# 1 深度学习

## 1.1 引言

### 1.1.1 深度学习的定义和特征

根据数据形态与反馈信息的不同, 机器学习大致可以分为监督学习, 无监督学习和强化学习三种类型. 深度学习则是一种方法与体系架构, 可以用于这些任务中. 作为神经网络的一种, 深度学习的深度体现在其神经网络包含多个隐藏层, 从而逐层提取数据特征[1].

深度学习是当前人工智能繁荣时期的主角. 互联网的发展创造了大量可用的训练数据, 而芯片技术的发展大大提高了计算能力, 这两者使得深度神经网络的计算与训练成为可能.

### 1.1.2 深度学习的实例

深度学习的一个热门应用就是语言模型, 通常由于模型的规模很大, 称作**大语言模型 (Large Language Model, LLM)**.

在著名的 Attention is all you need 这篇论文中, 人们提出了注意力机制以及被称作 Transformer 的通用架构网络, 是近几年最重要, 应用最广泛的深度学习架构.

## 1.2 深度神经网络的训练

由于层数的增加, 深度神经网络的训练异常困难 (这也是深度神经网络在早期不被看好的原因之一). 为此, 人们发展了各种适用于深度神经网络训练的方法.

### 1.2.1 ADAM 与小批量算法

在梯度下降法中, 我们采用下式来更新优化模型的权重 $\boldsymbol{w}$:

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} - \eta \nabla_{\boldsymbol{w}} E(\boldsymbol{w}^{(\tau)})$$

其中 $E(\boldsymbol{w})$ 是误差函数, $\eta$ 是学习率. 在深度神经网络中, 由于参数众多, $E(\boldsymbol{w})$ 是高维空间中的复杂曲面, 使用梯度下降法容易陷入到局部极小值点或鞍点中.

如果把 $E(\boldsymbol{w})$ 想象成能量, $\boldsymbol{w}$ 想象成位置, 则 $-\nabla_{\boldsymbol{w}} E(\boldsymbol{w})$ 相当于是力, 于是上式是让位置的变化与力成正比, 换言之就是沿着能量下降最快的方向前进. 这种想法在遇到病态曲率的情况会出现问题, 因此人们提出了一种更加符合物理思维的想法, 即将力转换为加速度, 先影响速度, 再由速度影响位置. 这就是 ADAM 算法, 其具体过程如下:

**i.** 计算梯度:

$$\boldsymbol{g}_t \leftarrow \nabla_{\boldsymbol{w}} E(\boldsymbol{w}_{t-1})$$

[1] 例如识别图像时, 先识别边缘和颜色, 再到纹理和形状, 最后到完整的物体.

**ii.** 计算速度:

$$\boldsymbol{v}_t \leftarrow \beta_1 \boldsymbol{v}_{t-1} + (1-\beta_1)\boldsymbol{g}_t$$

**iii.** 计算梯度平方:

$$r_t \leftarrow \beta_2 r_{t-1} + (1-\beta_2)||\boldsymbol{g}_t||^2$$

**iv.** 矫正偏差:

$$\hat{\boldsymbol{v}}_t \leftarrow \frac{1}{1-\beta_1^t}\boldsymbol{v}_t, \quad \hat{r}_t = \frac{1}{1-\beta_2^t} r_t$$

**v.** 更新参数:

$$\boldsymbol{w}_t = \boldsymbol{w}_{t-1} - \frac{\eta}{\sqrt{\hat{r}_t} + \varepsilon}\hat{\boldsymbol{v}}_t$$

ADAM 收敛速度快, 训练稳定, 对超参数的敏感度较低, 调参难度小, 广泛应用于各类深度学习任务, 是目前最流行的优化算法之一.

### 1.2.2 梯度消失与残差网络

深度神经网络训练过程所面临的另外一个挑战就是**梯度消失问题 (Vanishing Gradient Problem)**, 过多的隐藏层往往会使模型丧失学习到先前信息的能力, 从而让优化变得极其困难, 模型难以训练, 甚至无法收敛.

梯度消失的核心原因是在反向传播算法中, 参数的更新依赖于梯度的计算, 而梯度需要从网络的输出层反向传播到输入层. 当层数较深时, 梯度在传播的过程中可能会因为不断地乘以小于 1 的数而减小, 最终逐渐趋于 0, 这就是梯度消失现象.

