神经网络基础

目标

- 学完本课程后，您将能够：
  - 了解什么是人工神经网络与深度前馈网络
  - 掌握如何训练神经网络
  - 掌握梯度下降的方法以及反向传播的概念
  - 了解神经网络架构设计的因素

目录

数据集

- **特征（Feature）**：特征是用来描述机器学习系统处理的对象或事件的特性。
- **样本（Sample）**：样本是指我们从某些希望机器学习系统处理的对象或事件中收集到的已经**量化**的特征的集合。
- **数据集（Dataset）**：数据集是指很多样本组成的集合。有时我们也将样本称为数据集中的数据点（Data Point）。
- 大部分机器学习算法可以被理解为在**数据集**上获取经验。

学习方法分类 (1)

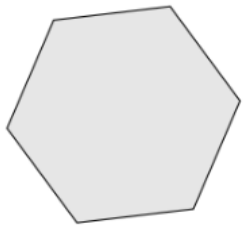
- **监督学习算法（Supervised Learning Algorithm）**：训练含有很多特征的数据集，不过数据集中的样本都有一个标签（Label）或目标（Target）。
- **无监督学习算法（Unsupervised Learning Algorithm）**：训练含有很多特征的数据集，然后学习出这个数据集上有用的结构性质。
- **监督**的理解：监督学习中教员或者老师提供标签给机器学习系统，指导其应该做什么。在无监督学习中，没有教员或者老师，算法必须学会在没有指导的情况下理解数据。

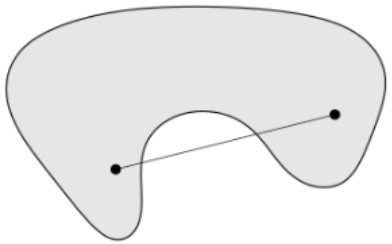
学习方法分类 (2)

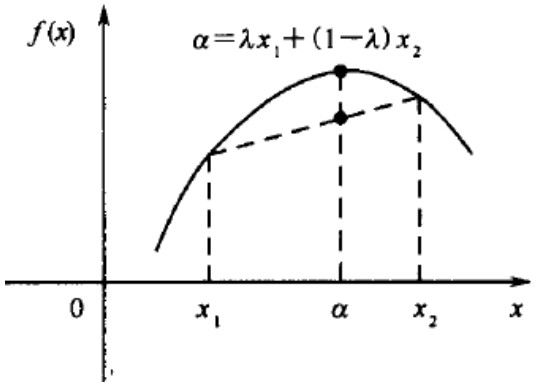
- 监督/非监督并非是严格分类的。很多机器学习技术会组合使用。
- **半监督学习（Semi-Supervised Learning）**：监督学习与无监督学习相结合的一种学习方法。半监督学习使用大量的未标记数据，以及同时使用标记数据作为学习的数据集。
- **强化学习（Reinforcement Learning）**：并不是训练于一个固定的数据集上。算法会和环境进行交互，所以学习系统和它的训练过程会有反馈回路。

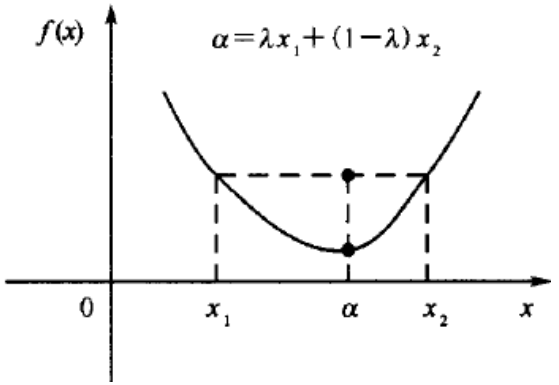
凹凸函数

- **凸集**：若集合中任意两点连线上的点都在该集合中，则称该集合为凸集。
- **凹集**：非凸集。
- **凸函数**：简单理解为在函数图像上任取两点，如果函数图像在这两点之间的部分总在连接着两点的线段上方且定义域为凸集的函数，为凸函数。
- **凹函数**：简单理解为在函数图像上任取两点，如果函数图像在这两点之间的部分总在连接这两点的线段的下方，则为凹函数。



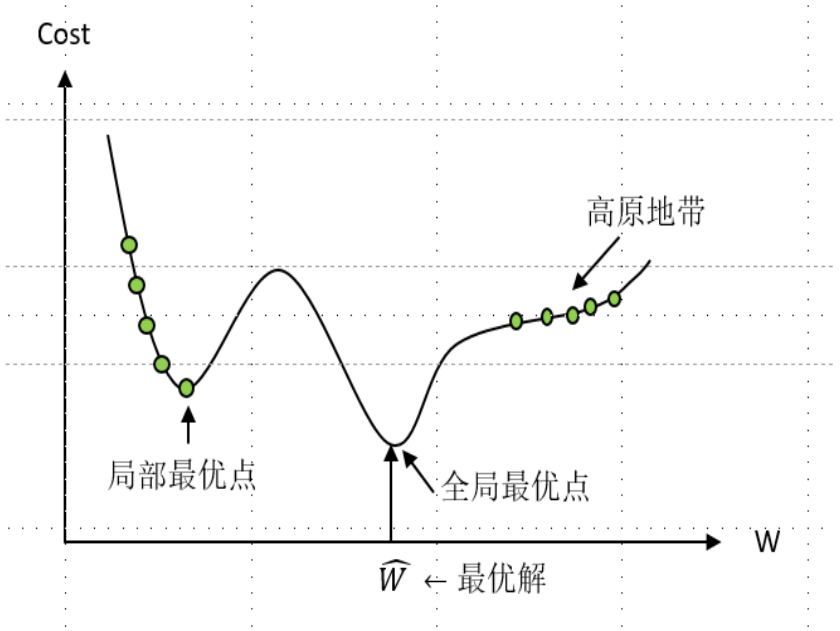






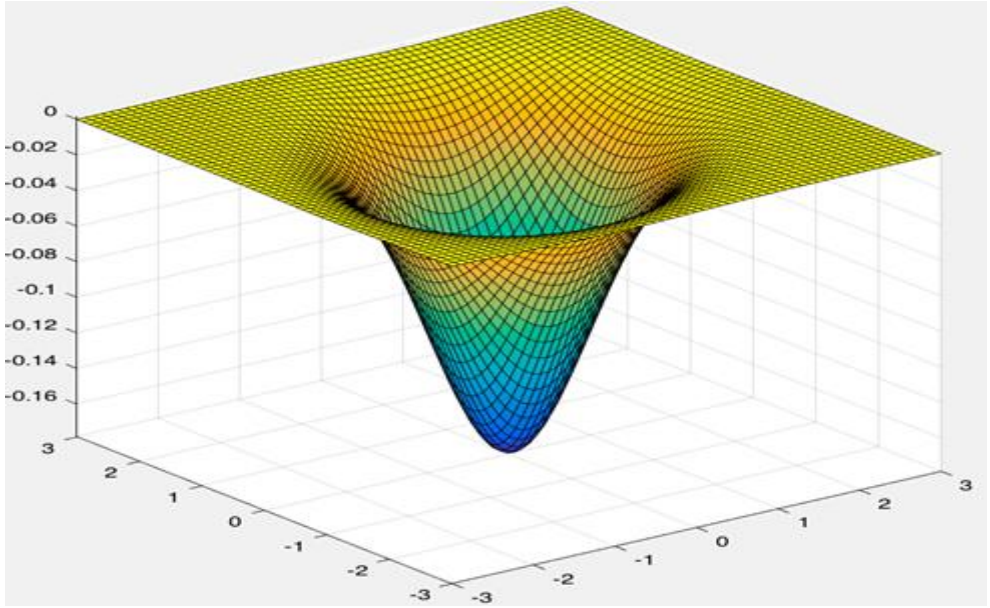
非凸函数

- 在一些最优化问题中很可能遇到复杂且不易求解的目标函数如高阶函数。如下图所示，其可能有多个局部极值点，而我们想找到的只有一个全局最优点。



凸函数

- 对于一个复杂函数，找其全局最优点无疑比较困难，在实际工程中可能很难求解，这时我们想将一个复杂的非凸函数转化为一个如下图所示的凸函数，这样其局部最优便是全局最优。凸函数所对应的优化便是凸优化。

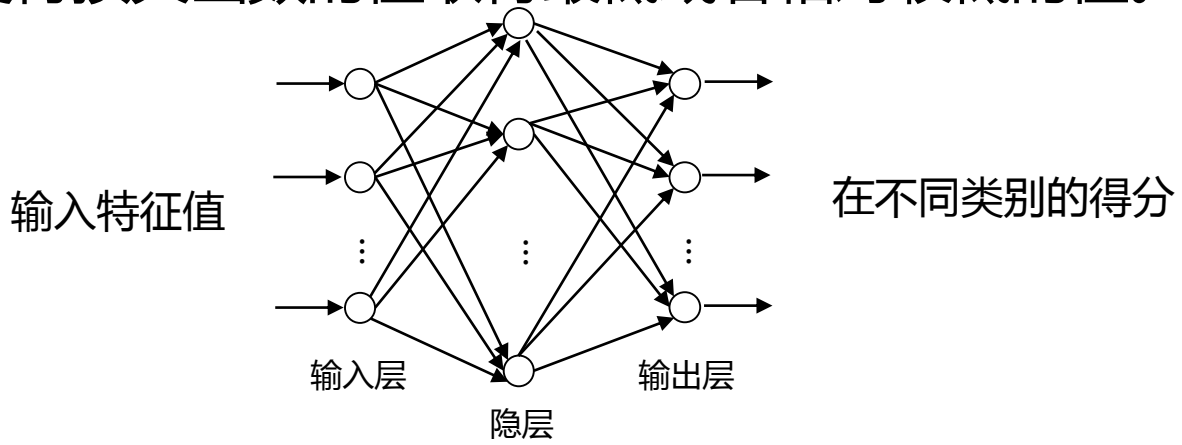


凸优化

- 凸优化是机器学习的一个根本性问题，在工程中很多问题可以抽象化为一个凸问题，很多非凸问题可以通过一定的手段或方法转化为一个凸问题，一旦转化为一个凸问题，理论上，这个问题便可以得到解决。
- **凸优化的定义**：只有满足以下两个条件才需要我们做凸优化的处理：
  - 条件一： 约束条件为凸集。
  - 条件二： 目标函数为凸函数。
- **非凸优化问题转化为凸优化问题的方法**：
  - 修改目标函数，使之转化为凸函数。
  - 抛弃一些约束条件，使新的可行域为凸集并且包含原可行域。

损失函数

- 神经网络中使用的代价函数被称作损失函数（Loss Function）。
  - 损失函数衡量了评分函数的预测与真实样本标签的吻合度。
  - Loss的值都会设置为和吻合程度负相关。如果算法公式是正相关，定义损失函数时候加负号，调整为负相关。
- 神经网络在数据集上学习，就是通过对训练数据集的学习，调整神经网络中每个神经元的参数，使得损失函数的值取得最低或者相对较低的值。



交叉熵损失函数

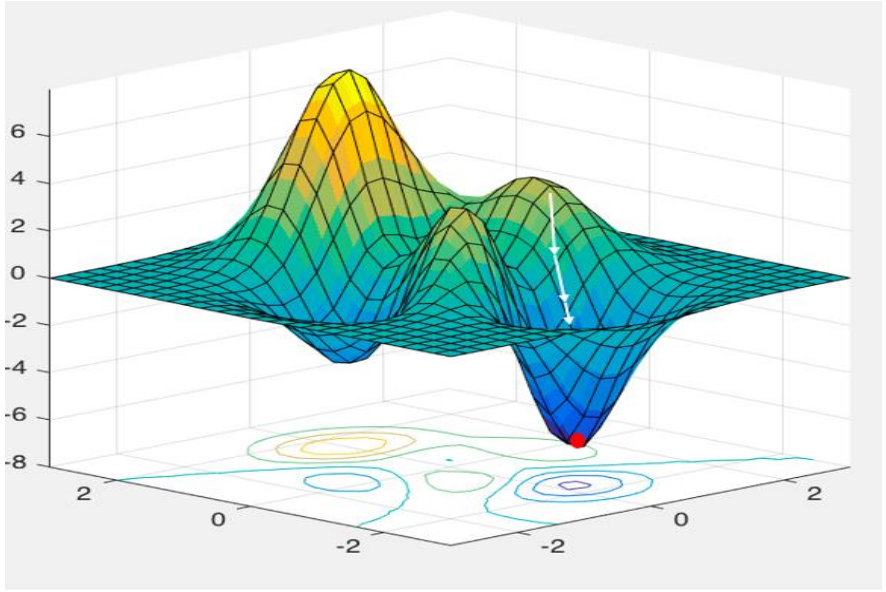
- 交叉熵损失（Cross Entropy / Softmax Loss）：对第i个样本x_i，神经网络W预测其为第k类的得分我们记作$f(x_i, W)_k$，第i个样本x_i数据的真实标签类别是y_i。损失函数：

$$Loss_i = -\sum_k p_k \log(q_k) = -\sum_k p_k \log\left(\frac{e^{f_k}}{\sum_j e^{f_j}}\right)$$

其中p_k是x_i属于k类的概率，$p_{k=y_i} = 1$，$p_{k\neq y_i} = 0$。

梯度下降法 (1)

