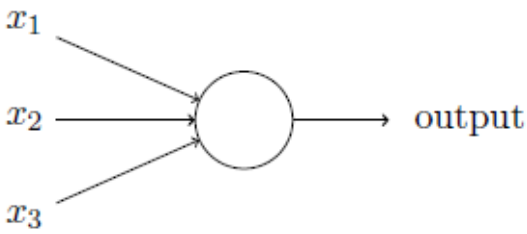
Michael Nielsen神经网络与深度学习
Neural networks and deep learning
Michael Nielsen
在线阅读链接：http://neuralnetworksanddeeplearning.com/index.html
中文版链接：https://pan.baidu.com/s/1bo0t7sz 密码: ecvf
代码下载链接：https://pan.baidu.com/s/1c1PuaGC 密码: 4w8n
（以识别手写数字为例）

Part 0 预备知识
0-0 感知器
（接受几个二进制输入，分配权重求和，产生一个二进制输出）



感知器

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

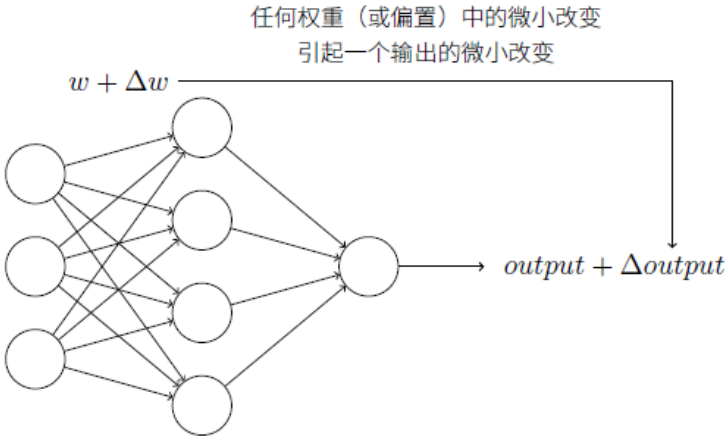
输出规则1

将阈值移到不等式另一边，用感知器的偏置b=-threshold代替。

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$

输出规则2

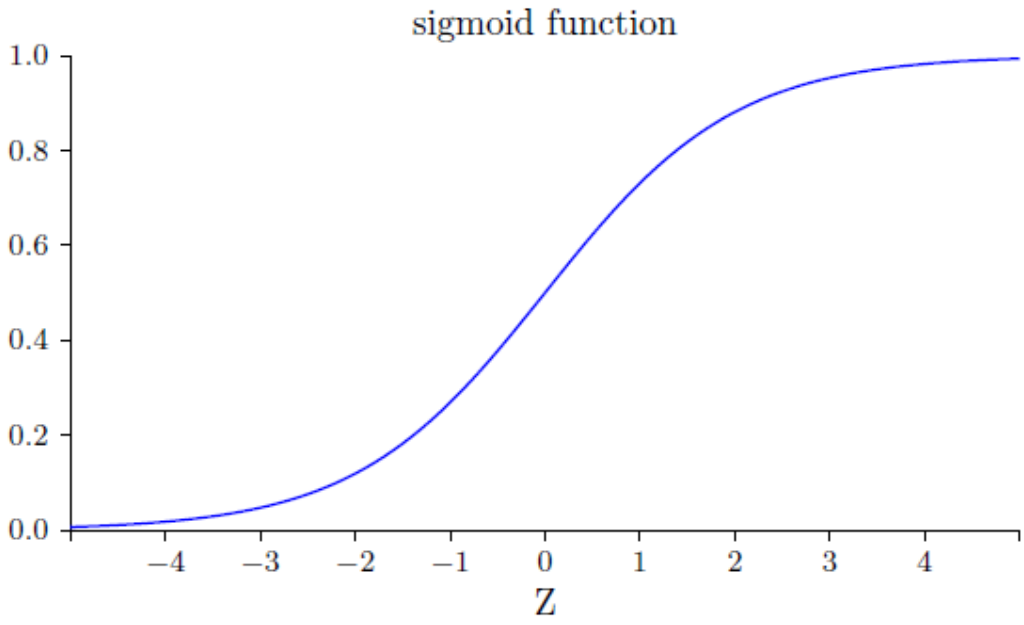
假如我们有一个感知器，那如果这个网络能学习权重和偏置，就能做出判断啦。
0-1 S型神经元



