机器学习

Machine Learning

主讲人：张敏 清华大学长聘副教授

ML 机器学习

MACHINE LEARNING-MIN ZHANG

Unit.10

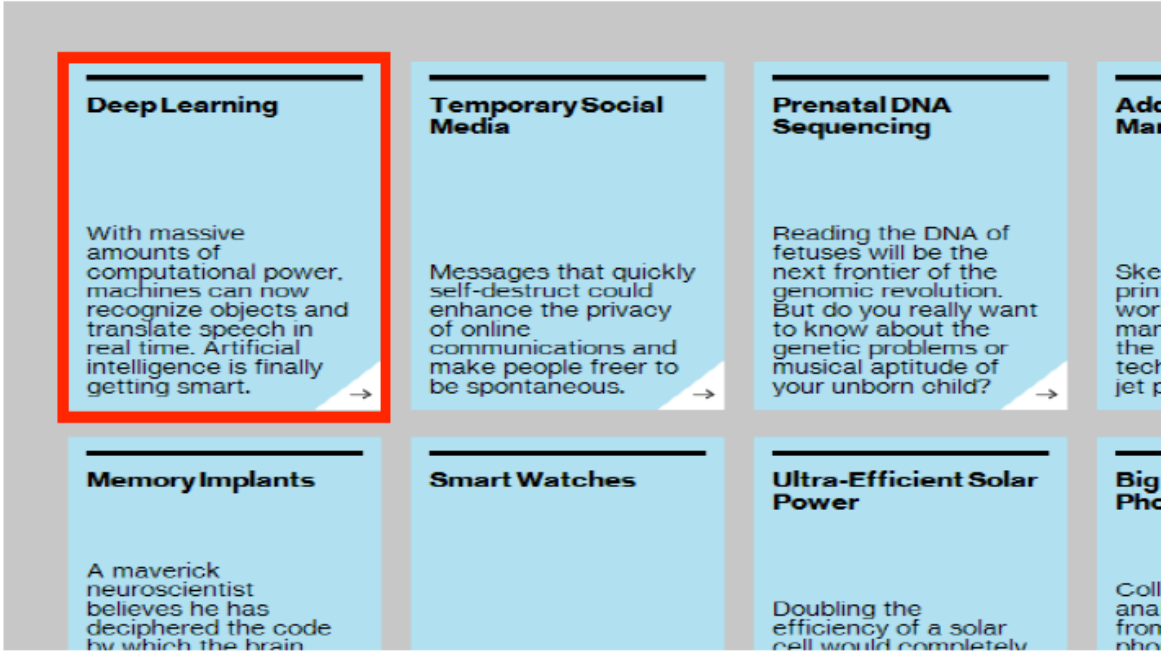
深度学习基础（I）

*图片均来自网络或已发表刊物

目录

10 BREAKTHROUGH TECHNOLOGIES 2013

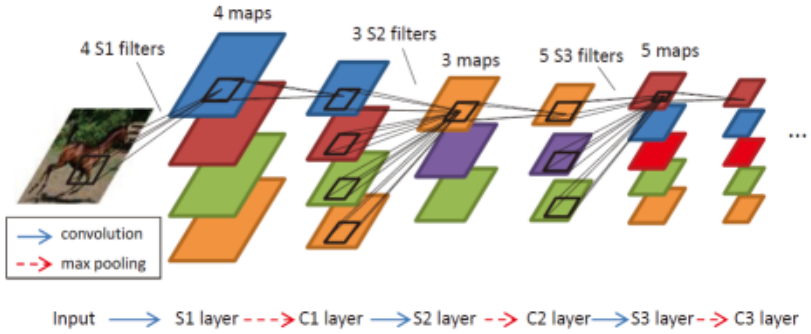
MIT Technology Review



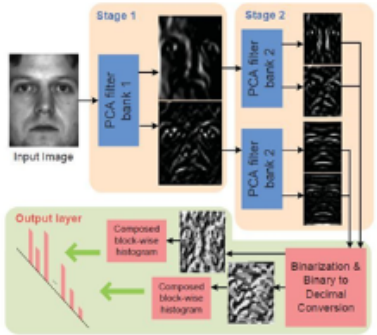
http://www.technologyreview.com/featuredstory/513696/deep-learning/

深度学习（Deep Learning）的定义

- 狭义：层数很深的神经网络
- 更扩展一点：神经网络
  - 前馈网络（Forward feedback network）:多层感知机(MLP) ,卷积神经网络 (CNN)等
  - 回馈网络（Feedback network）:长短时记忆网络(LSTM) 等
- 广义
  - 层次化的机器学习模型



