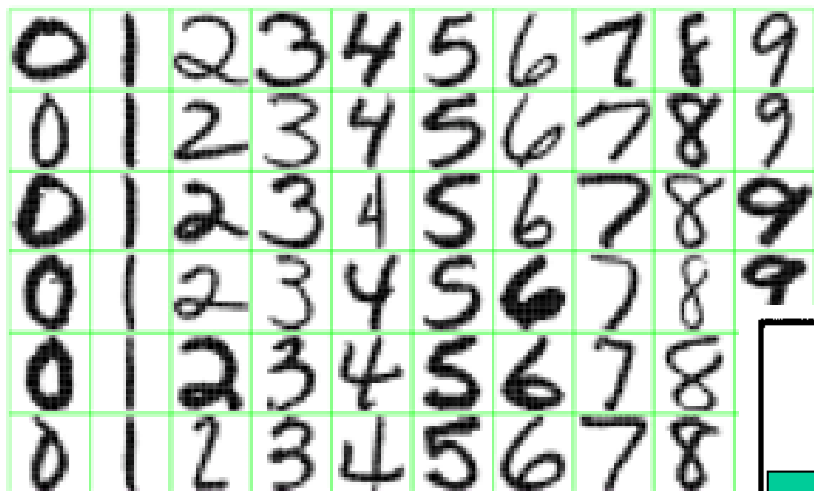


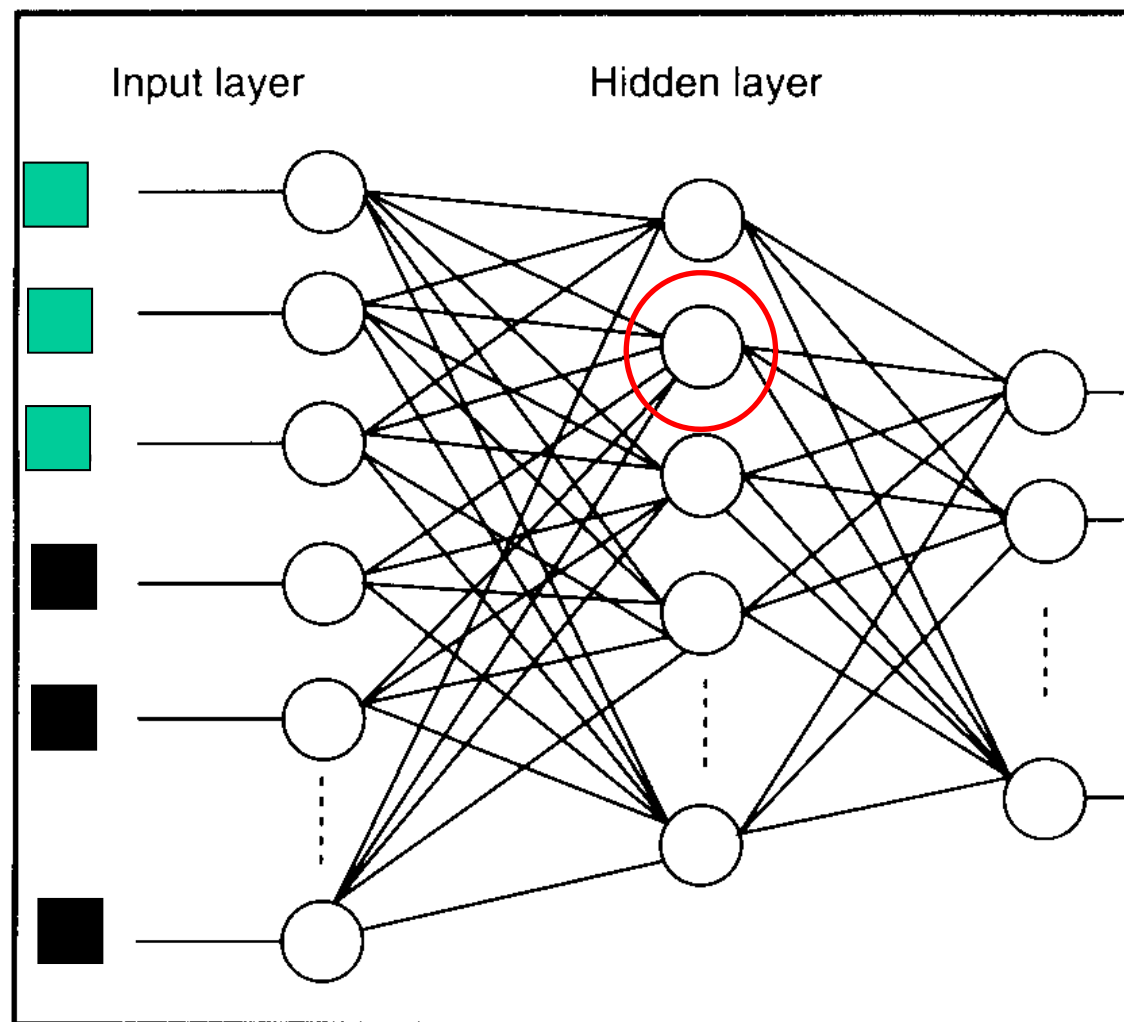
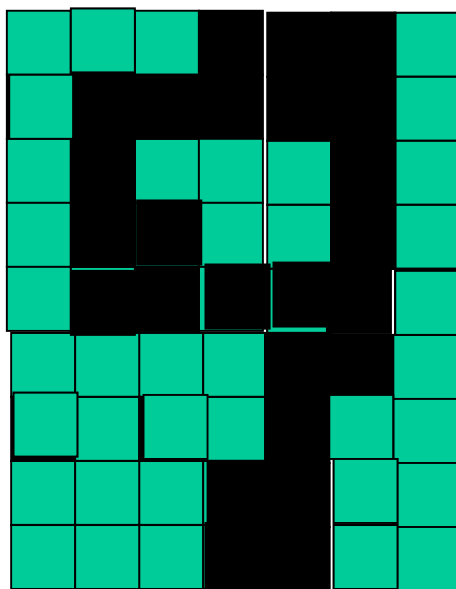
Figure 1.2: *Examples of handwritten digits from postal envelopes.*