为了使我们的权重和偏置的微小改动只引起微小变化，我们定义一个S型函数（求偏导就知道为什么可以这样啦；而且事实上还有很多不同的激活函数）：

$$\sigma(z) \equiv \frac{1}{1+e^{-z}}$$

S型函数



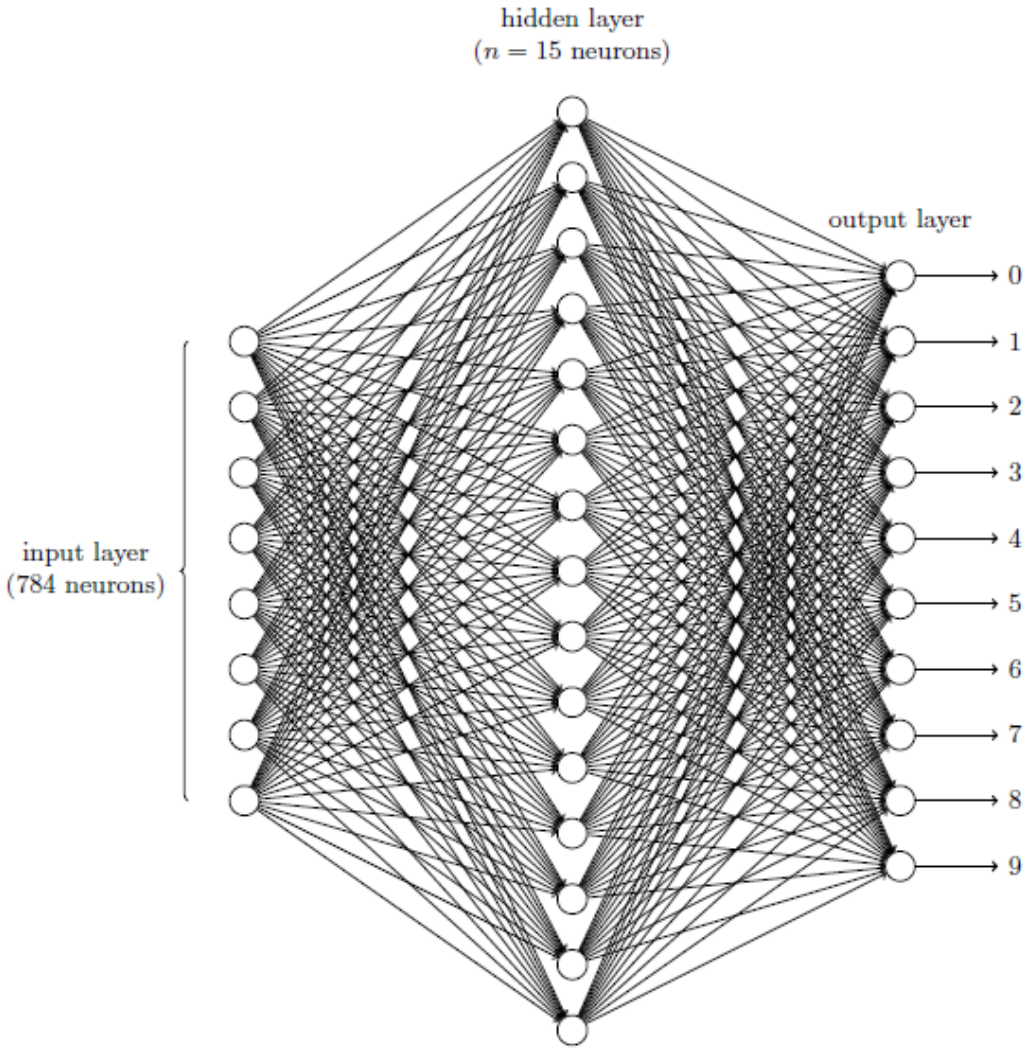
S型函数图像

那么此时输出变成：

$$\frac{1}{1+\exp(-\sum_j w_j x_j - b)}$$

输出

0-2 一个简单的手写数字分类神经网络结构



神经网络结构图

注：本文讨论的为前馈神经网络，即上一层输入作为下一层输出。（实际上也有递归神经网络~ ）

输入层：由于此网络的输入训练数据为扫描得到的2828的手写数字图像（*0,1,2,...,9*），因此输入层包含*784=2828*个神经元。

隐藏层：假设该层第一个神经元用于检测如下图像是否存在：

1

第二三四个分别检测以下是否存在：

234

那么若隐藏层的这四个神经元都被激活，则可以判断为0.

