神经网络读书报告

谭焱

June 26, 2020

摘要： 对神经网络自感知器起的 5 个历史发展阶段进行叙述, 提出其中 4 个重要并且基础的人工神经网络模型论文学习. 分别是人工神经网络的原型感知器; 最成功的神经网络算法反向传播算法; 神经网络在图像领域大放异彩的卷积神经网络; 最后是添加历史影响学习过程的循环神经网络.
关键词： 人工神经网络; 感知器; 反向传播算法; 卷积神经网络; 循环神经网络;

1 人工神经网络的发展历史

随着神经科学、认知科学的发展，我们逐渐知道人类的智能行为都和大脑活动有关. 人类大脑是一个可以产生意识、思想和情感的器官. 受到人脑神经系统的启发，早期的神经科学家构造了一种模仿人脑神经系统的数学模型，称为人工神经网络，简称神经网络. 在机器学习领域，神经网络是指由很多人工神经元构成的网络结构模型，这些人工神经元之间的连接强度是可学习的参数.

神经网络的发展大致经过五个阶段。

1.1 第一阶段：模型提出

1943 年至 1969 年是神经网络发展的第一个高潮期，科学家们提出了许多神经元模型和学习规则.

- 在 1943 年，心理学家 Warren McCulloch 和数学家 Walter Pitts 和最早描述了一种理想化的人工神经网络，并构建了一种基于简单逻辑运算的计算机制。他们提出的神经网络模型称为 MP 模型.
- 阿兰 · 图灵在 1948 年的论文中描述了一种 “B 型图灵机”. (赫布型学习)
- 1951 年, McCulloch 和 Pitts 的学生 Marvin Minsky 建造了第一台神经网络机，称为 SNARC

- Rosenblatt[6] 最早提出可以模拟人类感知能力的神经网络模型，并称之为**感知器 (Perceptron)**，并提出了一种接近于人类学习过程 (迭代、试错) 的学习算法.

在这一时期, 神经网络以其独特的结构和处理信息的方法, 在许多实际应用领域 (自动控制、模式识别等) 中取得了显著的成效。

1.2 第二阶段：冰河期

第二阶段为 1969 年至 1983 年, 是神经网络发展的第一个低谷期. 在此期间, 神经网络的研究处于长年停滞及低潮状态.

- 1969 年, Marvin Minsky 出版《感知器》一书, 指出了神经网络的两个关键缺陷: 一是感知器无法处理“异或”回路问题; 二是当时的计算机无法支持处理大型神经网络所需要的计算能力. 这些论断使得人们对以感知器为代表的神经网络产生质疑, 并导致神经网络的研究进入了十多年的“冰河期”.

- 但在这一时期, 依然有不少学者提出了很多有用的模型或算法. 1974 年, 哈佛大学的 Paul Werbos 发明**反向传播算法 (BackPropagation, BP)**[8], 但当时未受到应有的重视.

1.3 第三阶段: 反向传播算法引起的复兴

第三阶段为 1983 年至 1995 年, 是神经网络发展的第二个高潮期. 这个时期中, 反向传播算法重新激发了人们对神经网络的兴趣.

- 1986 年, David Rumelhart 和 James McClelland 提出了一种联结主义模型, 即分布式并行处理 (Parallel Distributed Pro-cessing, PDP) 模型 [4], 作为 PDP 的主要学习算法, 反向传播算法 (Error BackPropagation Algorithm, BP 算法) 的概念被首次提出. 这时, 神经网络才又开始引起人们的注意, 并重新成为新的研究热点.Rumelhart 等人提出的误差反向传播算法 [7].BP 算法系统解决了多层神经网络隐含层连接权学习问题，并在数学上给出了完整推导。人们把采用这种算法进行误差校正的多层前馈网络称为 **BP (back propagation) 神经网络**.

- 随后,LeCun 将反向传播算法引入了**卷积神经网络** [3], 并在手写体数字识别上取得了很大的成功.