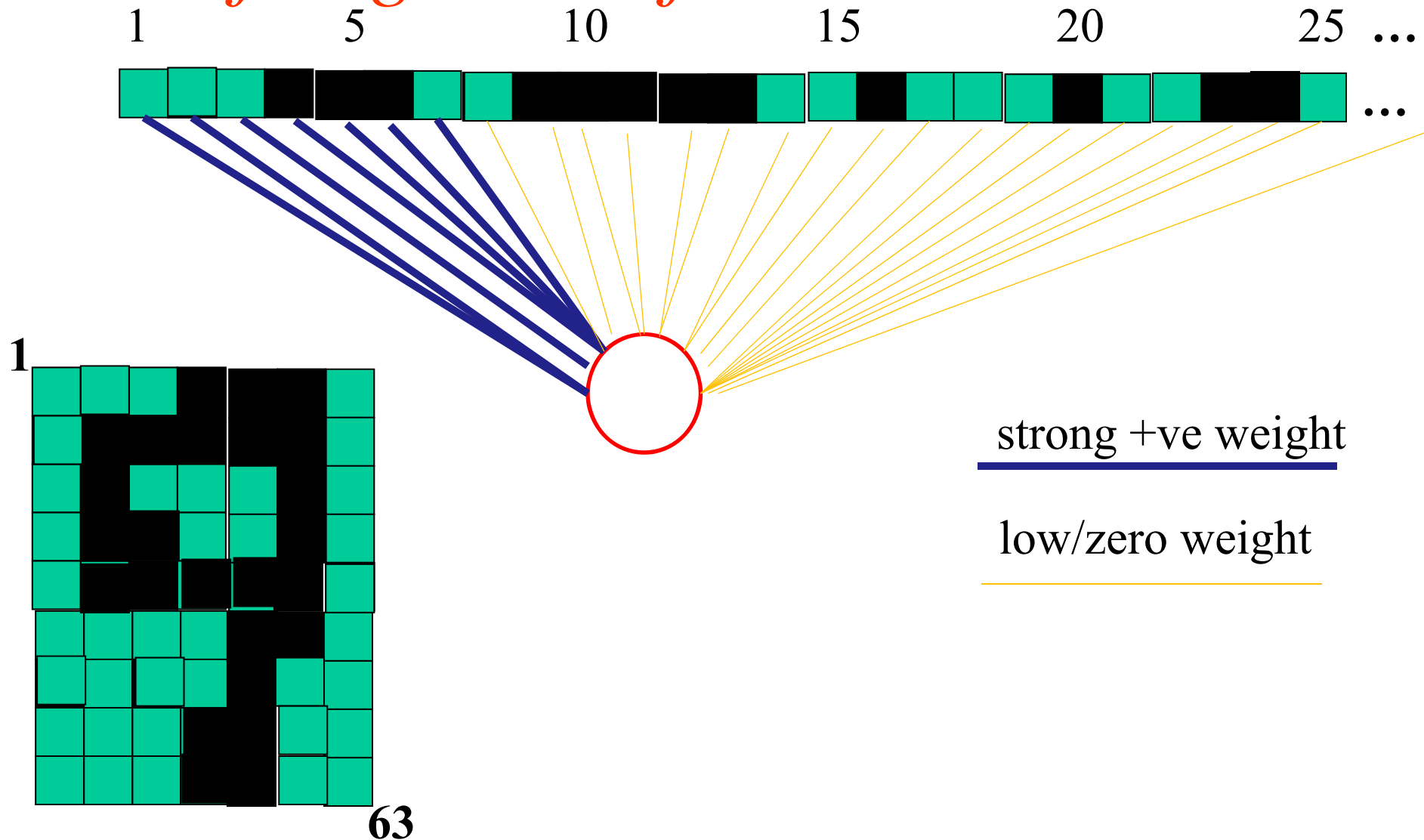


*what is this  
unit doing?*

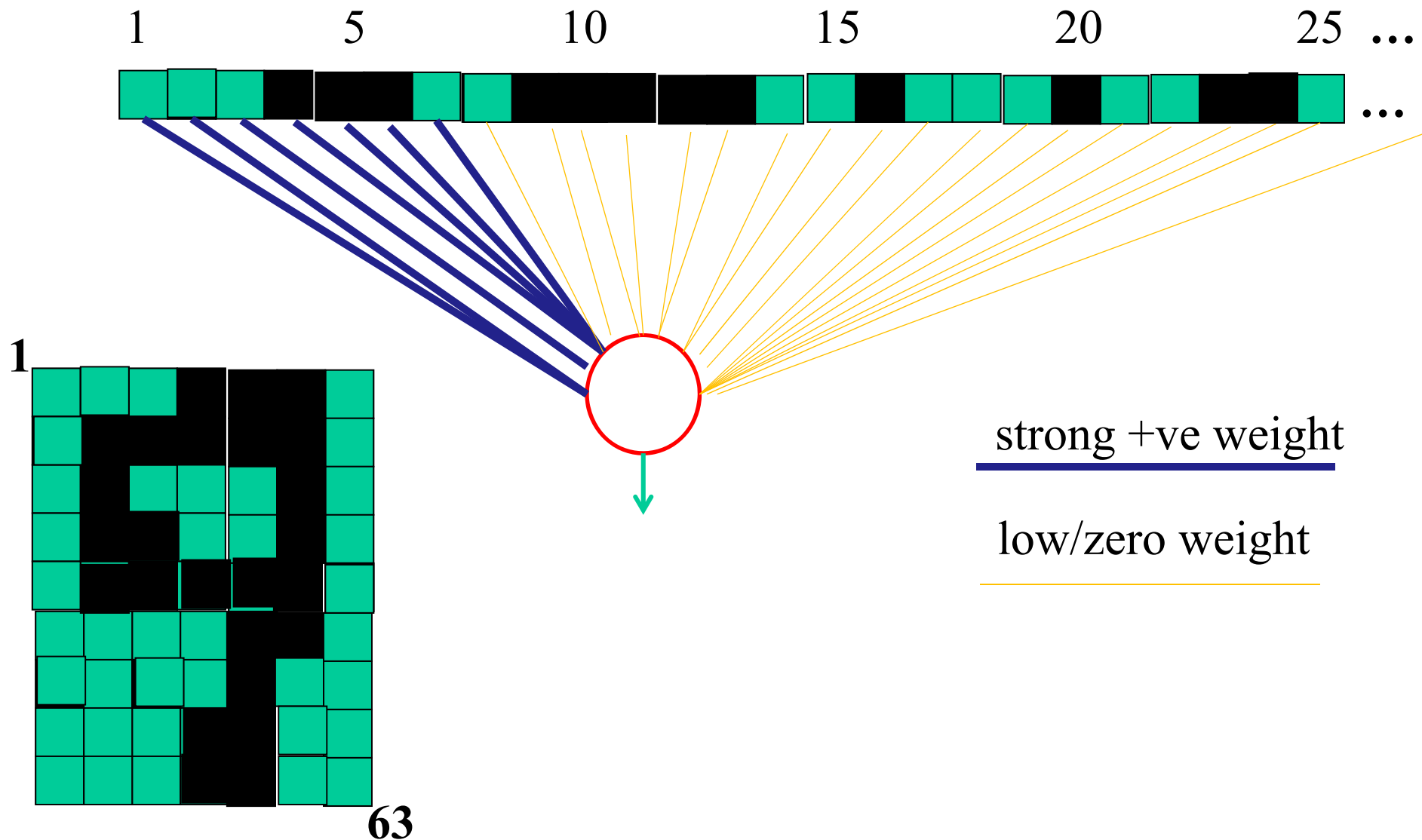
Figure 1.2: Examples of handwritten digits from postal envelopes.