- 蒙着眼睛下山 - 梯度下降

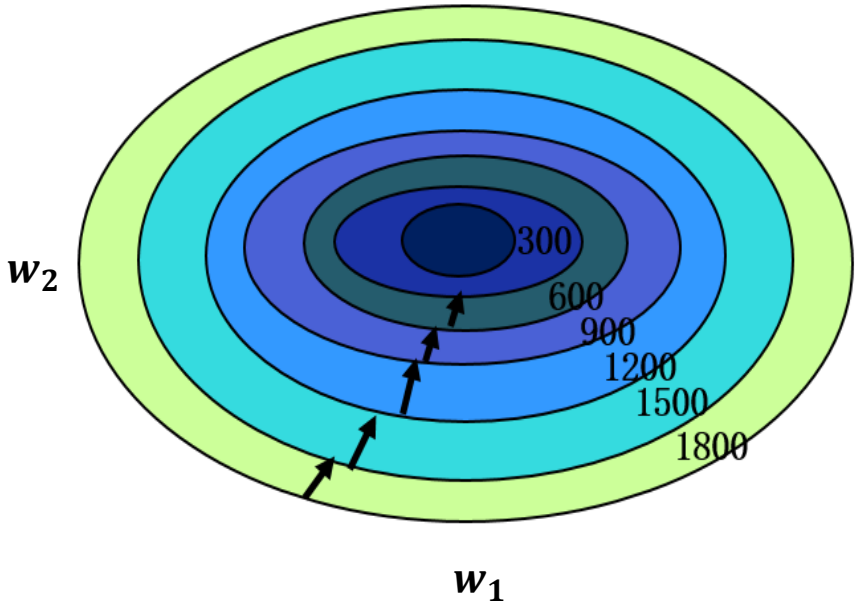


梯度下降法 (2)

- 学习率(Learning Rate,LR)，根据误差梯度调整权重数值的系数，通常记作η。

$$w^+ = w - \eta * \frac{\partial Loss}{\partial w}$$

- 通过学习率和梯度值更新所有参数值使得网络的损失函数值降低。



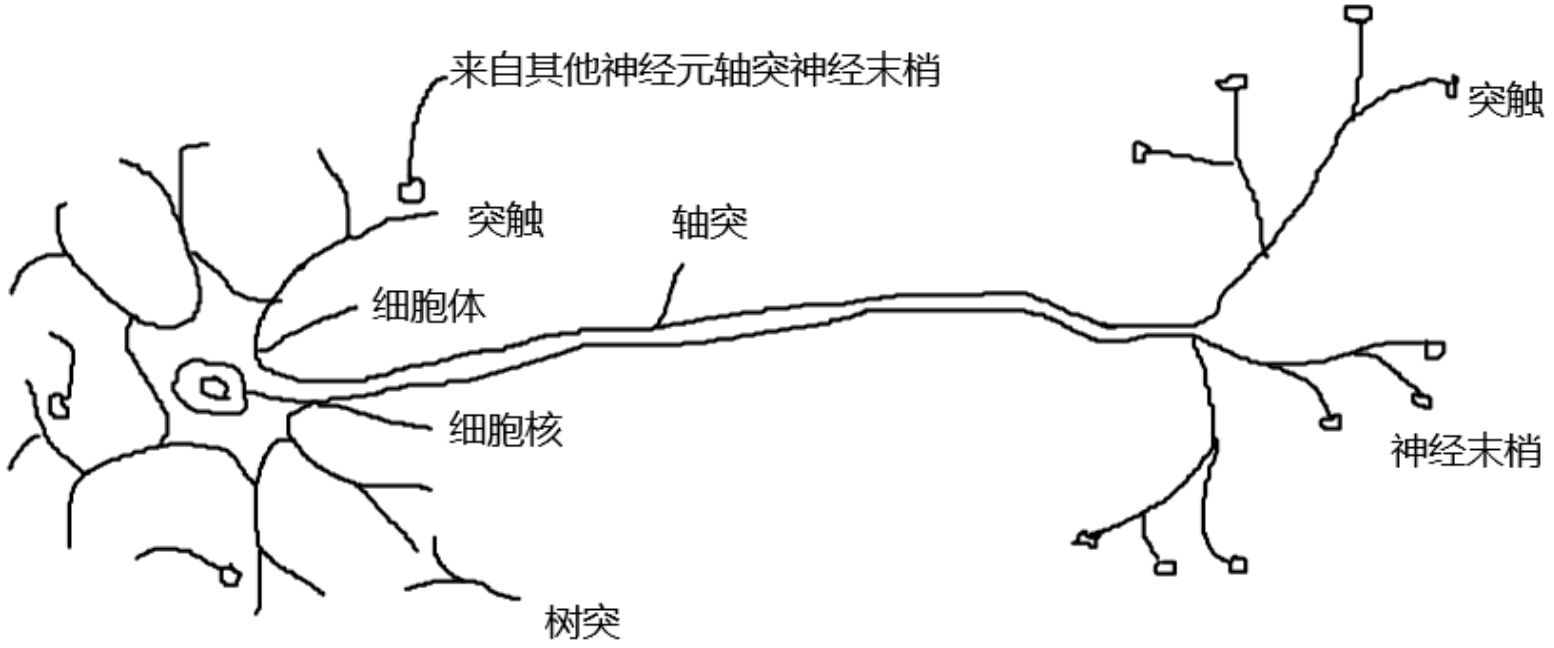
梯度下降法 (3)

- 梯度下降常用的方法有三种：
  - **批量梯度下降（BGD）**：每次更新使用所有的训练数据，最小化损失函数，如果只有一个极小值，那么批量梯度下降是考虑了训练集所有数据，**但如果样本数量过多，更新速度会很慢。**
  - **随机梯度下降（SGD）**：每次更新的时候只考虑了一个样本点，这样会大大加快训练速度，但是函数不一定是朝着极小值方向更新，且SGD对噪声也更加敏感。
  - **小批量梯度下降（MBGD）**：MBGD解决了批量梯度下降法的训练速度慢问题，以及随机梯度下降对噪声敏感的问题。

目录

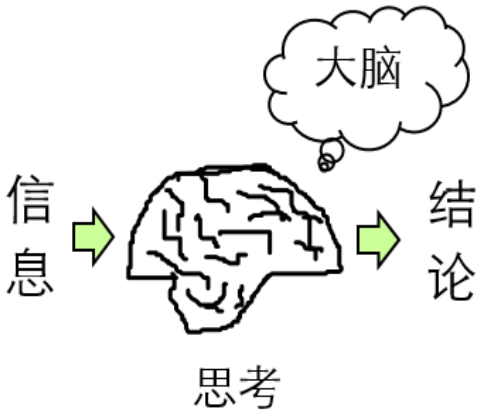
生物神经网络

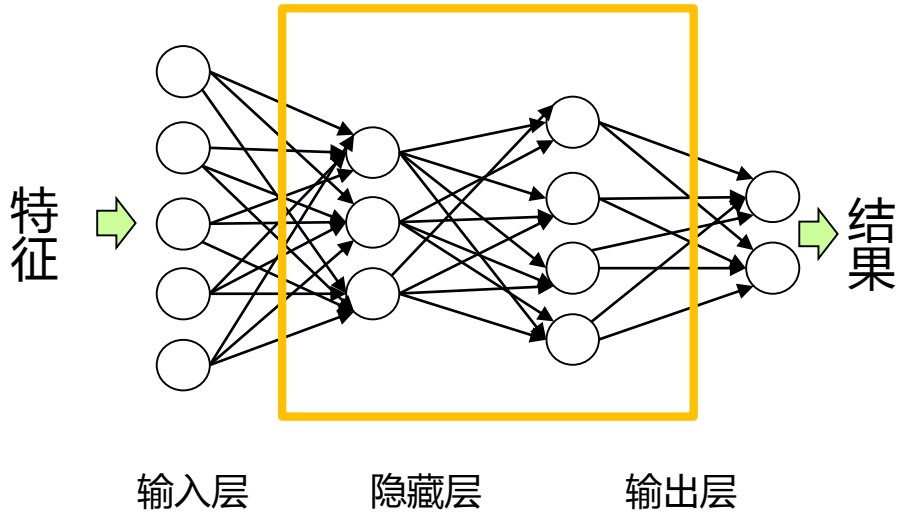
- **生物神经网络**（Biological Neural Networks）：一般指生物的大脑神经元，细胞，触点等组成的网络，用于产生生物的意识，帮助生物进行思考和行动。生物神经细胞功能比较简单，需要通过很多神经元一起协作完成复杂功能，通过一定的连接方式或信息传递方式进行协作的神经元可以看做是一个网络，就是神经网络。



人工神经网络

- 人工神经网络是一种旨在模仿人脑结构及其功能的信息处理系统。
- **人工神经网络，简称神经网络（Artificial Neural Network，ANN）**：是由人工神经元互连组成的网络，它是从微观结构和功能上对人脑的抽象、简化，是模拟人类智能的一条重要途径，反映了人脑功能的若干基本特征，如并行信息处理、学习、联想、模式分类、记忆等。

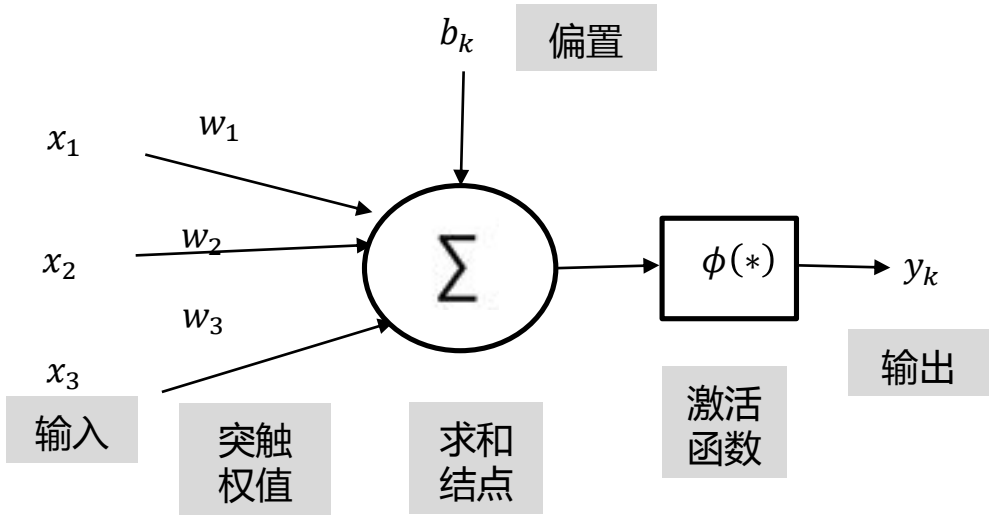




目录

神经元 (1)

- 神经网络的神经元单元由线性函数和激活函数构成。



- 线性函数：对于输入n维的特征向量X，计算方式为：

$f(X,W,b) = WX + b = \sum_n w_i x_i + b = [\mathrm{W};\mathrm{b}][\mathrm{X};1]$，

- 其中，$X = [x_0, x_1, \ldots, x_n]^T$为输入向量，$W = [\omega_0, \omega_1, \ldots, \omega_n]^T$为权重，$b$为偏置。

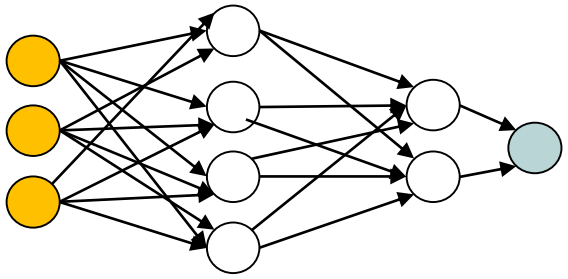
- 激活函数：$\phi(x) = sign(net) = \begin{cases} 1, net > 0, \\ -1, otherwise. \end{cases}$

神经元 (2)

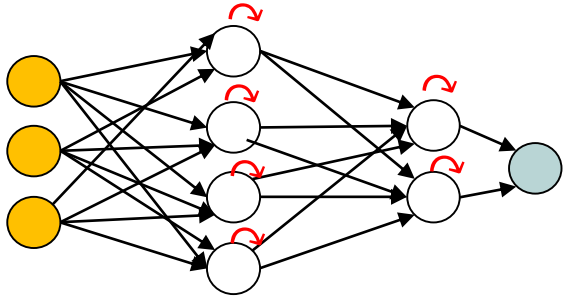
- 人工神经网络主要由大量的神经元以及它们之间的有向连接构成。因此考虑三方面：
  - 神经网络的**拓扑结构**：不同神经元之间的连接关系。
  - 神经元的**激活规则**：主要是指神经元输入到输出之间的映射关系，一般为非线性函数。
  - **学习算法**：通过训练数据来学习神经网络的参数。

神经网络的拓扑结构

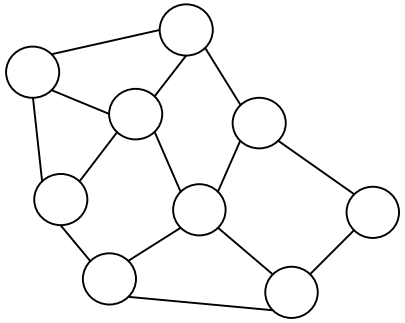
- 前馈网络
- 反馈网络
- 图网络



（a）前馈网络



（b）反馈网络



（c）图网络

目录

感知器 (算法说明)

- 1957年，美国学者Frank Rossenblatt（弗兰克·罗森布拉特）提出了感知器（感知机）算法。感知器用来接收多个信号，输出一个信号（由多个神经元组成）。每个输入信号具有一定的权重，计算多个输入信号的值与权重的乘积和，根据该结果（和）与指定的阈值进行比较，来决定该神经元是否被激活。