Sparse HMAX (Hu et al., 2014)

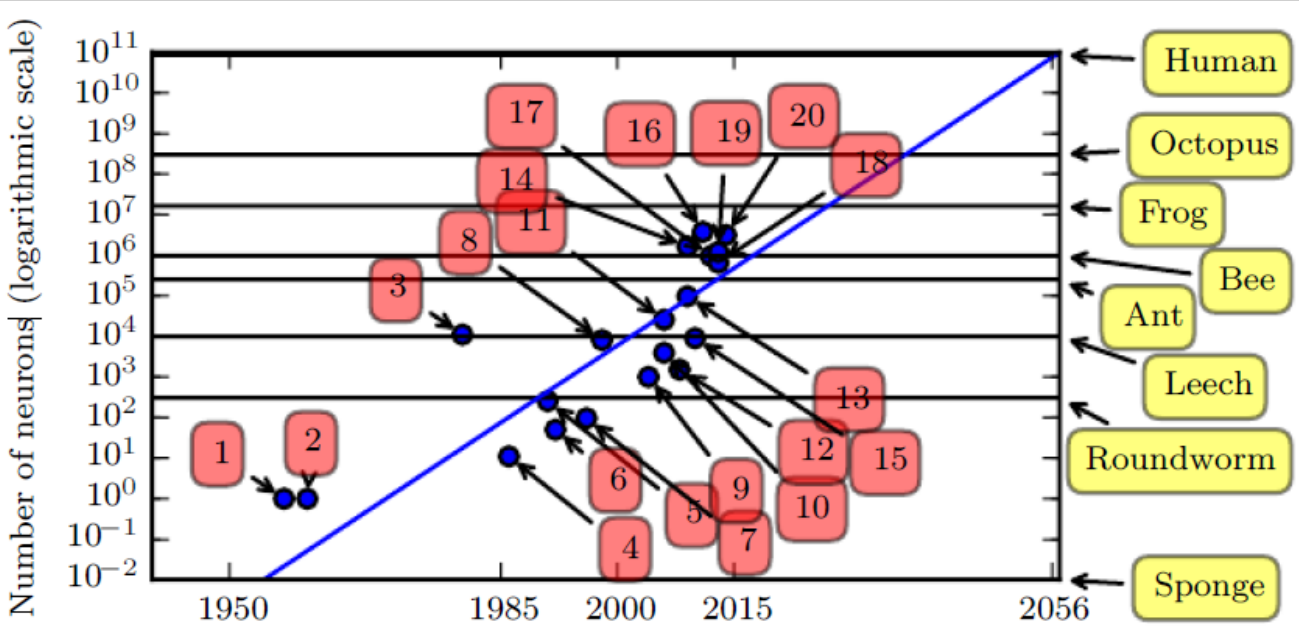


PCA net (Chan et al., 2014)