- 1989 年，Ronald Williams 和 David Zipser 提出了 RNN 的实时循环学习 (Real-Time Recurrent Learning, RTRL). 又名**循环神经网络 (Recurrent Neural Network, RNN)**[10].

反向传播算法是迄今最为成功的神经网络学习算法. 目前在深度学习中主要使用的自动微分可以看作反向传播算法的一种扩展. 然而, 梯度消失问题 (Vanishing Gradient Problem) 阻碍神经网络的进一步发展, 特别是循环神经网络.

1.4 第四阶段: 流行度降低

第四阶段为 1995 年至 2006 年, 在此期间, 支持向量机和其他更简单的方法 (例如线性分类器) 在机器学习领域的流行度逐渐超过了神经网络. 虽然神经网络可以很容易地增加层数, 神经元数量, 从而构建复杂的网络, 但其计算复杂性也会随之增长. 当时的计算机性能和数据规模不足以支持训练大规模神经网络. 因此神经网络的研究又一次陷入低潮.

1.5 第五阶段: 深度学习的崛起

第五阶段为从 2006 年开始至今, 在这一时期研究者逐渐掌握了训练深层神经网络的方法, 使得神经网络重新崛起.

Hinton 通过逐层预训练来学习一个深度信念网络, 并将其权重作为一个多层前馈神经网络的初始化权重, 再用反向传播算法进行精调 [2]. 这种 “预训练 + 精调” 的方式可以有效地解决深度神经网络难以训练的问题. 近年来, 随着大规模并行计算以及 GPU 设备的普及, 计算机的计算能力得以大幅提高. 此外, 可供机器学习的数据规模也越来越大. 在强大的计算能力和海量的数据规模支持下, 计算机已经可以端到端地训练一个大规模神经网络, 不再需要借助预训练的方式.

2 发展过程中的重要成果

2.1 前馈神经网络 (feedforward neural network, FNN)

2.1.1 感知器 [6]

1958 年, Rosenblatt 从人的眼睛光学系统中获得灵感, 提出的一种最简单形式的前馈式人工神经网络, 因此被称为感知器。感知器是一种二元线性分类器。

Rosenblatt 的感知器算法

一个训练集合 训练元素 $\mathcal{A}_k$ 是一个有序集合 $(x_k, \hat{y}_k), x_k = (x_{1k}, x_{2k}, \ldots, x_{nk})$ 是一些向量,$\hat{y}_k = 0$ 或 1 (即取决于 x_k 的所需的神经元输出).

如果 $\mathcal{A} = \{\mathcal{A}_k\}$ 满足

$$\mathcal{A} = \mathcal{A}^1 \cup \mathcal{A}^2$$
$$\mathcal{A}^1 = \{x_k; \mathcal{A}_k = (x_k, \hat{y}_k \in \mathcal{A} and \hat{y}_k = 1)\}$$
$$\mathcal{A}^0 = \{x_k; \mathcal{A}_k = (x_k, \hat{y}_k \in \mathcal{A} and \hat{y}_k = 0)\}$$

那么训练元素的集合 $\mathcal{A}$ 是一个训练集.

对于一个给定的训练集 $\mathcal{A}$, 学习任务就是找到一个权重向量 w^* 和一个闸值 b 满足

$$\forall x_k \in \mathcal{A}^1 : w^* \cdot x_k + b > 0$$
$$\forall x_k \in \mathcal{A}^1 : w^* \cdot x_k + b < 0$$

学习算法

- 令 $\mathcal{A}_k = (x_k, \hat{y}_k)$ 是一个元素, $w = (w_1, w_2, \ldots, w_n)$ 和 b 是当前的权重向量和闸值, 对于向量 x_k 计算神经元的输入值

$$\xi_k = \sum_{i=1}^{n} w_i x_{ik} + b.$$

- 神经元的输出值可以如下确定

$$y_k = \begin{cases} 1 & \text{if } \xi_k \geq 0 \\ 0 & \text{if } \xi_k < 0 \end{cases}$$