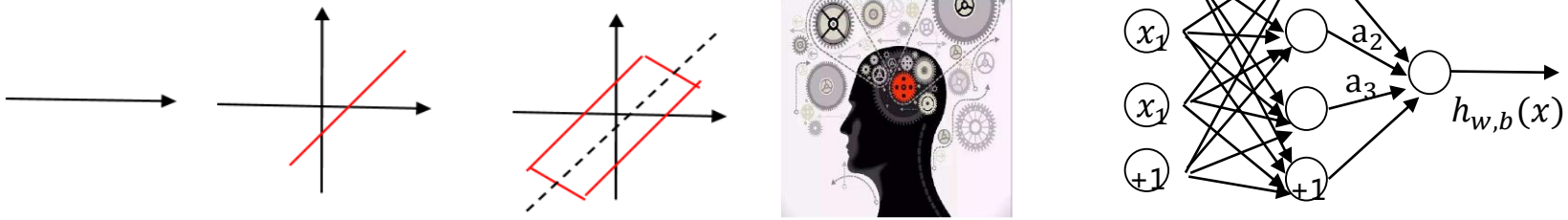
感知器 (算法公式)

- 根据感知器的定义可知，感知器可以实现二分类的任务。感知器的计算方式如下

  $Z = W_1X_1 + W_2X_2 + ... + W_nX_n = \sum_{n_i=1} W_iX_i = WX$

- 其中，Z称为净输入（Net Input）。然而，这样的计算结果是一个连续的值，我们需要将结果转换为离散的分类值，因此，这里，我们使用一个转换函数，该函数称为激励函数(激活函数)。

- 激活函数： $sign(\boldsymbol{Z}) = \begin{cases} 1 & Z \geq \theta \\ -1 & Z \leq \theta \end{cases}$ ，这里的θ就是阈值。



感知器 (损失函数)

- 感知机的训练过程其实就是求解W 和 b 的过程。 正确的W和b 所构成的超平面 $WX = 0$ 可以将两类数据点分割在这个平面的两侧。为了找出这样的超平面需定义目标函数。
- 目标函数使用误分类点到超平面s的总距离。即，简单的点到直线的距离。
- 损失函数：即期望使误分类的所有样本，到超平面的距离之和最小。
- 感知机的损失函数为：

$$L(w,b) = -\sum_{X_i \varepsilon M} y_i(w * x_0 + b)$$

感知器 (梯度下降优化)

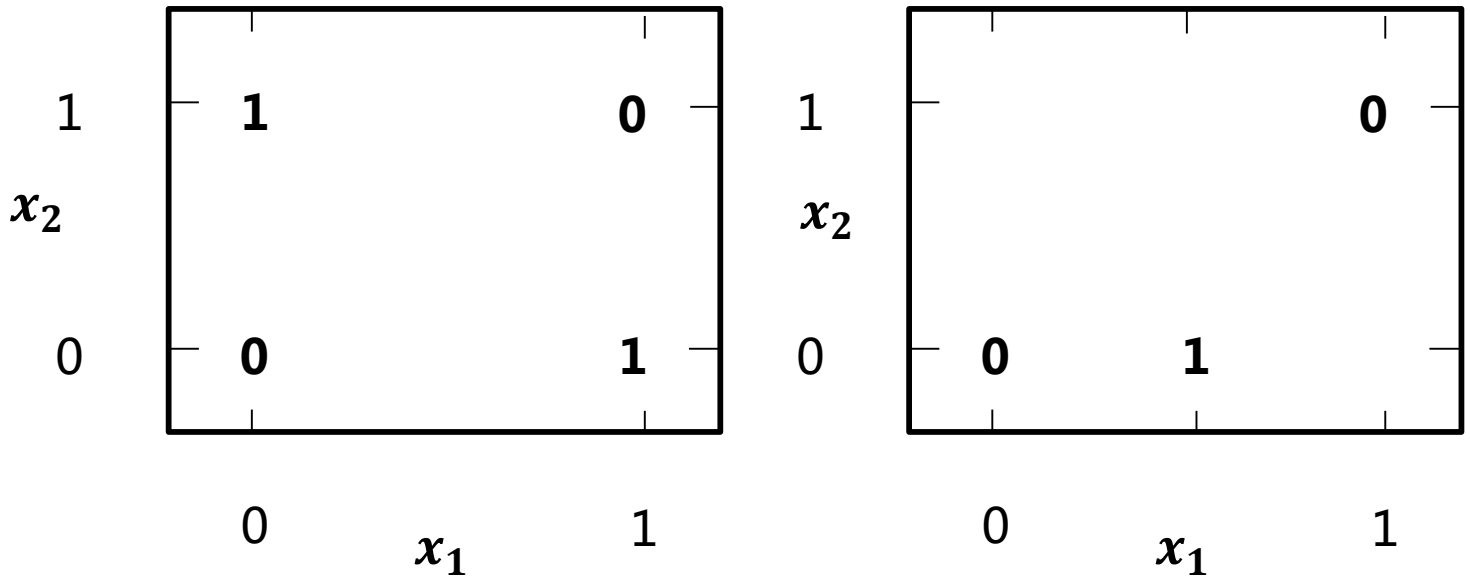
- 损失函数中的M是所有误分类的点的集合。这是一个凸函数，可以用梯度下降法或者拟牛顿法来解决，常用的是梯度下降法。
- 但是用普通的基于所有样本的梯度和的均值的批量梯度下降法（BGD）是行不通的，原因在于我们的损失函数里面有限定，只有误分类的M集合里面的样本才能参与损失函数的优化。所以我们不能用最普通的批量梯度下降,只能采用随机梯度下降（SGD）或者小批量梯度下降（MBGD）。
- 感知机选择的是随机梯度下降，这意味着我们每次仅仅需要使用一个误分类点来更新梯度。
- 更新原则：如果预测准确，则权重不进行更新，否则，增加权重，使其更趋向于正确的类别。

感知器 (实现步骤)

- 选择θ向量的初值和步长α的初值。可以将θ向量置为0向量，步长设置为1。要注意的是，由于感知机的解不唯一，使用的这两个初值会影响θ向量的最终迭代结果。
- 在训练数据集中选取点$(x(i), y(i))$，如果点这个点应该满足：$y(i)[\theta \cdot x(i)] \leq 0$，则更新参数。否则继续遍历数据集寻找误分类点。
- 对θ向量进行一次随机梯度下降的迭代：$\theta = \theta + \alpha \cdot y(i) \cdot x(i)$
- 检查训练集里是否还有误分类的点，如果没有，算法结束，此时的θ向量即为最终结果。如果有，继续第2步。
- 说明：
  - 如果两个类别线性可分，则感知器一定会收敛。因此，初始化的权值多少不会影响收敛。
  - 如果两个类别线性不可分，则感知器一定不会收敛。

XOR问题

- 异或问题是指对于给定的两个二进制输入，通过异或逻辑门得到一个预测输出，当输入不相等时输出1，否则输出0。
- 单层感知机无法解决XOR问题。

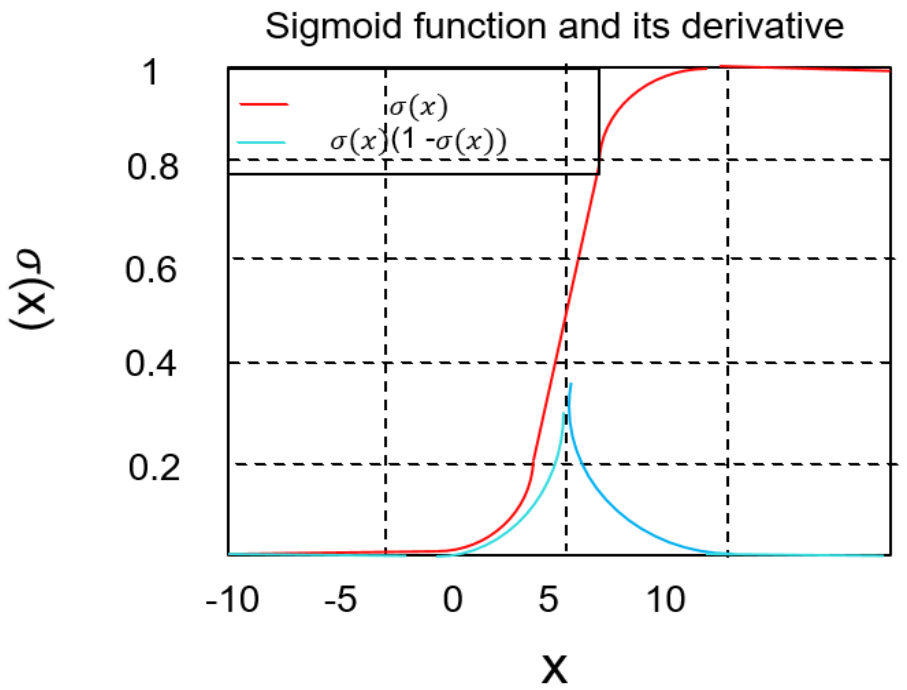


目录

激活函数 - Sigmoid函数

- Sigmoid函数：

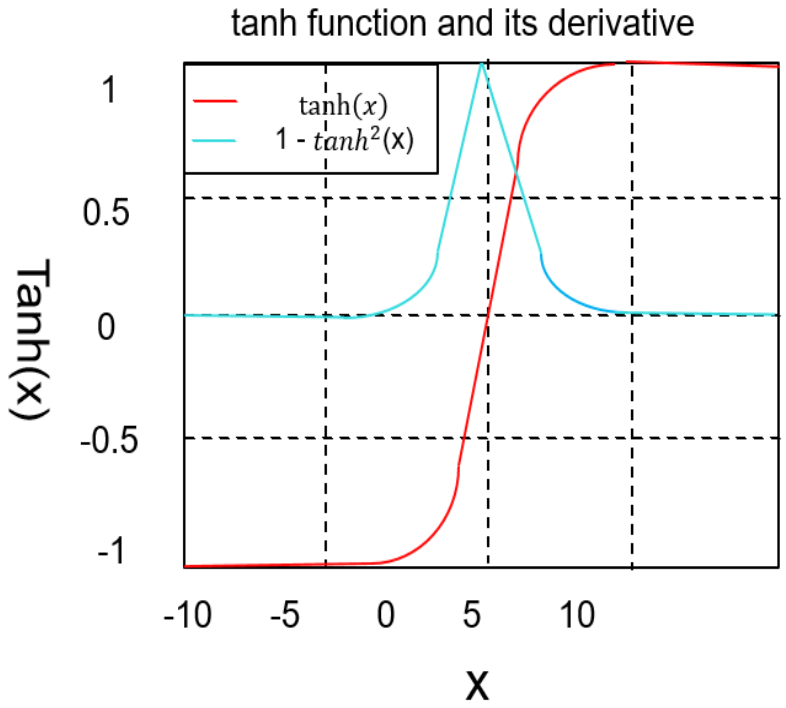
$$f(x) = \frac{1}{1 + e^{-x}}$$



激活函数 - tanh函数

- tanh函数:

$$\tanh x = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$



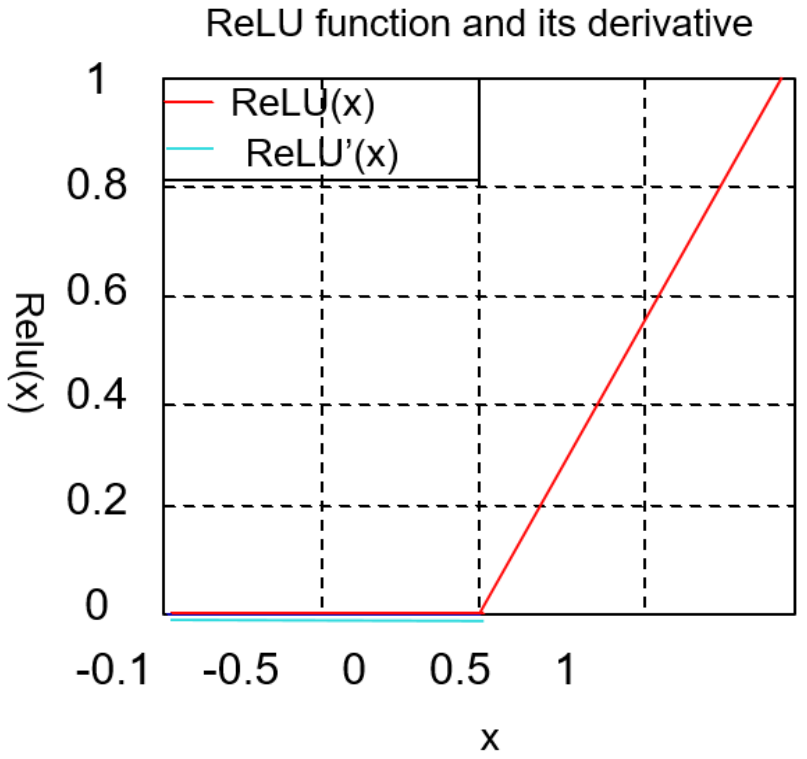
激活函数 - ReLU函数 (1)