人工神经网络与动物中的神经元个数

By Dmitry Fedyuk

https://dmitry.ai/t/topic/206

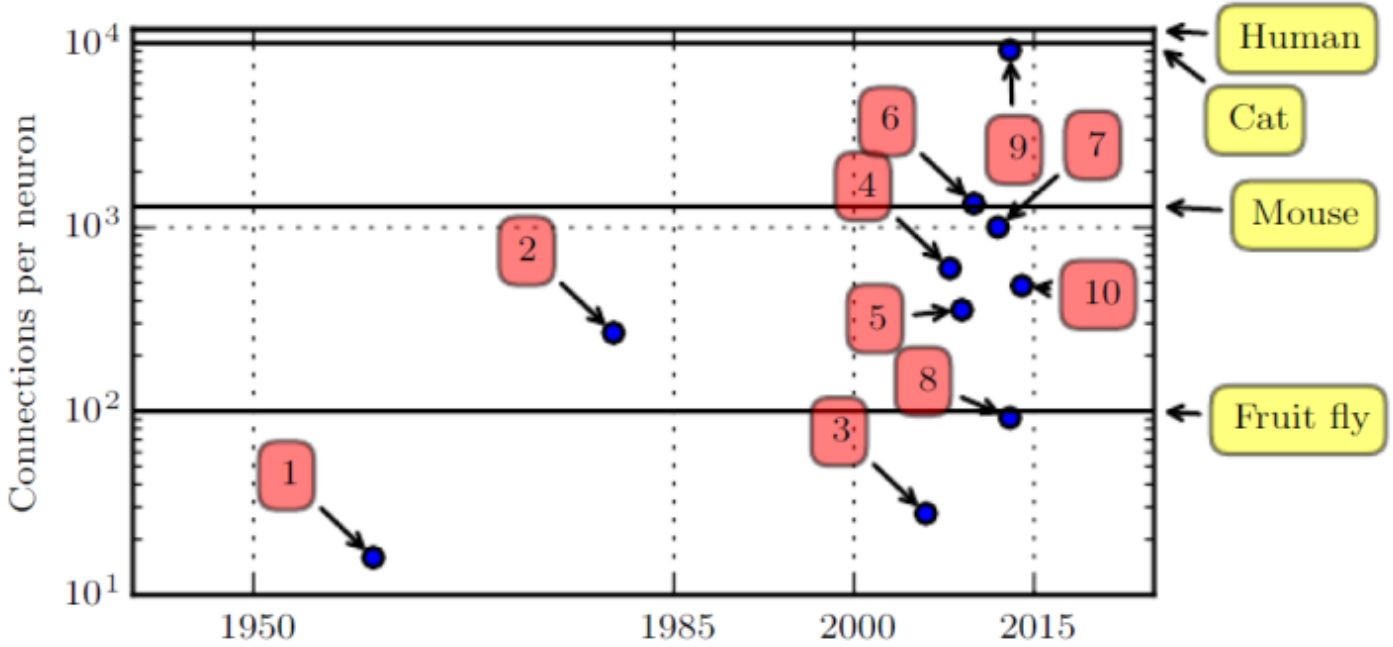


1. Perceptron (Rosenblatt, 1958, 1962)
2. Adaptive linear element (Widrow and Hoff, 1960)
3. Neocognitron (Fukushima, 1980)
4. Early back-propagation network (Rumelhart et al., 1986b)
5. Recurrent neural network for speech recognition (Robinson and Fallside, 1991)
6. Multilayer perceptron for speech recognition (Bengio et al., 1991)
7. Mean field sigmoid belief network (Saul et al., 1996)
8. LeNet-5 (LeCun et al., 1998b)
9. Echo state network (Jaeger and Haas, 2004)
10. Deep belief network (Hinton et al., 2006)
11. GPU-accelerated convolutional network (Chellapilla et al., 2006)
12. Deep Boltzmann machine (Salakhutdinov and Hinton, 2009a)
13. GPU-accelerated deep belief network (Raina et al., 2009)
14. Unsupervised convolutional network (Jarrett et al., 2009)
15. GPU-accelerated multilayer perceptron (Ciresan et al., 2010)
16. OMP-1 network (Coates and Ng, 2011)
17. Distributed autoencoder (Le et al., 2012)
18. Multi-GPU convolutional network (Krizhevsky et al., 2012)
19. COTS HPC unsupervised convolutional network (Coates et al., 2013)
20. GoogLeNet (Szegedy et al., 2014a)