# Hidden layer units become *self-organised feature detectors*

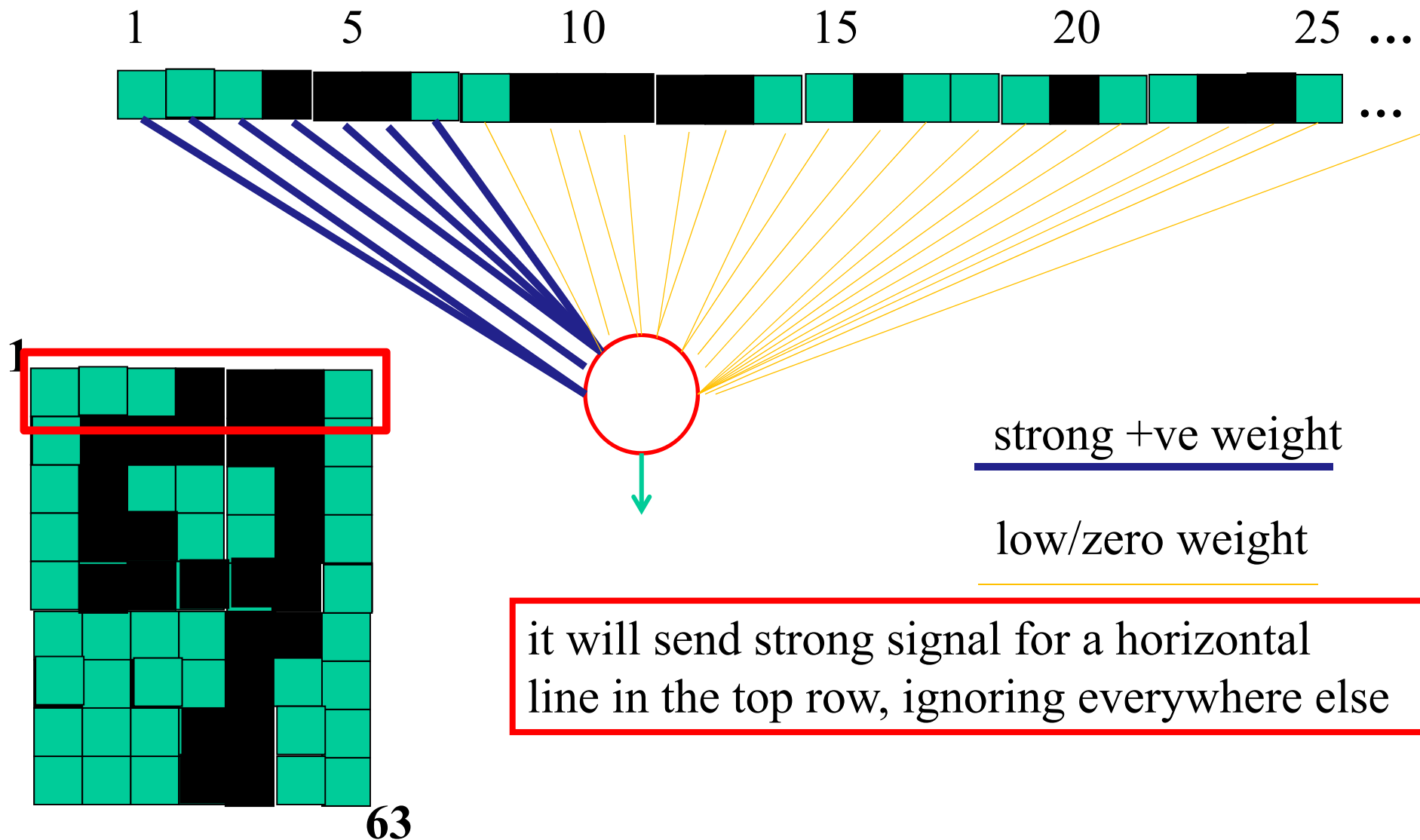


# What does this unit detect?

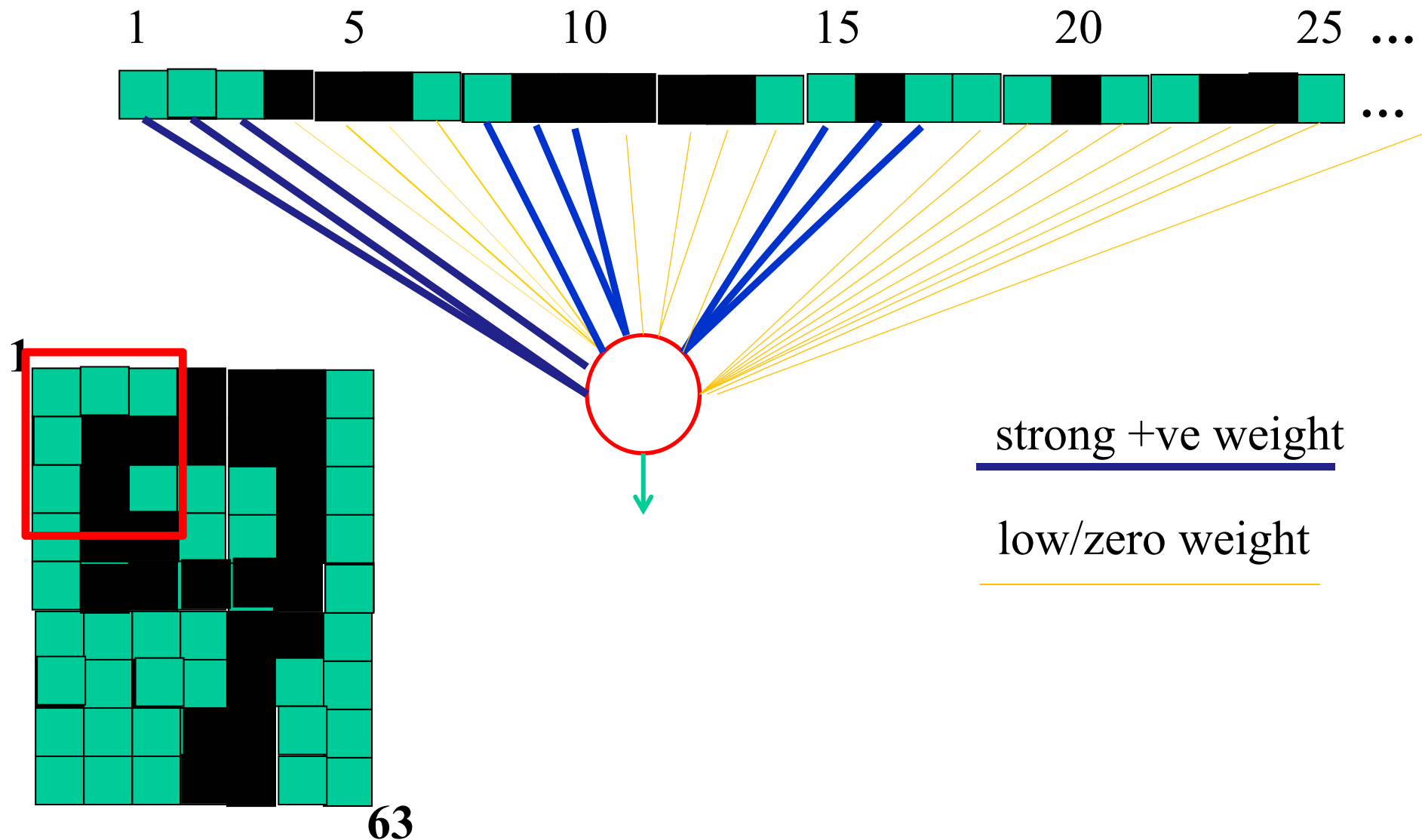




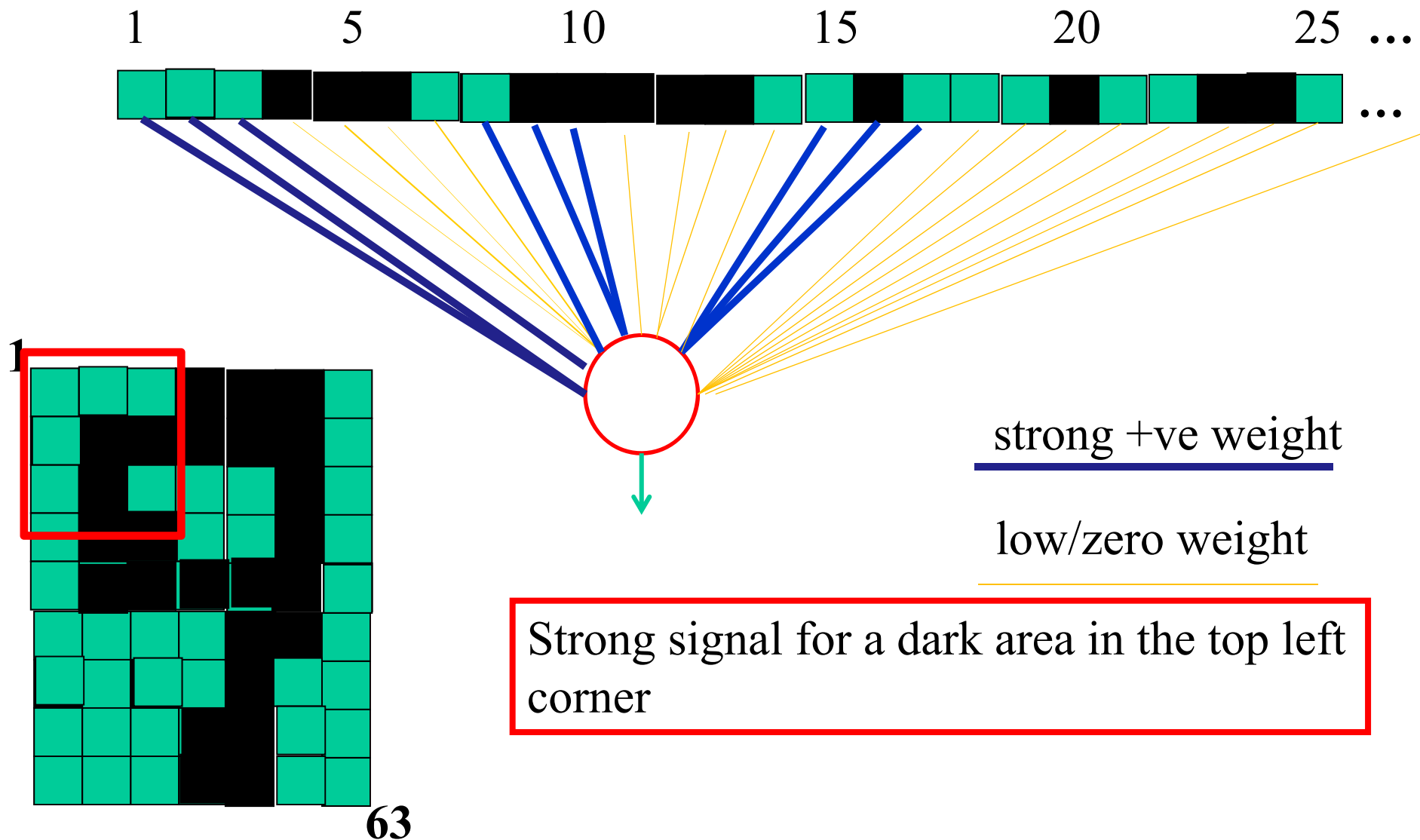
# What does this unit detect?



# What does this unit detect?



# What does this unit detect?



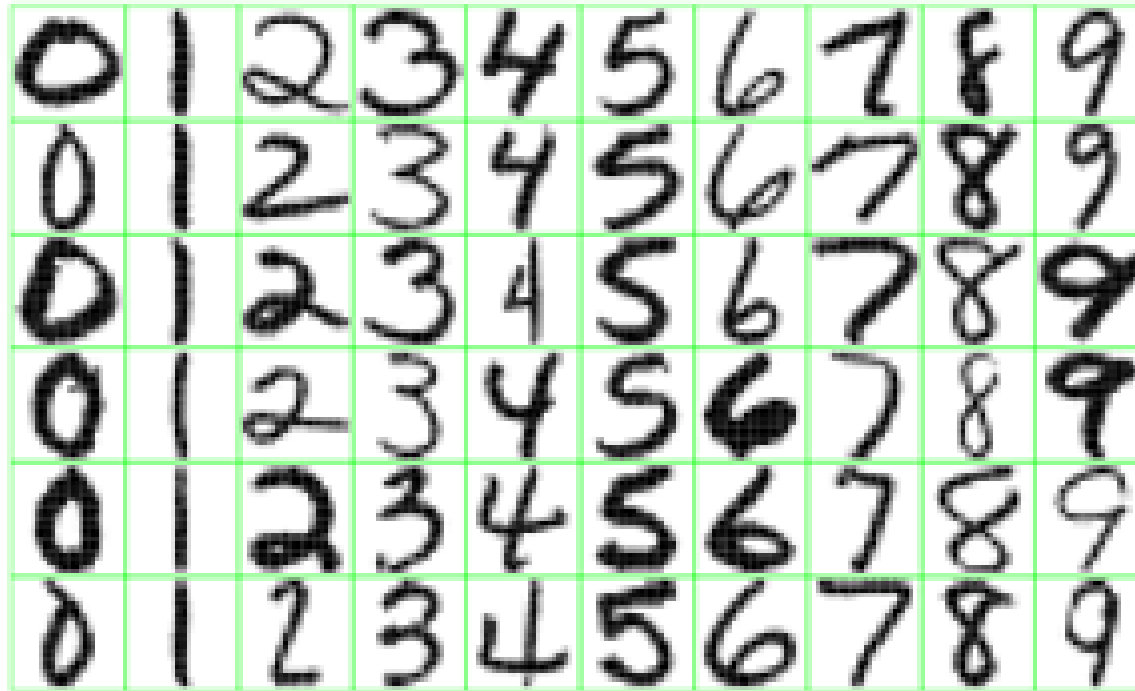


Figure 1.2: *Examples of handwritten digits from U.S. postal envelopes.*

What features might you expect a good NN to learn, when trained with data like this?