0

输出层：含有10个神经元。

Part 1 使用梯度下降算法学习权重和偏置
定义二次代价函数量化我们的目标：

$$C(w,b) \equiv \frac{1}{2n} \sum_x \|y(x) - a\|^2$$

代价函数

问题转化成求使C(w,b)最小的w和b。如下图，梯度下降方向则为使C最快减小的方向，eta称为学习速率。

$$w_k \to w_k' = w_k - \eta \frac{\partial C}{\partial w_k}$$
$$b_l \to b_l' = b_l - \eta \frac{\partial C}{\partial b_l}$$

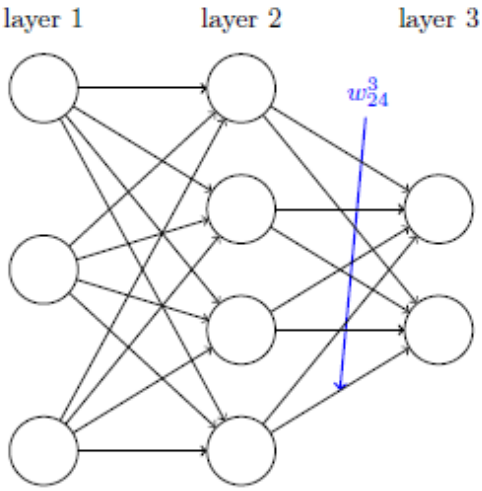
梯度下降学习规则

为了加快学习，采用随机梯度下降：通过随机选取小量训练样本计算deltaCx，求平均值即可得到deltaC的估算。

$$w_k \to w_k' = w_k - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial w_k}$$
$$b_l \to b_l' = b_l - \frac{\eta}{m} \sum_j \frac{\partial C_{X_j}}{\partial b_l}$$

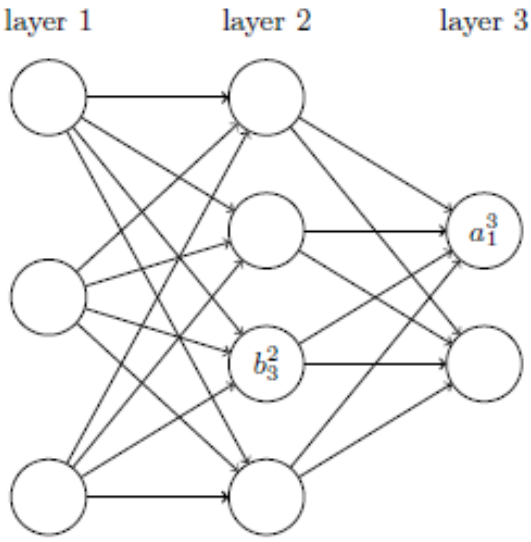
随机梯度下降学习规则

Part 2 反向传播算法计算代价函数的梯度
2-0 一些符号表示



w^l_{jk} 是从 $(l-1)^{\text{th}}$ 层的第 k^{th} 个神经元到 l^{th} 层的第 j^{th} 个神经元的连接上的权重

权重



隐藏层和输出层的值

代价函数表示为：

$$C = \frac{1}{2n}\sum_x \|y(x) - a^L(x)\|^2$$

代价函数

2-1 四个基本方程

输出层误差的方程，δ^L： 每个元素定义如下：

$$\delta^L_j = \frac{\partial C}{\partial a^L_j}\sigma'(z^L_j)$$

输出层误差

使用下一层的误差 δ^{l+1} 来表示当前层的误差 δ^l： 特别地，

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$$

当前层误差

代价函数关于网络中任意偏置的改变率： 就是

$$\frac{\partial C}{\partial b_j^l} = \delta_j^l$$

第l层第j个神经元的误差

代价函数关于任何一个权重的改变率： 特别地，

$$\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$$

2-2 反向传播算法

1. **输入 x：** 为输入层设置对应的激活值 a^1 。
2. **前向传播：** 对每个 $l = 2, 3, ..., L$ 计算相应的 $z^l = w^l a^{l-1} + b^l$ 和 $a^l = \sigma(z^l)$
3. **输出层误差 δ^L：** 计算向量 $\delta^L = \nabla_a C \odot \sigma'(z^L)$
4. **反向误差传播：** 对每个 $l = L-1, L-2, ..., 2$，计算 $\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l)$
5. **输出：** 代价函数的梯度由 $\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l$ 和 $\frac{\partial C}{\partial b_j^l} = \delta_j^l$ 得出

