人工神经网络理论及应用

尹达恒
（江南大学物联网工程学院，江苏　无锡）

摘要：人工神经网络 (Artificial neural network,ANN) 是现代人工智能研究的重要基石，它克服了过去人工智能中被认为难以解决的一些问题，且随着训练数据量的显著增长以及芯片处理能力的剧增，它在目标检测和计算机视觉、自然语言处理、语音识别和语义分析等领域成效卓然。本文介绍了人工神经网络的基本思想、发展历史和理论基础，综述了目前人工神经网络的主流技术并给出了一些范例，最后探讨了人工神经网络今后的研究方向。

关键词：回归分析，回归算法，机器学习

The Theory and Application of Artificial Neural Networks

YING Da-heng
(School of IoT, Jiangnan University, Wuxi Jiangsu, China)

Abstract: Artificial neural network (ANN) is an important foundation in the field of artificial intelligence. They overcome some problems that were considered difficult to solve for artificial intelligence in the past. And with the significant increase in the amount of training data and the surge in chip processing power, it has achieved outstanding results in the areas of target detection, computer vision, natural language processing, speech recognition and semantic analysis. This paper introduces the basic ideas, development history and theoretical basis of artificial neural network, summarizes the mainstream technology of artificial neural network with some examples, and finally discusses the future research direction of artificial neural network.

Keywords: regression analysis, regression algorithm, machine learning

1 简介

人工神经网络 (Artificial neural network,ANN)，又称连接系统 (Connectionist system) 是受生物神经网络启发而发明的一类信息处理系统。它是从信息处理角度对人脑神经元网络进行抽象而构建出的以有向图为拓扑结构、通过对连续或离散的输入作状态响应而进行信息处理的系统 [1]。这种信息处理系统集合了多种相互协作的机器学习算法，能够在不对特定任务规则（先验知识）进行编程的情况下从示例中“学习”到抽象的特征表示，并且这些特征表示具有良好的泛化能力 [2] [3]。

人工神经网络的基础是一系列互相连接的单元或节点，称为人工神经元 (Artificial neurons)。神经元之间的连接 (Connection) 能在神经元之间传递一些信号。当一个神经元从神经元连接中得到信号之后，这个神经元能对能对信号进行一

些处理并将处理后的信号通过神经元连接再发送给其他的神经元。在人工神经网络的计算过程中，每个神经元都重复着接收信号、处理信号、发送信号的过程，使得信息能在神经网络中传递并变化 [4]。这种信息的不断传递与变化就是人工神经网络进行信息处理的基本方式。

对人工神经网络的研究起始于 1943 年 McCulloch 和 Pitts 提出的神经元模型 [5]。1951 年，Donala O.Hebb 提出的连接权值强化的 Hebb 法则 [6] 为构造有学习功能的神经网络模型奠定了基础。然而到了 1969 年，Minsky 和 Papert 经过多年研究，指出当时的单层神经网络只能应用于简单的线性问题，而不能有效构建多层网络 [7]，由此开始了神经网络的第一个低谷期。直到 1986 年，Rumehart 和 Hinton 等人提出了用于训练多层神经网络的反向传播算法 [8]，解决了原来一些单层神经网络所不能解决的问题，由此掀起了第二次神经网络研究热潮。但传统的反向传播网络在增加深度时会遇到局部最优、过拟合及梯度扩散等问题，再加之 20 世纪 90 年代支持向量机 [9] 等一系列经典的浅层机器学习模型的提出，使得神经网络模型再次陷入低潮。2006 年，Hinton 等人在《Science》上发文指出 [10]：(1) 多隐层的人工神经网络具有优异的特征学习能力；(2) 可通过“逐层预训练”(layer-wise pre-training) 来有效克服深层神经网络在训练上的困难，从此引出了深度学习 (Deep Learning) 的研究，同时也掀起了人工神经网络研究的又一热潮 [11]。时至今日，随着计算机硬件处理能力的不断提高和神经网络训练数据量的不断增长，人工神经网络及其衍生出的各类深度学习模型正在越来越多的领域取得卓越成效。

本文将首先介绍人工神经网络的基本理论，随后从模型入手介绍一些主流的人工神经网络理论以及一些范例，并在此基础上对当今广泛使用的一些神经网络模型进行综述，最后简要探讨人工神经网络未来的发展方向。

2 人工神经网络的基本理论

2.1 人工神经元

人工神经元 (Artificial Neuron) 是人工神经网络的基本处理单元，是神经网络中最基本的组成部分。在生物神经网络中，每个神经元都与其他神经元相连，当神经元被激活 (activate) 时，就会向相连的神经元发送化学物质改变这些神经元内的电位；如果某神经元的电位超过一定阈值 (threshold)，那么它就会被激活，进而向其他神经元发送化学物质 [12]。1943 年，McCulloch 和 Pitts 对神经元的活动进行了一定程度的抽象，提出了人工神经元的基本模型 [5]，称为“MP 神经元模型”。MP 神经元模型是一个多自变量单因变量的函数，包含输入、偏置、

激活三个部分，其结构模型如图 1所示。

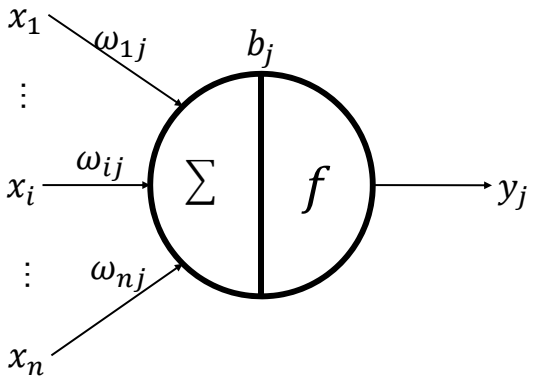


图 1. 人工神经元结构图

其中 x_i 表示输入信号，n 表示输入神经元 j 的信号个数，ω_{ij} 表示输入信号 x_i 与神经元 j 之间连接的权值，b_j 表示神经元 j 内部的偏置 (bias)，y_j 为神经元 j 的输出。在神经元 j 中，输入与输出之间的关系可以用如下公式表示：

$$y_j = f\left(b_j + \sum_{i=1}^{n} \omega_{ij} x_i\right) \tag{1}$$

其中 $f(\cdot)$ 为激活函数 (activate function)，它的形式多种多样。若将 b_j 视为一个恒为 1 的输入 x_0 通过一个 $\omega_{0j} = b_j$ 的权值与神经元相连，则输入与输出关系表达式可以简化为：

$$y_j = f\left(\sum_{i=0}^{n} \omega_{ij} x_i\right) \quad (x_0 = 1, \omega_{0j} = b_j) \tag{2}$$

进一步用向量形式表示为：

$$\begin{aligned} y_j &= f\left(\boldsymbol{\omega}^T \boldsymbol{x}\right) \\ \boldsymbol{\omega} &= [b_j, \omega_{1j}, \ldots, \omega_{ij}, \ldots, \omega_{nj}]^T \\ \boldsymbol{x} &= [1, x_1, \ldots, x_i, \ldots, x_n]^T \end{aligned} \tag{3}$$