- 最后权重向量和闸值关于训练元素 $\mathcal{A}_k$ 的更新如下进行 (Hebbian 学习)

$$w_{new} = w_{old} + \lambda(\hat{y}_k - y_k)x_k$$
$$b_{new} = b_{old} + \lambda(\hat{y}_k - y_k)$$

. 重复该过程, 直到正确解释了训练集的所有示例, 即所需的输出活动 $\hat{y}_k$ 等于计算的输出活动 y 参数 $\lambda > 0$ 称为学习率。

Rosenblatt 同时还给出了该算法的理论基础

理论基础

Theorem 1 对于一个训练集 $\mathcal{A} = \mathcal{A}^0 \cup \mathcal{A}^1$, 感知器学习算法可确保找到权重向量 w^*, 使得对于某些 $b > 0$

$$\forall \mathcal{A}_k \in \mathcal{A} : w^* \cdot x_k \geq b \ and \ \forall \mathcal{A}_k \in \mathcal{A} : w^* \cdot x_k \leq -b,$$

只要存在这样的解 w^*.

连续感知器 许多实际问题要求连续函数逼近. 这要求神经元能够计算实值函数 $f\colon \mathbf{R}_n \to (0,1)$. 为了我们即将考虑的目的，通过训练集 $\mathcal{A} = \{\mathcal{A}_k = (x_k, \hat{y}_k = f(x_k)); k = 1, 2, \ldots, P\}$; 确定连续阈值神经元

$$y = t\left(\sum_{i=1}^{n} w_i x_i + b\right) = \varphi(x_1, x_2, \ldots, x_n; w_1, w_2, \ldots, b)$$

$t\colon \mathbf{R} \to (0,1)$ 是一个转化函数满足它是可微并且单调递增的, 还满足 $t(-\infty) = 0, t(\infty) = 1$.

激活函数 t 是 sigmoid 函数, 作为激活函数是人工神经网络中的重要组成部分, 用于处理神经元的输入输出, 可以增强网络的表示能力和学习能力. 这个 t 是 Hard-Logistic 函数,sigmoid 函数还有一种 Hard-Tanh 函数, 区别在于 Tanh 函数是关于零点对称有负值输出的函数. 发挥和 sigmoid 函数类似作用的还有 ReLU (Rectified Linear Unit) 修正线性单元 [5], 也叫 Rectifier 函数, 是目前深度神经网络中经常使用的激活函数. 采用 ReLU 的神经元只需要进行加, 乘和比较的操作, 计算上更加高效. 但是 ReLU 函数的输出是非零中心化的, 给后一层的神经网络引入偏置偏移, 会影响梯度下降的效率. 激活函数还有 swish 函数,GELU 函数和 Maxout 单元等等.

损失函数 在连续感知器中要学习神经元功能，使其值将尽可能接近训练集中的所需值.Rosenblatt 利用二次逼近为第 k 个训练示例定义一个目标函数

$$E(w, b) = \frac{1}{2}\sum_{i=1}^{p} (y_k - \hat{y}_k)^2$$

为了找到使目标函数最小化的最佳权重和阈值，即具有这些最佳权重和阈值的连续神经元将训练集示例确定的函数 $f(x)$ 紧密建模, 定义了新的训练目标

$$(w_{opt}, b_{opt}) = \arg\min_{w,b} E(w, b).$$

这是一个非线性优化目标, 非线性目标函数的优化过程是通过所谓的最速下降法实现的，即通过基于以下权重和阈值更新的最简单梯度法来实现.

$$w^{t+1} = w^t - \lambda \frac{\partial E(w^t, b^t)}{\partial w^t}$$
$$b^{t+1} = b^t - \lambda \frac{\partial E(w^t, b^t)}{\partial b^t}$$

.

这个迭代公式与后来的反向传播算法非常接近. 新的训练目标也类似与损失函数.