算法步骤

Part 3 改进神经网络的方法

3-0 交叉熵代价函数

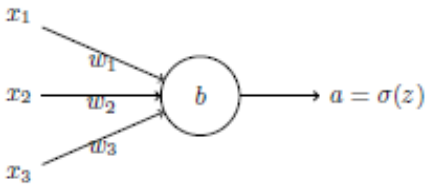
$$C = \frac{(y-a)^2}{2}$$

二次代价函数

$$\frac{\partial C}{\partial w} = (a-y)\sigma'(z)x = a\sigma'(z)$$
$$\frac{\partial C}{\partial b} = (a-y)\sigma'(z) = a\sigma'(z)$$

权重和偏置的偏导数

在神经元的输出接近于1时，代价函数对w和b的偏导很小，因此学习速率下降。为了解决这个问题引入交叉熵代价函数。



神经元的输出就是 $a = \sigma(z)$，其中 $z = \sum_j w_j x_j + b$ 是输入的带权和。我们如下定义这个神经元的交叉熵代价函数：

$$C = -\frac{1}{n} \sum_x [y \ln a + (1-y) \ln(1-a)] \tag{57}$$

交叉熵代价函数

$$\frac{\partial C}{\partial w_j} = \frac{1}{n} \sum_x \frac{\sigma'(z) x_j}{\sigma(z)(1-\sigma(z))} (\sigma(z) - y)$$

关于权重的偏导数

简化为：

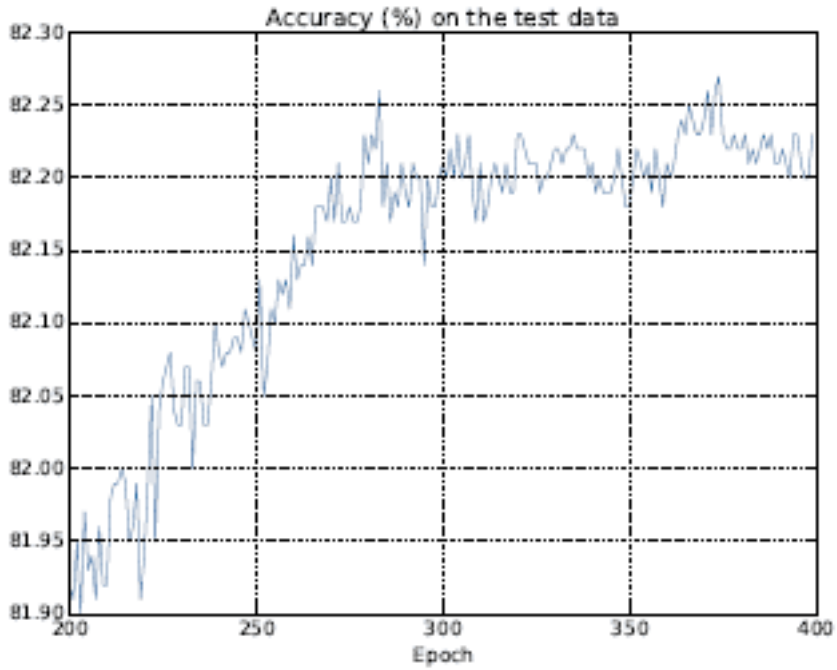
$$\frac{\partial C}{\partial w_j} = \frac{1}{n} \sum_x x_j (\sigma(z) - y)$$

上述算式表明权重的学习速度受到输出中的误差的控制，与S型函数的导数无关。
类似地，

$$\frac{\partial C}{\partial b} = \frac{1}{n} \sum_x (\sigma(z) - y)$$

关于权重的偏导数

3-1 过度拟合和规范化
（1）过拟合
下图为分类准确率在测试集上的表现，看到在280迭代期左右分类准确率停止增长，我们说网络在280迭代期后就过度训练了。



过拟合

检测过度拟合的方法：将全部数据分为test_data,validation_data,train_data，使用validation_data作测试，一旦验证数据的分类准确率已经饱和我们便停止训练，这个策略称为提前停止。

Q: 为什么不用test_data而是validation_data?

