Ch1 前馈神经网络和感知机

先介绍最简单、最原始的神经网络——前馈神经网络的概念，再讲解前馈神经网络的训练过程——反向传播（BP）和梯度下降，最后通过简单的实例——感知机解释前馈神经网络的一种应用

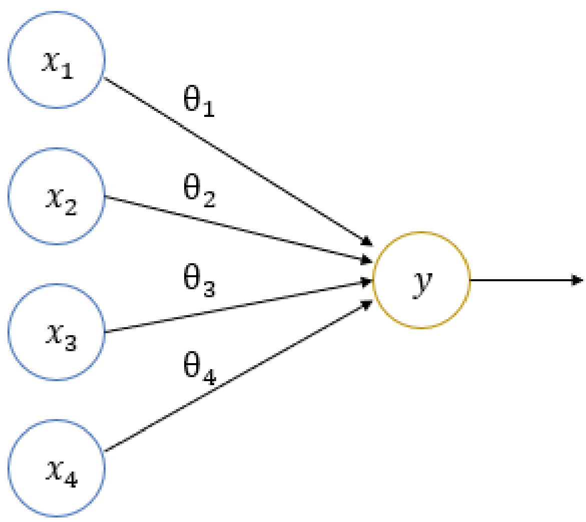
前馈神经网络

每一层神经元接受前一层神经元的输出作为输入，并输出到下一层神经元。网络中无反馈，信号单向传播。单独一层可以抽象为一个线性函数 $y = \omega x + b$。通过多个这样的层，可以实现输入空间到输出空间的复杂映射

单层前馈神经网络

不考虑激活函数，x 为输入（input），θ 为权重（weight）。通常 x_1 固定为 1，对应的 θ_1 称为偏置项

- 图像表示



- 矩阵表示

  y 和 x 满足这样的关系式

$$y = x_1 * \theta_1 + x_2 * \theta_2 + \cdots = \sum x_n * \theta_n$$

  表示为矩阵形式也就是

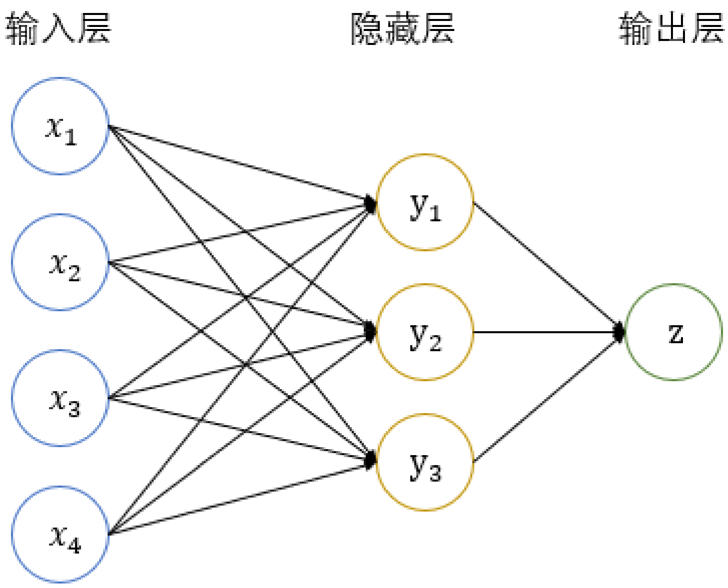
$$x = \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \end{Bmatrix} \quad \theta = \begin{Bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \vdots \end{Bmatrix}$$

$$y = \theta^T x$$

多层前馈神经网络

第零层称为输入层，最后一层称为输出层，中间其他层称为隐藏层

- 图像表示



- 矩阵表示

  y 和 x 满足这样的关系式

$$\begin{aligned} y_1 &= x_1 * \theta_{11} + x_2 * \theta_{21} + \cdots = \sum x_n * \theta_{n1} \\ y_2 &= x_1 * \theta_{12} + x_2 * \theta_{22} + \cdots = \sum x_n * \theta_{n2} \\ &\cdots \\ y_n &= x_1 * \theta_{1n} + x_2 * \theta_{2n} + \cdots = \sum x_n * \theta_{nn} \end{aligned}$$

  表示为矩阵形式也就是

$$x = \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \end{Bmatrix} \quad \theta = \begin{Bmatrix} \theta_{11} & \theta_{12} & \theta_{13} & \cdots & \theta_{1n} \\ \theta_{21} & \theta_{22} & \theta_{23} & \cdots & \theta_{2n} \\ \theta_{31} & \theta_{32} & \theta_{33} & \cdots & \theta_{3n} \\ \vdots \end{Bmatrix} \quad y = \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \end{Bmatrix}$$

$$y = \theta^T x$$

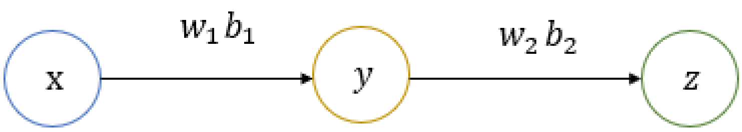
z 和 y 的关系与单层前馈神经网络相同

BP神经网络

用反向传播算法（BP）训练多层前馈神经网络，包括了信号前向传播和误差反向传播两个过程。BP算法的指导思想是梯度下降（$\omega = \omega - \alpha * \delta\omega$）