高阶感知器 作者文中已经提到感知器的局限性.Minsky 和 Papert 证明了感知器能够对由线性可分离的示例训练集确定的功能进行建模.Minsky 和 Paper 认为这一事实是感知器的非常严重的缺点，他们得出结论，感知器无法归类为通用计算设备. 为了克服感知器的这一缺点，他们引入了两个新概念，特别是高阶感知器和隐藏神经元. 这两个概念都极大地提高了感知器建模任意函数的计算能力，但是 Minsky 和 Papert 明确得出结论，由于它们的数值复杂性，这些概念不是归纳感知器的合适工具，并且感知器的理论无疑受到了严重的困扰限制，因此感知器并不代表寻找已知的通用计算设备的替代方案的视域. 其中隐藏神经元就是后来深度学习的进展方向, 体现了当时作者高度的前瞻性, 可惜由于历史客观条件限制使得人工神经网络沉寂了一段时间.

高阶感知器算法 他们提出的高阶感知器是通过推广简短神经元计算方式, 为了能够解决非线性问题, 他们加入了二次, 三次等更高阶次项,

$$y = t\left(\sum_{i=1}^{p} w_i x_i + \sum_{\substack{i,j=1 \\ i\le j}}^{p} w_{ij} x_i x_j + \sum_{\substack{i,j,k=1 \\ i\le j\le k}}^{p} w_{ijk} x_i x_j x_k + \cdots + b\right)$$

迭代算法类似连续感知器

$$w_i^{t+1} = w_i^t - \lambda \frac{\partial E(w^t, b^t)}{\partial w_i^t}$$
$$w_{ij}^{t+1} = w_{ij}^t - \lambda \frac{\partial E(w^t, b^t)}{\partial w_{ij}^t}$$
$$\cdots$$
$$b^{t+1} = b^t - \lambda \frac{\partial E(w^t, b^t)}{\partial b^t}$$

理论基础

Theorem 2 由参数函数 $\varphi(x; w, b) = t(\sum w_i p_i(x) + b)$ 实现的高阶感知器能够以规定的精度 ε 建模任意函数 $f\colon \mathbf{R}_n \to (0,1)$ 通过训练集 $\mathcal{A} = \{\mathcal{A}_k = (x_k, f(x_k)); k = 1, 2, \ldots, P\}$

$$\sum_{k=1}^{p} |\varphi(x_i; w, b) - f(x_i)| < \varepsilon.$$

换句话说，高阶阈值神经元关于函数 f 的通用逼近可以由训练集 $\mathcal{A}$ 确定.

2.1.2 BP 算法 [7]

与感知器类似定义了神经元输入 x_i 输出 y_j, 权重函数 w_{ji}, 神经元函数

$$x_j = \sum_i y_i w_{ji} \tag{1}$$

和激活函数

$$y_j = \frac{1}{1 + e^{-x_j}} \tag{2}$$

正向传递 分层网络其底部有一层输入单元. 任意数量的中间层; 在顶部有一层输出单元. 禁止在层内或从较高层到较低层的连接，但是连接可以跳过中间层. 通过设置输入单元的状态，可以将输入向量呈现给网络. 然后，通过将等式 (1) 和 (2) 应用于来自较低层的连接，可以确定每个层中单元的状态. 一层中的所有单元的状态并行设置，但不同层的状态则顺序设置，从底部开始直到向上确定直到确定输出单元的状态. 不必完全使用方程式 (1) 和 (2) 中给出的函数. 任何具有有界导数的输入输出函数都可以. 但是, 在应用非线性之前使用线性函数将输入组合到一个单元中, 可以大大简化学习过程.

因此, 当我们有一个分层网络和一组输入数据是就可以计算出一组输出数据, 并结合目标输出进行学习.

损失函数 学习的目标是找到一组权重，以确保对于每个输入向量，网络生成的输出向量与所需的输出向量相同 (或足够接近). 如果存在一组固定的有限的输入输出情况，则可以通过比较每种情况下的实际和期望输出矢量来计算具有特定权重集的网络性能的总误差 E. 总误差定义如下

$$E = \frac{1}{2} \sum_i \sum_c (y_{i,c} - d_{i,c})^2. \tag{3}$$

其中 c 是用于学习的数据序号,i 是每组数据的输出编号,d 是目标输出.