A: 如果我们设置超参数是基于test_data,最终我们得到过度拟合于test_data的超参数，但网络的性能并不能泛化到其他数据集合上，因此借助validation_data来克服这个问题。这种寻找好的超参数的方法称为hold out方法，因为validation data是从training_data拿出的一部分。

（2）规范化

减轻过拟合的方法有：增加训练样本数量、降低网络的规模等。

即使我们只有一个固定的网络和固定的训练集，我们可以利用规范化技术。最常用的为L2规范化（权重衰减）：即增加一个额外的项到代价函数上。

$$C = -\frac{1}{n}\sum_{xj}\left[y_j \ln a_j^L + (1-y_j)\ln(1-a_j^L)\right] + \frac{\lambda}{2n}\sum_w w^2$$

规范化的交叉熵

$$C = \frac{1}{2n}\sum_x \|y - a^L\|^2 + \frac{\lambda}{2n}\sum_w w^2$$

规范化的二次代价函数

两者均可以写成：

$$C = C_0 + \frac{\lambda}{2n}\sum_w w^2$$

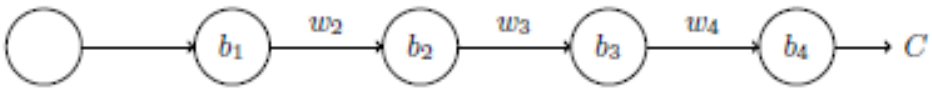
规范化代价函数

Q: 规范化项为什么可以降低过拟合？
A: 联想噪声线性模型某些情况下比多项式模型具有更强大广泛的预测，但这样的优势不是绝对的。
规范化的其他技术：
L1规范化：

$$C = C_0 + \frac{\lambda}{n}\sum_w |w|$$

Dropout：弃权
人为扩展训练数据
3-2 权重初始化
假设我们有Nin个输入权重的神经元，使用均值为0，方差为1/Nin的高斯随机分布初始化权重；使用均值为0，标准差为1的高斯分布初始化偏置。
Part 4 神经网络可以计算任何函数
（1）通用逼近性质
http://www.dartmouth.edu/~gvc/Cybenko_MCSS.pdf
（2）某个即时反馈神经网络训练模型结果的网站
http://cs.stanford.edu/people/karpathy/convnetjs/demo/regression.html
Part 5 深度神经网络
5-0 消失的梯度
（某些深度神经网络中，我们隐藏层BP的时候梯度倾向于变小，意味着前面隐藏层中神经元的学习速度小于后面的隐藏层）

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1) \times w_2 \times \sigma'(z_2) \times w_3 \times \sigma'(z_3) \times w_4 \times \sigma'(z_4) \times \frac{\partial C}{\partial a_4}$$