在人工神经元连接而成的神经网络中，特征并非集中表征于某个函数或变量，而是分散在各个神经元连接的权值中，这种表征方式被称为分布式表征 (distributed representation) [13]。和传统的局部表征 (localized representation) 相比，分布式表征对特征存储效率较高。线性增加的神经元数目可以表达指数级增加的大量不同概念。这使得类似于神经网络的分布式表征系统大都具有较高的适应性。除此之外，分布式表征的另一个优点是，即使局部出现硬件故障，信息的表达也不会受到根本性的破坏。这使得神经网络对于样本中出现的错误具有较高的容错性。

但是，单有人工神经元组成的神经网络是不能自行改变神经网络中的连接权值的，要想使得人工神经网络具有“学习”能力，调节人工神经元连接权值的算法必不可少。对人工神经网络权值进行调节的算法多种多样，主要分为两大类：监督学习 (Supervised Learning) 与无监督学习 (Unsupervised Learning)。

2.2 监督学习的基础——Delta 规则

Delta 规则 (Delta rule) 又称 Widrow-Hoff 规则 (Widrow-Hoff rule) [14]，是一种简单的有导师学习算法。该算法根据神经元的实际输出与期望输出差别来调整连接权，其数学表示如下：

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \alpha(y_i(t) - \hat{y}_i(t))x_j(t) \tag{4}$$

其中 $\omega_{ij}(t)$ 表示计算第 t 个样本时神经元 j 到神经元 i 的连接权值，$y_i(t)$ 表示神经元 i 期望输出，$\hat{y}_i(t)$ 表示神经元 i 的实际输出，$x_j(t)$ 表示神经元 j 的状态，若神经元 j 处于激活态则 $x_j(t)$ 为 1，若处于抑制状态则 $x_j(t)$ 为 0 或-1（根据激活函数而定）；α 是表示学习速度的常数，称为学习率 (Learning Rate)。假设 $x_j(t)$ 为 1，若 $\hat{y}_i(t) > y_i(t)$，那么 ω_{ij} 将增大，反之 ω_{ij} 将变小。

简言之，Delta 规则就是：若神经元实际输出比期望输出大，则减小所有输入为正的连接的权重，增大所有输入为负的连接的权重，从而减小神经元实际输出。反之，若神经元实际输出比期望输出小，则增大所有输入为正的连接的权重，减小所有输入为负的连接的权重，从而增大神经元实际输出。这个增大或减小的幅度由学习率 α 和神经元实际输出与期望输出的差距共同决定。

2.3 无监督学习的基础——赫布理论

赫布理论 (Hebbian theory) 又称细胞结集理论 (cell assembly theory) 是一个神经科学理论，解释了在学习的过程中脑中的神经元所发生的变化。赫布理论描述了突触可塑性的基本原理，即突触前神经元向突触后神经元的持续重复的刺激，可以导致突触传递效能的增加 [6]。赫布理论可以用于解释“联合学习” (associative learning)，在这种学习中，由对神经元的重复刺激，使得神经元之间的突触强度增加。这样的学习方法被称为赫布型学习 (Hebbian learning)，是最古老的也是最简单的神经元学习规则。赫布理论也由此成为了非监督学习的生物学基础。

描述赫布学习规则的公式非常简洁：

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \alpha y_i(t) y_j(t) \tag{5}$$

其中 $\omega_{ij}(t)$ 表示计算第 t 个样本时神经元 j 到神经元 i 的连接权值，$y_i(t)$ 与 $y_j(t)$ 表示计算第 t 个样本时神经元 i 和神经元 j 的输出，输出为正表示激活，输出为负表示抑制；α 是表示学习速度的常数。若 y_i 与 y_j 同时被激活，即 y_i 与 y_j 同时为正，那么 ω_{ij} 将增大，表示神经元之间的突触强度增加。若 y_i 被激活，而 y_j 处于抑制状态，即 y_i 为正 y_j 为负，那么 ω_{ij} 将变小，表示神经元之间的突触强度减小。

3 多层感知机

多层感知机 (Multilayer Perceptron,MLP) 是一种由人工神经元组成网络结构，由输入层、隐藏层和输出层构成。多层感知机每一层都包含一个或多个神经元，各层的神经元都以上一层的输出作为输入，并将该层的输出传递到下一层的输入中 [15]。例如，一个典型的含有两个隐藏层的的多层感知机结构如图 2。

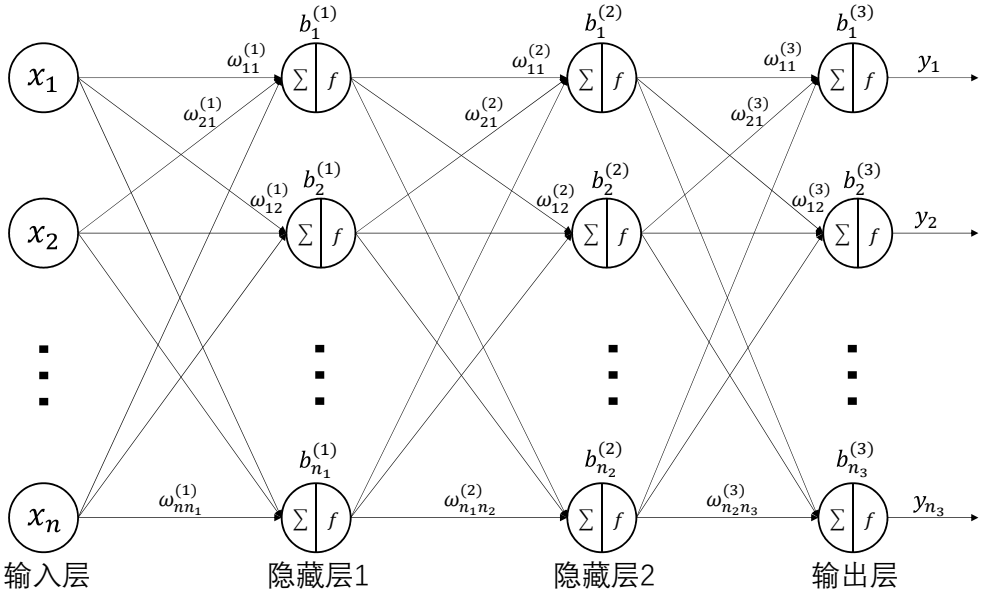


图 2. 多层感知机结构图

其中 n 表示输入层的变量个数，x_i 表示输入层各变量的值，n_l 表示第 l 层的神经元个数，$\omega_{ij}^{(l)}$ 表示第 $(l-1)$ 层的第 i 个神经元和第 l 层的第 j 个神经元之间的连接权值，y_i 表示输出层各神经元的输出值，通常也作为整个多层感知机的输出。如果以 $y_j^{(l)}$ 表示隐藏层 l 中的第 j 个神经元的输出，且令输出层为第 L 层，则多