每个神经元的连接数（人工神经网络与动物的对比）

By Dmitry Fedyuk

https://dmitry.ai/t/topic/202

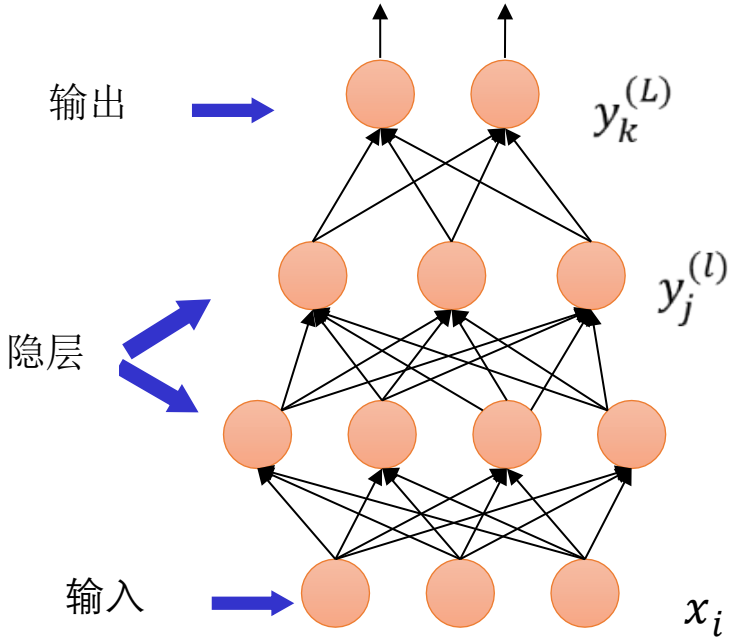


1. Adaptive linear element (Widrow and Hoff, 1960)
2. Neocognitron (Fukushima, 1980)
3. GPU-accelerated convolutional network (Chellapilla et al., 2006)
4. Deep Boltzmann machine (Salakhutdinov and Hinton, 2009a)
5. Unsupervised convolutional network (Jarrett et al., 2009)
6. GPU-accelerated multilayer perceptron (Ciresan et al., 2010)
7. Distributed autoencoder (Le et al., 2012)
8. Multi-GPU convolutional network (Krizhevsky et al., 2012)
9. COTS HPC unsupervised convolutional network (Coates et al., 2013)
10. GoogLeNet (Szegedy et al., 2014a)

目录

- 背景
- 多层感知机（MLP）
- 卷积神经网络（CNN）
- 序列神经网络
  - 循环神经网络（RNN）
  - 长短期记忆网络（LSTM）
  - 门控循环单位网络（GRU）
- 应用举例

Multi-layer Perceptron (MLP) 多层感知机



- 除了输入共有 L 层
- 连接：
  - 层与层之间全连接
  - 层与层之间无反向连接
  - 同一层中无横向连接
- 每一个神经元接收来自前一层的输入并经过一个激活函数后输出

Activation functions 激活函数

- 逻辑激活函数Logistic function (sigmoid)

$$f(z) = \frac{1}{1 + \exp(-z)}$$

- 双曲正切函数Hyperbolic tangent function

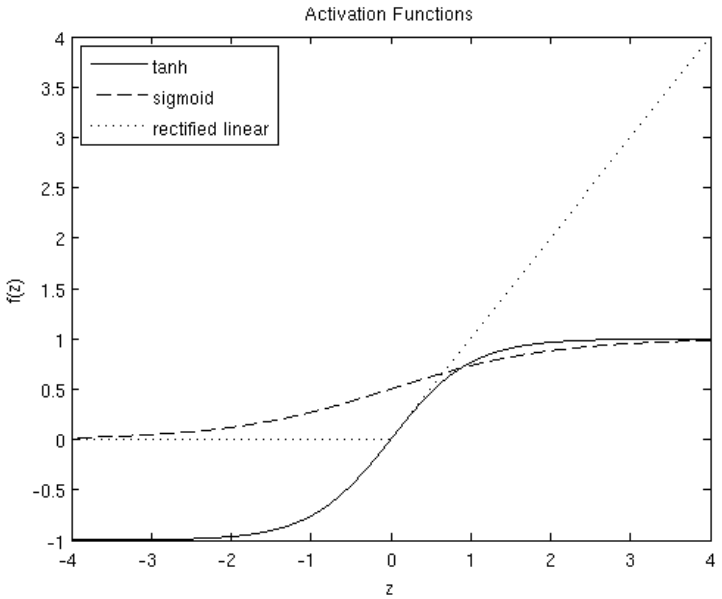
$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$tanh(x) = 2sigmoid(2x) - 1$$

- 校正线性激活函数Rectified linear activation function (ReLU)