解决梯度消失问题的办法很多, 列举如下:

**批归一化** **批归一化 (Batch Normalization)** 可以稳定各层输入分布, 减少梯度波动. 其主要想法是在神经网络训练时, 随着参数更新, 各层输入的分布会不断变化, 这会导致在后续层需要不断适应新的分布, 增加训练难度.

**残差网络** 何恺明等人提出的**残差网络 (Residual Network, ResNet)** 是解决梯度消失问题的重要方法. 从理论上讲, 增加隐藏层只会使模型更好, 不会更差. 基于这种朴素思想, 可以通过残差方式, 将输入直接跳过部分网络层传递到输出, 形成短路连接:

$$\boldsymbol{y} = F(\boldsymbol{x}, \boldsymbol{w}_i) + \boldsymbol{x}$$

其中 $F(\boldsymbol{x}, \boldsymbol{w}_i)$ 是原来的变换模块; 这被称作一个残差模块. 通过这样的残差连接, 输入中的信息总是可以被传递到任意一层, 不容易发生信息的丢失. 通过残差网络, 我们更容易构建多层的网络, 从而得到性能更好的模型.

### 1.2.3 过拟合

与其它的机器学习方法类似, 深度神经网络也会发生过拟合. 除了一些常见的防止过拟合的方法 (例如增加数据量, 正则化) 以外, 在深度学习中还发展了一些特有的方法.

**早停法** 早停法 **(Early Stopping)** 是一种简单有效的防止过拟合的策略. 其核心思想是监控模型在验证集上的性能, 提前终止训练过程, 避免出现对训练集的过拟合.

在训练的初期, 训练集和验证集的误差都随着训练进行而下降, 表明模型正在学习数据的普遍规律; 当训练到一定程度时, 可能出现训练集的误差逐渐下降, 但验证集的误差停止下降甚至上升的现象, 这表明模型开始对训练集产生过拟合, 此时就应当停止训练了.

**丢弃法** 丢弃法 **(Dropout)** 也是一种常见的防止过拟合的策略. 其核心思想是通过随机屏蔽部分神经元, 减少模型对特定神经元的依赖, 增强其泛化能力. 从组合模型的角度考虑, 丢弃法提供了一种简单易行的 Bagging 方法.

实际操作中, 只需在每轮训练时按照对于神经网络的某一层按照一定概率随机地将部分神经元的输出设为 0, 达到丢弃的效果, 使得每次训练的网络结构都不同. 通常在输入层或隐藏层进行如此处理, 输出层则一般不使用, 避免破坏最终预测结果.

## 1.3 生成对抗模型

### 1.3.1 生成对抗模型的基本原理

利用概率分布讨论体系性质是机器学习的一种重要思路. 在分类任务中, 如果能从已知数据集中推断出每个类别 $\mathcal{C}$ 的分布曲线 $p(\boldsymbol{x},\mathcal{C})$, 就可以据此进行预测, 也可以据此用来产生类别 $\mathcal{C}$ 的新样本. 后者被称作样本生成任务, 属于无监督学习的范畴.

概率函数 $p(\boldsymbol{x},\mathcal{C})$ 和 $p(\boldsymbol{x})$ 可以用马尔可夫网络或神经网络描述, 但因为归一化因子 (换言之, 通常的神经网络并不是归一化的) 的存在而不容易训练. **生成对抗网络 (Generative Adversarial Nets, GAN)** 提供了在不需要显式计算 $p(\boldsymbol{x})$ 的情况下产生新样本的有效方法.

GAN 由两个网络组成: 生成器网络 $\boldsymbol{x}=G(z)$ 和判别器网络 $y=D(\boldsymbol{x})$. $D$ 用于判断一个输入 $\boldsymbol{x}$ 是来源于原有数据集还是由 $G$ 生成的, 而 $G$ 用于生成尽量使 $D$ 判断错误的 $\boldsymbol{x}$. 两者对抗优化, 最后收敛于稳定的解. 在训练时, $G$ 不接触训练集数据, 只接受来自 $D$ 的反馈.

