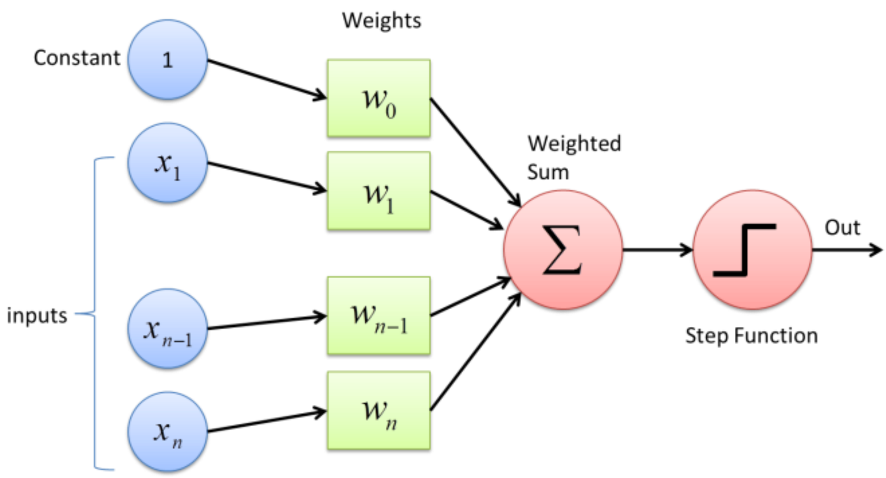
Lecture3-1 Multilayer Perceptron MLP

1. 感知机 Perceptron

感知机（Perceptron）：在深度学习中，感知器也被指为**单层的人工神经网络**，以区别于较复杂的**多层感知机（Multilayer Perceptron）**

- 作为一种**线性分类器**，（单层）感知机可说是最简单的**前向**人工神经网络形式
- 感知机主要的本质缺陷是它不能处理**线性不可分**问题，比如 XOR 问题



定义

如上图所示，这是一个有 n 维输入的单个感知机

- $x_1, ..., x_n$ 为输入的各个分量
- $w_1, ..., w_n$ 为各个输入分量连接到感知机的权重
- w_0：为偏置
- Step Function：激活函数
  - 对于简单的感知器来说，激活函数可以是简单的符号算法 $sgn()$
    - 即大于零的取 1，小于零的取 -1，以此做二元分类

$$\text{out} = \text{Sgn}(\sum_{i=1}^{n} w_i x_i + w_0)$$

学习算法

Define $\text{Training}(\mathbb{D}, \boldsymbol{w}, \lambda)$

Input

- $\mathbb{D}$：给定训练数据，其中元素为$(\boldsymbol{x}^{(i)}, y^{(i)})$
    - $\boldsymbol{x}^{(i)} \in \mathbb{R}^n$
    - $y^{(i)} \in \{-1, 1\}$
- $\boldsymbol{w}$：感知机的权重向量（$\boldsymbol{w} \in \mathbb{R}^n$）
- λ：学习率

Algorithm

1. $\textbf{Initialize } \boldsymbol{w} = 0 \in \mathbb{R}^n$
2. $\textbf{for } epoch \text{ from } 1 \text{ to } \mathrm{T} \textbf{ do}$
3. 　　打乱数据集 $\mathbb{D}$
4. 　　$\textbf{for}$ 每一个训练样本 $(\boldsymbol{x}^{(i)}, y^{(i)}) \in \mathbb{D}$ $\textbf{do}$
5. 　　　　$\textbf{if } y^{(i)}\boldsymbol{w}^T\boldsymbol{x}^{(i)} \leq 0 \textbf{ then}$
6. 　　　　　　更新 $\boldsymbol{w} \leftarrow \boldsymbol{w} + \lambda y^{(i)}\boldsymbol{x}^{(i)}$
7. 　　　　$\textbf{end if}$
8. 　　$\textbf{end for}$
9. $\textbf{end for}$

---

2. 多层感知机 MLP

定义

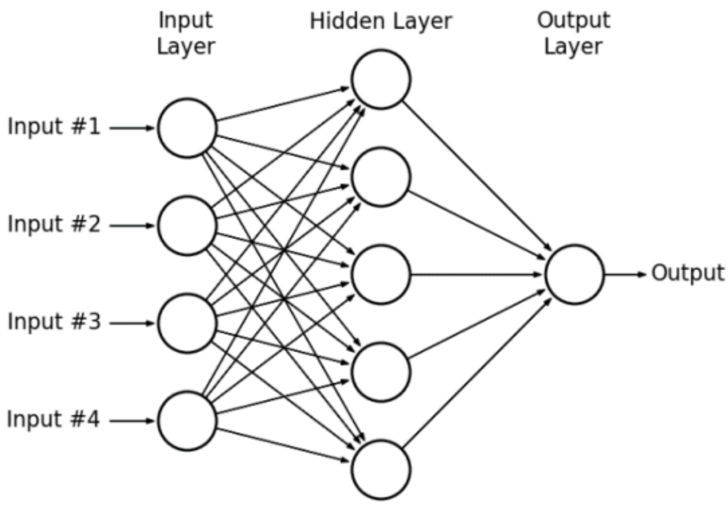
深度前馈网络（deep feedforward network）/前馈神经网络（feedforward neural network）/多层感知机（multilayer perceptron, MLP）：是同一个意思，是典型的**前向**深度学习模型