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

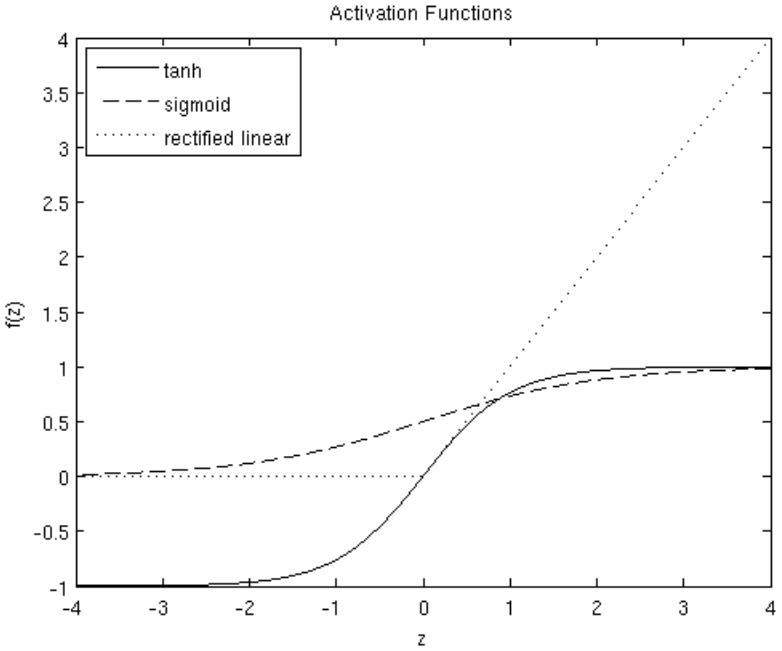
最简单的logistic regression也可看成没有隐含层的神经网络



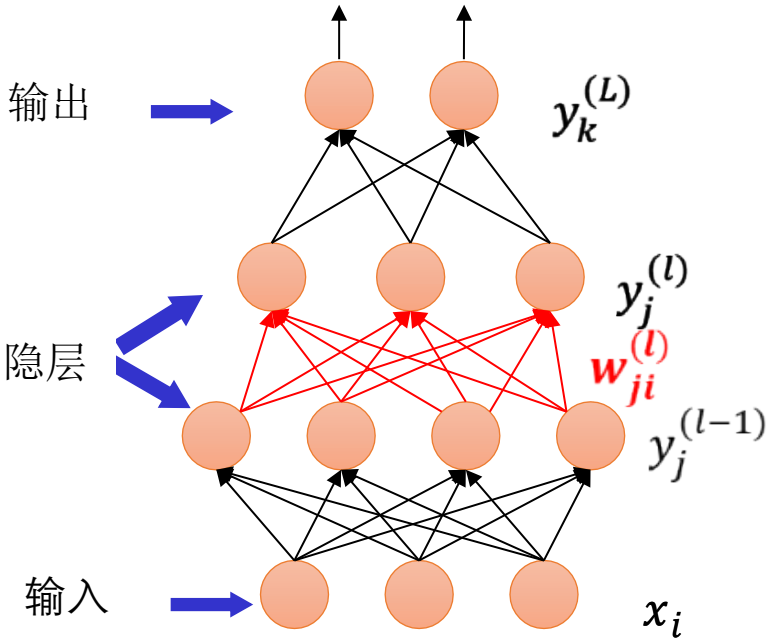
Activation functions 激活函数

为什么要用激活函数？

- 如果不用激活函数，每层的输出都是来自上层的输入的线性函数
  - 无论神经网络有多少层，输出都是输入的线性组合
- 使用激活函数，为神经元引入了非线性因素
  - 使得神经网络可以逼近任何非线性函数
  - 方便应用到众多非线性模型中



前向传播(Forward pass)



