Ch2 神经网络基础

1. 基本概念

人工智能是最大的范畴，人工智能当中包括机器学习，计算机视觉、符号逻辑等不同的分支

机器学习当中又包括很多分支，人工神经网络、贝叶斯网络、决策树、线性回归

其中机器学习最主流的学习方法是人工神经网络

人工神经网络当中最先进的技术是深度学习

综上所述

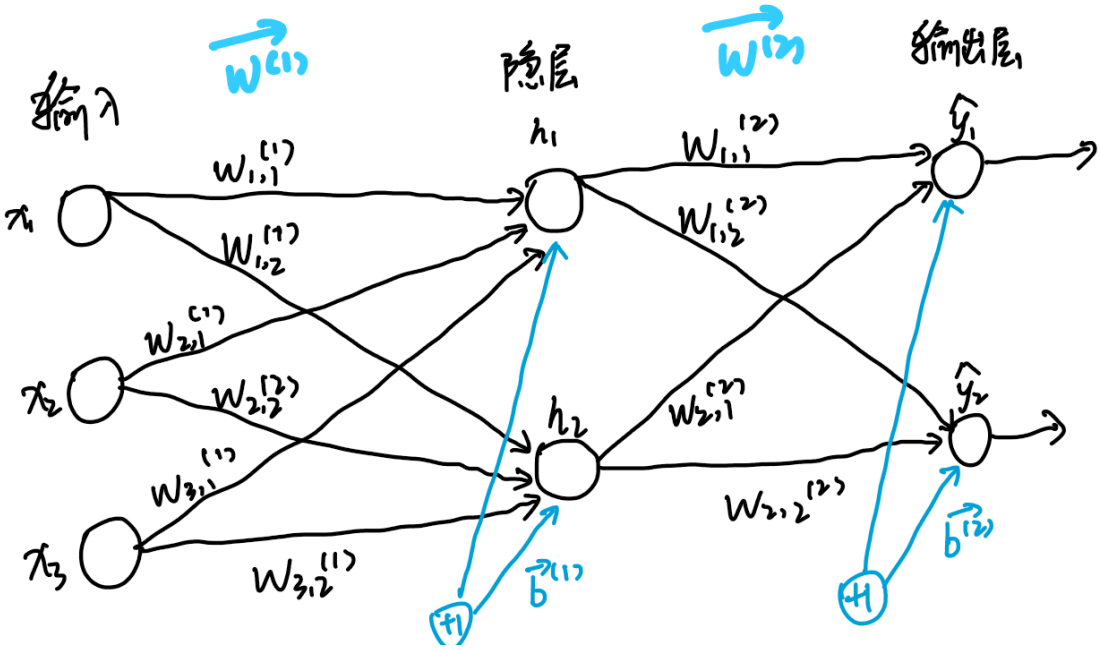
人工智能 contains 机器学习 contains 神经网络 contains 深度学习