之前已经描述了正向传播过程可以计算得到输出, 再结合损失函数得出的总误差进行学习. 为了通过梯度下降最小化 E，有必要针对网络中的每个权重计算 E 的偏导数, 此即反向传播过程, 反向传播算法的核心进展.

反向传播 该过程从对 y 递归求 $\partial E/\partial y$ 开始. 推导过程如下

$$\begin{aligned}
\partial E/\partial y_j &= y_j - d_j \\
\partial E/\partial x_j &= \partial E/\partial y_j \cdot \partial y_j/\partial x_j \\
&= \partial E/\partial y_j \cdot y_j(1 - y_j) \\
\partial E/\partial w_{ji} &= \partial E/\partial x_j \cdot \partial x_j/\partial w_{ji} \\
&= \partial E/\partial x_j \cdot y_i \\
\partial E/\partial y_j &= \partial E/partial x_j \cdot \partial x_j/\partial y_j \\
&= \partial E/\partial x_j \cdot w_{ji}
\end{aligned}$$

现在，我们已经看到了在为倒数第二层的任何单位提供最后一个单位的 $\partial E/\partial y$ 时如何计算 $\partial E/\partial y$. 因此，我们可以重复此过程来为较早的各层计算该项，计算每一层关于权重的偏微分 $\partial E/\partial w$. 得到所有偏微分后可以根据梯度修改权重.

递归逼近权重 逼近权重过程中, 使用 $\partial E/\partial w$ 的一种方法是在每种输入输出情况之后更改权重. 这样做的优点是, 导数不需要单独的存储器. 我们在这里报告的研究中使用的一种替代方案是在更改权重之前，在所有输入/输出情况下累计 $\partial E/\partial w$. 梯度下降的最简单形式是将每个权重改变与累积的 $\partial E/\partial w$ 成正比的量

$$\Delta w = -\varepsilon \partial E/\partial w.$$

该方法的收敛速度不如使用二阶导数的方法快，但是它更简单，并且可以通过并行硬件中的本地计算轻松实现。通过使用一种加速方法，可以在不牺牲简单性和局部性的情况下显着改善它，在该方法中，梯度用于修改权重空间中点的速度而不是其位置

$$\Delta w(t) = -\varepsilon \partial E/\partial w(t) + \alpha \Delta w(t-1)$$

其中，对于整个输入-输出情况集合中的每次扫描，t 都会增加 1，而 α 是 0 与 1 之间的指数衰减因子，它决定了当前梯度和较早的梯度对权重变化的相对贡献。

数据实验结论 学习程序最明显的缺点是误差表面可能包含局部最小值，因此梯度不能保证找到全局最小值. 然而, 根据作者数据实验的经验表明，网络很少被困在糟糕的局部最小值中全局最小值.

2.2 卷积神经网络 [3]

卷积神经网络从与 CNN 结构类似, 由于图像数据的高维性, 和为了减少运算量不能将图像数据直接转化为 1 维数据输入神经网络进行学习. 卷积神经网络从图片中提取数据信息后进行神经网络学习分类. 将图片信息提取压缩的过程是卷积算法和进行压缩过程的部件是卷积神经网络的主要不同.

2.2.1 卷积神经网络一般部件

导致部件区分的是部件之间的神经元算法. 如下将主要部件及他们之间的算法介绍

卷积层 (convolutional layer) 卷积层是卷积神经网络的主要结构, 算法是用一个小矩阵滤波器在卷积层上滑动计算提取得到下一层数据, 不同的滤波器能对图像进行不同的处理.

第一个例子是如下的平滑滤波器，它将使得图像变得更为平滑，因为它联合周边像素进行了平均处理。也可以改为更大的矩阵，不过需要保证矩阵的所有元素和为 1. 平滑滤波器 size 越大，则平均的范围越大，平滑效果越强.

$$g = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

第二个例子是如下的增强边缘滤波器，它使得像素值变化明显的地方更为明显，强化边缘，而变化平缓的地方没有影响，达到提取边缘的目的.

$$g = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

因此，滤波器函数 g 的不同可以对图片 f 会产生不同的结果，人们可以改变不同的滤波器来达到不同的目的.