对于层数 $l = 1, \ldots, L$ 计算第 l 层神经元 j 的输入

$$u_j^{(l)} = \sum_i w_{ji}^{(l)} y_i^{(l-1)} + b_j^{(l)}$$

以及它的输出

$$y_j^{(l)} = f(u_j^{(l)})$$

其中 $f(\cdot)$ 是激活函数

- $y^{(0)} = x$
- 每个输入样例都有期望输出 t
- 对于 $l = L$，$f(\cdot)$ 可以是激活函数或者

$softmax$ 函数：$y_i = \frac{e_i}{\sum_j e_j}$

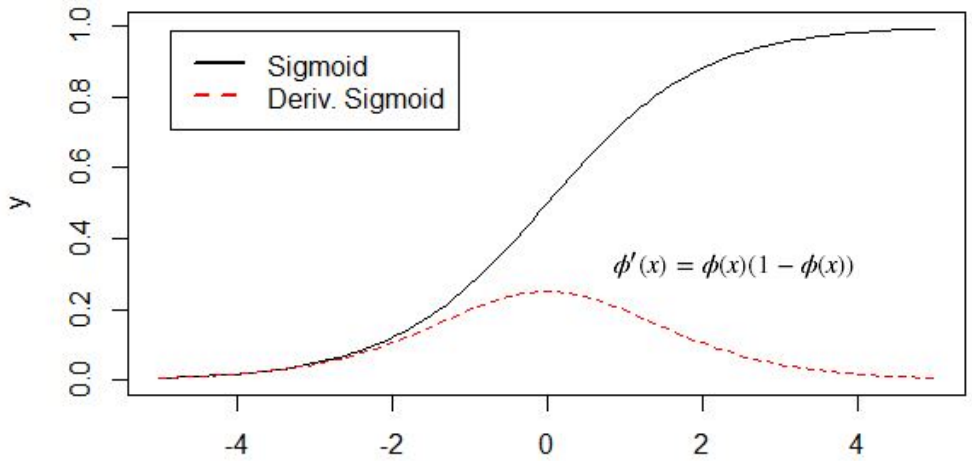
误差函数

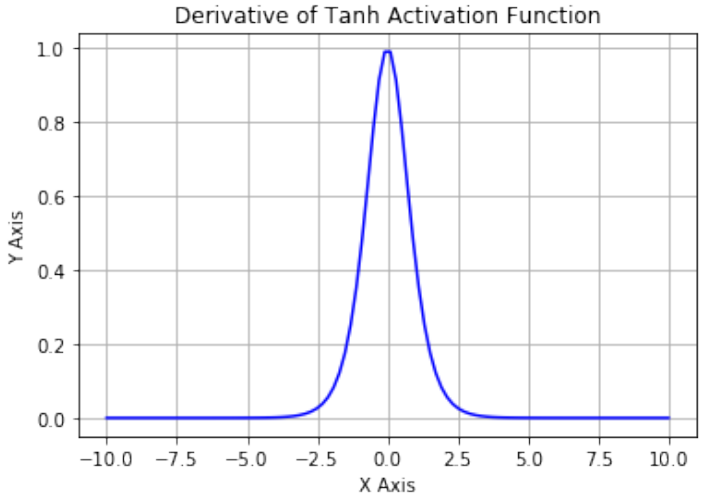
- 误差函数 $$E = \frac{1}{N}\sum_{n=1}^{N} E^{(n)}$$ 其中 $E^{(n)}$ 是每个输入样例 n 的误差函数
- 最小平方误差 $$E^{(n)} = \frac{1}{2}\sum_{k=1}^{K}(t_k - y_k^{(L)})^2,$$ 其中 t 是形式如 $(0,0,\ldots,1,0,0)^T$ 的目标

（不适合与sigmoid配合使用: 当自变量非常大或非常小时，导数接近0 →梯度消失问题）

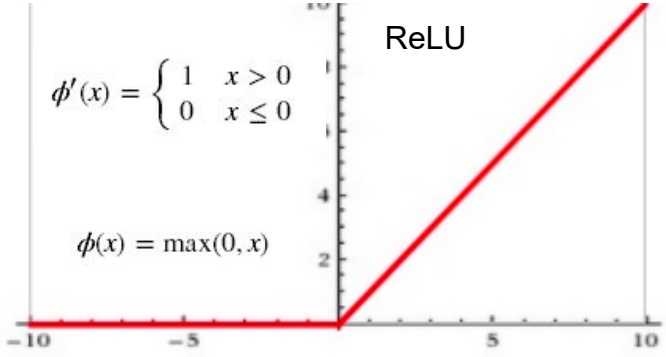
$$y_k^{(L)} = \frac{1}{1+\exp(-w_k^{(L)\top} y^{(L-1)}) - b_k^{(L)})}$$

梯度消失





- Sigmoid: 梯度消失问题
- Tanh: 稍微好一点
- ReLU: 解决梯度消失方面表现良好 →



误差函数

- 误差函数 $E = \frac{1}{N}\sum_{n=1}^{N} E^{(n)}$ 其中 $E^{(n)}$ 是每个输入样例 n 的误差函数
- 最小平方误差 $E^{(n)} = \frac{1}{2}\sum_{k=1}^{K}(t_k - y_k^{(L)})^2$, 其中 t 是形式如 $(0,0,...,1,0,0)^T$ 的目标

（不适合与sigmoid配合使用: 当自变量非常大或非常小时，导数接近0 →梯度消失问题）

$$y_k^{(L)} = \frac{1}{1+\exp(-w_k^{(L)\top} y^{(L-1)}) - b_k^{(L)})}$$

MSE适合回归问题

- 交叉熵

$$H(p,q) = -\sum_{i=1}^{n} p(x_i) log(q(x_i)) \qquad E^{(n)} = -\sum_{k=1}^{K} t_k \ln y_k^{(L)}, \quad y_k^{(L)} = \frac{\exp(w_k^{(L)\top} y^{(L-1)} + b_k^{(L)})}{\sum_{j=1}^{K}\exp(w_j^{(L)\top} y^{(L-1)} + b_j^{(L)})}$$

- 与softmax配合使用非常简洁
- 也可与sigmoid配合使用
- 但不能用ReLU，因为理论上不通

交叉熵只对正确的分类看中，因此非常适合分类问题，不适合回归问题

交叉熵+Softmax

$$Loss = -\sum_i t_i ln y_i$$

$$y_i = \frac{e^i}{\sum_j e^j} = 1 - \frac{\sum_{j \neq i} e^j}{\sum_j e^j}$$

$$\begin{aligned}
\frac{\partial Loss_i}{\partial_i} &= -\frac{\partial ln y_i}{\partial_i} \\
&= -\frac{\sum_j e^j}{e^i} \cdot \frac{\partial(\frac{e_i}{\sum_j e^j})}{\partial_i} \\
&= -\frac{\sum_j e^j}{e^i} \cdot (-\sum_{j \neq i} e^j) \cdot \frac{\partial(\frac{1}{\sum_j e^j})}{\partial_i} \\
&= \frac{\sum_j e^j \cdot \sum_{j \neq i} e^j}{e^i} \cdot \frac{-e^i}{(\sum_j e^j)^2} \\
&= -\frac{\sum_{j \neq i} e^j}{\sum_j e^j} \\
&= -(1 - \frac{e^i}{\sum_j e^j}) \\
&= y_i - 1
\end{aligned}$$