层感知机的输出 y_j 可以由如下递推公式给出：

$$\begin{aligned} y_j &= y_j^{(L)} && (1 \le j \le n_L) \\ y_j^{(l)} &= f\left(b_j^{(l)} + \sum_{i=1}^{n_{l-1}} \omega_{ij}^{(l)} y_i^{(l-1)}\right) && (2 \le l \le L, 1 \le j \le n_l) \\ y_j^{(1)} &= f\left(b_j^{(1)} + \sum_{i=1}^{n} \omega_{ij}^{(1)} x_i\right) && (1 \le j \le n) \end{aligned} \tag{6}$$

与 2.1节类似，若将 $b_j^{(l)}$ 视为一个恒为 1 的输入 $y_0^{(l-1)}$ 通过一个 $\omega_{0j}^{(l)} = b_j^{(l)}$ 的权值与神经元相连，并且将输入层看作第 $l = 0$ 层，则递推公式 (6) 可化为向量形式：

$$\begin{aligned} y_j^{(l)} &= f\left((\boldsymbol{\omega}_j^{(l)})^T \boldsymbol{y}^{(l-1)}\right) && (1 \le j \le n_l, 1 \le l \le L) \\ \boldsymbol{\omega}_j^{(l)} &= \left[b_j^{(l)}, \omega_{1j}^{(l)}, \ldots, \omega_{ij}^{(l)}, \ldots, \omega_{n_{l-1}j}^{(l)}\right]^T && (1 \le j \le n_l, 1 \le l \le L) \\ \boldsymbol{y}^{(l)} &= \left[1, y_1^{(l)}, \ldots, y_i^{(l)}, \ldots, y_{n_l}^{(l)}\right]^T && (1 \le l \le L) \\ \boldsymbol{y}^{(0)} &= [1, x_1, \ldots, x_i, \ldots, x_n]^T \end{aligned} \tag{7}$$

对于一个多层感知机，在已知输入和各层神经元连接权值的情况下，通过递推公式 (7)，可以从 $l = 0$ 开始逐层计算神经元的输出，直到最后得到多层感知机输出层的的输出值 $\boldsymbol{y}^{(L)}$，这个过程也被称为前向传播 (forward propagation) [16]。更进一步，若将层与层之间的权值 $\boldsymbol{\omega}_j^{(l)}$ 用权值矩阵 $\boldsymbol{W}$ 的形式表示，则递推公式 (7) 可化为矩阵形式：

$$\begin{aligned} \boldsymbol{y}^{(l)} &= f\left(\boldsymbol{W}^{(l)} \boldsymbol{y}^{(l-1)}\right) && (1 \le l \le L) \\ \boldsymbol{W}^{(l)} &= \begin{bmatrix} b_1^{(l)} & \omega_{11}^{(l)} & \ldots & \omega_{i1}^{(l)} & \ldots & \omega_{n_{l-1}1}^{(l)} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ b_j^{(l)} & \omega_{1j}^{(l)} & \ldots & \omega_{ij}^{(l)} & \ldots & \omega_{n_{l-1}j}^{(l)} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ b_{n_l}^{(l)} & \omega_{1n_l}^{(l)} & \ldots & \omega_{in_l}^{(l)} & \ldots & \omega_{n_{l-1}n_l}^{(l)} \end{bmatrix} && (1 \le l \le L) \\ \boldsymbol{y}^{(0)} &= [1, x_1, \ldots, x_i, \ldots, x_n]^T \end{aligned} \tag{8}$$

递推公式 (8) 即多层感知机前向传播公式的矩阵形式。

多层感知机是神经网络最基本、使用最广泛的形式。对于一系列给定输入 $\boldsymbol{x}$ 和对应的输出 $\boldsymbol{y}$，多层感知机可以通过改变各层的连接权值，使得对于每个给定的输入 $\boldsymbol{x}$，多层感知机给出输出向量 $\hat{\boldsymbol{y}}$ 和给定输出 $\boldsymbol{y}$ 尽可能的逼近 (approximate)。George Cybenko 曾从数学上证明：包含一个及以上隐藏层的神经网络可以被用来按照任意给定的精度来逼近任何连续函数，且近似精度随隐藏层神经元个数

的增加而上升 [17]。这个定理表明多层感知机是一种具有普适性的函数拟合器，这也是众多深度学习算法有效性的基础。

图 3和图 4展示了 8 个输入层有两个神经元，输出层有一个神经元的多层感知机的输入和输出关系。这 8 个多层感知机的权值矩阵均为随机生成的在 0-1 之间的浮点数，各层激活函数均为 ReLU 函数。图 3中所示的 4 个多层感知机隐藏层由 20 个神经元组成，而图 3中所示的 4 个多层感知机隐藏层由 200 个神经元组成。由图中我们可以看出，增加了隐藏层的神经元个数后，多层感知机的输出变得更加“灵活”，这也印证了 George Cybenko 所证明的神经网络的普适性定理。

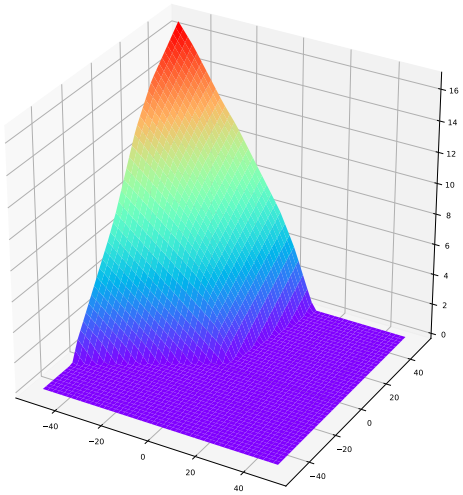


图 3. 隐藏层 20 神经元的多层感知机

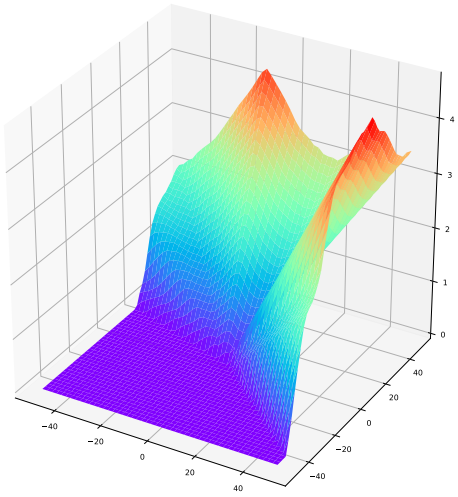


图 4. 隐藏层 200 神经元的多层感知机

4 反向传播

反向传播 (Back propagation) 是 Rumehart 和 Hinton 等人于 1986 年提出的用于训练多层神经网络的算法 [8]，其本质类似 Delta 规则，也是一种按照输出误差对权值进行迭代调整的算法。进行反向传播的方法是梯度下降 (Gradient Descent)，即在每次迭代过程中，对于网络中的每个权值，都按照它相对于误差的梯度方向进行调整，从而递归性地逼近具有最小误差的模型。在一个多层感知机中，梯度下降有如下公式：

$$\omega_{ij}^{(l)*} = \omega_{ij}^{(l)} - \alpha \frac{\partial E}{\partial \omega_{ij}^{(l)}} \quad (1 \leq l \leq L) \tag{9}$$

