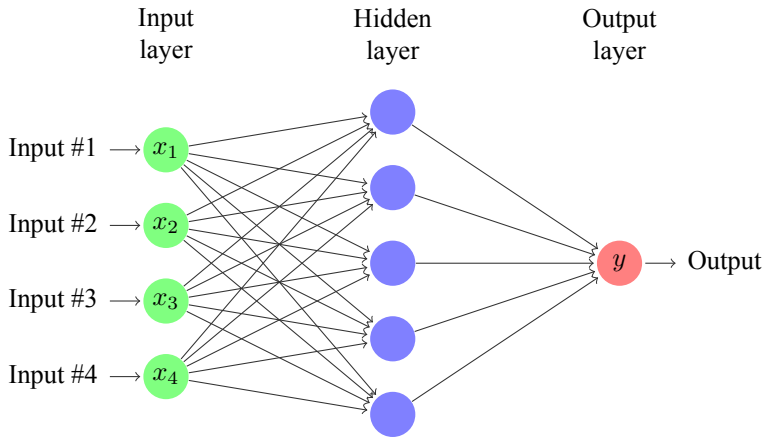
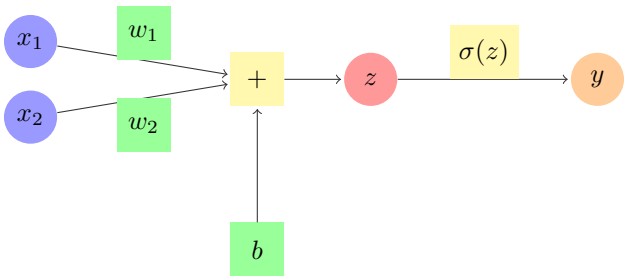
深度学习笔记

一、深度学习模型

1. 神经网络示意图



一个 2 层的 full-connect 神经网络



$f(x) = wX + b$ 线性模型 $+ \sigma(z)$ 激活函数

2. 深度学习基本步骤

(1) 搭建神经网络

搭建神经网络需要确定神经网络类型、隐藏层的神经元个数、神经网络的

Figure 1: 步骤图（来自李宏毅）

层数、初始化参数、选择合适的激励函数和学习率（learning rate）等等。

(2) 调整参数（learning）

通过梯度下降法，找到使得 Cost Function（即 $\sum_{i=1}^{m} L_i$）最小的参数；这里需要使用 BP 算法算出每个参数的梯度值。

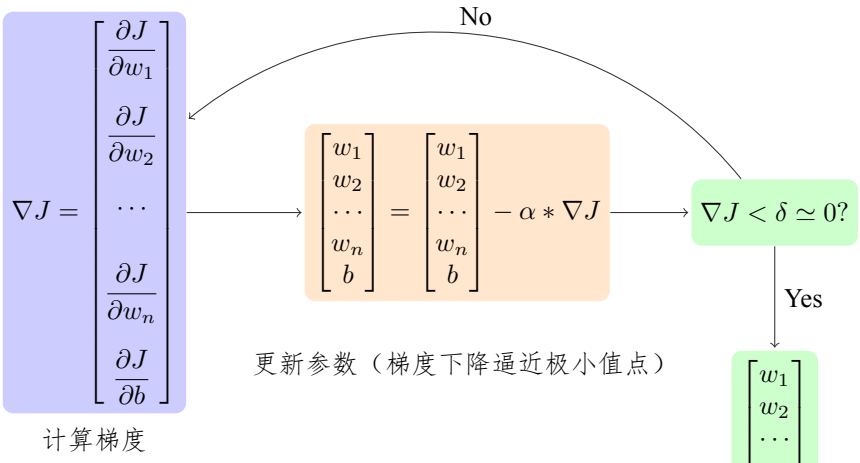
二、技巧

1. Backpropagation

(1) 梯度下降法

梯度下降法 **(gradient descent)** 是一个最优化算法，通常也称为最速下降法。梯度下降法的计算过程就是沿梯度下降的方向求解极小值（也可以沿梯度上升方向求解极大值）。常用于机器学习和人工智能当中用来递归性地逼近最小偏差模型。