MLP 的目标是**近似某个函数** f^*

- 对于分类器，$y = f^*(\boldsymbol{x})$ 将输入 $\boldsymbol{x}$ 映射到一个类别 y
- MLP 定义了一个映射 $\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$，学习参数 $\boldsymbol{\theta}$ 的值，使它能够得到最佳的函数近似

> **前向（feedforward）**：输入 $\boldsymbol{x}$ 流经用于定义 f 的中间计算过程，最终到达输出 $\boldsymbol{y}$，在模型的输出和模型本身之间没有**反馈（feedback）**连接
>
> - 当前馈神经网络被扩展成包含反馈连接时，它被称为**循环神经网络（recurrent neural network）**

MLP 被称为前馈神经**网络（network）**，是因为它们通常用许多不同函数复合在一起来表示



- 例如：有三个函数 $f^{(1)}, f^{(2)}, f^{(3)}$ 连接在一个链上形成 $f(\boldsymbol{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\boldsymbol{x})))$
  - $f^{(1)}$：**输入层（input layer）/ 隐藏层（hidden layer）**
  - $f^{(2)}$：**隐藏层（hidden layer）**
    - 训练数据并没有给出这些层中的每一层所需的输出
  - $f^{(3)}$：**输出层（output layer）**（MLP 的最后一层）
  - 模型的**深度（depth）**：链的全长
  - 模型的**宽度（width）**：隐藏层的维数

学习 XOR

任务

我们知道感知机是无法学习 XOR 函数的，对于 MLP 来说，是可以解决这个任务的

XOR 函数提供了我们想要学习的目标函数 $y = f^*(\boldsymbol{x})$，对于这个问题，我们希望网络只在四个点 $\mathbb{X} = \{[0,0]^T, [0,1]^T, [1,0]^T, [1,1]^T\}$ 上表现正确

性能评估/损失函数

可以把这个问题作为**回归**问题，并且使用**均方误差损失函数**，即评估整个训练集上的表现的 MSE 损失函数为

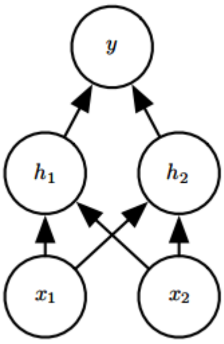
$$J(\boldsymbol{\theta}) = \frac{1}{4} \sum_{\boldsymbol{x} \in \mathbb{X}} (f^*(\boldsymbol{x}) - f(\boldsymbol{x}; \boldsymbol{\theta}))^2$$

架构

现在我们选择 MLP 的形式，我们引入一个简单的前馈神经网络，它有一个隐藏层并且隐藏层中包含了两个单元

- 第一层通过函数 $f^{(1)}(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c})$ 计算得到隐藏单元向量 $\boldsymbol{h}$，隐藏单元随后作为第二层的输入
  - $\boldsymbol{W}$：线性变换的权重矩阵
  - $\boldsymbol{c}$：偏置
- 激活函数 g 通常选择对每个元素分别其作用的函数，有 $\boldsymbol{h} = g(\boldsymbol{x}^T\boldsymbol{W} + \boldsymbol{c})$

- 默认的推荐激活函数为 $g(z) = \max\{0, z\}$，或者称为 ReLU
- 第二层为 $y = f^{(2)}(\boldsymbol{h}; \boldsymbol{w}, b)$



现在整个网络的架构为

$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c}, \boldsymbol{w}, b) = \boldsymbol{w}^\top \max\left\{0, \boldsymbol{W}^\top \boldsymbol{x} + \boldsymbol{c}\right\} + b$$

通过学习可以获得 XOR 问题的一个解

- $\boldsymbol{W} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$，$\boldsymbol{c} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$
- $\boldsymbol{w} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$，$b = 0$

计算

- 输入：$\boldsymbol{X} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$
- 第一层：$\boldsymbol{X}\boldsymbol{W} + \boldsymbol{c}^T = \begin{bmatrix} 0 & -1 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$
- 经过一层激活后可以得到 $\boldsymbol{H} = g(\boldsymbol{X}\boldsymbol{W} + \boldsymbol{c}^T) = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix}$
- 第二层：$\boldsymbol{H}\boldsymbol{w} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$

神经网络对这一批次中的每个样本都给出了正确的结果

3. 历史小记

理论

- 17 世纪：反向传播算法的根本链式法则的发现
- 20 世纪 40 年代：线性模型，感知机的使用
    - 无法学习 XOR 函数
- 20 世纪 60 年代到 70 年代：学习非线性函数需要多层感知机的发展和计算该模型梯度的方法
    - 基于动态规划的链式法则
- 20 世纪 90 年代：神经网络研究得到普及，其它机器学习基础变得更加欢迎

算法

- 交叉熵损失函数替代均方误差损失函数
    - 提高了 sigmoid 和 softmax 的性能
    - 均方误差存在饱和和学习缓慢的问题
- 最大似然原理的想法在统计学界和机器学习界的广泛传播
- ReLU 隐藏单元替代 sigmoid 隐藏单元