GAN 的误差函数为

$$V(D,G)=\mathbb{E}_{\boldsymbol{x}\sim p_{\mathcal{D}}}[\log D(\boldsymbol{x})]+\mathbb{E}_z[\log(1-D(G(z)))]$$

其中 $z$ 是生成器 $G$ 的输入, 一般服从简单的分布. $D$ 的目标是使得上式尽可能大, 而 $G$ 的目标是使得上式尽可能小, 即

$$\min_G\max_D V(D,G)$$

数学上可以证明, 当极值收敛是, $\boldsymbol{x}=G(z)$ 的分布将与原有数据集的概率分布 $p(\boldsymbol{x})$ 一致, 且 $D(\boldsymbol{x})=\dfrac{1}{2}$.

## 1.4 扩散模型

在 GAN 之后, 另一类深度学习生成模型快速崛起, 即**扩散模型 (Diffusion Models)**.

扩散模型采用了非平衡统计的思想, 目的是从一些样本中来学习其分布概率 $p(\boldsymbol{x})$ 并据此产生新的采样. 它的主要想法大致理解如下: 对于某个分布, 可以定义带随机性的变换使得分布随时间发生变化, 在时间较长时变换到某一简单分布.

例如, 对于外力 $\boldsymbol{f}(\boldsymbol{x})$ 下的扩散方程, 粒子流为

$$\boldsymbol{j} = -D\nabla p(\boldsymbol{x}) + \frac{D}{k_{\mathrm{B}}T} p(\boldsymbol{x})\boldsymbol{f}(\boldsymbol{x})$$

其中 $D$ 为扩散系数. 如果将外力替换为

$$\boldsymbol{f}'(\boldsymbol{x}) = -\boldsymbol{f}(\boldsymbol{x}) + 2k_{\mathrm{B}}T\nabla \ln p(\boldsymbol{x})$$

则粒子流将分享, 末态会随时间演化成初态.

类似地, 扩散模型的工作过程分为两个阶段:

**i.** 前向扩散: 将 $p_0(\boldsymbol{x})$ 用训练数据集 $\{\boldsymbol{x}_n\}$ 所组成的 $\delta$ 函数之和来近似:

$$p_0(\boldsymbol{x}) = \frac{1}{N}\sum_n \delta(\boldsymbol{x} - \boldsymbol{x}_n)$$

在扩散过程中, $\delta$ 函数有简单的解析解 $\mathcal{N}(\boldsymbol{x}|\boldsymbol{x}_n, \sigma^2)$(其中 $\sigma^2 = 2Dt$ 是扩散所导致的方差, $D$ 是 $\boldsymbol{x}$ 的维度), 其对逆过程外力的贡献也可以解析地给出. 因此, 我们可以通过对 $\{\boldsymbol{x}_n\}$ 采样, 即从真实数据 $\boldsymbol{x}_n$ 开始逐步添加高斯噪声模拟扩散过程, 计算其在任一时刻对外力的贡献, 并训练一个深度学习网络来近似总外力, 即

$$\boldsymbol{f}_{\mathrm{ext}}(\boldsymbol{x}, t) = 2k_{\mathrm{B}}T\nabla \ln \frac{1}{N}\sum_n \mathcal{N}(\boldsymbol{x}|\boldsymbol{x}_n, \sigma_t^2) = -\frac{2k_{\mathrm{B}}T}{\sigma_t^2}\frac{\sum_n (\boldsymbol{x} - \boldsymbol{x}_n)\mathcal{N}(\boldsymbol{x}|\boldsymbol{x}_n, \sigma_t^2)}{\sum_n \mathcal{N}(\boldsymbol{x}|\boldsymbol{x}_n, \sigma_t^2)}$$

经过足够长时间 ($t \gg 1$) 的扩散后, $p_t(\boldsymbol{x})$ 将变为完全随机的高斯分布 $\mathcal{N}(\boldsymbol{x}|\boldsymbol{0}, \sigma_T^2)$.

**ii.** 反向扩散: 利用前一阶段得到的外力 $\boldsymbol{f}_{\mathrm{ext}}(\boldsymbol{x}, t)$ 从随机的高斯分布 $p_T(\boldsymbol{x})$ 中选取一个初值 $\boldsymbol{x}(T)$, 利用外力下的扩散公式还原至初态, 生成与真实数据分布一致的样本.

扩散模型的典型例子就是图像生成与编辑.