若定义 Cost Function 为：$J(w,b) = \sum_{i=1}^{m} L(y_i, \hat{y}_i)$，则通过梯度下降学习参数的过程为：

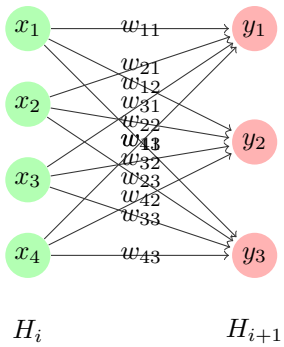


梯度下降法简易流程图

(2) 反向传播

反向传播，顾名思义就是沿着与神经网络前进相反的方向进行数据传播，可以理解为神经元的反馈；BP 可用于计算神经网络中每层参数的梯度值，利用的是求导链式法则；

可以这么理解，假设神经网络中相邻的两层 H_i, H_{i+1}，H_i 作为 H_{i+1} 的 input，那么相对而言 output 就是 H_{i+1}，因此两者间有线性关系；



因此，计算相邻的参数梯度即为其对应的参数 w_{ij}，如：$\frac{\partial y_3}{\partial x_4} = w_{43}$；然后依据链式法则，即可算出从最后的 Cost Function 到任意一层的任一参数的偏微分 $\frac{\partial J}{\partial w_{ij}}$，所谓的反向传播在这里就是从最后一层向前进行连续的链式求导求出各参数的偏微分！

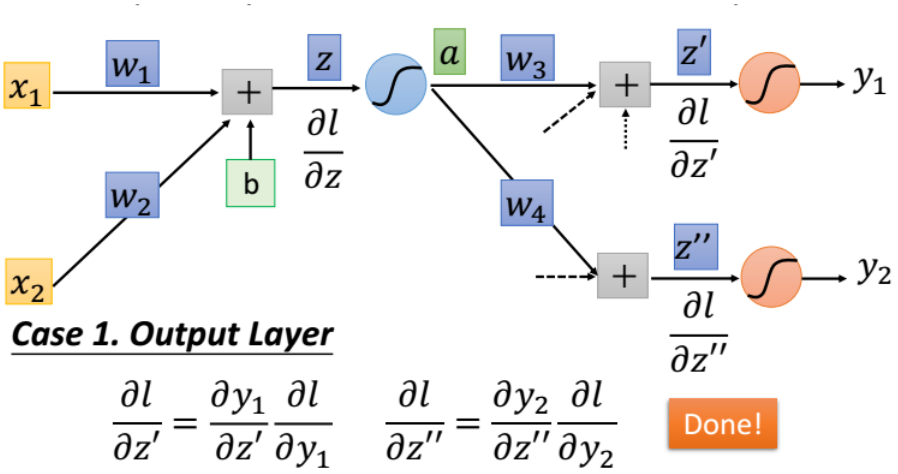


Figure 2: 反向链式求导（来自李宏毅）

如果 H_i, H_{i+1} 并非直接的相邻关系，中间多了一个激励函数 $\sigma(z)$ 作为输出的话，那么根据链式求导法则再多求一次就好了！（见上图）

2. 激励函数（activation function）

(1) Sigmoid 函数

Sigmoid 定义为：$\sigma(z) = \frac{1}{1+e^{-z}}$，所以可以看出其输出值在 [0,1] 之间，因此比较适合来表示概率！但是其有一个比较明显的缺点，从其函数图像可以看出两端的值比较『缓和』，因此梯度接近于 0，不利于 learning；此外，其图像并非中心对称，而且 e 的指数幂计算起来比较耗时！