其中，$\omega_{ij}^{(l)}$ 表示网络中的某个权值，$\omega_{ij}^{(l)*}$ 表示一次迭代后的权值，E 表示网络的误差，α 表示学习率。反向传播的关键是求出权值的梯度 $\frac{\partial E}{\partial \omega_{ij}^{(l)}}$。首先考察最后一

层的情况，若已知误差 E 与输出层输出值 $\boldsymbol{y}^{(L)}$ 之间的函数关系式 $E = E(\boldsymbol{y}^{(L)})$，即损失函数 (Loss Function)，则可以由链式法则求得最后一层权值相对于误差的梯度：

$$\begin{aligned}\frac{\partial E}{\partial \omega_{ij}^{(L)}} &= \frac{\partial E}{\partial y_j^{(L)}} \frac{\partial y_j^{(L)}}{\partial \omega_{ij}^{(L)}} \\ &= \frac{\partial E}{\partial y_j^{(L)}} \frac{\partial f\left((\boldsymbol{\omega}_j^{(L)})^T \boldsymbol{y}^{(L-1)}\right)}{\partial (\boldsymbol{\omega}_j^{(L)})^T \boldsymbol{y}^{(L-1)}} \frac{\partial (\boldsymbol{\omega}_j^{(L)})^T \boldsymbol{y}^{(L-1)}}{\partial \omega_{ij}^{(L)}} \\ &= \frac{\partial E}{\partial y_j^{(L)}} f'\left((\boldsymbol{\omega}_j^{(L)})^T \boldsymbol{y}^{(L-1)}\right) y_i^{(L-1)}\end{aligned} \tag{10}$$

若令 $t_j^{(l)} = (\boldsymbol{\omega}_j^{(l)})^T \boldsymbol{y}^{(l-1)}$ 表示第 l 层第 j 个神经元激活前的值，其中 $1 \le l \le L$，则公式 (10) 可以简写为：

$$\frac{\partial E}{\partial \omega_{ij}^{(L)}} = \frac{\partial E}{\partial t_j^{(L)}} \frac{\partial t_j^{(L)}}{\partial \omega_{ij}^{(L)}} = \frac{\partial E}{\partial y_j^{(L)}} f'\left(t_j^{(L)}\right) y_i^{(L-1)} \tag{11}$$

类似的，对于网络中的每一个连接的权值，可以得到：

$$\frac{\partial E}{\partial \omega_{ij}^{(l)}} = \frac{\partial E}{\partial t_j^{(l)}} \frac{\partial t_j^{(l)}}{\partial \omega_{ij}^{(l)}} = \frac{\partial E}{\partial t_j^{(l)}} y_i^{(l-1)} \quad (1 \le l \le L) \tag{12}$$

公式 (12) 表明，连接权值误差的梯度可以表示为神经元激活前的值相对于误差的偏导与该神经元所连接的前一个神经元值的积，其中，神经元激活前的值相对于误差的偏导 $\frac{\partial E}{\partial t_j^{(l)}}$ 又被称为误差信号 [18] [19]，用 $E_j^{(l)}$ 表示。因此连接权值误差的梯度表达式 (12) 可以进一步简化为：

$$\frac{\partial E}{\partial \omega_{ij}^{(l)}} = E_j^{(l)} y_i^{(l-1)} \quad (1 \le l \le L) \tag{13}$$

当已知第 $l+1$ 层的误差信号 $E_j^{(l+1)}$ 时，由链式法则可以求得第 l 层 $(1 \leq l \leq L-1)$ 的误差信号 $E_i^{(l)}$ 为：

$$\begin{aligned} E_i^{(l)} &= \frac{\partial E}{\partial t_i^{(l)}} \\ &= \frac{\partial E}{\partial y_i^{(l)}} \frac{\partial y_i^{(l)}}{\partial t_i^{(l)}} \\ &= \sum_{j=0}^{n_j} \left(\frac{\partial E}{\partial t_j^{(l+1)}} \frac{\partial t_j^{(l+1)}}{\partial y_i^{(l)}} \right) \frac{\partial f\left(t_i^{(l)}\right)}{\partial t_i^{(l)}} \\ &= \sum_{j=0}^{n_j} \left(E_j^{(l+1)} \omega_{ij}^{(l+1)} \right) f'\left(t_i^{(l)}\right) \end{aligned} \tag{14}$$

公式 (14) 表明，第 l 层的误差信号可以用 $l+1$ 层的误差信号和权值表示，同时说明误差信号在网络中从最后一层开始逐步向第一层传播。这也是“反向传播”名称的由来。此外，由公式 (10) 可知，第 L 层的误差信号表达式为：

$$E_j^{(L)} = \frac{\partial E}{\partial y_j^{(L)}} f'\left(t_j^{(L)}\right) \tag{15}$$

综合公式 (9)、公式 (13)、公式 (14) 和公式 (15)，可以得到完整的反向传播公式：

$$\begin{cases} \omega_{ij}^{(l)*} = \omega_{ij}^{(l)} - \alpha E_j^{(l)} y_i^{(l-1)} & (1 \leq l \leq L) \\ E_i^{(l)} = \sum_{j=0}^{n_j} \left(E_j^{(l+1)} \omega_{ij}^{(l+1)} \right) f'\left(t_i^{(l)}\right) & (1 \leq l \leq L-1) \\ E_j^{(L)} = \frac{\partial E}{\partial y_j^{(L)}} f'\left(t_j^{(L)}\right) \end{cases} \tag{16}$$

更进一步，若用公式 (8) 中所示的矩阵方法表示权值和各层输出值，并以向量形式表示误差信号和神经元激活前的值，则可以得到如下所示的反向传播的矩阵形式：

$$\begin{cases} \boldsymbol{W}^{(l)*} = \boldsymbol{W}^{(l)} - \alpha \boldsymbol{E}^{(l)} (\boldsymbol{y}^{(l-1)})^T \\ \boldsymbol{E}^{(l)} = \left((\boldsymbol{W}^{(l+1)})^T \boldsymbol{E}^{(l+1)} \right) * f'\left(\boldsymbol{t}^{(l)}\right) \\ \boldsymbol{E}^{(L)} = \frac{\partial E}{\partial \boldsymbol{y}^{(L)}} * f'\left(\boldsymbol{t}^{(L)}\right) \end{cases} \tag{17}$$

其中 $*$ 表示向量按位相乘。在实际计算过程中，由于计算机是不能直接求偏导的，因此反向传播公式中输出层对损失函数偏导 $\frac{\partial E}{\partial \boldsymbol{y}^{(L)}}$ 的表达式和激活函数的导数 $f'(\cdot)$ 表达式一般由人为预先指定。