- ReLU(Rectified Linear Unit)，修正线性单元。定义为：

$$ReLU(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases} = \max(0, x)$$

- ReLU的导数为：

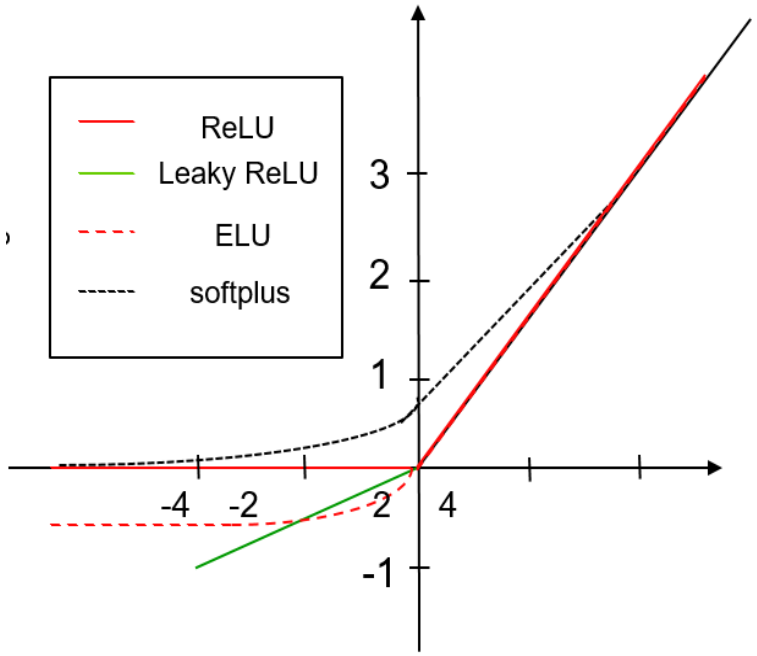
$$ReLU(x) = \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases}$$



激活函数 - ReLU函数 (2)

- 整流线性单元的扩展基于当 $z_i < 0$ 时使用一个非零的斜率 $\alpha_i h_i = g(z, \alpha)_i = \max(0, z_i) + a_i \min(0, z_i)$。
  - 绝对值整流：$g(z) = |z|$.
  - 渗漏整流线性单元（Leaky ReLU）：所有负值赋予一个非斜率。
  - ELU（指数线性单元），Softplus函数:

$$y = \log(1 + e^x)$$



激活函数 - Softmax函数

- Softmax 函数：

$$\sigma(\mathrm{z})_j = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

  - Softmax函数的功能就是将一个K 维的任意实数向量映射成另一个K维的实数向量，其中向量中的每个元素取值都介于（0，1）之间。新的向量所有维度模长之和为1。
  - Softmax函数经常用作多分类任务的输出层。

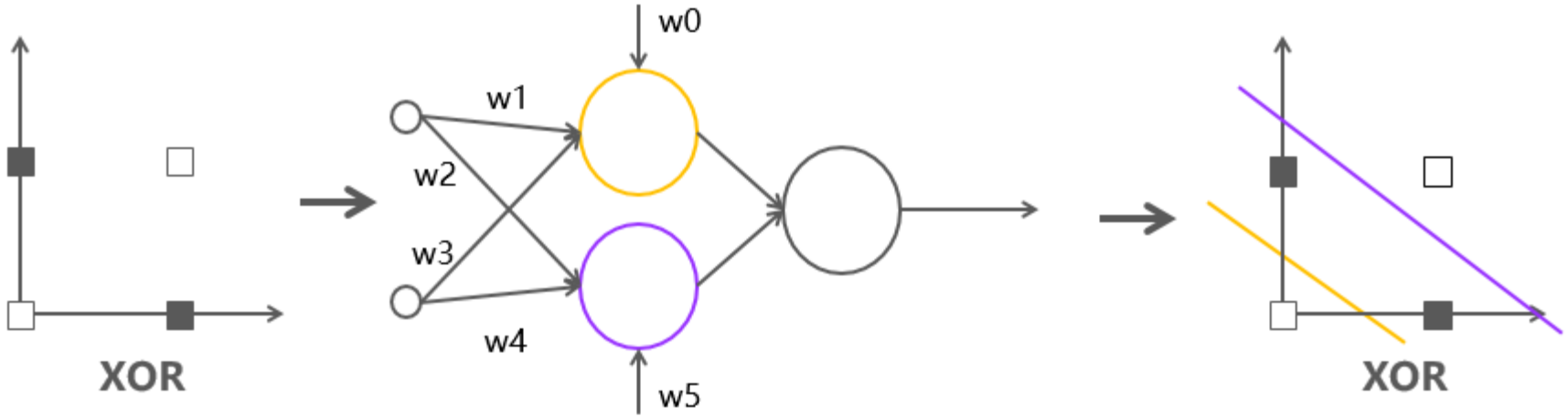
激活函数设计需考虑的因素

- **非线性**：当激活函数是非线性的，一个两层神经网络可以证明是一个通用函数近似值，如果失去了非线性，整个网络就相当于一个单层的线性模型。
- **连续可微性**：这个属性对基于梯度优化方法是必要的，如果选择了一些具有局部不可微的函数，则需要强行定义此处的导数。
- **有界性**：如果激活函数有界的，基于梯度的训练方法往往更稳定；如果是无界的，训练通常更有效率，但是训练容易发散，此时可以适当减小学习率。
- **单调性**：如果激活函数是单调的，与单层模型相关的损失函数是凸的。
- **平滑性**：有单调导数的平滑函数已经被证明在某些情况下泛化效果更好。

目录

全连接人工神经网络

- 单个感知器的表达能力有限，它只能表达线性决策面（超平面）。如果我们把众多的感知器互联起来，就像人的大脑一样，我们就可以表达种类繁多的非线性曲面。



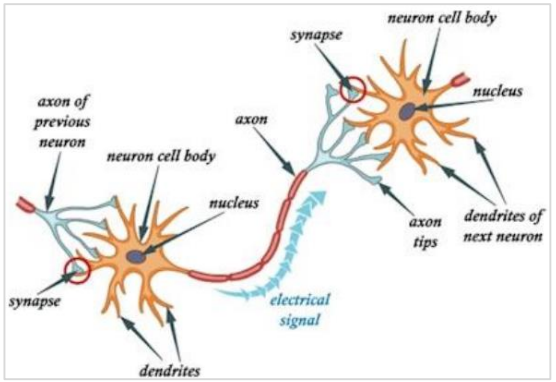
神经网络

- 神经网络在感知器的模型上做了以下三点的扩张：
  - 加入隐藏层，隐藏层可以有多层，增强模型的表达能力。
  - 输出层的神经元也可以不止一个，可以有多个输出，模型就可以灵活的应用于分类回归，以及其他的机器学习领域比如降维和聚类等。
  - 对激活函数做扩展，包括Sigmoid函数，Softmax和ReLU等。

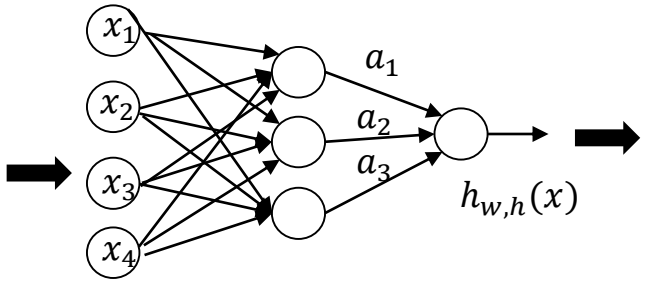
目录

什么叫深度学习

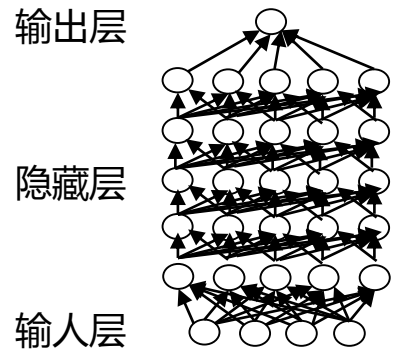
- 隐藏层比较多（大于2）的神经网络叫做**深度神经网络（Deep Neural Network，DNN）**。而深度学习，就是使用深度神经网络架构的机器学习方法。



人类神经网络



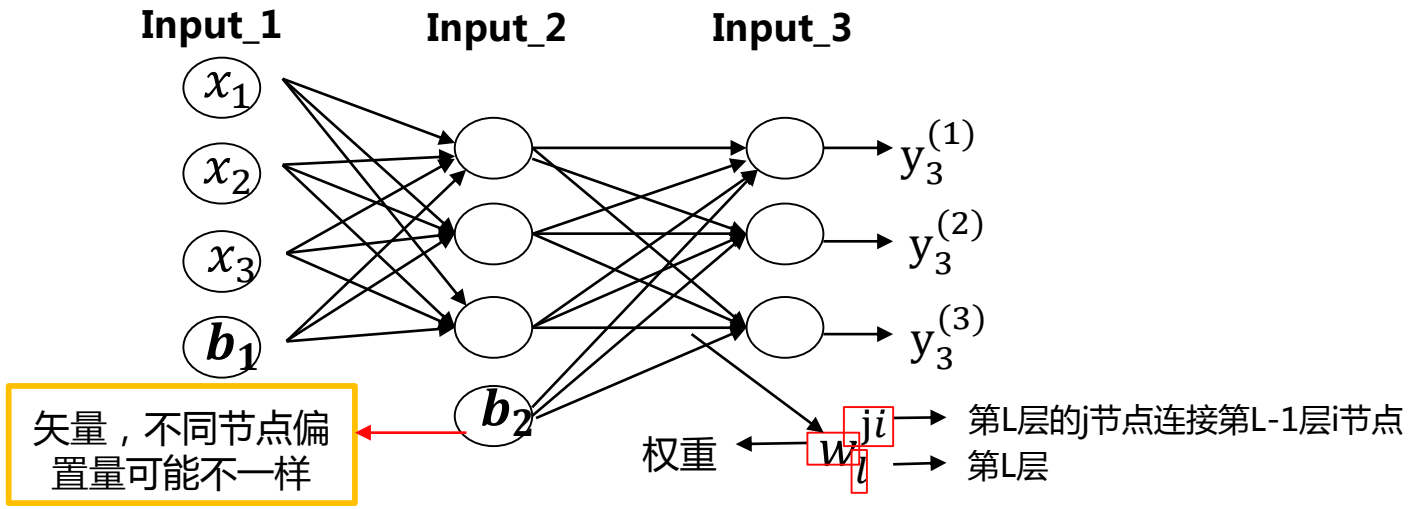
感知器



深度前馈网络

深度前馈网络

- **深度前馈网络**（Deep Feedforward Network），也叫作**前馈神经网络（Feedforward Neural Network）**或者**多层感知机**（Multilayer Perceptron, MLP），是典型的深度学习模型。
  - 输入节点并无计算功能，只是为了表征输入矢量各元素值。
  - 各层节点表示具有计算功能的神经元，称为计算单元。每个神经元只与前一层的神经元相连；接收前一层的输出，并输出给下一层，采用一种单向多层结构；每一层包含若干个神经元，同一层的神经元之间没有互相连接，层间信息的传送只沿一个方向进行。



深度前馈网络推导

- 对于第二层的三个神经元的输出则有：
  - $y_2^{(1)} = f\left(\mu_2^{(1)}\right) = f\left(\sum_{i=1}^{n} w_2^{1i} x_i + b_2^{(1)}\right) = f(w_2^{(11)} x_1 + w_2^{(12)} x_2 + w_2^{(13)} x_3 + b_2^{(1)})$
  - $y_2^{(2)} = f\left(\mu_2^{(2)}\right) = f\left(\sum_{i=1}^{n} w_2^{2i} x_i + b_2^{(2)}\right) = f(w_2^{(21)} x_1 + w_2^{(22)} x_2 + w_2^{(23)} x_3 + b_2^{(2)})$
  - $y_2^{(3)} = f\left(\mu_2^{(3)}\right) = f\left(\sum_{i=1}^{n} w_2^{3i} x_i + b_2^{(3)}\right) = f(w_2^{(31)} x_1 + w_2^{(32)} x_2 + w_2^{(33)} x_3 + b_2^{(3)})$
- 将上述的式子转换为矩阵表达式：

$$y_2 = \begin{bmatrix} y_2^{(1)} \\ y_2^{(2)} \\ y_2^{(3)} \end{bmatrix} = f\left(\begin{bmatrix} w_2^{11} w_2^{12} w_2^{13} \\ w_2^{21} w_2^{22} w_2^{23} \\ w_2^{31} w_2^{32} w_2^{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_2^{(1)} \\ b_2^{(2)} \\ b_2^{(3)} \end{bmatrix}\right) = f(W_2 X + b_2)$$

  - 将第二层的前向传播计算过程推广到网络中的任意一层，则：

$$\begin{cases} y_l^{(j)} = f(u_l^{(j)}) \\ u_l^{(j)} = \sum_{i \in L_{l-1}} w_l^{(ji)} y_{l-1}^{(i)} + b_l^{(j)} \\ y_l = f(u_l) = f(W_l y_{l-1} + b_l) \end{cases}$$