(2) ReLU

Rectified Linear Unit，简称为 ReLU，是目前最常用的激励函数之一；其函

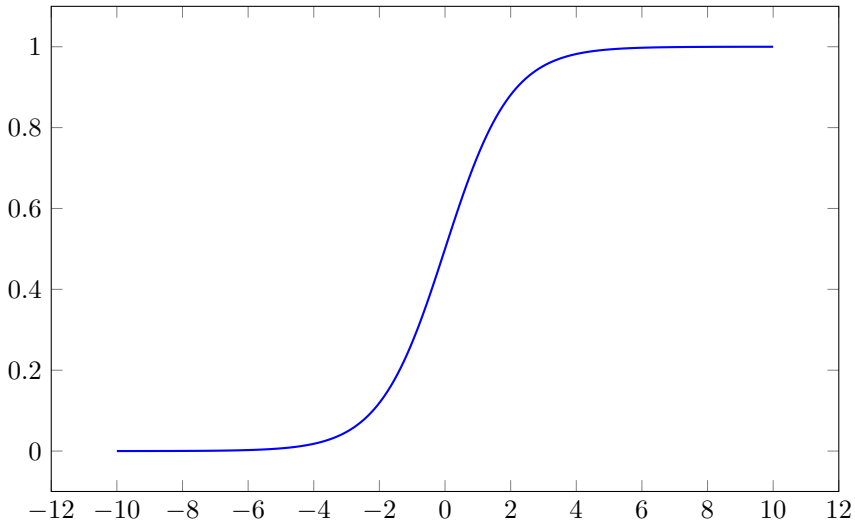


Figure 3: Sigmoid 函数-$\sigma(z) = \frac{1}{1+e^{-z}}$

数形式为 $f(x) = max(0, x)$，因此可以看出其左半部分（$x < 0$）是一个常数 0，右半部分是一个一元线性函数 $y = x$；

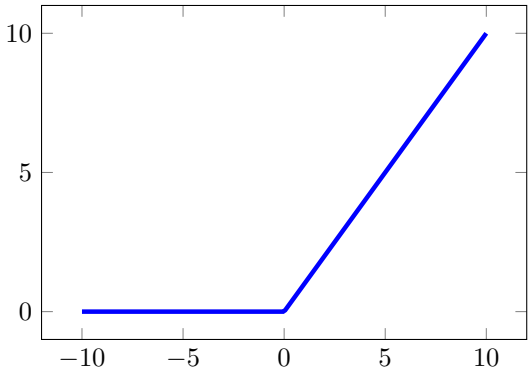


Figure 4: ReLU-$f(x) = max(0, x)$

这样设计的好处有：（1）便于计算（线性）；（2）解决梯度消失（Vanishing gradient）的问题；（3）除去负值？

(3) Leaky ReLU

为了弥补 ReLU 左半部分的不足（小于 0 会使神经元被『抛弃』，即不影响后续神经元），Leaky ReLU 在左半部分做了小小的修正，在左半部分改为一个斜率很小的线性函数，如：$max(0.1x, x)$；

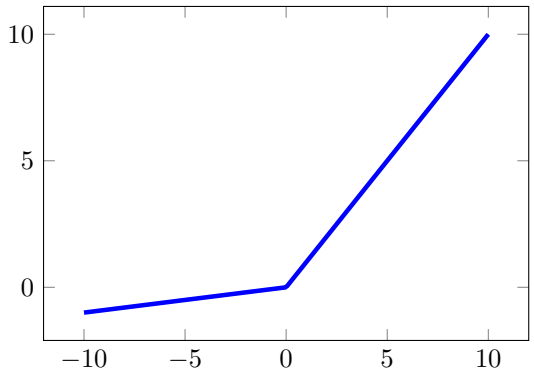


Figure 5: Leaky ReLU-$f(x) = max(0.1x, x)$

(4) ELU

Exponential Linear Units (ELU)，跟 Leaky ReLU 的修改不同，ELU 把左半部分修改为一个非线性的函数——即 $f(x) = \begin{cases} \alpha * (e^x - 1), & x < 0 \\ x, & x \geq 0 \end{cases}$；

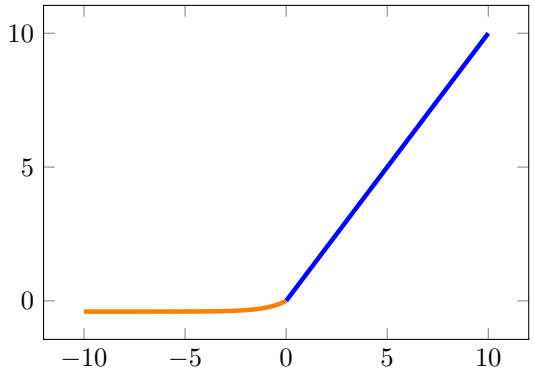


Figure 6: ELU-$\alpha = 0.4$

(5) maxout

maxout，顾名思义就是输出最大值，一般将 k 个神经元的输出值进行比较选出其中最大的输出值，即：$max(w_1x+b_1,\cdots,w_kx+b_k)$；关键是在于比较的神经元的数量 k 以及它们之间的位置如何选择，一般是选取相邻的神经元？