常用的损失函数有：

(1) 用于二分类的交叉熵 (Corss Entropy)：

$$\begin{aligned} E(\boldsymbol{y}^{(L)}) &= -\left(\boldsymbol{y}_0^T ln(\boldsymbol{y}^{(L)}) + (1-\boldsymbol{y}_0)^T ln(1-\boldsymbol{y}^{(L)})\right) \\ \frac{\partial E}{\partial \boldsymbol{y}^{(L)}} &= -\left(\boldsymbol{y}_0^T \frac{1}{\boldsymbol{y}^{(L)}} - (1-\boldsymbol{y}_0)^T \frac{1}{1-\boldsymbol{y}^{(L)}}\right) \end{aligned} \tag{18}$$

其中 $\boldsymbol{y}^{(L)}$ 表示输出层的输出值，$\boldsymbol{y}_0$ 表示期望的输出，若分类为真则 $y_0 = 1$，反之 $y_0 = 0$。

(2) 用于函数拟合的均方误差 (Mean Square)：

$$\begin{aligned} E(\boldsymbol{y}^{(L)}) &= (\boldsymbol{y}^{(L)} - \boldsymbol{y}_0)^T (\boldsymbol{y}^{(L)} - \boldsymbol{y}_0) \\ \frac{\partial E}{\partial \boldsymbol{y}^{(L)}} &= 2(\boldsymbol{y}^{(L)} - \boldsymbol{y}_0) \end{aligned} \tag{19}$$

常用的激活函数有：

(1) Sigmoid 函数 [20]：

$$\begin{aligned} f(x) &= \frac{1}{1-e^{-x}} \\ f'(x) &= f(x)(1-f(x)) \end{aligned} \tag{20}$$

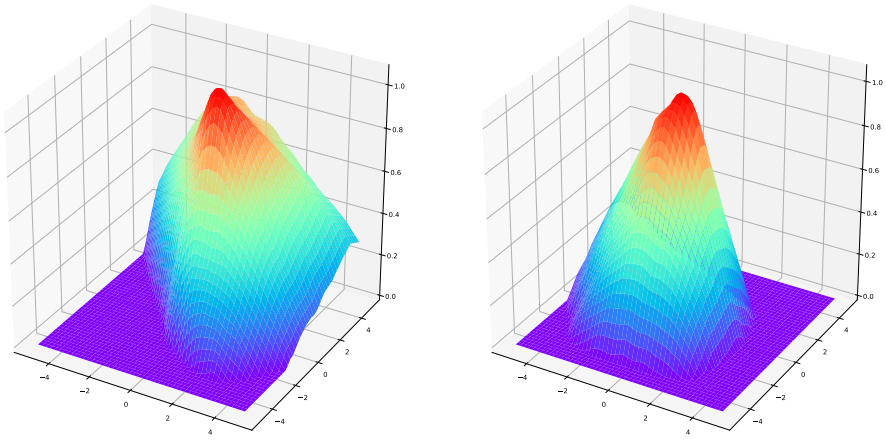
(2) tanh 函数 [21]：

$$\begin{aligned} f(x) &= tanh(x) \\ f'(x) &= 1 - f(x)^2 \end{aligned} \tag{21}$$

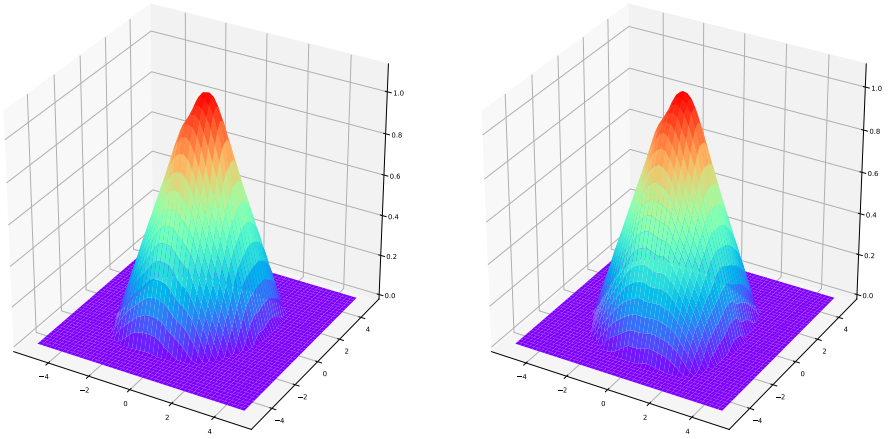
(3) 线性纠正函数 (Rectified Linear Unit,ReLU) [22]：

$$\begin{aligned} f(x) &= max(x, 0) \\ f'(x) &= \begin{cases} 1 & x \geq 0 \\ 0 & x < 0 \end{cases} \end{aligned} \tag{22}$$

图 5表示 3节中介绍的隐藏层包含 200 个神经元的多层感知机在学习率 $\alpha = 0.01$ 的情况下使用均方误差和 ReLU 函数拟合目标函数 $y = cos\left(x_1^2 + x_2^2\right)$ 时（图 7）输入输出关系的变化。(a)、(b)、(c)、(d) 依次表示第 100 步、第 200 步、第 300 步和第 400 步时多层感知机的输入输出关系。从图中可以看出，尽管多层感知机内部不包含任何与目标函数相关的计算，但随着训练的进行，多层感知机整体的输出越来越接近目标函数（图 6）。这印证了神经网络能在没有任何先验知识的条件下，能通过调节神经元之间的连接权值对非线性函数进行拟合。



(a) (b)



(c) (d)

图 5. 训练过程中输入输出关系变化图

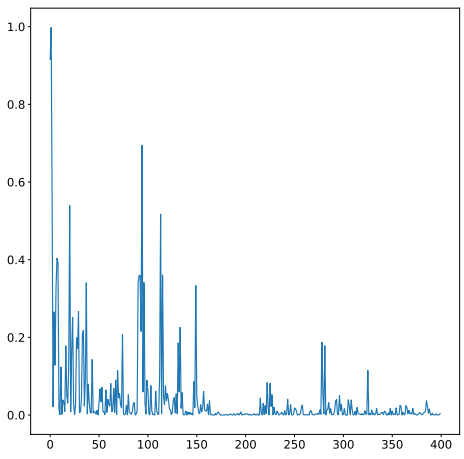


图 6. 训练过程中输出误差变化图

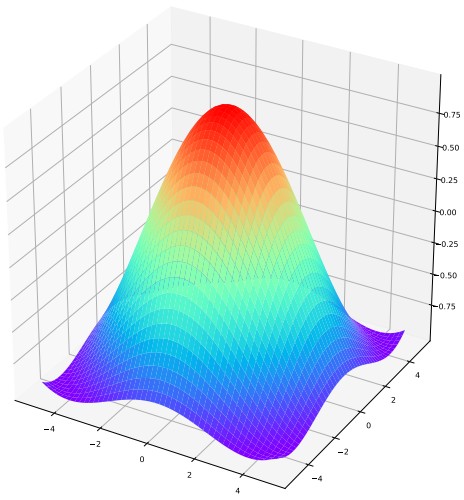


图 7. 期望的输入输出关系

5 正则化

正则化 (Regularization) 原本是用在线性回归中用于对不适定问题 (ill-posed problem) 进行回归的一种分析方法，后来被推广到人工智能领域，用于解决机器学习中的过拟合 (over-fitting) 问题 [23]，即神经网络在训练集上表现很好而在测试集上表现较差的现象。正则化的基本思想是通过在训练中利用一些与过拟合有关的额外信息，对网络的过拟合进行抑制。在人工神经网络中使用的正则化方法主要有利用权值大小信息的 L1 正则化 [24] 和 L2 正则化 [25]、利用神经元之间依赖性信息的 Dropout [26]、利用神经网络测试集表现的提前停止 (early stopping) [27] 等。