可参考：
https://blog.csdn.net/bitcarmanlee/article/details/105619286

权重更新

- 权重更新

学习率

$$w_{ji}^{(l)} = w_{ji}^{(l)} - \alpha \frac{\partial E}{\partial w_{ji}^{(l)}} \qquad b_j^{(l)} = b_j^{(l)} - \alpha \frac{\partial E}{\partial b_j^{(l)}}$$

- 更新 $w_{ji}^{(l)}$ 时（$b_j^{(l)}$ 不必要）经常还会用到权重衰减，即在损失函数上添加额外的一项

$$J = E + \frac{\lambda}{2} \sum_{i,j,l} (w_{ji}^{(l)})^2$$

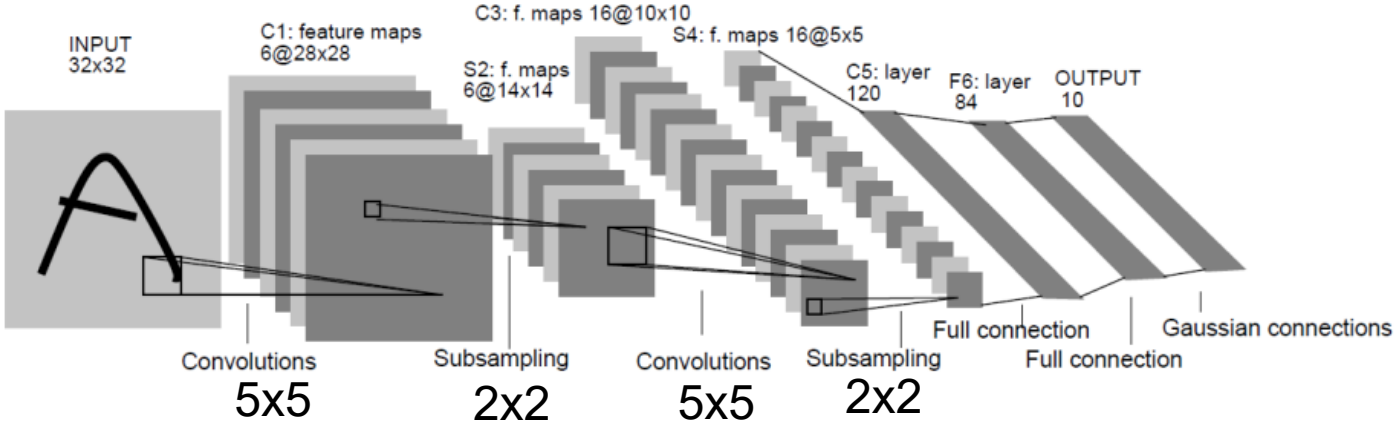
- w 的权重更新变为

$$w_{ji}^{(l)} = w_{ji}^{(l)} - \alpha \frac{\partial J}{\partial w_{ji}^{(l)}} = w_{ji}^{(l)} - \alpha \frac{\partial E}{\partial w_{ji}^{(l)}} - \alpha \lambda w_{ji}^{(l)}$$

目录

- 背景
- 多层感知机（MLP）
- 卷积神经网络（CNN）
- 序列神经网络
  - 循环神经网络（RNN）
  - 长短期记忆网络（LSTM）
  - 门控循环单位网络（GRU）
- 应用举例

卷积神经网络 (Convolutional neural network, CNN)