1

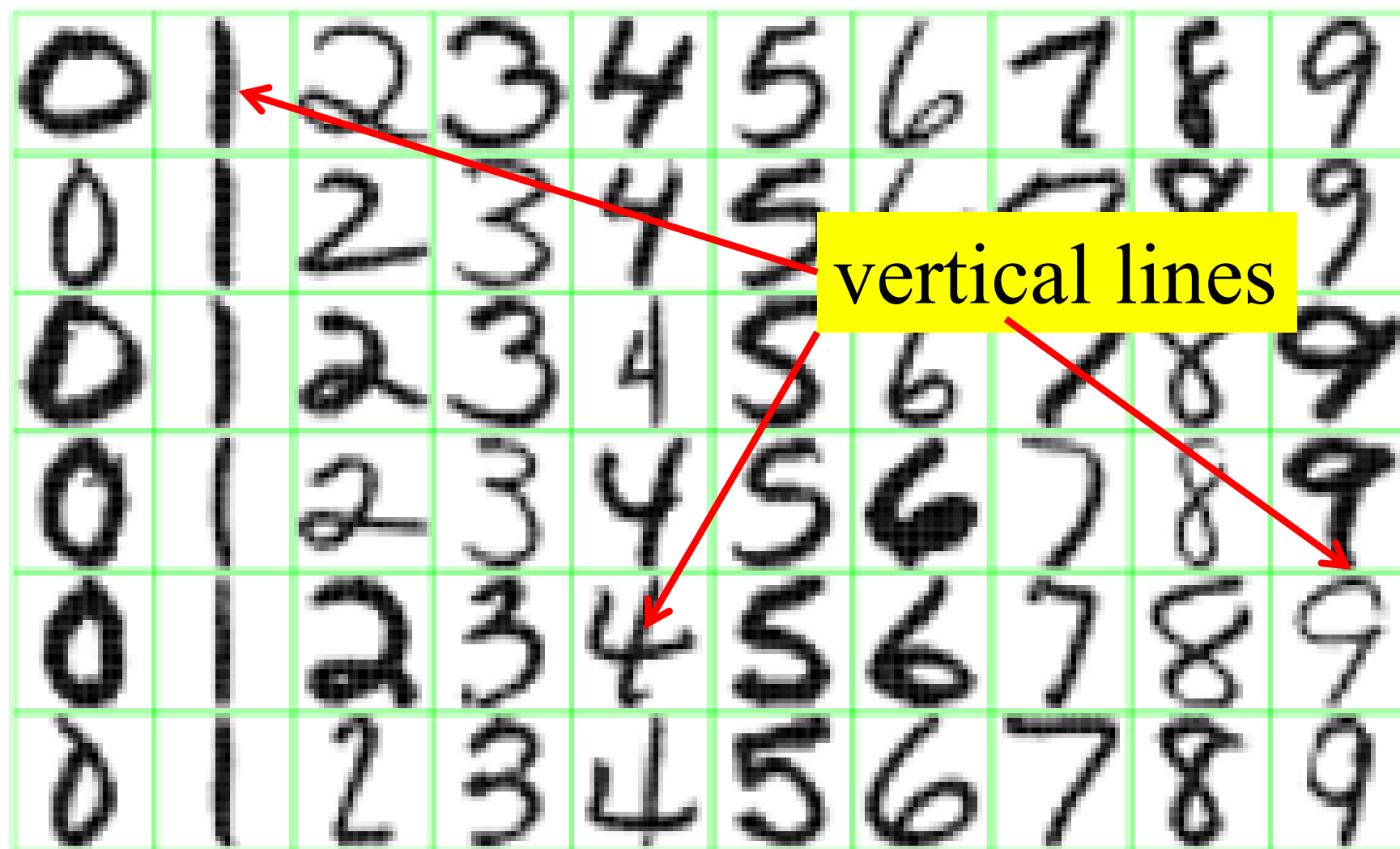


Figure 1.2: *Examples of handwritten digits from U.S. postal envelopes.*



Figure 1.2: *Examples of handwritten digits from U.S. postal envelopes.*

1

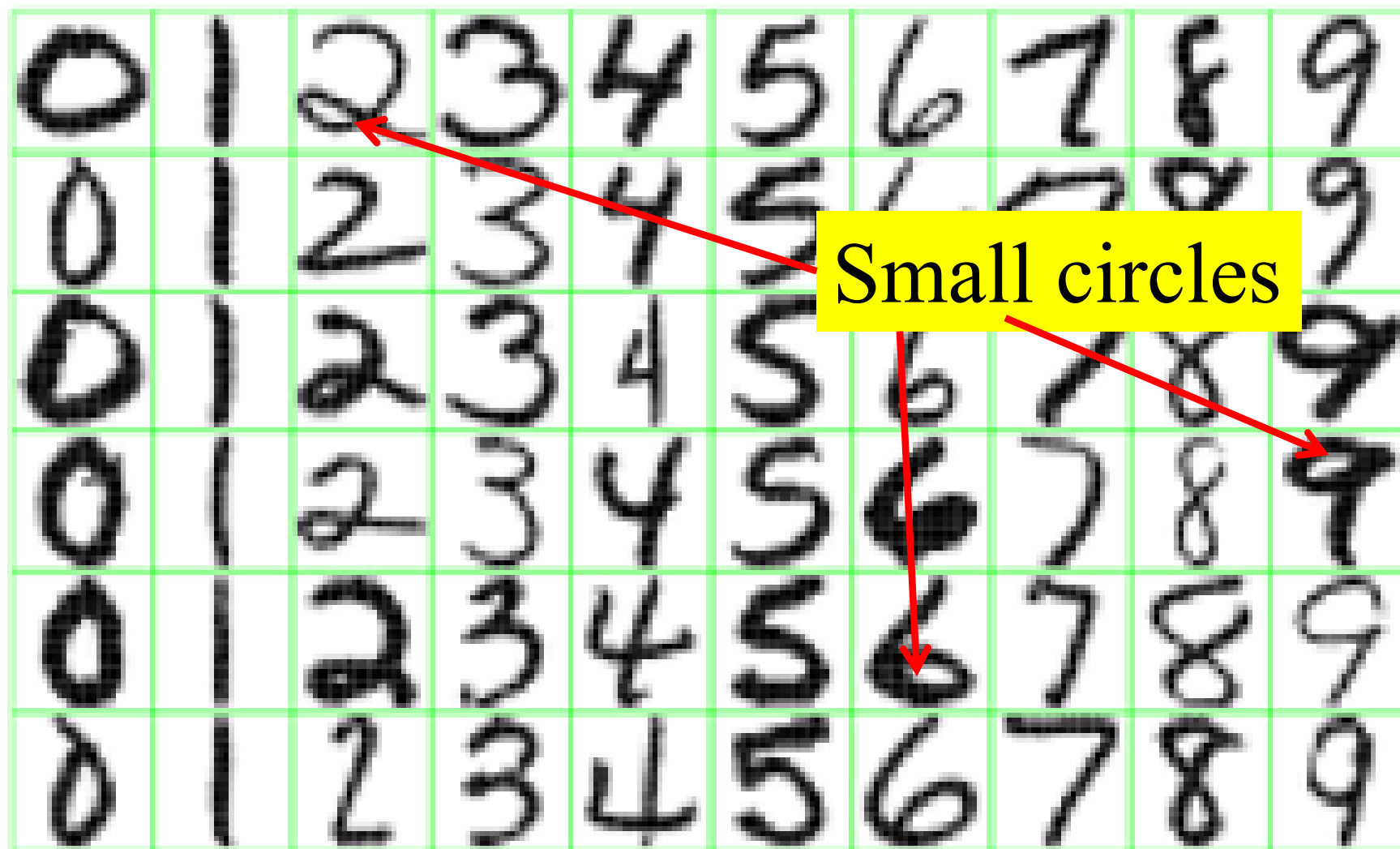


Figure 1.2: *Examples of handwritten digits from U.S. postal envelopes.*

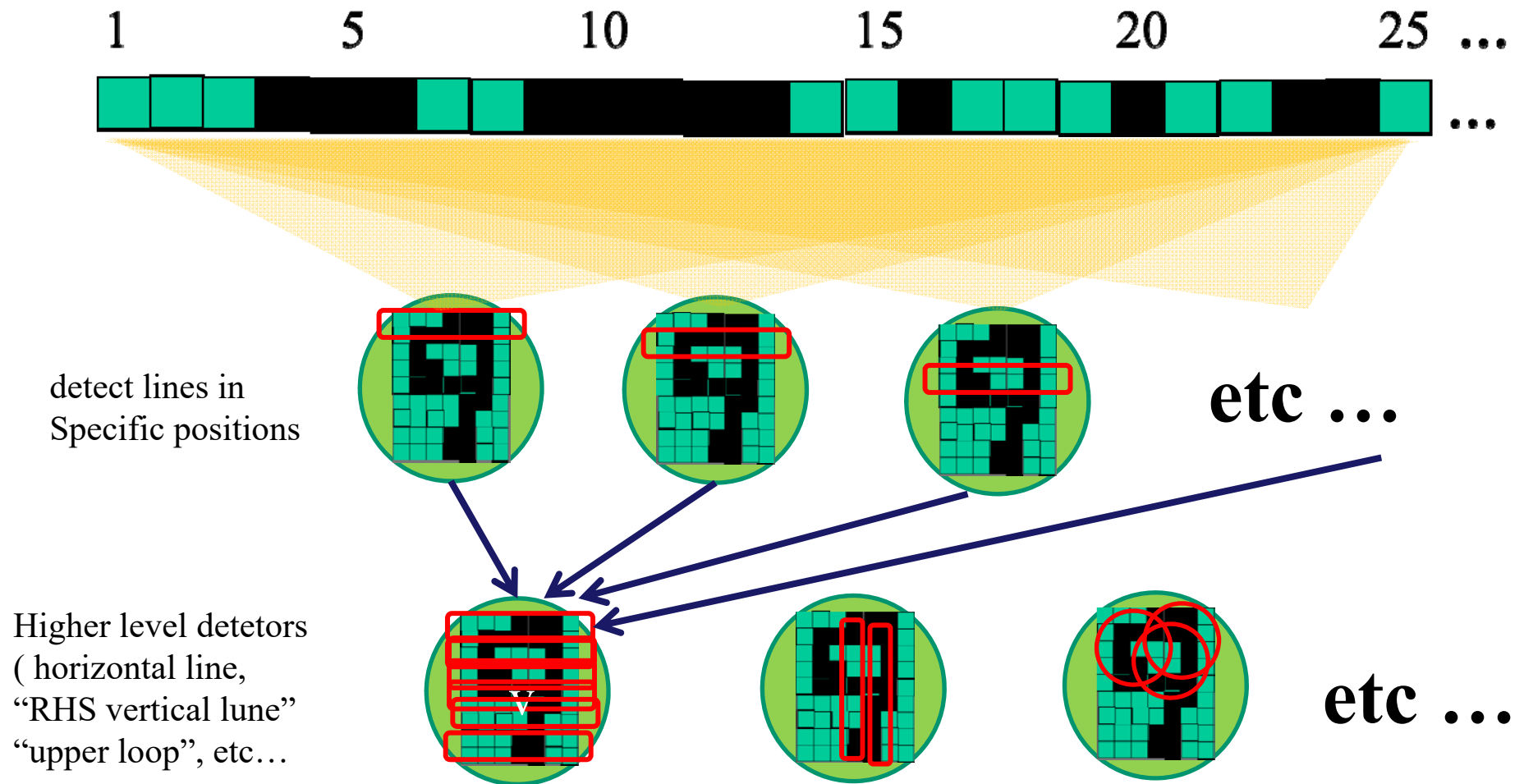
1



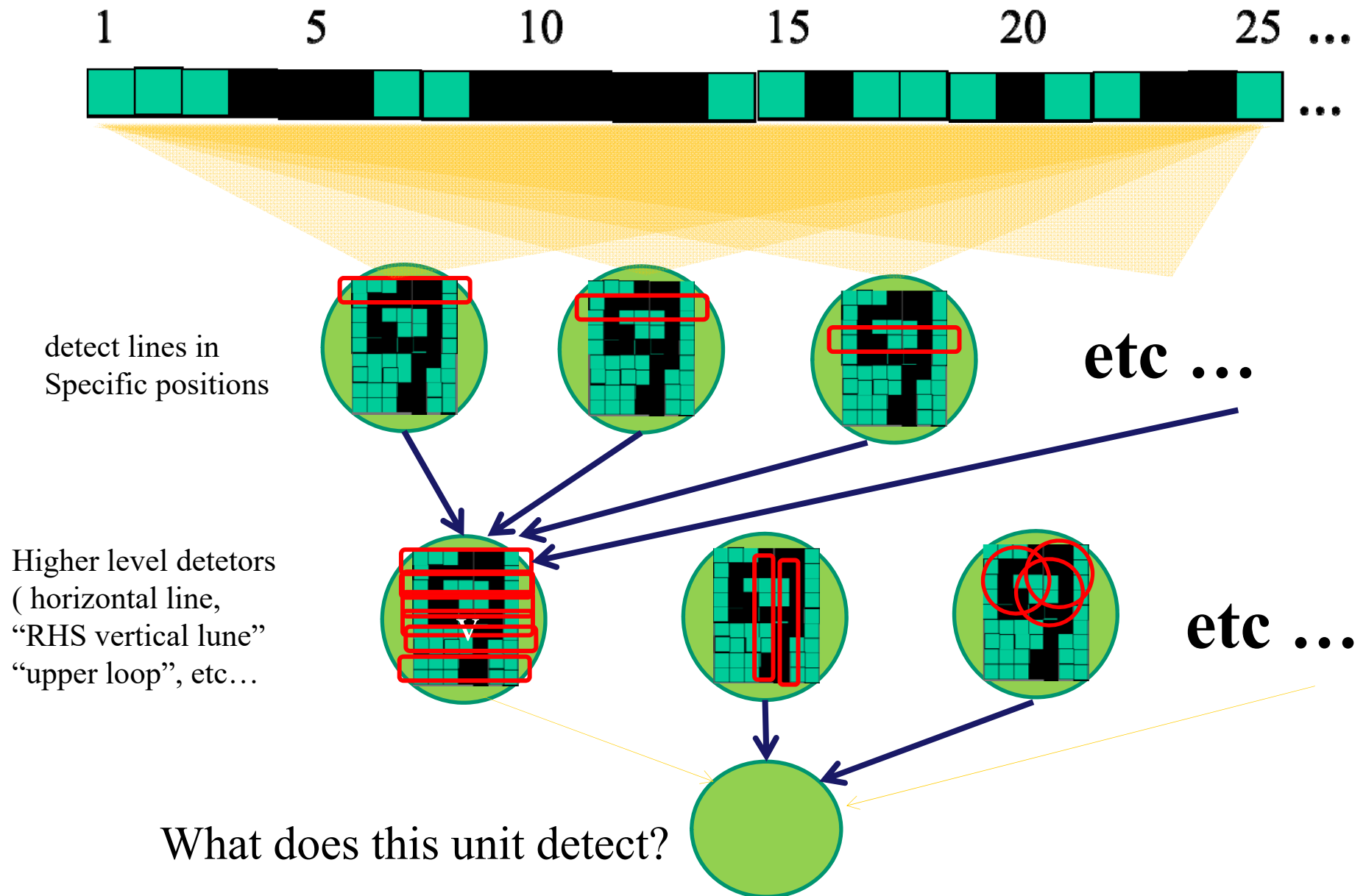
But what about position invariance ???  
our example unit detectors were tied to  
specific parts of the image



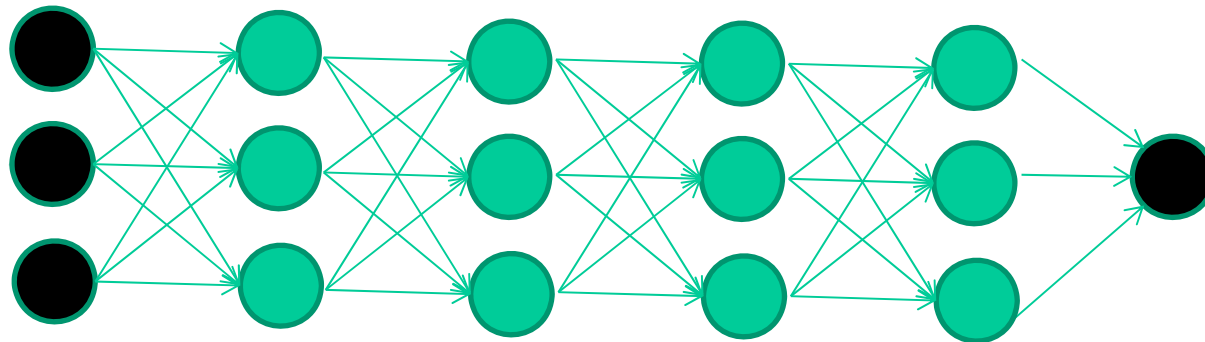
successive layers can learn higher-level features ...



successive layers can learn higher-level features ...