目录

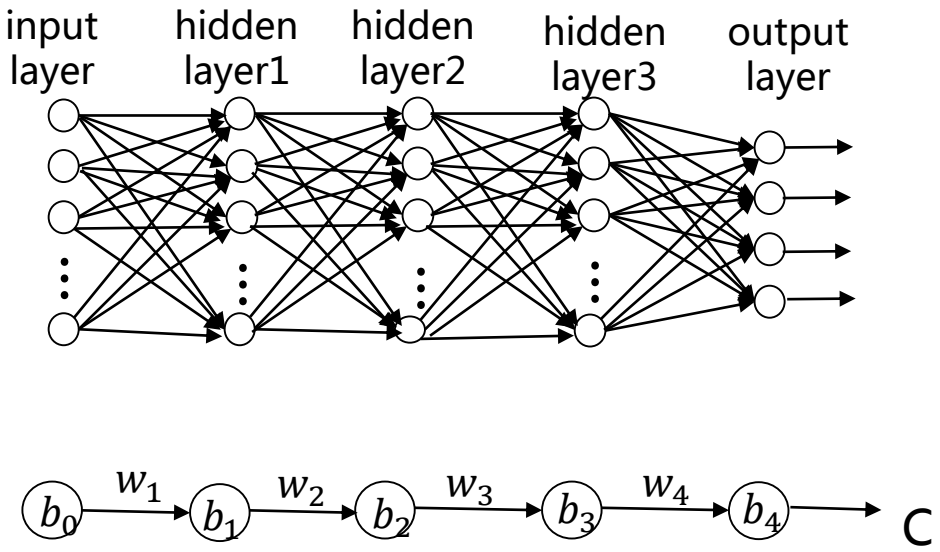
如何训练神经网络

- 神经网络有大量的参数需要学习，每个神经元的权重w和偏置b。
  - 先随机地给参数赋值。
  - 计算网络在训练数据集上的损失函数值，并不断地调整参数，使得损失函数值最小。这个学习的过程称为最优化。
- 神经网络的最优化算法一般都使用梯度下降法。

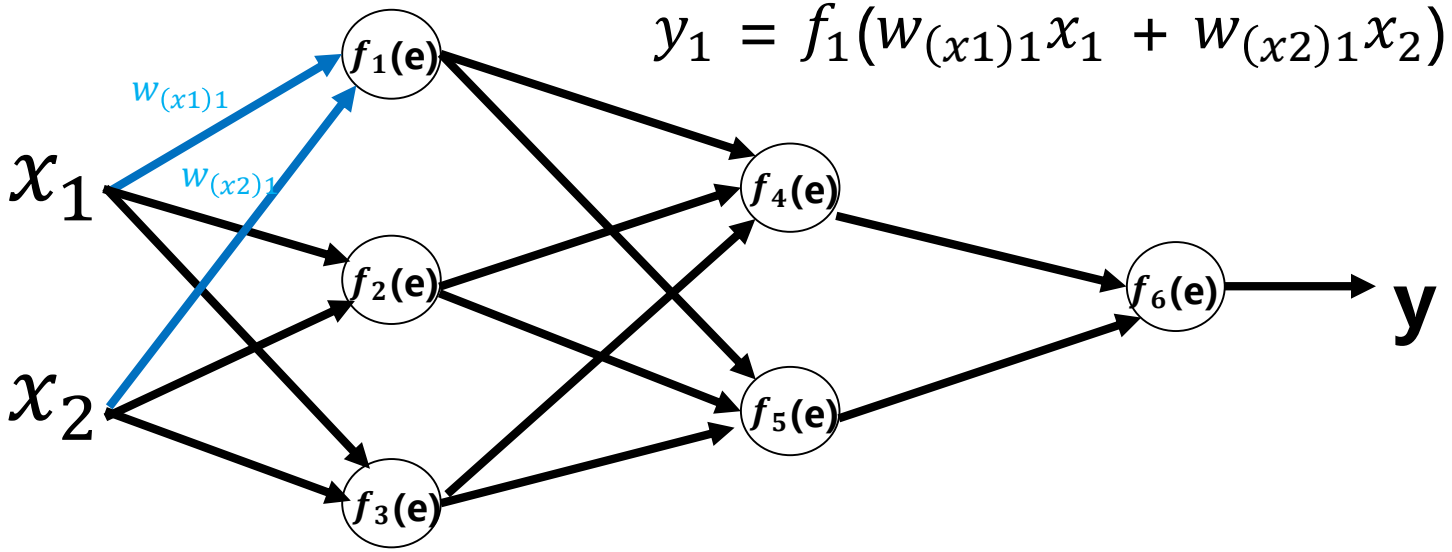
反向传播原理

- 反向传播算法（Back Propagation）是神经网络的重要算法。它使用链式求导法则将输出层的误差反向传回给网络，使神经网络的权重有了较简单的梯度计算实现方法。
  - 根据反向传播算法可以编程实现神经网络权重的梯度计算，从而通过梯度下降法完成神经网络的训练。
  - 成熟的深度学习框架，如TensorFlow，会自动完成反向传播和求导的部分。我们在使用时只需要定义前向运算。

反向传播

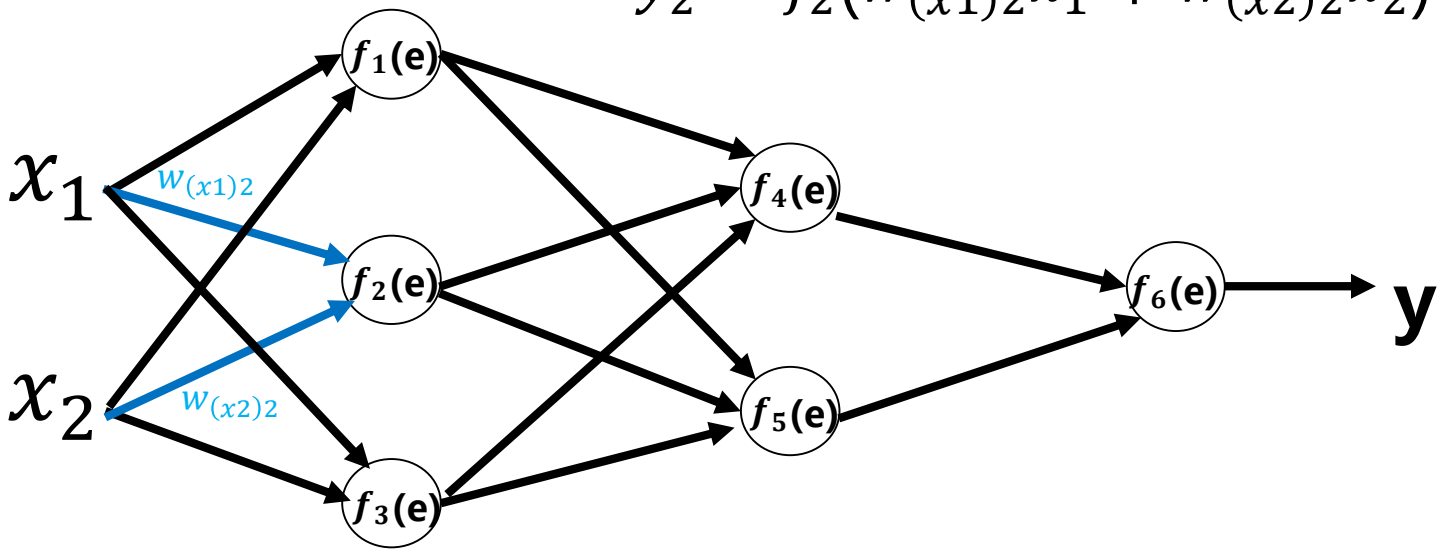


反向传播算法 – 正向传播 (1)



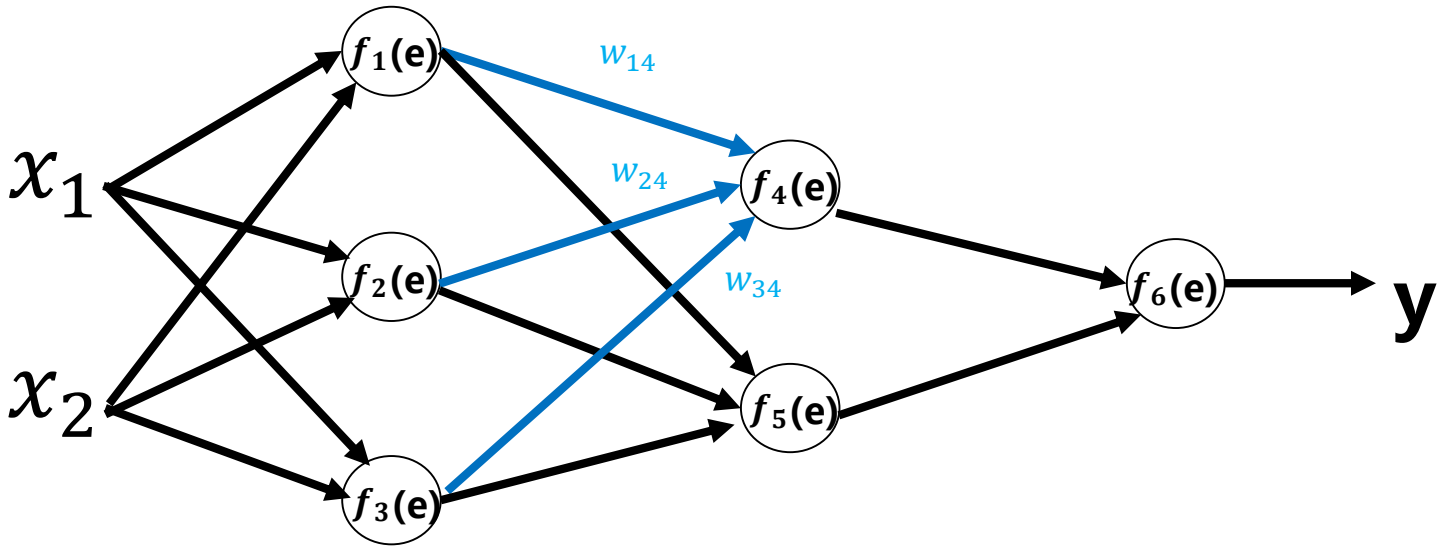
反向传播算法 –正向传播 (2)

$$y_2 = f_2(w_{(x1)2}x_1 + w_{(x2)2}x_2)$$



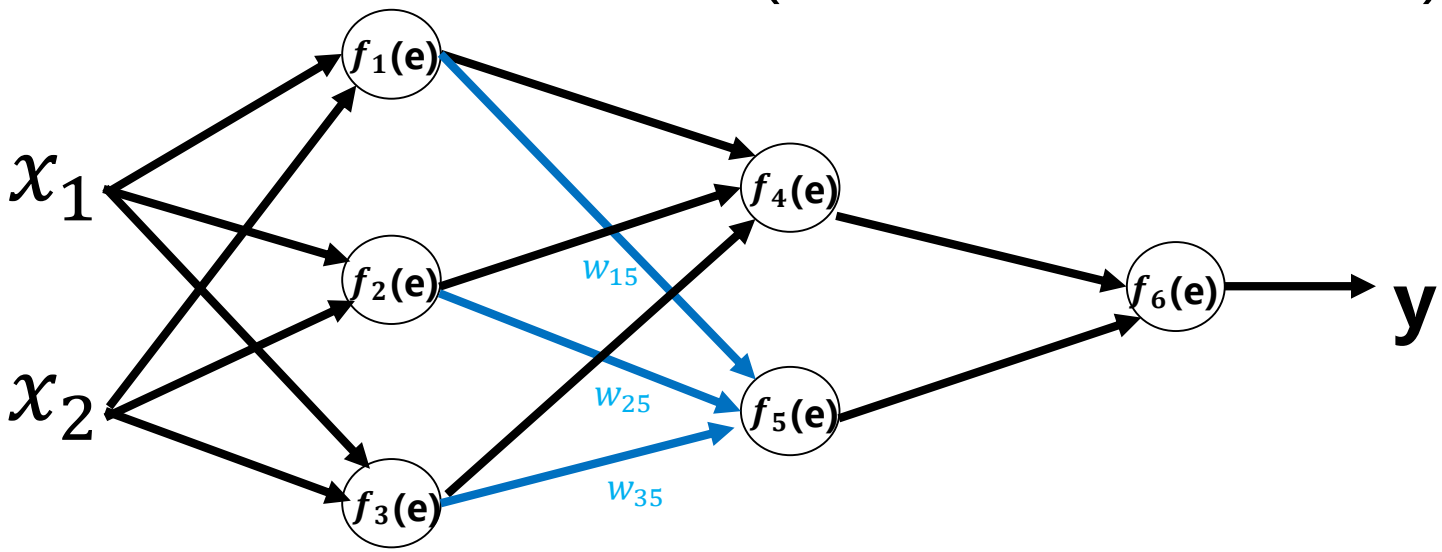
反向传播算法 –正向传播 (3)

$$y_4 = f_4(w_{14}y_1 + w_{24}y_2 + w_{34}y_3)$$

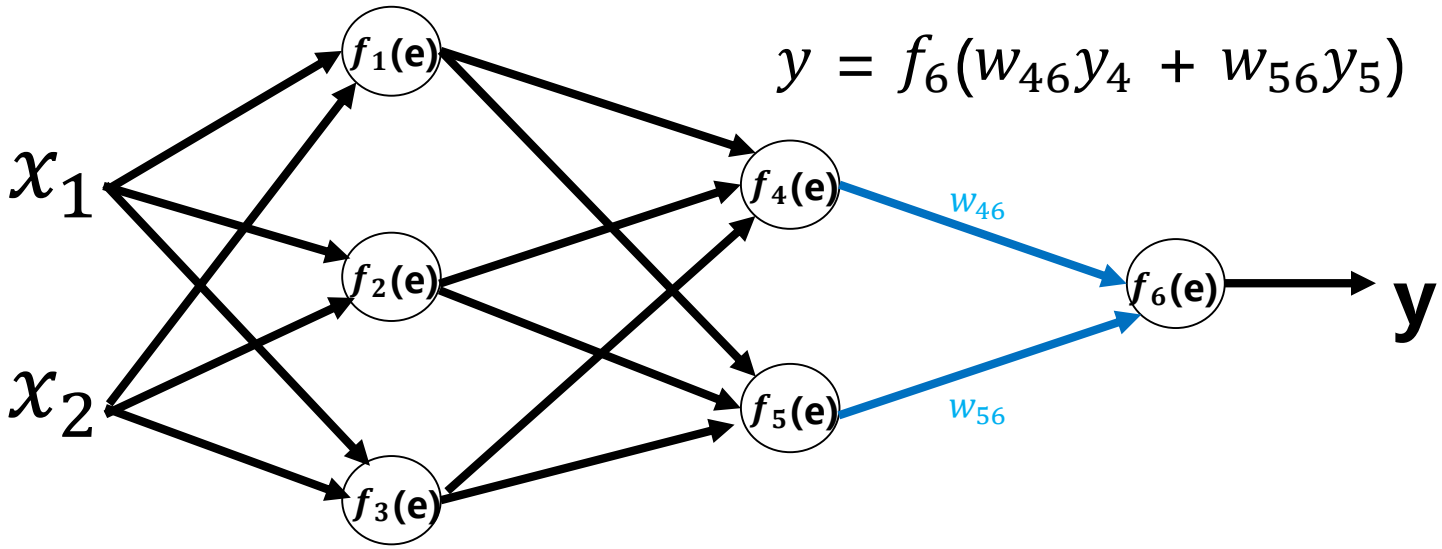


反向传播算法 –正向传播 (4)