5.1 L1 正则化和 L2 正则化

L1 正则化和 L2 正则化都是通过修改神经网络的损失函数实现对神经网络中过大的连接权值进行抑制，进而达到使神经网络正则化的目的。

以 3中的多层感知机为例，加入 L1 正则化和 L2 正则化操作后，神经网络的损失函数定义如下：

$$E_{Regularization} = E + R(\boldsymbol{W}) \tag{23}$$

其中 E 为不加正则化时的网络损失函数，$R(\boldsymbol{W})$ 为正则化项。在 L1 正则化时，$R(\boldsymbol{W})$ 表达式为：

$$R(\boldsymbol{W}) = \sum_{i,j,l} |\omega_{ij}^{(l)}| \tag{24}$$

即网络中所有神经元连接权值的绝对值之和。在 L2 正则化时，$R(\boldsymbol{W})$ 表达式为：

$$R(\boldsymbol{W}) = \sum_{i,j,l} (\omega_{ij}^{(l)})^2 \tag{25}$$

即网络中所有神经元连接权值的平方之和。可以看出，不论是 L1 正则化还是 L2 正则化，基本的思想都是希望通过在损失函数中对过大的神经元连接权值添加“惩罚”项，减小使得神经网络的输入对输出的影响，进而使模型不能任意拟合训练数据中的随机噪声，从而达到减少过拟合的目的。

5.2 Dropout

Dropout 方法最先由 Hinton 于 2012 年提出，是一种通过阻止一个较为复杂的前馈神经网络中有过多神经元的共同作用来防止神经网络过拟合的方法 [26]。

Dropout 可以作为训练深度神经网络的一种优化方法供选择。在每个训练批次中，通过忽略一定数量的神经元输出，可以明显地减少过拟合现象。这种方式可以减少隐藏层神经元间的依赖性（一个神经元需要靠其他神经元的激活才能发挥作用）。实际应用中，Dropout 方法以 Dropout 层的形式存在于神经网络中，一个 Dropout 层由多个 Dropout 单元构成，有如下性质 [28]：

(1) Dropout 层的 Dropout 单元个数与上一层的神经元个数相同；

(2) 当前向传播算法计算到 Dropout 层时，Dropout 单元的输出为（设 Dropout 层为第 l 层）：

$$\begin{aligned} y_j^{(l)} &= y_j^{(l-1)} \frac{d_j^{(l)}}{p} \\ d_j^{(l)} &= \begin{cases} 1 & (\text{概率为} p) \\ 0 & (\text{概率为} 1-p) \end{cases} \end{aligned} \tag{26}$$

其中，$d_j^{(l)}$ 为 Dropout 单元的清除标记，p 为预先指定的 Dropout 层的失活概率。

(3) 当反向传播算法计算到 Dropout 层时，Dropout 单元的误差信号为：

$$E_j^{(l)} = E_j^{(l-1)} \frac{d_j^{(l)}}{p} \tag{27}$$

其中，清除标记 $d_j^{(l)}$ 在上一次前向传播时确定。

Dropout 层的性质可以看出，Dropout 层实际上是用于在训练阶段使前一层的神经元随机“失活”，“失活”的神经元在前向传播时输出为 0，其余神经元的输出按设定失活概率的倒数增大，从而使得传递到下一层的输出总和与 Dropout 层时保持一致；反向传播时与失活神经元之间相连的神经元连接权值不进行更新（变化为 0）。

5.3 提前停止

提前停止方法的基本假设是：一个会出现过拟合的神经网络，其过拟合程度随训练次数的增加而增加 [27]。在这个假设条件下，一定存在某一个时刻，神经网络在训练集上的表现和其过拟合的程度达到某种折中的状态，即网络训练得恰到好处，既有较好的性能，又没有出现很强的过拟合。提前停止的核心即是要在训练过程中找到这种恰到好处的状态，并停止训练，防止网络出现过拟合从而达到正则化的目的。

神经网络的过拟合一般通过神经网络在训练集和测试集上的性能差异来评价，当神经网络在训练集上表现较好，但在测试集表现较差，说明神经网络已经出现了过拟合。按照这种方法，使用提前停止方法对神经网络进行训练的步骤如下：

(1) 将原始的训练数据集划分成训练集和测试集；

(2) 只在训练集上进行训练，并每隔一个固定周期计算模型在测试集上的误差；

(3) 当模型在测试集上的误差比上一次训练结果差的时候停止训练；

(4) 使用上一次迭代结果中的参数作为模型的最终参数。

6 卷积神经网络

卷积神经网络 (Convolutional Neural Network,CNN) 是对一般多层感知机的一种扩展，广泛应用于计算机视觉、自然语言处理等领域 [29]。在卷积神经网络中，除了输出部分的多层感知机（又称全连接层）还包含一个或多个卷积层 (convolutional layer)、激活层 (Rectified Linear Units layer,ReLU layer) 和池化层 (pooling layer) [30]。

6.1 卷积层

卷积神经网络中每层卷积层由若干卷积核组成，每个卷积核的参数都对应一个权重系数和一个偏置，与人工神经元类似。卷积层内每个卷积核都与前一层中位置接近的区域的多个神经元输出相连，区域的大小取决于卷积核的大小，连接区域又称“感受野 (receptive field)”，卷积核在工作时，会有规律地扫过输入特征，在感受野内对输入特征做矩阵元素乘法求和并叠加偏置，其表达式如下 [30]：

$$
\begin{aligned}
\boldsymbol{Y}^{(l+1)}(i,j) &= \left[\boldsymbol{Y}^{(l)} \otimes \boldsymbol{\omega}^{(l)}\right](i,j) + \boldsymbol{b} \\
&= \sum_{k=1}^{K_l} \sum_{x=1}^{f} \sum_{y=1}^{f} \left[\boldsymbol{Y}_k^{(l)}(s_0 i + x, s_0 j + y)\boldsymbol{\omega}_k^{(l+1)}(x,y)\right] + \boldsymbol{b} \\
i,j &\in [0, L_{l+1}] \\
L_{l+1} &= \frac{L_l + 2p - f}{s_0} + 1
\end{aligned}
\tag{28}
$$

其中 $\boldsymbol{b}$ 为偏差量，$\boldsymbol{Y}^{(l)}$ 表示第 l 层的卷积输出，也被称为特征图 (feature map)，L_{l+1} 为 $\boldsymbol{Y}^{(l)}$ 的尺寸，$\boldsymbol{Y}^{(l)}(i,j)$ 对应特征图中的像素，K_l 为特征图 $\boldsymbol{Y}^{(l)}$ 的通道数，f、s_0 和 p 分别对应卷积核大小、卷积步长 (stride) 和填充 (padding) 层数 [30]。一个典型的卷积核大小为 3、卷积步长为 1、填充层数为 0 的单通道二维卷积运算如图 8所示。