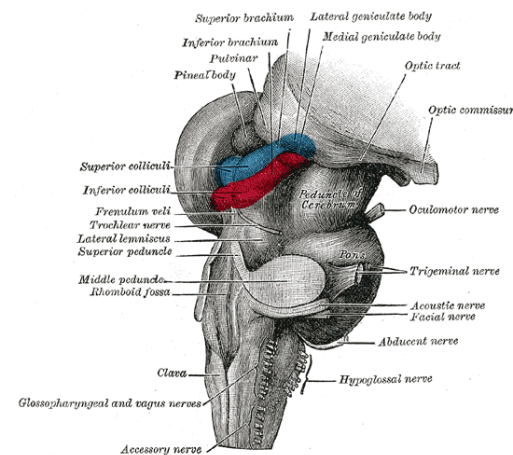
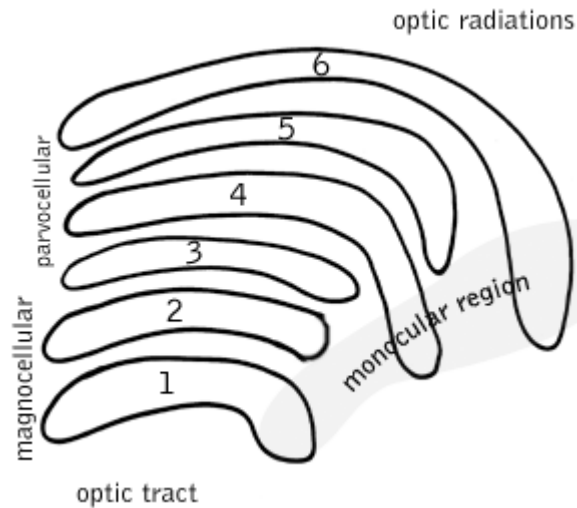


*So: multiple layers make sense*



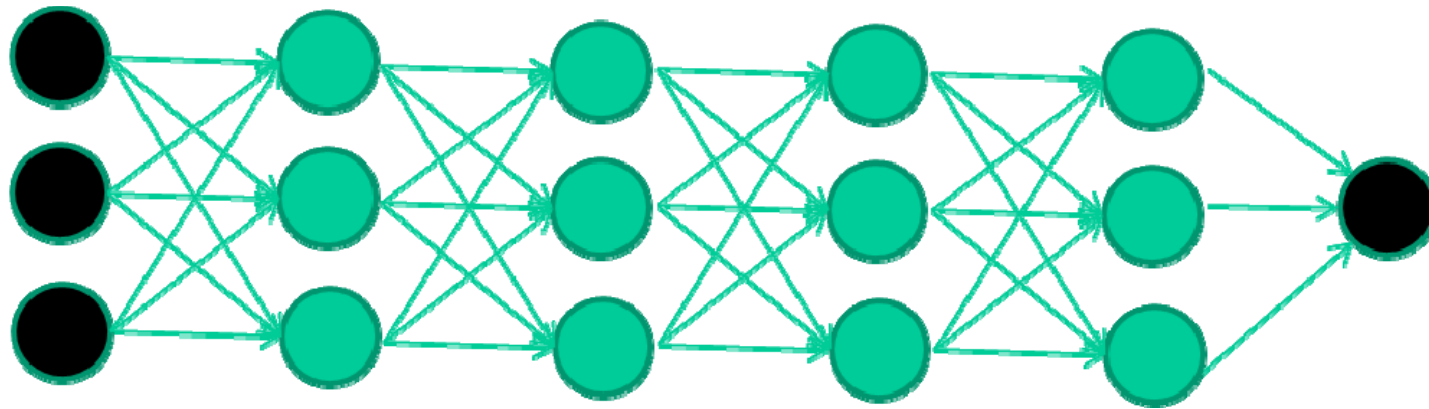
*So: multiple layers make sense*

Your brain works that way

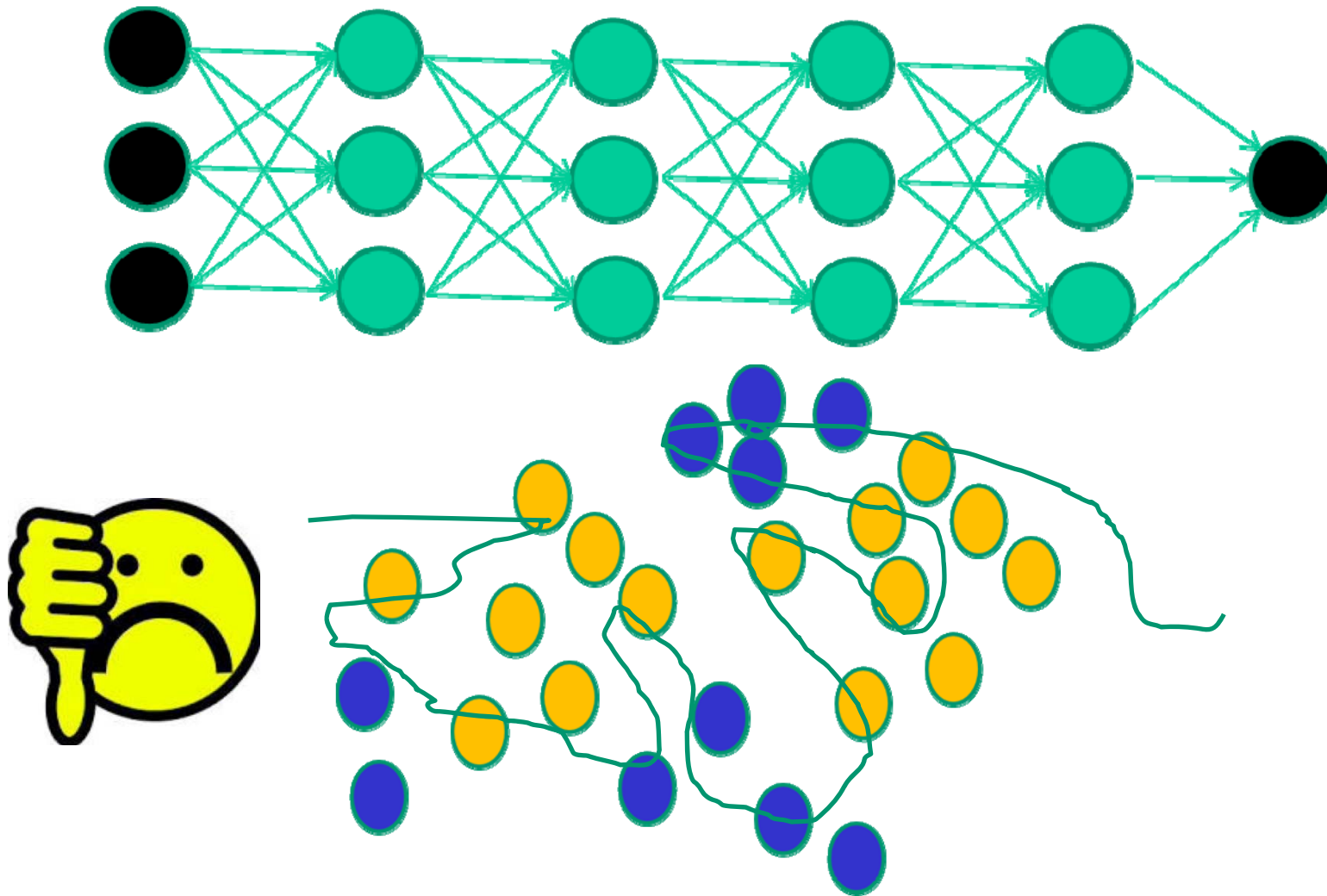


# *So: multiple layers make sense*

**Many-layer neural network architectures should be capable of learning the true underlying features and ‘feature logic’, and therefore generalise very well ...**

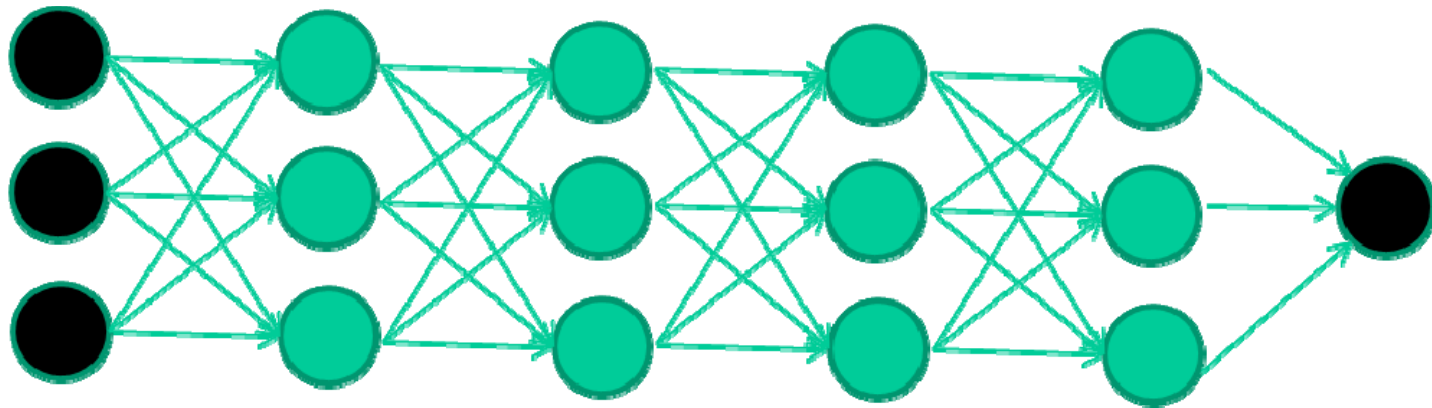


But, until very recently, our weight-learning algorithms simply did not work on multi-layer architectures



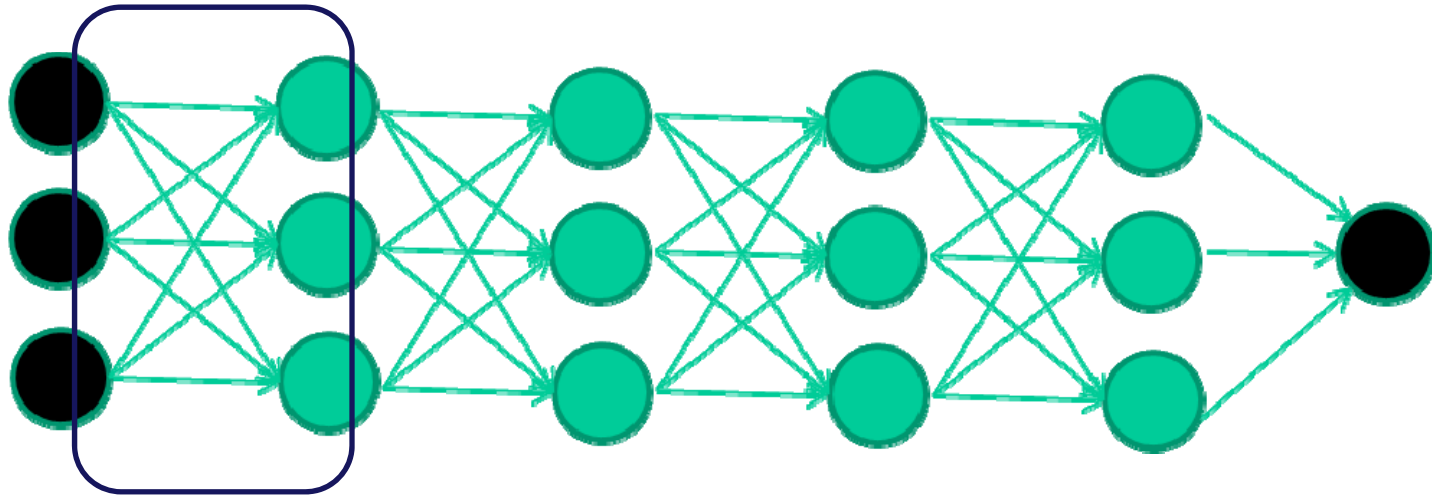
Along came deep learning ...

The new way to train multi-layer NNs...