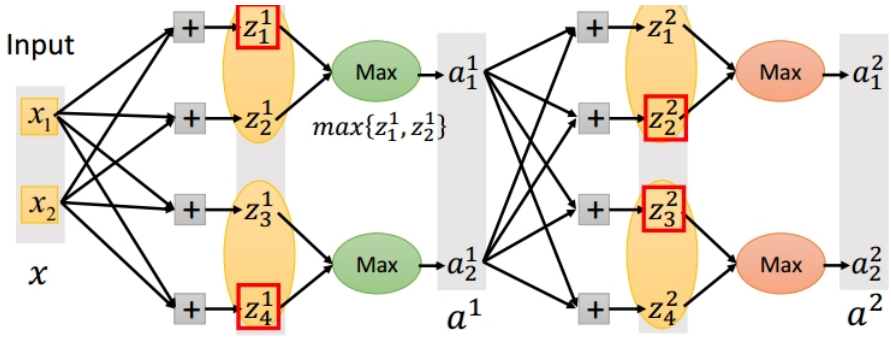


Figure 7: maxout 训练过程（来自李宏毅）

3. Dropout

Dropout 的大致思想是在 learning 过程中对每一层神经层中神经元以概率 p 进行『舍弃』(注意：这里的舍弃并非真正舍去，而是仅仅在这一次迭代更新中将这些神经元不加入计算行列而已，即每次迭代更新参数过程中都进行一次随机 dropout，因此每次『舍去』的神经元都不一样)；至于为何要这么做，大概是因为以下几个原因：

1. 减少计算量：因为每次 learning 的过程中，减去了很多神经元，所以神经网络的复杂度大大降低，计算量肯定是减少的；(精度能保证吗?)

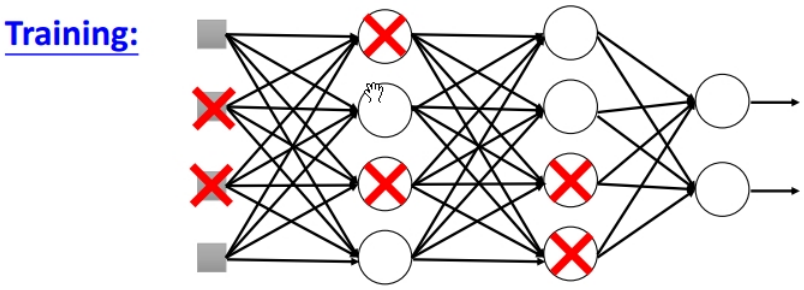


Figure 8: dropout 训练过程（来自李宏毅）

2. 集成（ensemble）：因为每次的迭代过程『舍去』的神经元都不一样，所以每次迭代的神经网络的构造肯定是不同的，因此就相当于对多个神经网络同时进行 learning，也就是所谓的 ensemble learning（?）；