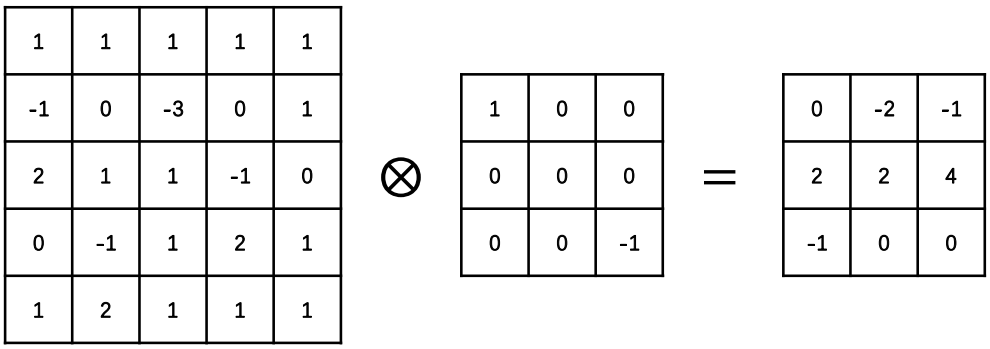


图 8. 二维卷积运算

卷积运算的目的是提取输入的不同特征，第一层卷积层可能只能提取一些低级的特征如边缘、线条和角等层级，更多层的网路能从低级特征中迭代提取更复杂的特征。

6.2 激活层

卷积神经网络的激活层和多层感知机的激活函数类似，都是将线性输入映射到非线性输出，以协助表达复杂特征。其表达式如下 [30]：

$$\boldsymbol{Y}^{(l)}(i,j) = f\left(\boldsymbol{Y}^{(l-1)}(i,j)\right) \tag{29}$$

其中 $f(\cdot)$ 为激活函数,通常采用 ReLU 函数及它的一些变体如有斜率的 ReLU(Leaky ReLU, LReLU)、参数化的 ReLU(Parametric ReLU, PReLU)、指数线性单元 (Exponential Linear Unit, ELU) 等。

6.3 池化层

在卷积层进行特征提取并经过激活层后，输出的特征图会被传递至池化层进行特征选择和信息过滤。卷积神经网络中一般采用最大值池化 (max pooling)，其表达式如下：

$$\boldsymbol{Y}^{(l)}(i,j) = max_{x=1\to f, y=1\to f}\left[\boldsymbol{Y}_k^{(l-1)}(s_0 i + x, s_0 j + y)\right] \tag{30}$$

一个典型的 2×2 最大值池化操作如图 9所示。

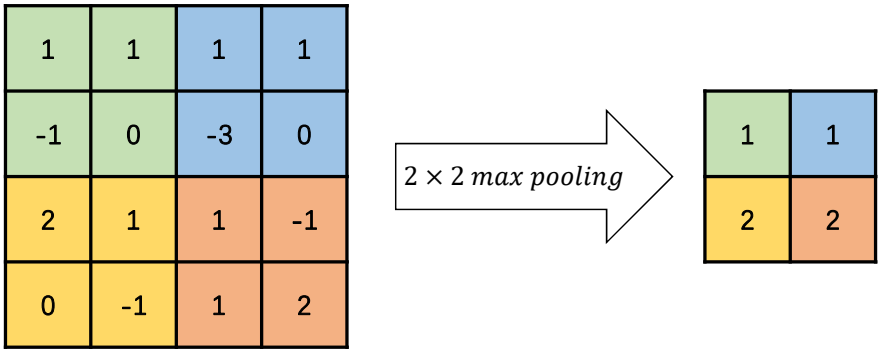


图 9. 最大值池化操作

7 人工神经网络应用实例

MNIST 是来自美国国家标准与技术研究所 (National Institute of Standards and Technology,NIST) 的一个手写数字识别库，是世界上最权威的数字手写体数据库；CIFAR-10 是由 Hinton 的学生 Alex Krizhevsky 和 Ilya Sutskever 整理的一个用于识别普适物体的小型数据集。这两个数据集是人工智能领域常用的基准测试。本文将以 MNIST 和 CIFAR-10 为例，结合前文所述的各类人工神经网络相关技术构建一个简单的图像识别模型，并对其进行评价。

7.1 构建数据集

MNIST 数据集中包含 70000 个 28×28 大小的数字手写体灰度图像，这些数据已经预先分为两个部分：包含 60000 个样本的训练集和包含 10000 个样本的测试集。CIFAR-10 数据集中包含 6 个样本集共计 60000 张 32×32 大小的 3 通道彩色图像，共有 10 个类别：飞机 (airplane)、汽车 (automobile)、鸟类 (bird)、猫 (cat)、鹿 (deer)、狗 (dog)、蛙类 (frog)、马 (horse)、船 (ship) 和卡车 (truck)。6 个样本集中 5 个用作训练集，一个作为测试集。训练集用于训练神经网络，测试集用于对训练完成的神经网络进行测试。

7.2 构建模型

手写体识别模型和 CIFAR-10 识别模型都基于经典的 LeNet 网络 [31]，其结构如图 10所示。模型包含一个边长为 5，输出通道数为 20 的卷积层、一个边长为 5，输出通道数为 50 的卷积层、两个激活层、两个 2×2 池化层、一个隐藏层

神经元数目为 50 的全连接层和一个 Softmax 层。其中全连接层和激活层所使用的激活函数为 ReLU 函数。手写体识别模型和 CIFAR-10 识别模型不同点在于，前者输入通道数为 1，后者输入通道数为 3。

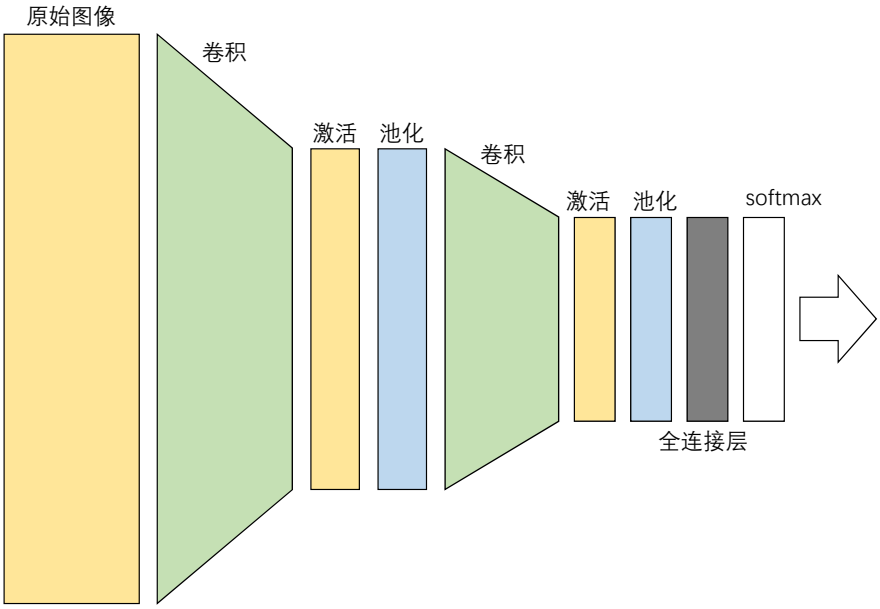


图 10. LeNet 网络结构

7.3 测试模型

7.3.1 基本模型测试

(1) 使用 MNIST 训练集中的全部 60000 个样本进行训练，使用测试集中的全部 10000 个样本对模型进行测试。训练-测试的过程重复 10 次可以得到 LeNet 在 MNIST 测试集上的平均正确率为 98.279%。