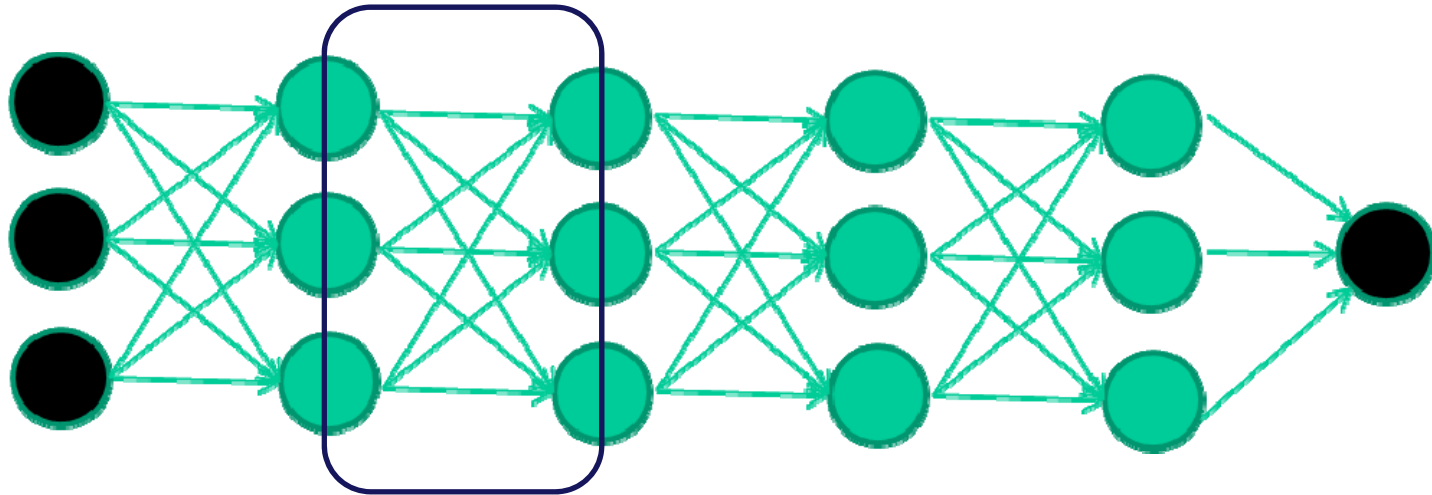


# The new way to train multi-layer NNs...



Train **this** layer first

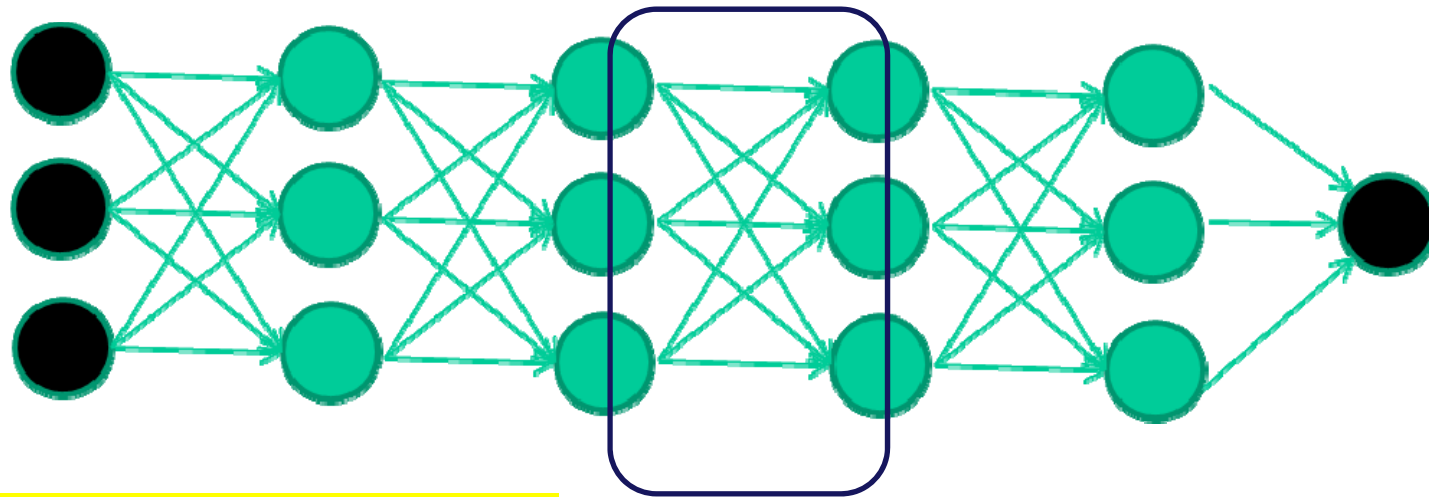
# The new way to train multi-layer NNs...