池化层 (ooling layer), 采样层 图像的数据有时太大, 全部进行卷积提取计算量无法接受. 从池化层能够迅速减少数据量, 并且保留所需数据. 池化层算法就是从一个二维正方形区域提取一个数据到下一层, 速度快迅速缩小数据量. 池化方法有平均池化 (即保留区域类所有数据的平均值到下一层) 和最大池化 (直接保留区域内的最大值到下一层). 下采样与池化类似, 上采样与之相反操作.

全连接层 进行卷积和池化操作后, 图像数据被压缩到可以接受的大小时, 将 3 维图像数据乘以一定权重拉成一维数据, 之后进行神经网络训练权重. 全连接层也包含传统神经网络中的 sigmoid 函数 softmax 函数等过程.

2.2.2 典型的卷积神经网络结构

- LeNet-5 虽然提出的时间比较早, 但它是一个非常成功的神经网络模型. 基于 LeNet-5 的手写数字识别系统在 20 世纪 90 年代被美国很多银行使用, 用来识别支票上面的手写数字.LeNet-5 共有 7 层, 开始卷积层和池化层交错 5 层, 然后全连接层后输出.

- AlexNet 是第一个现代深度卷积网络模型, 其首次使用了很多现代深度卷积网络的技术方法, 比如使用 GPU 进行并行训练, 采用了 ReLU 作为非线性激活函数, 使用 Dropout 防止过拟合, 使用数据增强来提高模型准确率等.AlexNet, 包括 5 个卷积层,3 个汇聚层和 3 个全连接层.

- 在卷积网络中, 如何设置卷积层的卷积核大小是一个十分关键的问题. 在 Inception 网络中, 一个卷积层包含多个不同大小的卷积操作, 称为 Inception 模块. Inception 网络是由有多个 Inception 模块和少量的汇聚层堆叠而成.

2.3 循环神经网络 [10]

在之前看过的人工神经网络中, 信息的传递是单向的, 这种限制虽然使得网络变得更容易学习, 但在一定程度上也减弱了神经网络模型的能力. 在生物神经网络中, 神经元之间的连接关系要复杂得多. 前馈神经网络可以看作一个复杂的函数,

每次输入都是独立的, 即网络的输出只依赖于当前的输入. 但是在很多现实任务中, 网络的输出不仅和当前时刻的输入相关, 也和其过去一段时间的输出相关.

循环神经网络是一类具有短期记忆能力的神经网络. 在循环神经网络中, 神经元不但可以接受其他神经元的信息, 也可以接受自身的信息, 形成具有环路的网络结构. 和前馈神经网络相比, 循环神经网络更加符合生物神经网络的结构.

2.3.1 循环神经网络的特点

循环神经网络的变量 论文中与之前类似定义输入 $x_k(t)$ 输出 $y_k(t)$ 和损失函数 $J(t)$. 但是有一点不同, 输入输出数据的数量是可变的. 这在传统神经网络中每一个训练好的网络的输入输出维数是固定的. 对于不同数量的输入输出数据, 传统神经网络必须调整神经网络结构, 但是调整结构后的神经网络需要整体重新开始训练, 这显然是不效率的. 除此之外, 论文中还定义了一个新变量 $p_{ij}^l(t) = \frac{\partial y_l(t)}{\partial w_{ij}}$ 用以存储过去一段时间的输入-输出状态, 影响对当前循环学习效果.