(2) 使用 CIFAR-10 训练集中的全部 50000 个样本进行训练，使用测试集中的全部 10000 个样本对模型进行测试。训练-测试的过程重复 10 次可以得到 LeNet 在 MNIST 测试集上的平均正确率为 61%。

7.3.2 增加全连接层神经元数量

将隐藏层神经元个数增加到 200，训练方案和测试方案同 7.3.1，可以得到 LeNet 在 MNIST 测试集上的平均正确率为 98.566%，在 CIFAR-10 测试集上的平均正确率为 69%。

7.3.3 添加 Dropout 层

在 7.3.2的基础上，在网络全连接层输出添加一个 $p = 0.5$ 的 Dropout 层，训练方案和测试方案同 7.3.1，可以得到 LeNet 在 MNIST 测试集上的平均正确率为 98.566%，在 CIFAR-10 测试集上的平均正确率为 63%。

7.4 结果分析

7.3.2中的结果印证了 3中增加隐藏层神经元数量能提高神经网络拟合精度的结论，但也证明在数据集较为简单时，增加隐藏层神经元数量对神经网络精度的提升不明显，而在较为复杂的 CIFAR-10 数据集上，增加 3 倍的隐藏层神经元个数能将网络的性能提升提升约 10%；7.3.3中的结果表明，隐藏层神经元数量为 200 时，LeNet 模型在较为简单的 MNIST 数据集训练的过程中存在一定的过拟合，通过添加 Dropout 层能进一步提高 LeNet 模型在 MNIST 测试集上的精度。但是在更为复杂的 CIFAR-10 数据集上，LeNet 模型中由 200 个神经元组成的隐藏层并不足以产生明显的过拟合现象，添加 Dropout 层反而会影响网络的训练效果，降低模型的测试精度。

8 总结

在人工神经网络提出至今的半个多世纪里，对于人工神经网络理论和应用的研究取得了长足的发展。神经元模型的多样化、神经网络结构的多样化、训练方式的多样化、以及突破分层的神经网络模型开发新的网络结构，是目前人工神经网络研究的突出发展方向。近年来，各类新兴神经元模型和神经网络模型的发展有效促进了回归分析技术的提升，例如递归神经网络 (Recursive neural network) [32]，循环神经网络 (Recurrent Neural Network,RNN) [33]、编码解码模型 (Encoder-Decoder model) [34]、长短时记忆网络 (Long Short-Term Memory,LSTM) [35]、残差网络 (ResNet) [36]、生成对抗网络 (Generative Adversarial Networks,GAN) [37] 等；与神经网络相关的各种训练与优化算法也在蓬勃发展，例如注意力 (Attention) 机制 [38]、Boosting 方法 [39]、各种改进的梯度下降法 [40] [41] [42] 等。人工神经网络及其相关领域的研究进展使人工智能在越来越多的场景中表现出超越人类的能力，也使得与人工神经网络和相关的训练与优化技术成为目前人工智能领域最受关注的发展方向。

本文对人工神经网络的基本思想方法进行了综述研究，并将人工神经网络相关的一些常见技术进行了归类介绍，并适当描述了各类方法之间、同类方法之

间的一些差别与联系。本文还详细分析了各类方法的部分理论细节，给出了一个简单的应用示例，最后分析了人工神经网络的主要发展趋势。希望本文能对相关科研人员了解人工神经网络并开展相关研究起到微薄的作用。

参考文献

[1] Robert Hecht-Nielsen. *Theory of the backpropagation neural network*, pages 65–93. Elsevier, 1992.

[2] Marcel van Gerven and Sander Bohte. *Artificial neural networks as models of neural information processing*. Frontiers Media SA, 2018.

[3] 周飞燕, 金林鹏, and 董军. 卷积神经网络研究综述. 计算机学报, 40(06):1229–1251, 2017.

[4] 韩力群. 人工神经网络教程, volume 12. 北京邮电大学出版社, 2006.

[5] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[6] Donald O Hebb. *The organization of behavior*. na, 1961.

[7] Marvin Minsky and Seymour Papert. An introduction to computational geometry. *Cambridge tiass., HIT*, 1969.

[8] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[9] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[10] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[11] 余凯, 贾磊, 陈雨强, and 徐伟. 深度学习的昨天、今天和明天. 计算机研究与发展, 50(09):1799–1804, 2013.

[12] 周志华. 机器学习. 清华大学出版社, 2016.

[13] Geoffrey E Hinton. Learning distributed representations of concepts. In *Proceedings of the eighth annual conference of the cognitive science society*, volume 1, page 12. Amherst, MA.

[14] Richard S. Sutton and Andrew G. Barto. Toward a modern theory of adaptive networks: Expectation and prediction. *Psychological Review*, 88(2):135–170, 1981.

[15] Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Report, CORNELL AERONAUTICAL LAB INC BUFFALO NY, 1961.

[16] Kotaro Hirasawa, Masanao Ohbayashi, Masaru Koga, and Masaaki Harada. Forward propagation universal learning network. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, volume 1, pages 353–358. IEEE.

[17] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[18] 李友坤. *BP* 神经网络的研究分析及改进应用. 硕士, 2012.

[19] 刘天舒. *BP* 神经网络的改进研究及应用. 硕士, 2011.

[20] Yoshifusa Ito. Representation of functions by superpositions of a step or sigmoid function and their applications to neural network theory. *Neural Networks*, 4(3):385–394, 1991.

[21] Barry L Kalman and Stan C Kwasny. Why tanh: choosing a sigmoidal function. In *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, volume 4, pages 578–581. IEEE.

[22] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

[23] Theofanis Sapatinas. Statistics for high-dimensional data, 2012.

[24] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[25] Andrei Nikolaevich Tikhonov. On the solution of ill-posed problems and the method of regularization. In *Doklady Akademii Nauk*, volume 151, pages 501–504. Russian Academy of Sciences.

[26] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[27] Federico Girosi, Michael Jones, and Tomaso Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7(2):219–269, 1995.

[28] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[29] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[30] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[31] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[32] Jordan B Pollack. Recursive distributed representations. *Artificial Intelligence*, 46(1-2):77–105, 1990.

[33] Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–868, 2009.

[34] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[35] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks.

In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM.

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

[37] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[38] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[39] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[40] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[41] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[42] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.