Train **this** layer first

then **this** layer

# The new way to train multi-layer NNs...

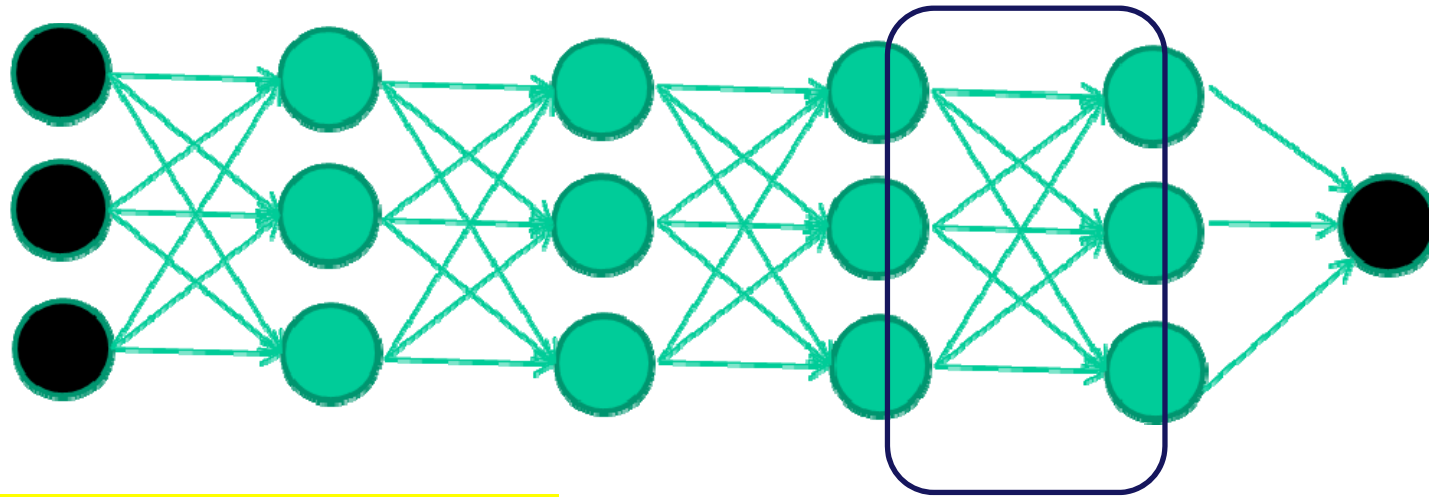


Train **this** layer first

then **this** layer

then **this** layer

# The new way to train multi-layer NNs...



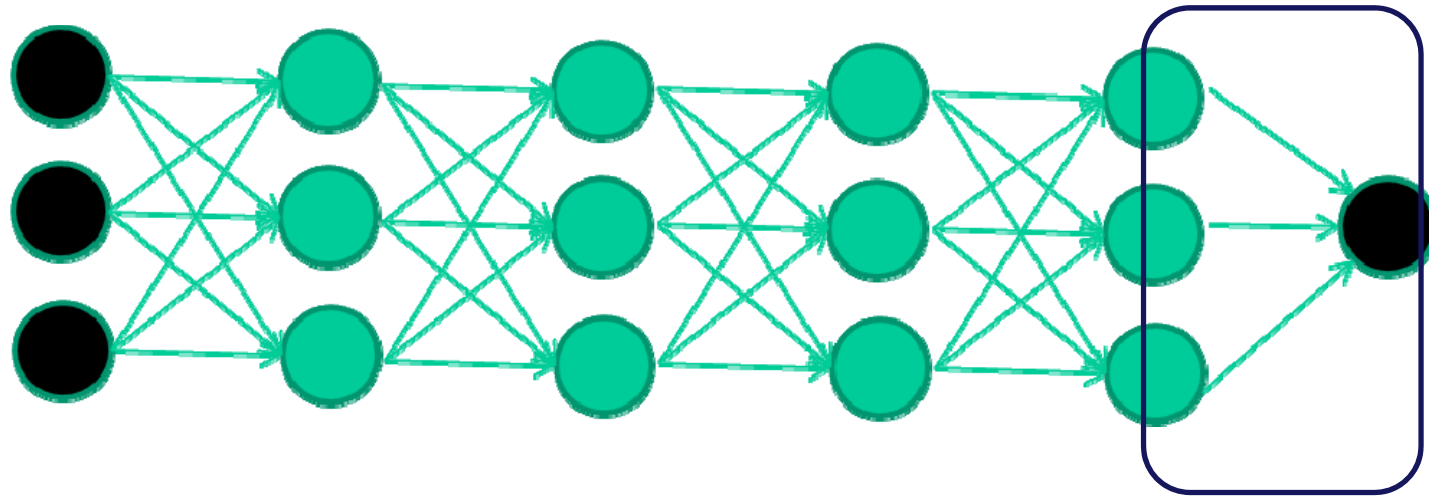
Train **this** layer first

then **this** layer

then **this** layer

then **this** layer

# The new way to train multi-layer NNs...



Train **this** layer first

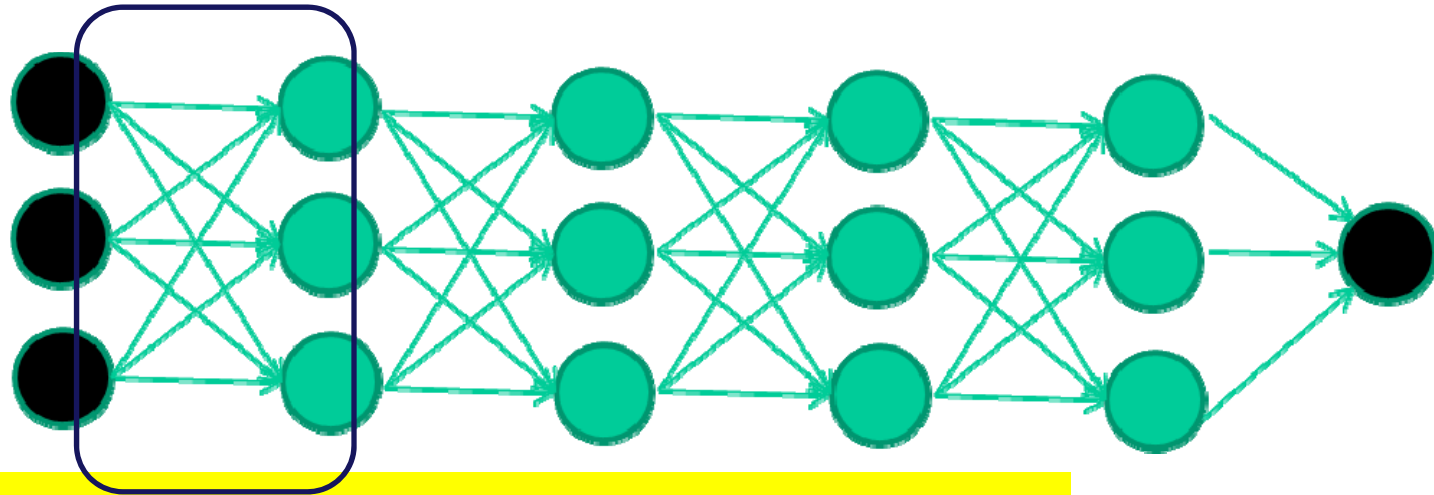
then **this** layer

then **this** layer

then **this** layer

finally **this** layer

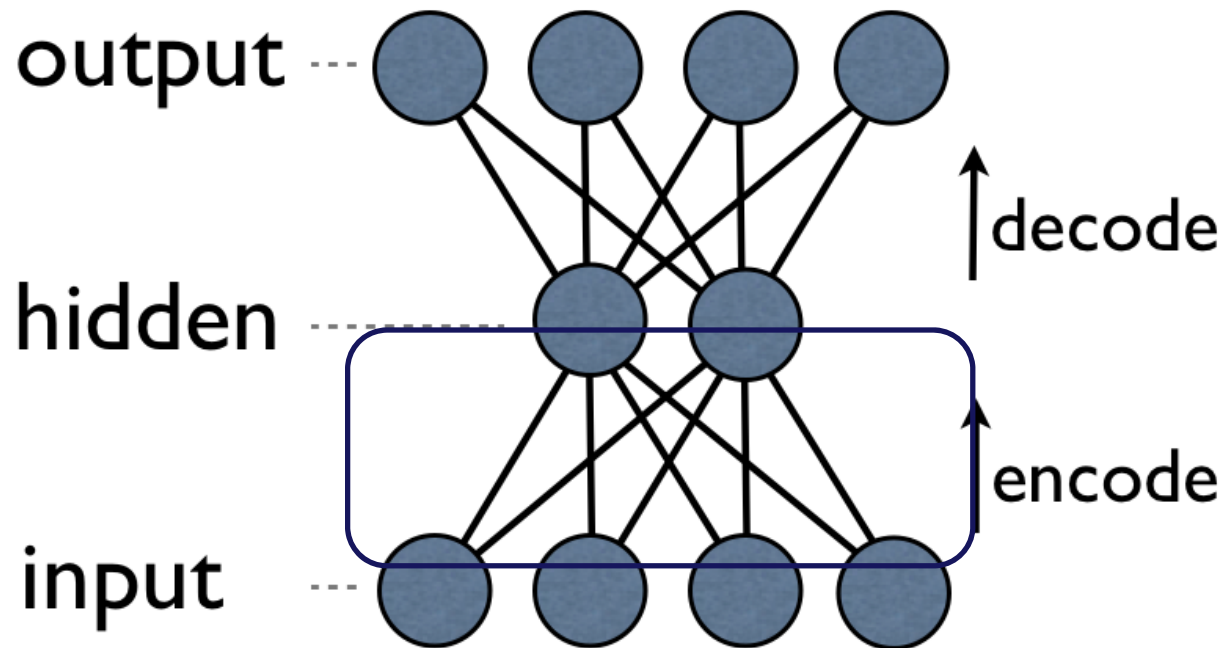
# The new way to train multi-layer NNs...



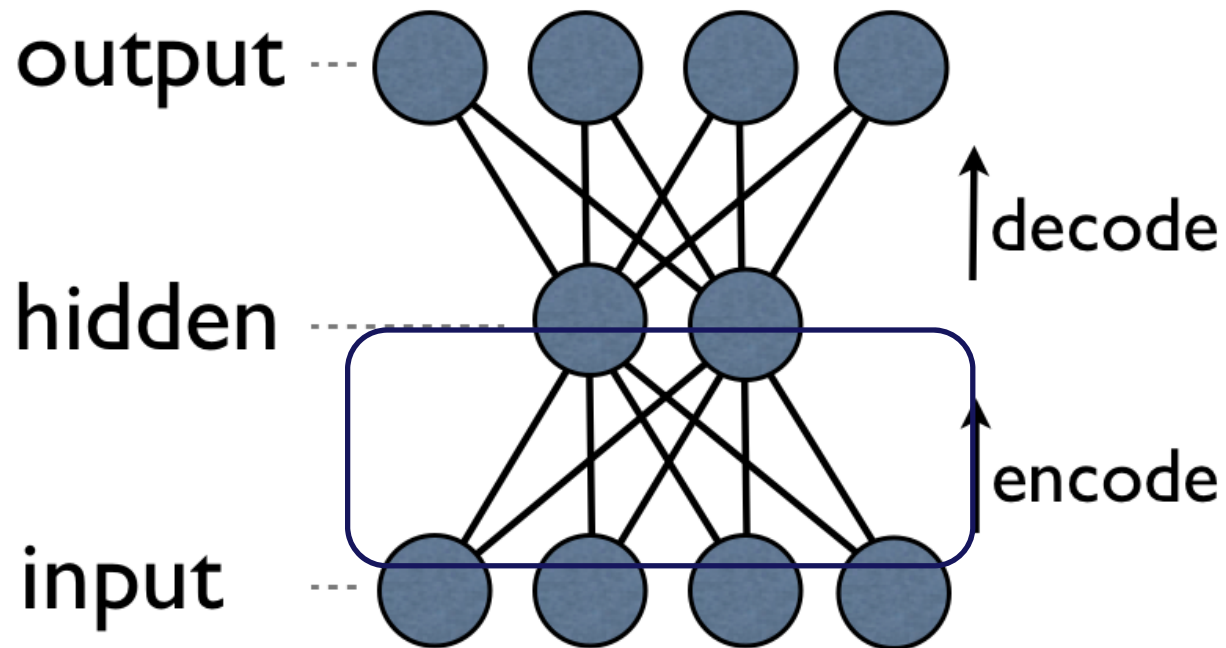
*EACH of the (non-output) layers is trained to be an **auto-encoder***

*Basically, it is forced to learn good features that describe what comes from the previous layer*

**an auto-encoder is trained, with an absolutely standard weight-adjustment algorithm to reproduce the input**



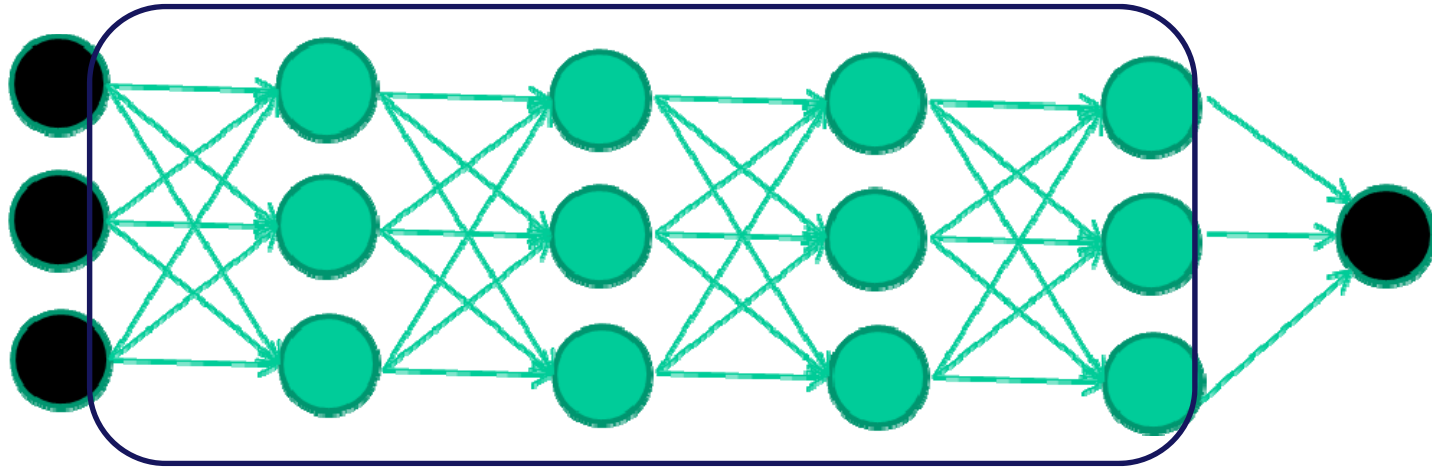
**an auto-encoder is trained, with an absolutely standard weight-adjustment algorithm to reproduce the input**



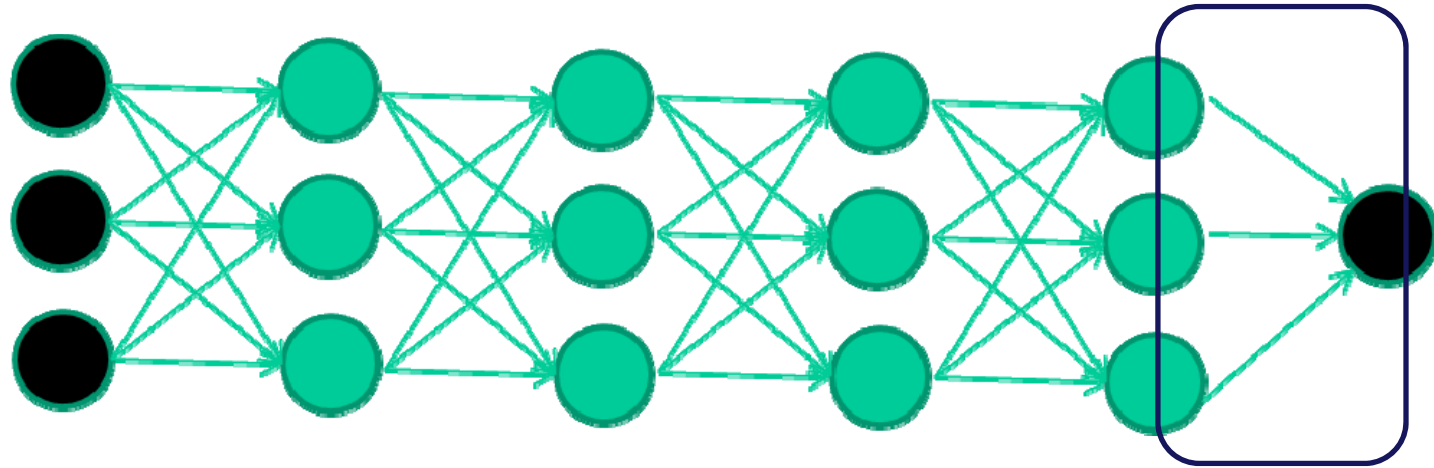
**By making this happen with (many) fewer units than the inputs, this forces the ‘hidden layer’ units to become good feature detectors**



intermediate layers are each trained to be auto encoders (or similar)

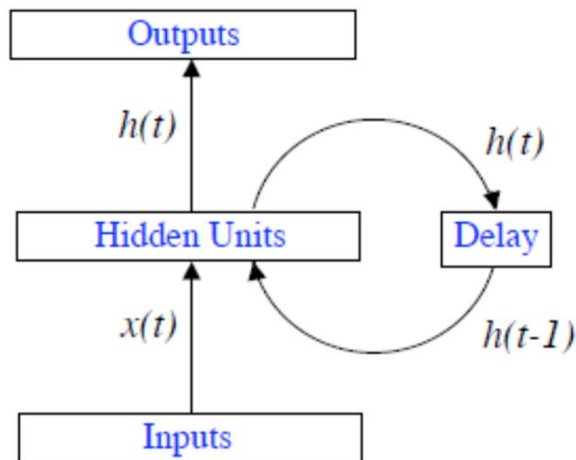


Final layer trained to predict class based  
on outputs from previous layers



# 循环神经网络

Recurrent neural networks (RNNs) are connectionist models with the ability to selectively pass information across sequence steps, while processing sequential data one element at a time.