机器学习可以训练数据有无标记（label）分为 监督学习 和 无监督学

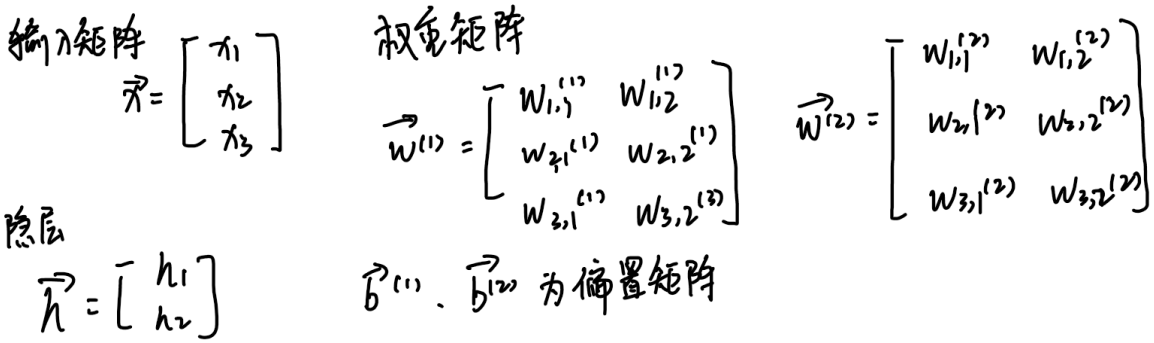
2. （重点）两层神经网络 -- 多层感知机

多层感知机（multi-layer perceptron） MLP

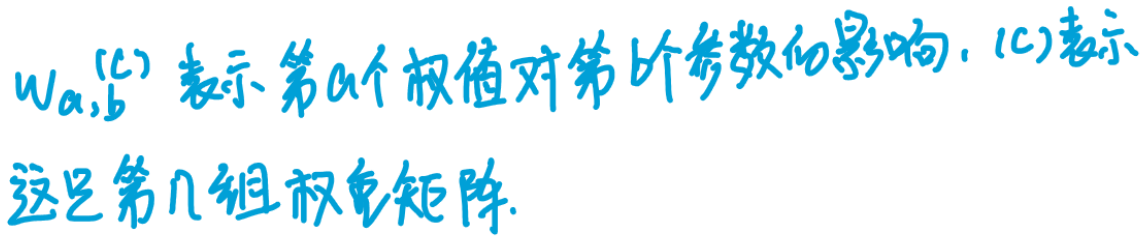
2.1 模型抽象视图



2.2 各参数说明



其中



2.3 计算流程

① 权重矩阵 $\vec{w}^{(1)}$ 取转置与输入矩阵相乘，得到中间值T

$$T = [\vec{w}^{(1)}]^T \cdot \vec{x} = \begin{bmatrix} w_{1,1}^{(1)} & w_{2,1}^{(1)} & w_{3,1}^{(1)} \\ w_{1,2}^{(1)} & w_{2,2}^{(1)} & w_{3,2}^{(1)} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} w_{1,1}^{(1)}x_1 + w_{2,1}^{(1)}x_2 + w_{3,1}^{(1)}x_3 \\ w_{1,2}^{(1)}x_1 + w_{2,2}^{(1)}x_2 + w_{3,2}^{(1)}x_3 \end{bmatrix}$$

② 将①中得到的 T 值加上偏置向量 $\vec{b}^{(1)}$，组合与非线性激活函数G计算得到隐层值

$$\vec{r} = G(\vec{w}^{(1)T}\vec{x} + \vec{b}^{(1)})$$

激活函数　权值矩阵　输入值x　偏置值b

③ 将权值矩阵 $\vec{w}^{(2)}$ 的转置乘以②中得到的隐层 r 值，加上偏置向量 $\vec{b}^{(2)}$，再使用激活函数计算

$$y = G(\vec{w}^{(2)T}\vec{r} + \vec{b}^{(2)})$$

2.4 深度学习（深层神经网络）

相对于浅层的神经网络，深度学习（深层神经网络可以超过1层），比如多个隐层（h层）

深度学习的工作原理 通过对信息的多层抽取和加工来完成复杂的功能。

神经网络发展历史

神经网络的发展可以分为三个阶段

- M-P神经元模型，最早的神经网络。感知机模型（Perceptron），是基于M-P的神经元模型的单层神经网络，可以解决数据的线性可分问题，同时，它不能解决非线性可分的问题

- 反向传播算法（back-propagation）的提出，通过不断调整网咯连接的权重来最小化实际输出向量和预期结果向量之间的差值，即由训练的结果不断纠正参数的权重
- 深度置信网络（deep belief Network，DBN）使用贪婪逐层预训练法大幅提高了训练深层神经网络的效率