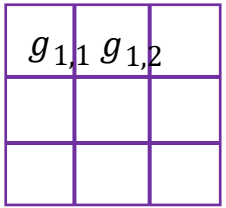


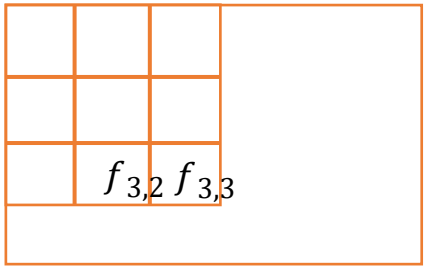
- 局部连接以及权重共享
- C layers: 卷积层(convolutions)
  - 输出 $y_i = f(\sum_{\Omega} w_j x_j + b)$， Ω :区域块大小， $f(\cdot)$: sigmoid 函数， w , b: 参数
- S layers: 采样层(subsampling)（池化层, 以平均池化为例average pooling）
  - 输出 $y_i = f\left(\frac{1}{|\Omega|}\sum_{\Omega} x_j\right)$， Ω :池化大小

2D 卷积

- 假设有两个大小分别为$M \times N$和$K_1 \times K_2$的矩阵f和g，其中$M \geq K_1, N \geq K_2$（g通常称为“filter”，“卷积核”，“局部感受野”）
- 两个矩阵的离散卷积

$$h[m,n] = (f * g)[m,n] \triangleq \sum_{k_1=1}^{K_1} \sum_{k_2=1}^{K_2} f[m-k_1, n-k_2]\, g[k_1, k_2]$$





当 $m = 4, n = 4$

$$(f * g)_{m,n} = f_{3,3}g_{1,1} + f_{3,2}g_{1,2} + f_{3,1}g_{1,3} + f_{2,3}g_{2,1} + \cdots$$

相当于把filter旋转180度，然后依次滑动，对与原始f中的重叠区域取按位点积求和

- valid shape: h 大小为 $(M - K_1 + 1) \times (N - K_2 + 1)$ ← CNN采用的模式
- full shape: h 大小为 $(M + K_1 - 1) \times (N + K_2 - 1)$ > 边界上补0
- same shape: h 大小为 $M \times N$

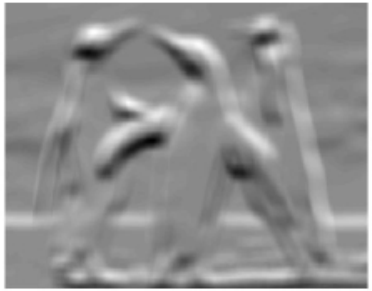
举例

原始图像

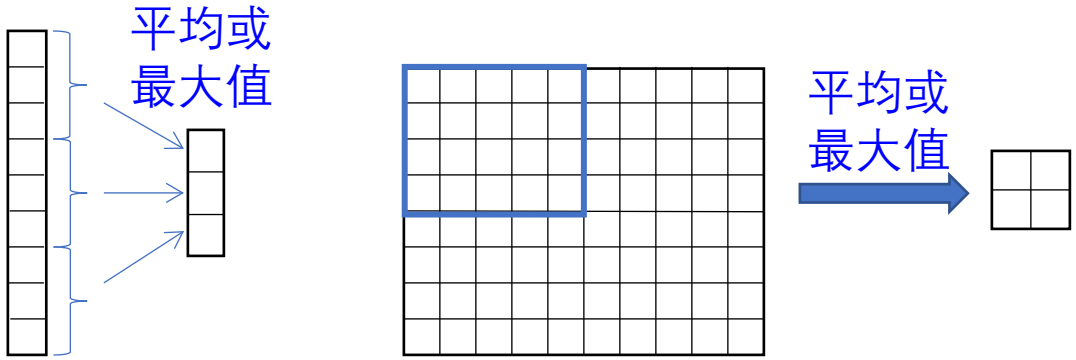
*

filter

特征映射
feature map



局部池化（Pooling in local regions）



Pooling size=3x1　　Pooling size=4x5

- 把卷积特征分成不重合的 $m \times n$ 个区域，然后在这些区域上取平均（或最大值）来得到池化后的特征
- 另一个常用的池化方法是 L2 pooling

$$p = \sqrt{\sum_{y_{ij} \in \Omega} y_{ij}^2}$$

CNN的特点及优点

- 通过卷积层提取图像的各种特征；
- 通过池化层对原始特征信号进行抽象 (减少训练参数，减轻模型过拟合)
- 输入的图像与网络的拓扑结构很好地吻合
- 权重共享极大减少了网络的训练参数，使NN结构更简单，适应性更强

CNN 的缺点

- 计算开销依然很大
- 包括 LeCun 在内的许多研究者认为这不是一个通用的模式识别方法

GPU 可以缓解这个问题

目录

- 背景
- 多层感知机（MLP）
- 卷积神经网络（CNN）
- 序列神经网络
  - 循环神经网络（RNN）
  - 长短期记忆网络（LSTM）
  - 门控循环单位网络（GRU）
- 应用举例