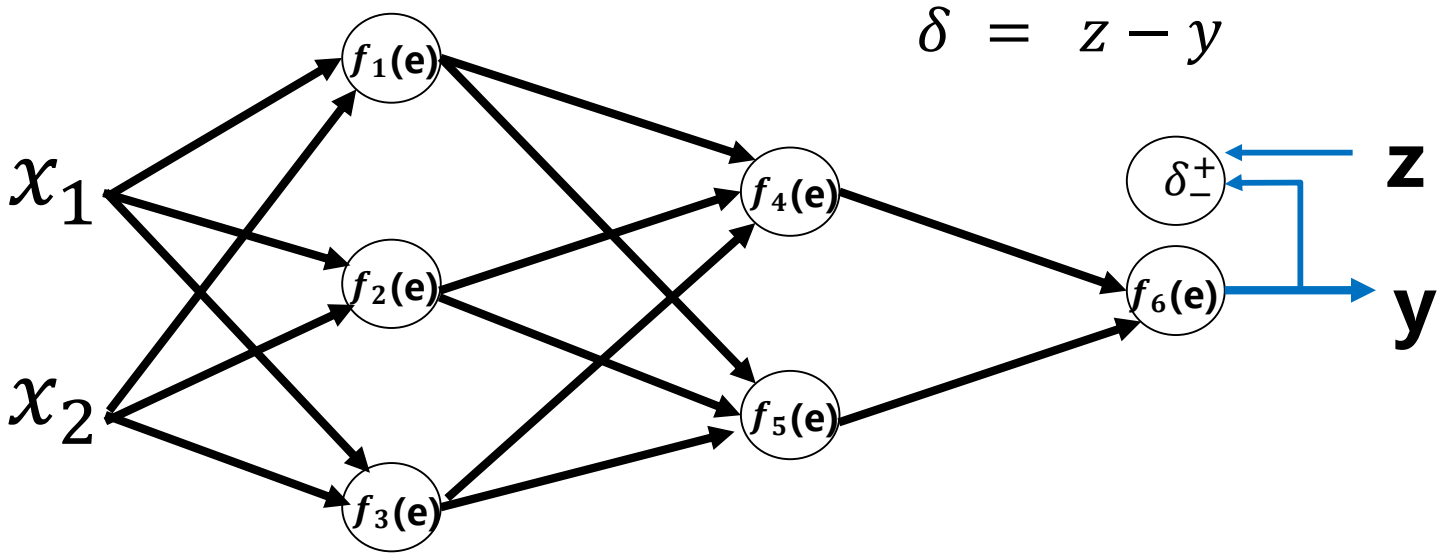
$$y_5 = f_5(w_{15}y_1 + w_{25}y_2 + w_{35}y_3)$$



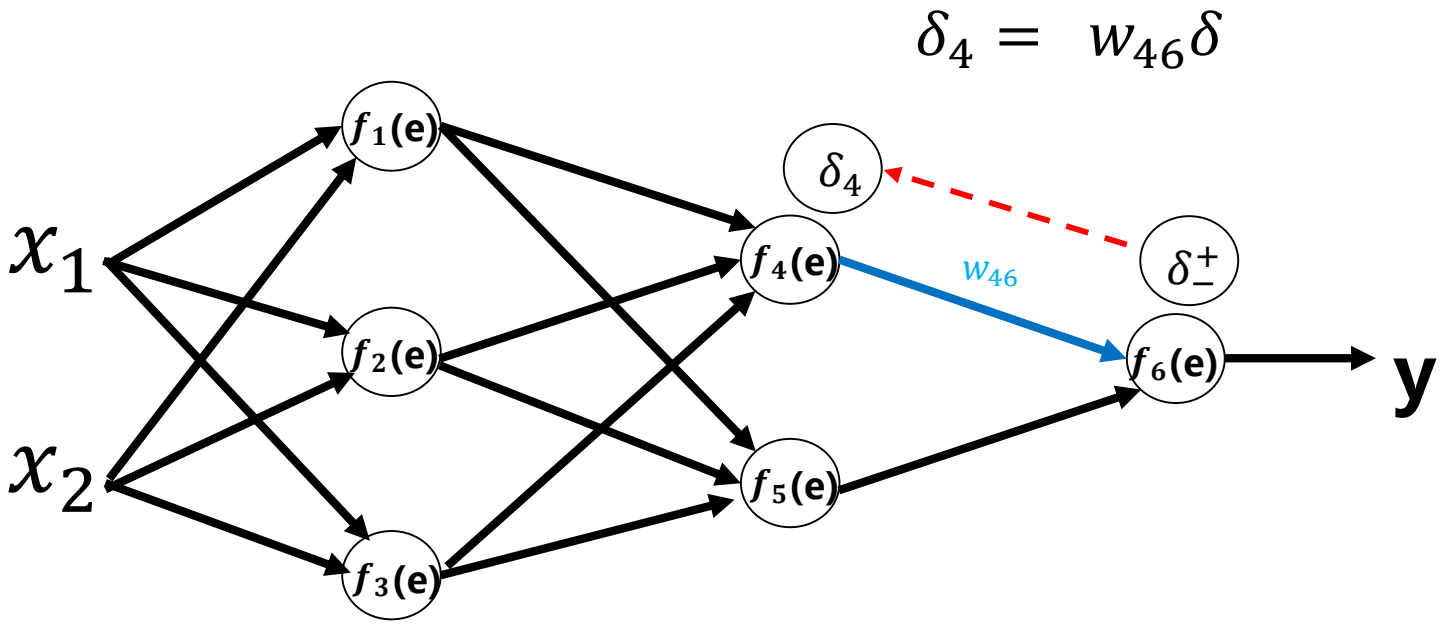
反向传播算法 –正向传播 (5)



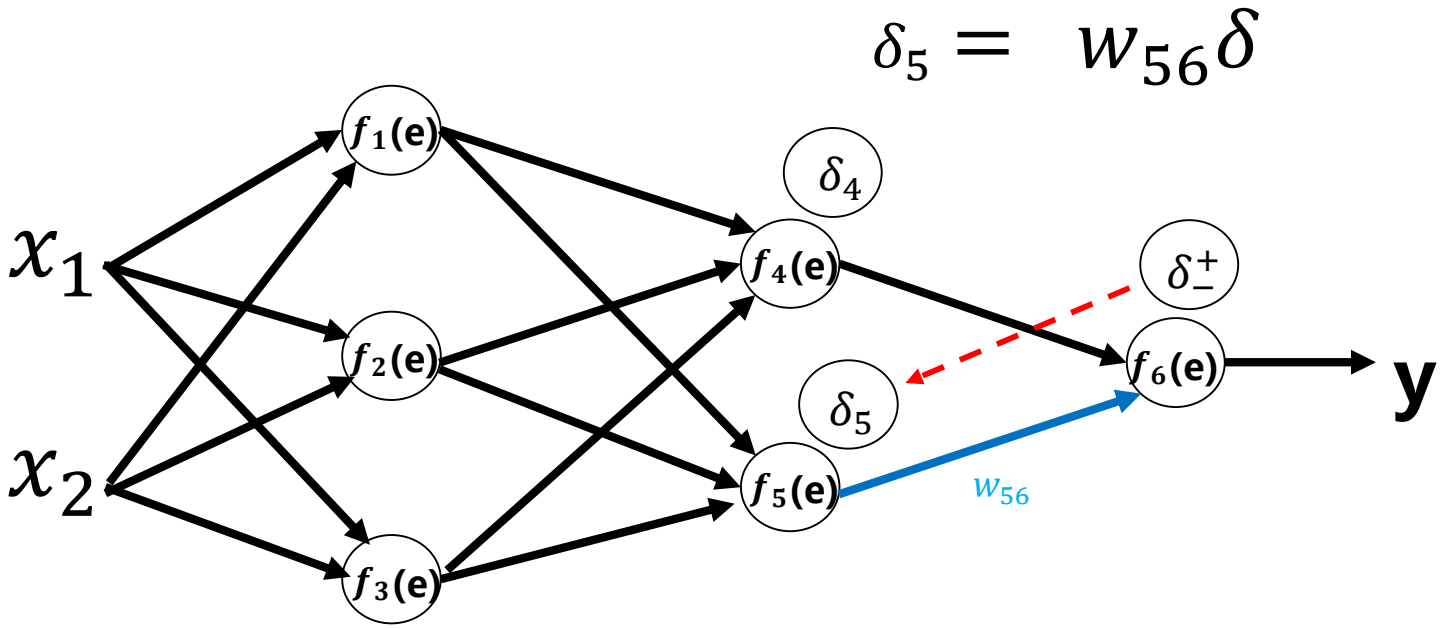
反向传播算法 –误差反向传播 (1)



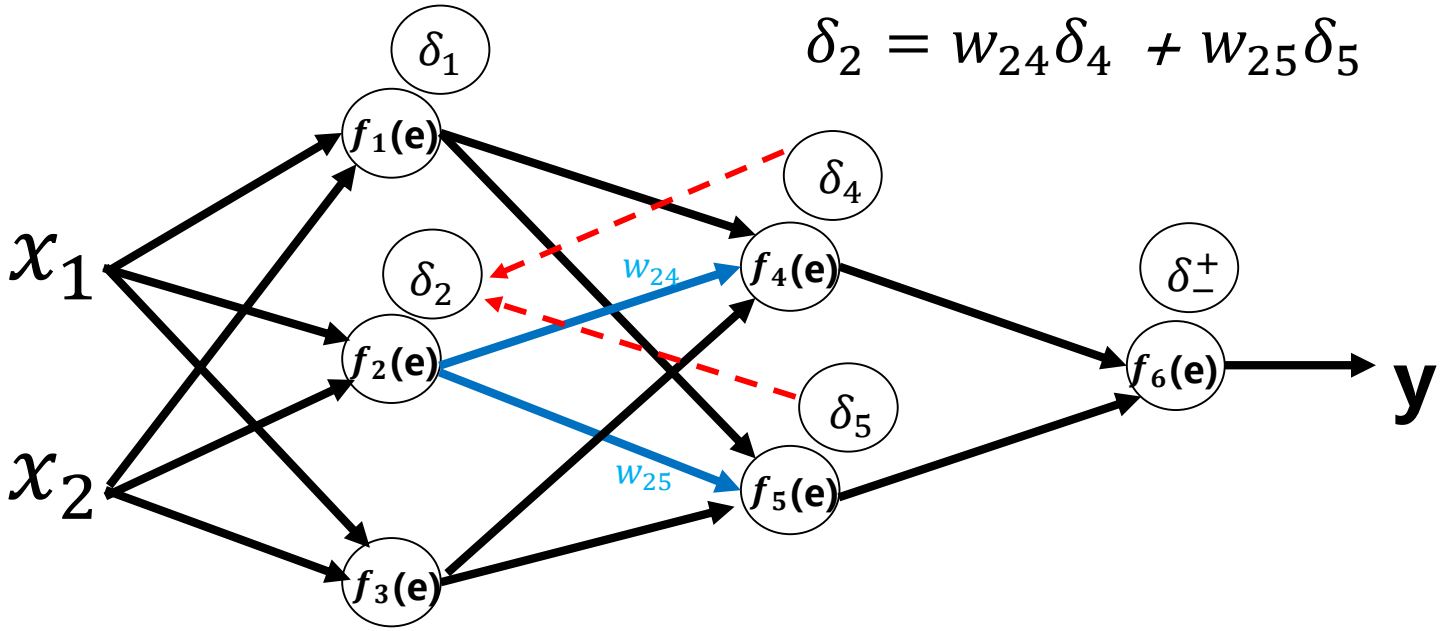
反向传播算法 –误差反向传播 (2)



反向传播算法 –误差反向传播 (3)