反向传播

通过链式法则计算损失函数对每一个参数的导数



根据前馈神经网络输入和输出的关系，我们可以得到这样的信息

$$y = \omega_1 * x + b_1$$
$$z = \omega_2 * y + b_2$$

假设损失函数是根据 z 定义的 $Loss(z)$，为了找到能够最小化损失函数的参数 ω 和 b，就需要求出 $Loss(z)$ 对 ω 和 b 的偏导数来确定梯度下降的方向

$$\frac{\delta Loss(z)}{\delta\omega_2} = Loss'(z) * y$$

$$\frac{\delta Loss(z)}{\delta b_2} = Loss'(z) * 1$$

$$\frac{\delta Loss(z)}{\delta\omega_1} = Loss'(z) * \frac{\delta z}{\delta y} * \frac{\delta y}{\delta\omega_1} = Loss'(z) * \omega_2 * x$$

$$\frac{\delta Loss(z)}{\delta b_1} = Loss'(z) * \frac{\delta z}{\delta y} * \frac{\delta y}{b_1} = Loss'(z) * \omega_2$$

使用链式求导的方法，计算损失函数对每个参数的偏导数的过程就是反向传播

梯度下降

顾名思义，梯度下降就是沿着梯度向下的方向调整参数，以达到使目标函数取到极小值的目标

因为实数的分布是连续的，而实现梯度下降的过程是离散的，因此只能按照一定的步长逐步逼近损失函数极小值，即按照 $a_{k+1} = a_k + \rho_k \vec{s}$ 的规律不断更新参数以逼近极小值点

其中 ρ 称为步长，就是参数朝梯度方向移动的距离，过大的步长会导致函数值发散，过小的步长又会导致函数值迟迟不能取到极小值，因此应该对每个参数 a_k 选择合适的 ρ_k

根据梯度下降的样本选取方式，分为三种

- 批量梯度下降（Batch Gradient Descent，BGD）
  每次迭代使用梯度下降中，取所有的样本计算出来的数据来更新参数，优点是充分利用全数据集，缺点是训练过程很慢
- 随机梯度下降（Stochastic Gradient Descent，SGD）
  每次迭代使用梯度下降中，随机取一个样本来更新参数，优点是训练速度加快，缺点是准确度下降
- 小批量梯度下降（Mini-Batch Gradient Descent，MBGD）
  每次迭代使用梯度下降中，取 $batch\_size$ 个样本来更新参数，平衡 BGD 和 SGD 的优缺点

感知机

感知机概述

感知机是用于二分类的线性分类模型，输入实例的特征向量，输出实例的类别（取值为±1）

感知机和输入空间中将实例划分为正负两类的超平面相对应，属于一种判别模型

感知机的数学模型

假设输入空间 $x \subseteq R^n$，输出空间 $y = \{-1, +1\}$。感知机即为输入空间到输出空间的如下函数

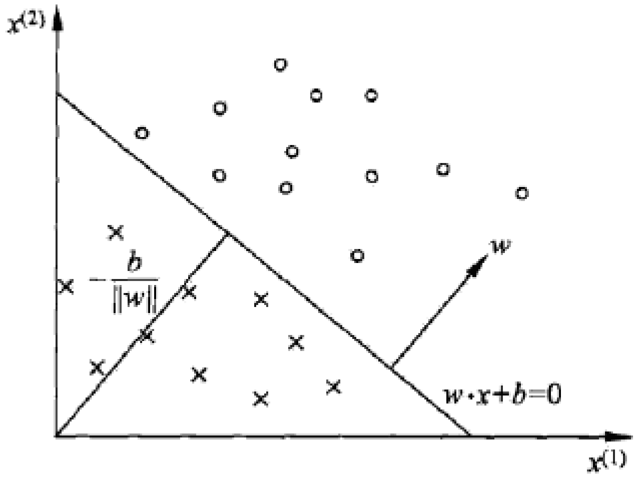
$$y = sign(\omega * x + b) = \begin{cases} +1, \omega * x + b \geq 0 \\ -1, \omega * x + b < 0 \end{cases}$$

ω 为权值（weight），b 为偏置（bias）

感知机的几何解释

线性方程 $\omega * x + b = 0$ 对应于特征空间 R^n 中的一个超平面，其中 ω 对应的是法向量，b 对应的是截距

任何特征空间上的实例，会落在超平面的两侧，以此为依据可以划分为正、负两类



感知机的学习过程

假设已有训练数据集 $T = \{(x_1, y_1), (x_2, y_2), (x_3, y_3) \cdots, (x_n, y_n)\}$

求解最合适的参数 ω 和 b 的问题，等效于求损失函数极小值 $min(Loss(\omega, b)) = min(-\sum y_i(\omega * x_i + b))$

损失函数 $Loss(\omega, b) = -\sum y_i(\omega * x_i + b)$

- 假设分类正确，y_i 和 $\omega * x_i + b$ 符号相同，则 $-y_i(\omega * x_i + b)$ 使得 $Loss(\omega, b)$ 减小
- 假设分类错误，y_i 和 $\omega * x_i + b$ 符号不同，则 $-y_i(\omega * x_i + b)$ 使得 $Loss(\omega, b)$ 增大

采用随机梯度下降的方式，每次选出一个误分类点来更新 ω 和 b

$$\frac{\delta Loss(\omega, b)}{\delta \omega} = -y_i x_i \quad \omega_{k+1} = \omega_k + \lambda * y_i x_i$$

$$\frac{\delta Loss(\omega, b)}{\delta b} = -y_i \quad \omega_{k+1} = \omega_k + \lambda * y_i$$

不断迭代，直到损失函数减小到一定的值，就可以视为学习完成