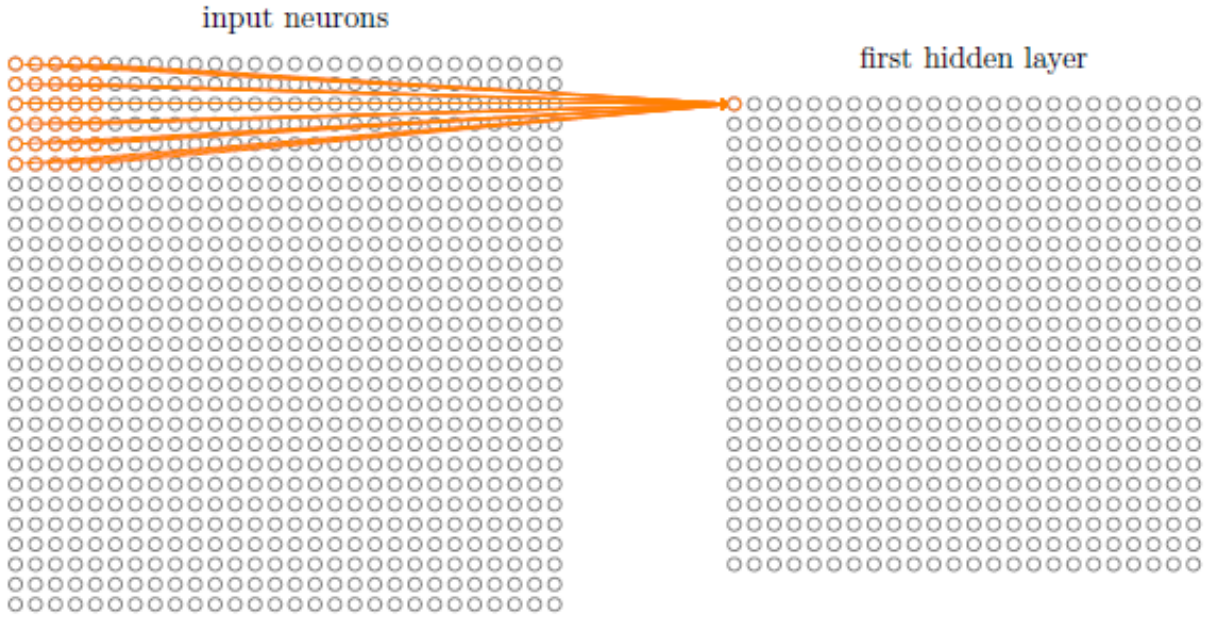
梯度消失

$$\frac{\partial C}{\partial b_1} = \sigma'(z_1)\overbrace{w_2\sigma'(z_2)}^{<\frac{1}{4}}\overbrace{w_3\sigma'(z_3)}^{<\frac{1}{4}}\underbrace{w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}}$$

common terms

$$\frac{\partial C}{\partial b_3} = \sigma'(z_3)\overbrace{w_4\sigma'(z_4)\frac{\partial C}{\partial a_4}}$$

5-1 卷积神经网络

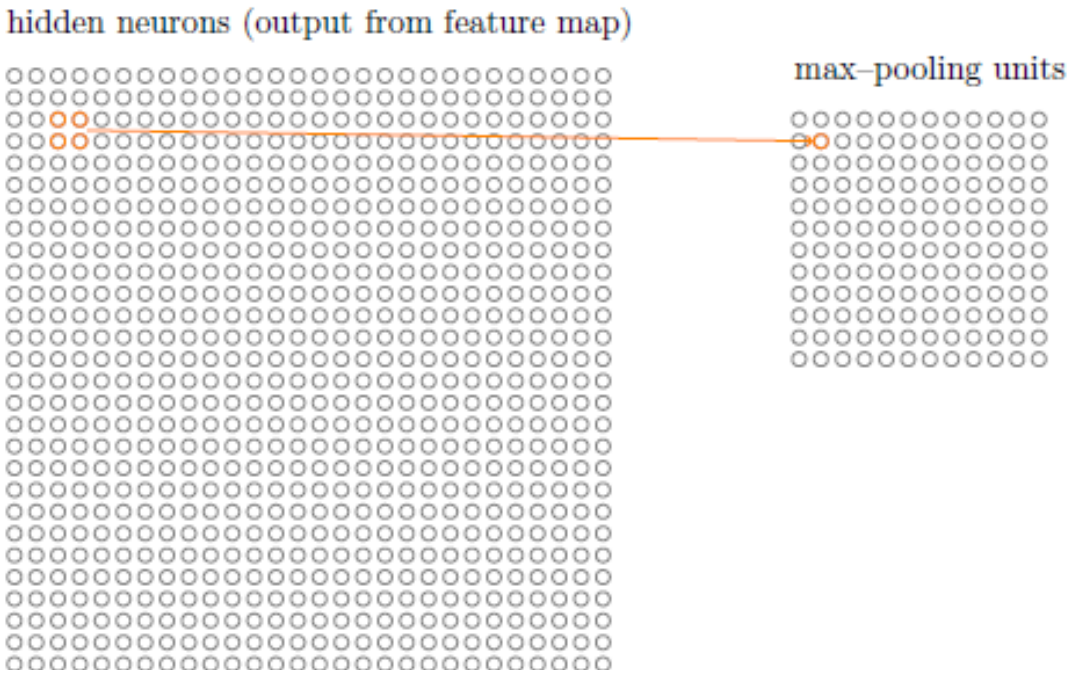


局部感受野

对第 j, k 个隐藏神经元，输出为：

$$\sigma\left(b + \sum_{l=0}^{4}\sum_{m=0}^{4} w_{l,m}a_{j+l,k+m}\right)$$

共享权重和偏置



混合层

5-2 其他深度学习模型

RNN、Boltzmann Machine、生成式模型、迁移学习、强化学习等。