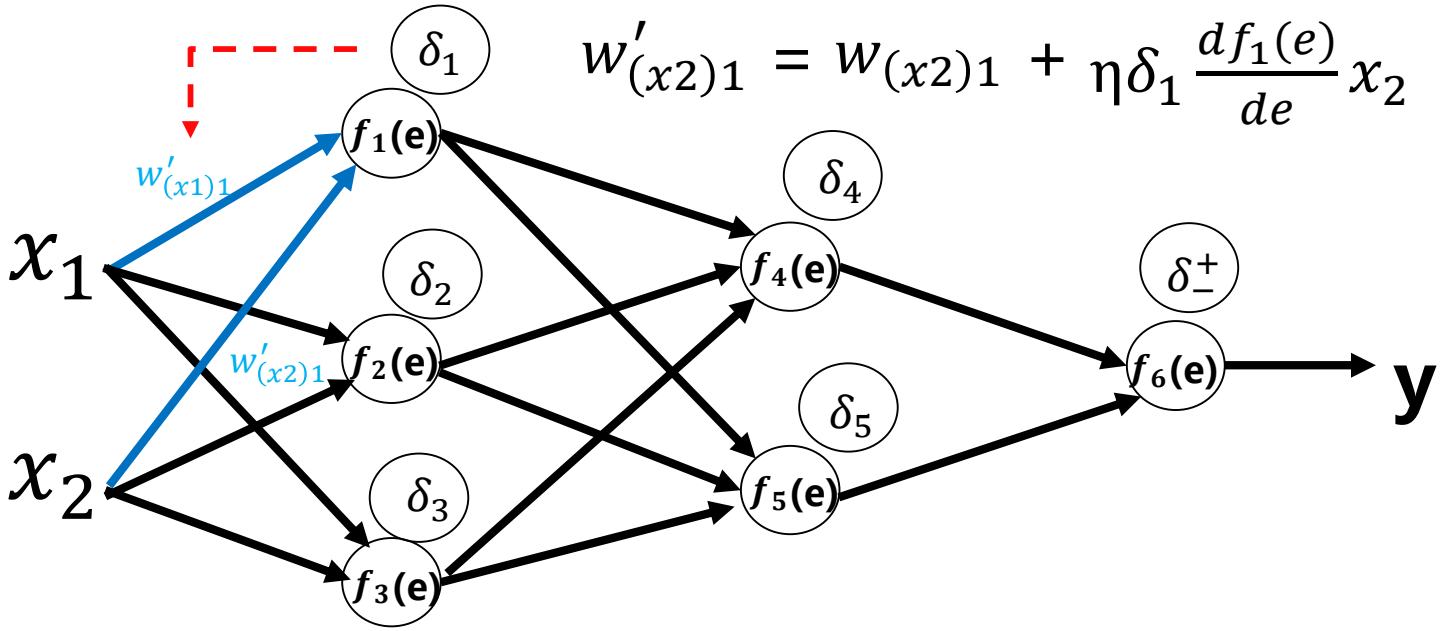
反向传播算法 –误差反向传播 (4)



反向传播算法 – 权重更新 (1)

$$w'_{(x1)1} = w_{(x1)1} + \eta\delta_1 \frac{df_1(e)}{de} x_1$$

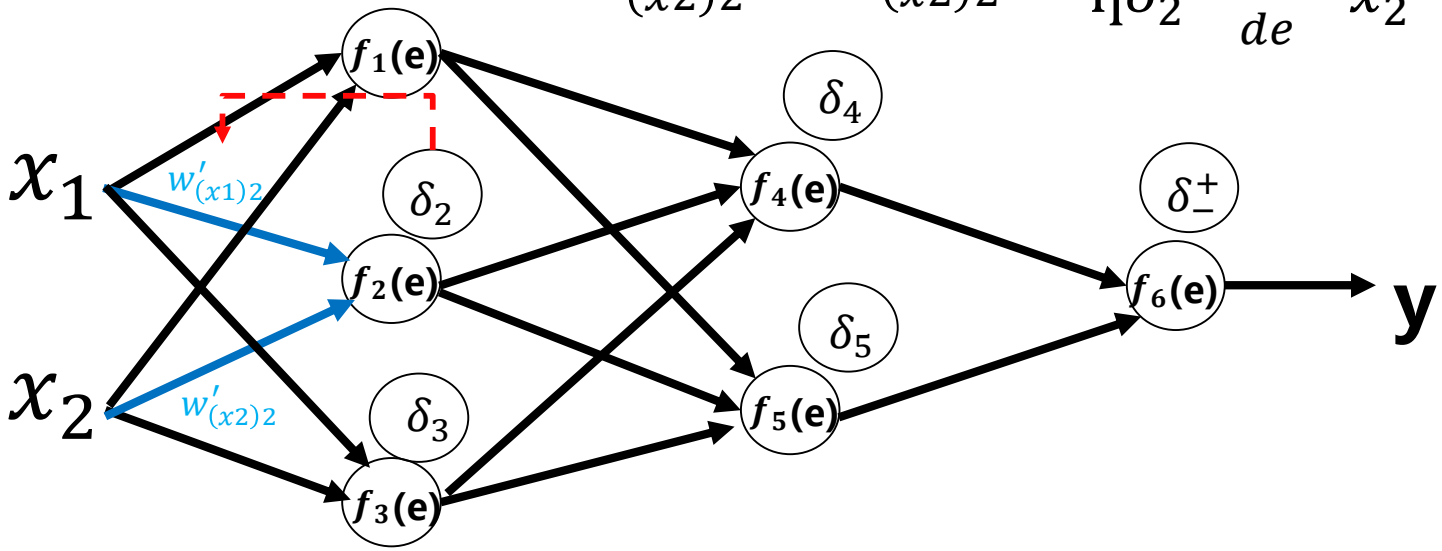
$$w'_{(x2)1} = w_{(x2)1} + \eta\delta_1 \frac{df_1(e)}{de} x_2$$



反向传播算法 – 权重更新 (2)

$$w'_{(x1)2} = w_{(x1)2} + \eta\delta_2 \frac{df_2(e)}{de} x_1$$

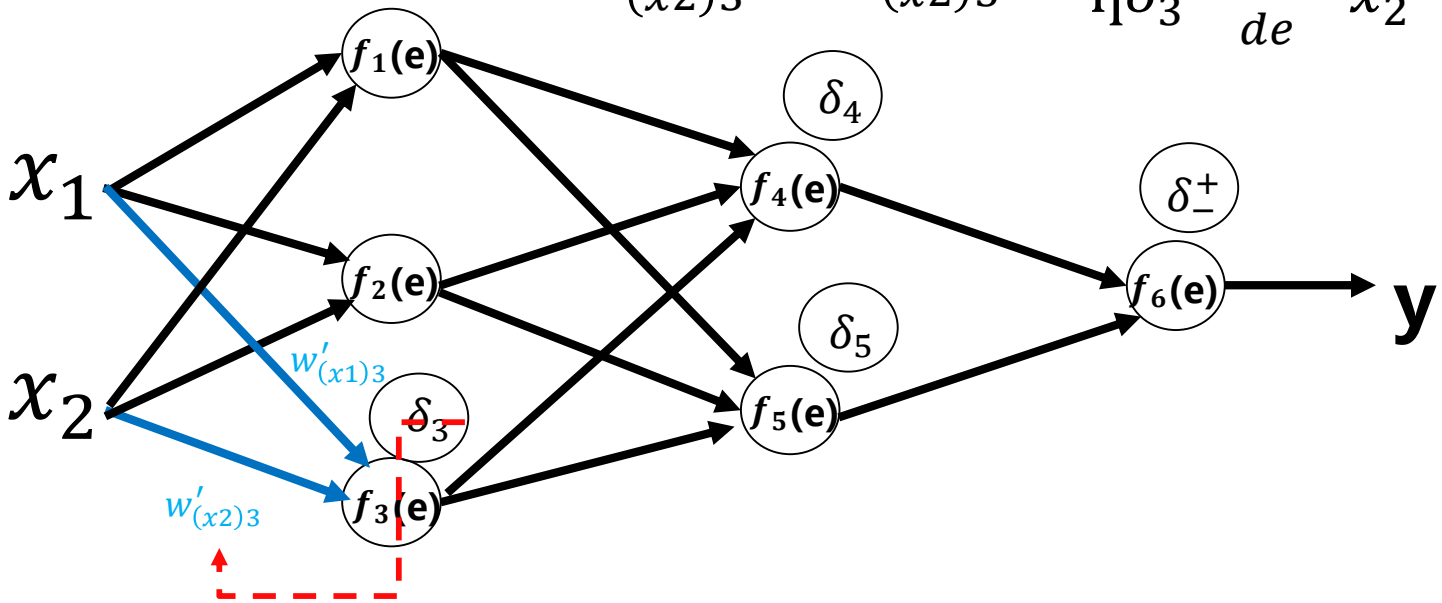
$$w'_{(x2)2} = w_{(x2)2} + \eta\delta_2 \frac{df_2(e)}{de} x_2$$



反向传播算法 – 权重更新 (3)

$$w'_{(x1)3} = w_{(x1)3} + \eta\delta_3 \frac{df_3(e)}{de} x_1$$

$$w'_{(x2)3} = w_{(x2)3} + \eta\delta_3 \frac{df_3(e)}{de} x_2$$

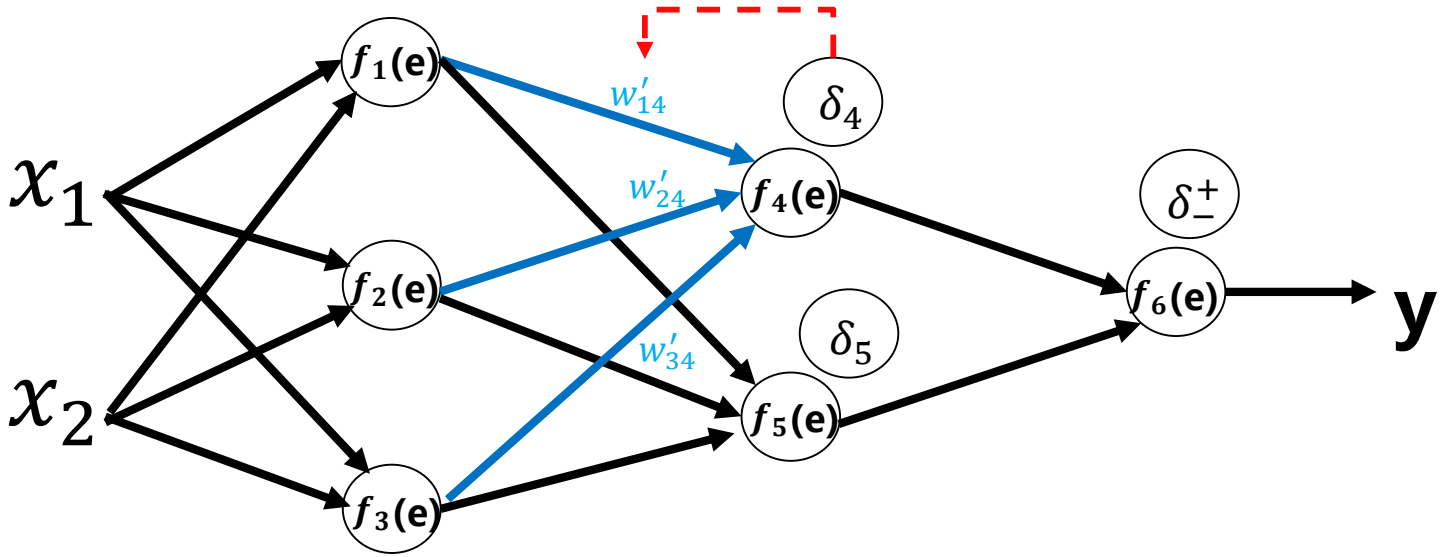


反向传播算法 – 权重更新 (4)

$$w'_{14} = w_{14} + \eta\delta_4 \frac{df_4(e)}{de} y_1$$

$$w'_{24} = w_{24} + \eta\delta_4 \frac{df_4(e)}{de} y_2$$

$$w'_{34} = w_{34} + \eta\delta_4 \frac{df_4(e)}{de} y_3$$

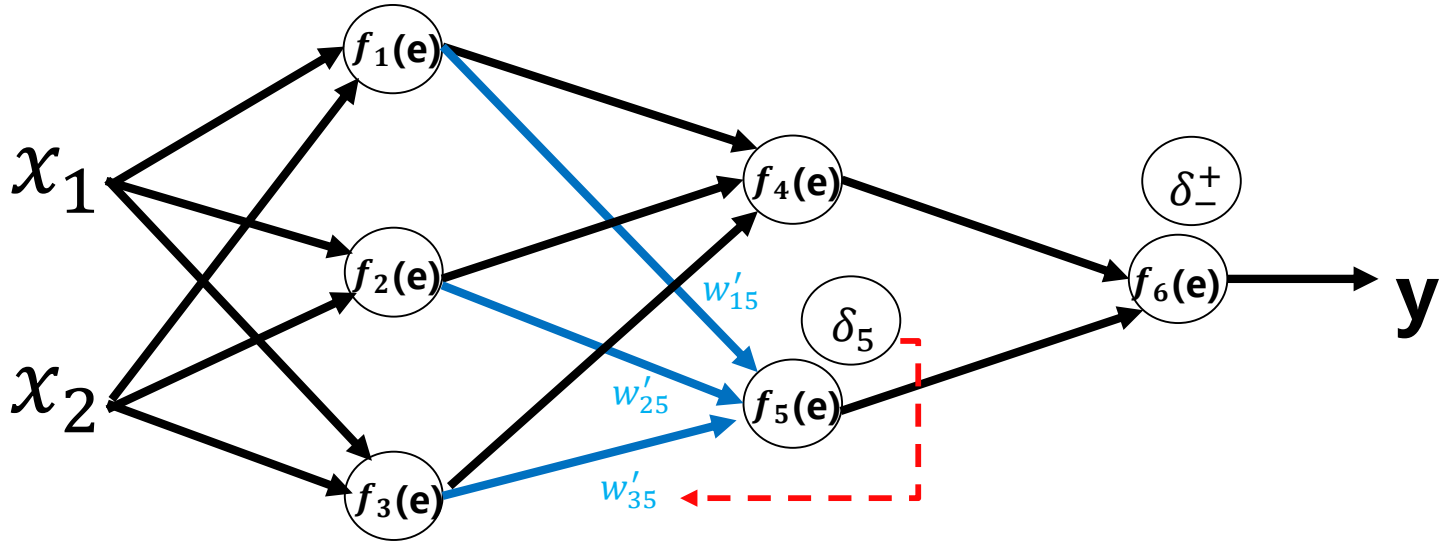


反向传播算法 – 权重更新 (5)