The simplest form of **fully recurrent neural network** is an MLP with the previous set of hidden unit activations feeding back into the network along with the inputs

Allow a 'memory' of previous inputs to persist in the network's internal state, and thereby influence the network output

$$h(t) = f_H(W_{IH}x(t) + W_{HH}h(t-1))$$

$$y(t) = f_O(W_{HO}h(t))$$

$f_H$  and  $f_O$  are the activation function for hidden and output unit;  $W_{IH}$ ,  $W_{HH}$ , and  $W_{HO}$  are connection weight matrices which are learnt by training

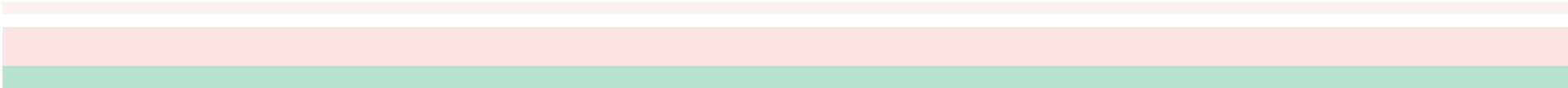
# 循环神经网络

---

The **limitations** of the Neural network (CNNs)

- ❑ Rely on the assumption of independence among the (training and test) examples.
  - After each data point is processed, the entire state of the network is lost
- ❑ Rely on examples being vectors of fixed length

We need to model the data with temporal or sequential structures and varying length of inputs and outputs

- Frames from video
  - Snippets of audio
  - Words pulled from sentences
- 

## 7.6 深度学习

---

### □ 深度学习常用软件包

- CAFFE

The Berkeley Vision and Learning Center (BVLC)

<http://caffe.berkeleyvision.org/>

- MatConvNet

The Oxford Visual Geometry Group (VGG)

<http://www.vlfeat.org/matconvnet/>

- Torch

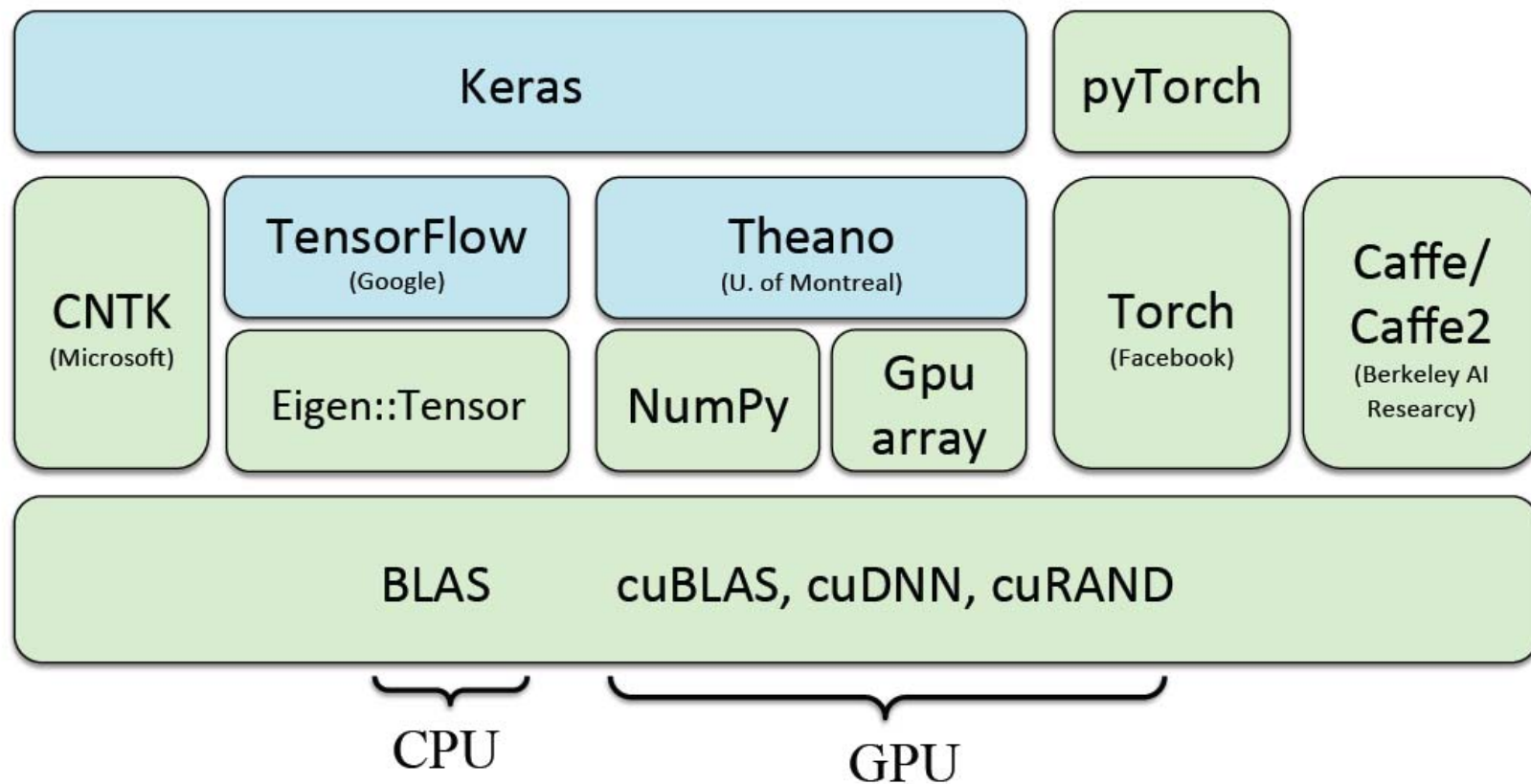
<http://torch.ch/>

### □ 神经网络基础教材

<http://hagan.okstate.edu/>

## 7.6 深度学习

### 深度学习常用软件包



## 7.7 深度森林

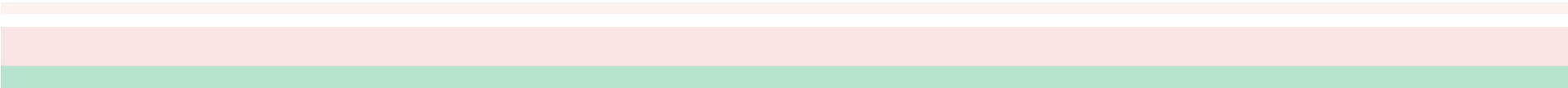
---

### □ 深度学习模型的成功因素

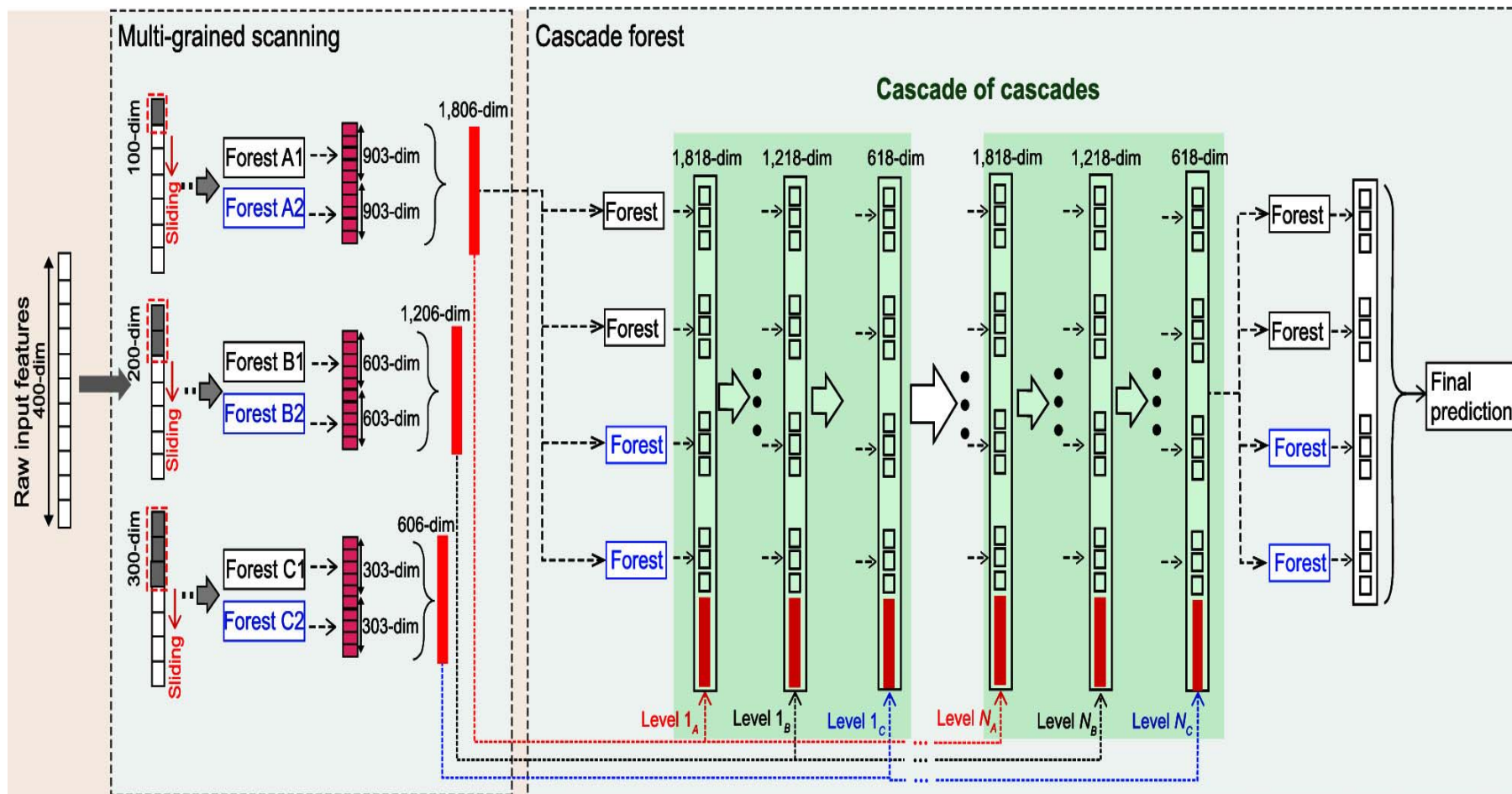
- 逐层加工 (layer-by-layer processing)
- 内部特征变换 (in-model feature transformation)
- 充分的模型复杂度 (sufficient model complexity)

□ 猜想：设计出满足上述条件的模型，就可能成为有效的深度学习模型，未必一定要使用神经网络

### □ 深度森林

- 验证上述猜想
  - 第一个基于不可微构件的深度学习模型
  - 训练过程不需BP，不依赖梯度计算
  - 神经网络的补充（涉及符号、离散建模或混合建模的任务上）
- 

## 7.7 深度森林





## 7.8 阅读材料

---

### □ 主流学术期刊

- Neural Computation
- Neural Networks
- IEEE Transactions on Neural Networks and Learning Systems

### □ 国际会议

- 国际神经信息处理系统会议 (NIPS)
- 国际神经网络联合会议 (IJCNN)

### □ 区域性国际会议

- 欧洲神经网络会议 (ICANN)
  - 亚太神经网络会议 (ICONIP)
- 