3. 神经网络的训练

主要包括 **正向传播** 和 **反向传播**

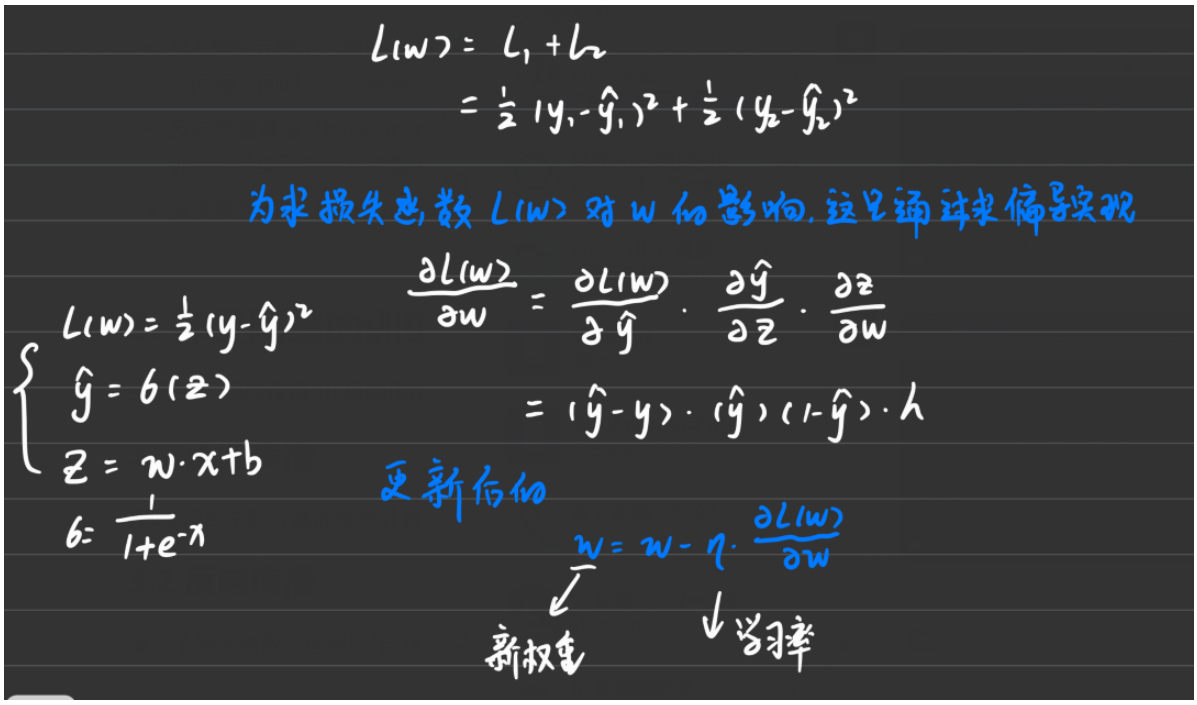
3.1 正向传播

按照2.3当中的计算流程来计算最后的输出结果，一个从输入到输出的过程

3.2 反向传播

取一个损失函数，这里记作L(x)，**损失函数有不同的取法，但是必须保证和激活函数一起使用时不能出现梯度为0的情况**

下面计算过程是基于损失函数是1/2的平方差,激活函数使用的是sigmod函数



3.3 神经网络设计原则

网络的拓扑结构

神经网络的结构包括输入、隐层和输出层，隐层用于提取输入特征中的隐藏规律，隐层太少导致提取结果不准确，隐层太多会导致模型泛化能力变差

激活函数

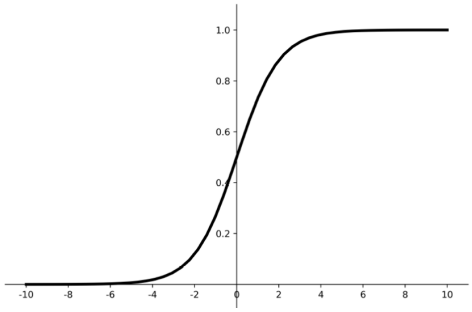
激活函数有两点要求

- 是可微的
- 输出值的范围是有限的

常见的激活函数

sigmoid函数

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

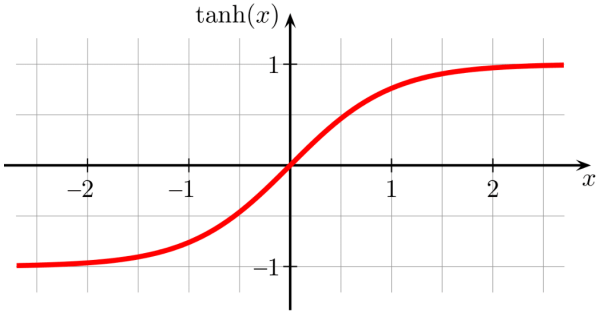


存在缺点:

- 1.输出的均值不是0，即下一层的输入会有一个偏移（>0），可能会影响神经网络收敛性
- 2.计算复杂度高，主要是指数运算
- 3.饱和性问题，两边趋势平缓，提供的数值在两边容易导致梯度为0

tanh函数

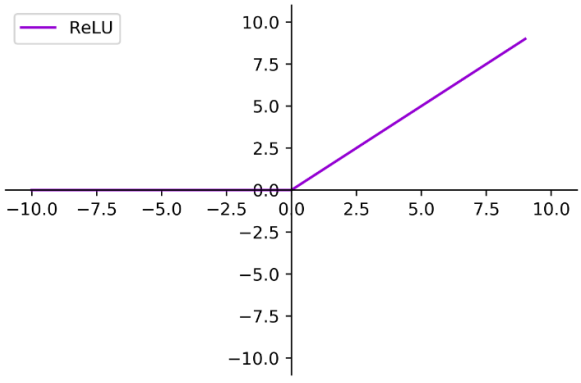
$$tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$$



tanh改善了sigmoid函数均值不为0的情况，但是两边仍然很缓，**即当给的值趋向于两边时，仍然会出现梯度减为0的情况**

ReLU函数

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



- 均值不是0
- 容易导致梯度为0 进而导致ReLU死掉
- 输出范围无限 容易随着层数加大而加大

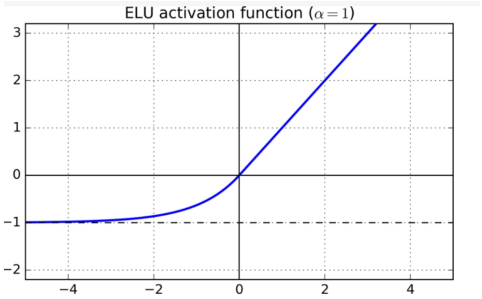
Leaky ReIU函数

$$LeakyReLU(x) = max(\alpha x, x) \qquad \alpha \subset (0,1)$$

- 优化ReLU函数在输入为负值的时候死掉的现象

ELU函数

$$f(x) = \begin{cases} x, & \text{x>0} \\ \alpha(e^x - 1), & \text{x} \geq 0 \end{cases}$$



输出均值接近0 收敛速度较快，但是设计指数计算 复杂度较高

损失函数

均方差损失函数

交叉熵损失函数

3.4 过拟合

过拟合 当网络层数很多时，神经网络会学到一些并不重要甚至错误的特征。**神经网络的泛化能力比较差**

欠拟合 训练的特征少，拟合函数无法逼近训练集 误差较大

正则化

在损失函数中对不想要的部分加入惩罚项，

L1 正则化

L1正则化，公式为：

$$L(w) = L_0(w) + \lambda \sum_i |w_i|$$

其中

$$L_0(w)\text{是原始的损失函数，} \sum_i |w_i| \text{是权重向量的} L1 \text{范数之和，} \lambda \text{是正则化系数。}$$

L2 正则化

对于L2正则化，公式为：

$$L(w) = L_0(w) + \frac{\lambda}{2}||w||_2^2 \text{其中，} L_0(w) \text{是原始的损失函数，} ||w||_2^2 \text{是权重向量的} L2 \text{范数平方，} \lambda \text{是正则化系数。}$$

稀疏化

让神经网络当中很多重或神经元为0

Baggging集成学习

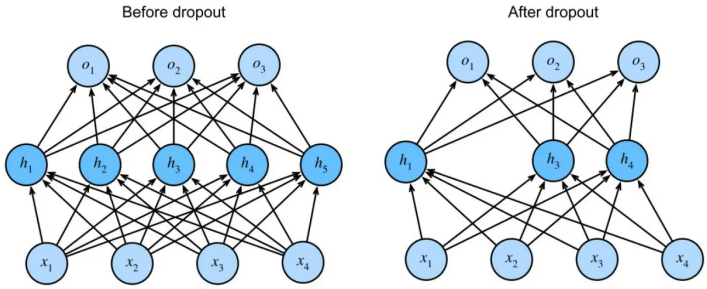
数据集: 原始数据集中重复采样，比如有1，2，3这三个数据集，实际使用可以1，1，2或者2，2，3这种组合

训练模型: 可以有多个

输出: 可以取所有模型均值 或者另外训练一个网络去choose

Dropout

在训练阶段随机 (掩码向量) 删除一些隐层的节点



4. 交叉验证

definition：要求把机器学习用到的数据集分成两部分，一部分是训练集，一部分是测试集，在训练集上训练模型，在测试集上测试对应的结果

常用方法**K-fold** 折叠交叉验证，将数据集分成k份，每次保留一份做测试集，剩下的做训练集，共k次

$$N = 33 \times 512 + 512 + 512 \times 10 + 10 = 22528$$