$$w'_{15} = w_{15} + \eta\delta_5 \frac{df_5(e)}{de} y_1$$

$$w'_{25} = w_{25} + \eta\delta_5 \frac{df_5(e)}{de} y_2$$

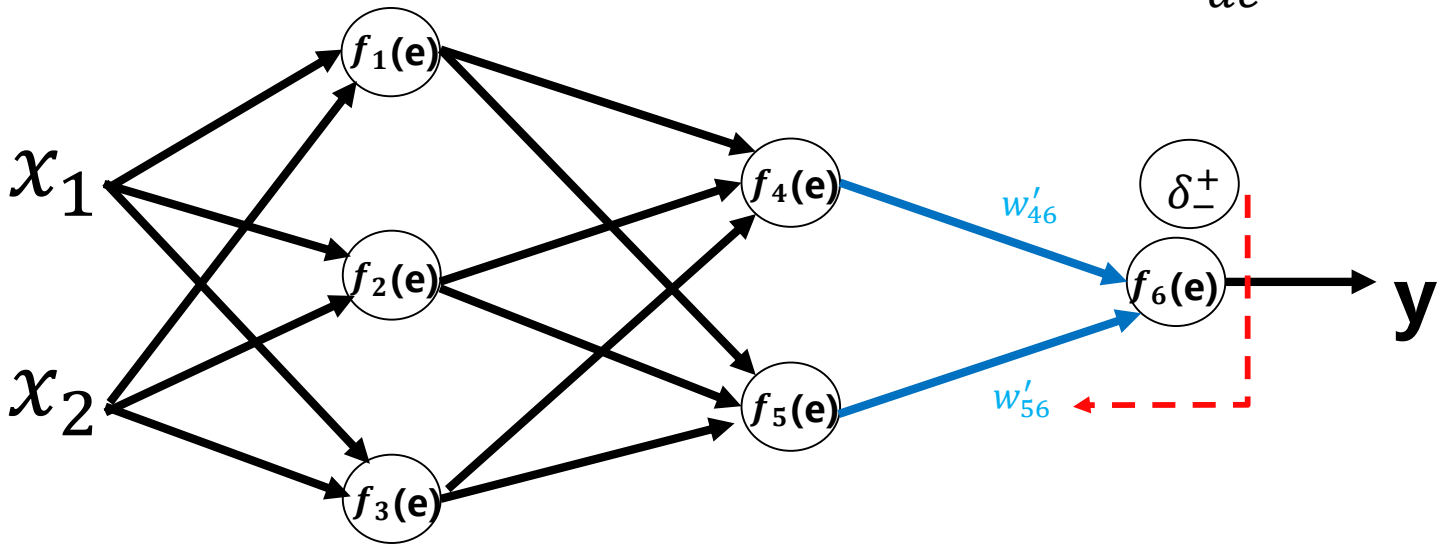
$$w'_{35} = w_{35} + \eta\delta_5 \frac{df_5(e)}{de} y_3$$



反向传播算法 – 权重更新 (6)

$$w'_{46} = w_{46} + \eta\delta \frac{df_6(e)}{de} y_4$$

$$w'_{56} = w_{56} + \eta\delta \frac{df_6(e)}{de} y_5$$

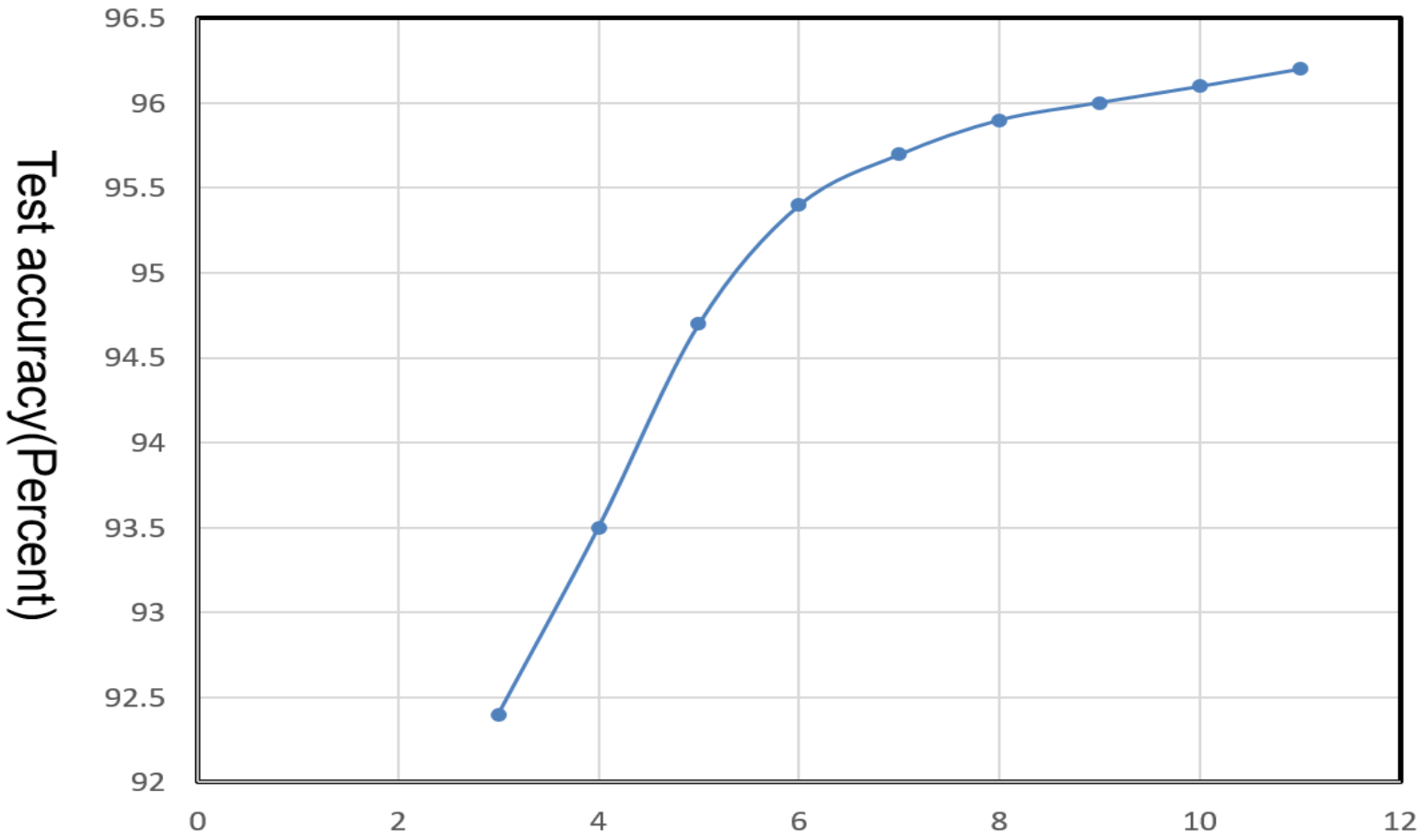


梯度消失与梯度爆炸

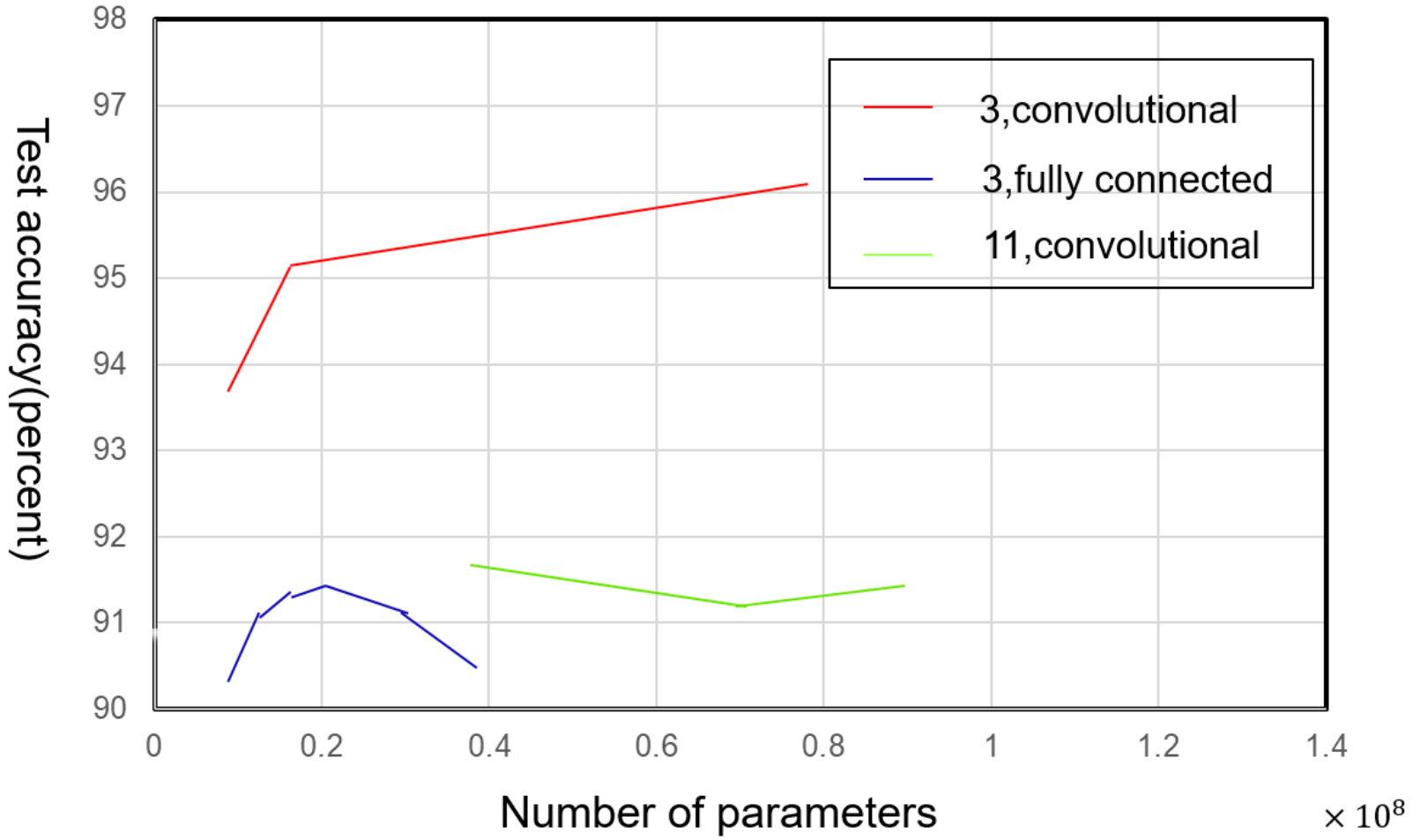
- **梯度消失**：$\sigma'(x)$的最大值为$\frac{1}{4}$，而我们初始化的网络权值$|w|$通常都小于1，因此$|\sigma'(z)w| \leq \frac{1}{4}$，对于上面的链式求导，层数越多，$\frac{\partial C}{\partial b_1}$求导结果越小，因而导致梯度消失的情况出现。
- **梯度爆炸**： $|\sigma'(z)w| > 1$，也就是w比较大的情况。但对于使用sigmoid激活函数来说，这种情况比较少。因为$\sigma'(z)$的大小也与w有关（$z = wx + b$），除非该层的输入值x在一直一个比较小的范围内。

目录

神经网络深度的影响



神经网络参数数量的影响



神经网络连接的影响

- 如何层与层之间连接起来？
  - 默认的神经网络层采用矩阵 W 描述的线性变换，每个输入单元连接到每个输出单元。许多专用网络具有较少的连接，使得输入层中的每个单元仅连接到输出层单元的一个小子集。这些用于减少连接数量的策略减少了参数的数量以及用于评估网络的计算量，但通常高度依赖于问题。

本章总结

- 本章节介绍了人工神经网络与隐藏单元，进而介绍了深度前馈网络，最后介绍了如何训练神经网络和如何设计神经网络。

思考题

1. 我们可以利用以下哪种方法实现反向传播？(　　)

A.计算图

B.链式法则

C.代价函数

D.高阶微分

更多信息

- 参考书目：
- [1]Ian Goodfellow, Yoshua Bengio, Aaron Courville 著，赵申剑，黎彧君，符天凡，李凯，译. 深度学习 2017.


谢 谢
www.huawei.com