循环神经网络的算法 循环神经网络改变了传统神经网络中对权重函数的递归修改方法. 首先, 如下递归定义了 $p_{ij}^l(t)$ 的计算方法

$$p_{ij}^k(t+1) = f_k'(s_k(t)) \left[\sum_{l \in U} w_{kl} p_{ij}^l(t) + \delta_{ik} z_j(t) \right].$$

并且满足 $p_{ij}^k(t_0) = 0$. 然后递归得到权重函数

$$\delta w_{ij}(t) = \alpha \sum_{k \in U} e_k(t) p_{ij}^k(t).$$

最后循环神经网络的参数学习可以通过随时间反向传播算法 [9] 来学习.

2.3.2 循环神经网络的理论支持 [1]

Theorem 3 *如果一个完全连接的循环神经网络有足够数量的 sigmoid 型隐藏神经元, 它可以以任意的准确率去近似任何一个非线性动力系统*

$$s_t = g(s_{t-1}, x_t)$$
$$y_t = o(x_t)$$

其中 s_t 为每个时刻的隐状态, x_t 是外部输入, $g(\cdot)$ 是可测的状态转换函数,$o(\cdot)$ 是连续输出函数, 并且对状态空间的紧致性没有限制.

循环神经网络通过添加了一个时间分量 t 和 $p_{ij}^k(t)$ 存储隐藏状态将最近的输入输出对神经网络的学习的影响添加了进来, 并且消除了神经网络对输入-输出数据的维数限制. 但是在随时间反向传播算法即按照时间的逆序将错误信息一步步地往前传递. 当输入序列比较长时, 会存在梯度爆炸和消失问题, 也称为长程依赖问题.

3 总结

通过按人工神经网络的历史发展和其中关键成果的阅读, 可以发现人工神经网络的成果很多但是变化不大. 从最初的感知器到反向传播算法只是增加神经网络的深度并同时推广梯度下降算法; 再到卷积神经网络, 由于对图片处理的需要, 和图片数据量冗余, 于是在使用神经网络学习前使用卷积和池化提取必要信息, 减小数据规模之后进行神经网络学习. 到最后的循环神经网络在面对前向反馈神经网络对输入-输出数据维数限制和对历史数据利用不足时, 增加一个因变量时间 t 来消除维数限制和一个递归变量 $p_{ij}^{k}(t)$ 存储历史信息的隐藏状态增添历史信息的影响因子.

这 3 个人工神经网络举足轻重成果实际上作出的改变不多, 并且从之前的研究成果中有迹可循 (1958 年,Minsky 和 papert 提出隐藏神经元层是感知器下一步发展方向). 启示了科学研究要找细节, 和从过往别人的研究中发现能够继续的方向. 例如 20 世纪由于数据量和计算能力限制, 很多人工神经网络思想沉寂一段时间后, 在 21 世纪异军突起.

参考文献

[1] S. S. Haykin and R. Gwynn, *Neural Networks and Learning Machines*, China Machine Press,, 2009.

[2] G. E. Hinton and R. R. Salakhutdinov, *Reducing the dimensionality of data with neural networks*, Science, 313 (2006), pp. 504–507.

[3] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, *Backpropagation Applied to Handwritten Zip Code Recognition*, Neural Computation, 1 (1989), pp. 541–551.

[4] J. L. McClelland, D. E. Rumelhart, and J. L. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 2: Psychological and Biological Models*, vol. 1, MIT Press, 1986.

[5] V. Nair and G. E. Hinton, *Rectified linear units improve Restricted Boltzmann machines*, in ICML 2010 - Proceedings, 27th International Conference on Machine Learning, 2010, pp. 807–814.

[6] F. ROSENBLATT, *The perceptron: a probabilistic model for information storage and organization in the brain.*, Psychological review, 65 (1958), pp. 386–408.

[7] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning representations by back-propagating errors*, Nature, 323 (1986), pp. 533–536.

[8] P. Werbos, *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, PhD Thesis, Harvard U., PhD thesis (1974).

[9] P. J. Werbos, *Backpropagation Through Time: What It Does and How to Do It*, Proceedings of the IEEE, 78 (1990).

[10] R. J. Williams and D. Zipser, *A Learning Algorithm for Continually Running Fully Recurrent Neural Networks*, Neural Computation, 1 (1989), pp. 270–280.