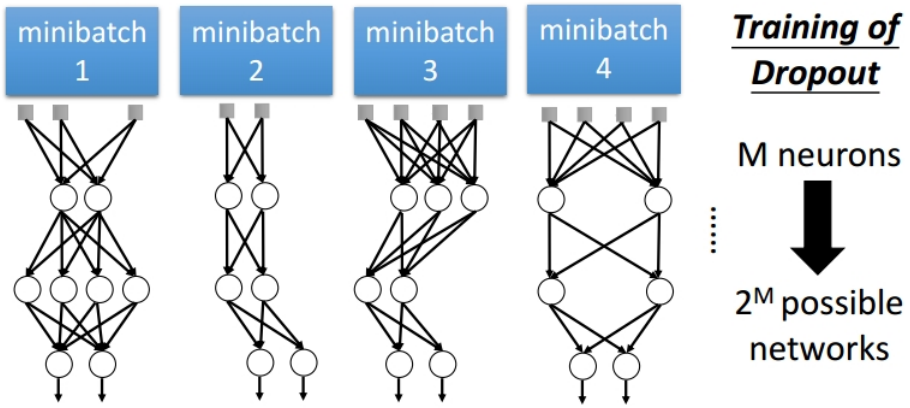


Figure 9: 集成化（来自李宏毅）

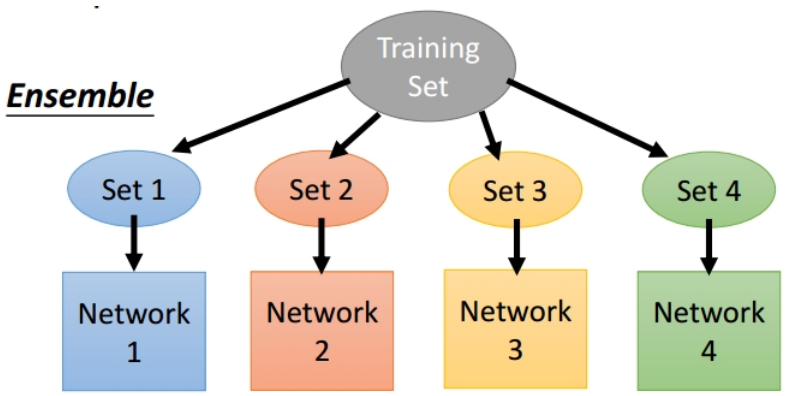


Figure 10: ensemble learning 示意图（来自李宏毅）

3. 防止过拟合：

需要注意的是，在使用了 Dropout 后，在之后的 testing 过程中需对所有的参数乘上 $1-p$！(为何？)

4. Regularization（正则化）

正则化就是通过调整代价函数（Cost Function）来防止参数产生过拟合的现象，主要方式为：$J'(w) = J(w) + \lambda g(w)$，其中 $g(w)$ 是关于所有参数的约束函数，λ 为一个常数；常用的约束函数有 $L^1(\|w\|_1)$ 和 $L^2(\|w\|_2)$ 等；因此在梯度下降过程中要使用 $\frac{\partial J'}{\partial w}$ 来代替原来的 $\frac{\partial J}{\partial w}$；

5. mini-batch

所谓的 batch，就是一次处理的数据的规模，用于学习过程中对损失函数的更新方式；mini-batch 就是将所有数据划分为若干个 batch，然后每次用一个 batch 的数据更新损失函数，然后用梯度下降更新所有参数；

三、卷积神经网络

1. 卷积

(1) 概念

以前只知道图像中有个卷积的运算，而且很常用，后来没想到在考研复习的时候发现概率论也有卷积运算，我想图像中的离散卷积运算应该是从连续的卷积运算演化过来的吧。连续卷积运算定义为：

$$s(t) = \int x(a)w(t-a)da = (x * w)(t)$$

一般使用 $*$ 符号作为卷积符号。通常，w 被称作核函数（kernel function）。

单通道的图像数据可以看做一个二维的数据，图像中卷积的操作形式如下：

$$s(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i-m, j-n) * K(m,n)$$

与卷积很相近的一种运算叫做相关（correlation）运算（$\otimes$），可以这么定义：

$$s(i,j) = (I \otimes K)(i,j) = \sum_m \sum_n I(i+m, j+n) \otimes K(m,n)$$

但是在实际运用过程中，这两种操作都叫做『卷积』?（deeplearning book 一书是这么说的）

(2) 作用

为何要在神经网络中使用卷积运算？大概是『参数共享』使每层的参数相比全连接（full-connect）时大大减少，每层的参数仅有核函数的大小。但是很明显，并不是所有的神经网络都能使用卷积操作。对于图像而言，每个核函数就像是一个局部特征检测器，检测一定的与位置无关的特征。

2. 池化（pooling）

所谓的池化就是对输入层相邻的一定数据进行筛选，最常用的就是最大池化（max pooling）；所谓的最大池化就是选择输入层一定邻域内最大的数据作为输出（有点像 maxout?）。

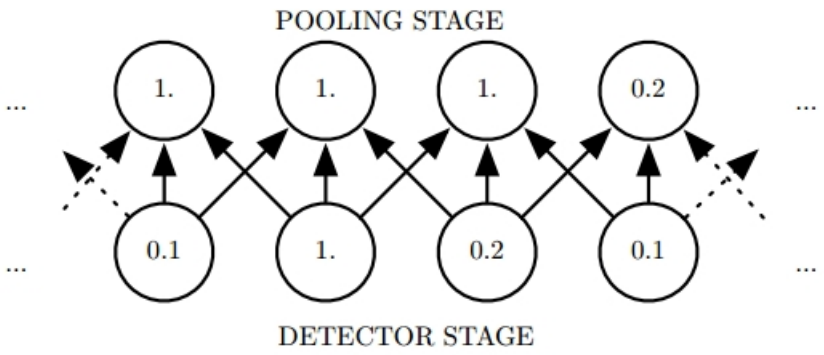


Figure 11: max pooling 示意图（来自 dl-book）

池化后的输出具有近似的平移不变性，即少量的位移不会影响大多数的输出结果，这有利于局部特征的检测。