technische universität
dortmund

硕士论文

领域专用神经网络架构设计的结构化方法

Gerrit
Nolte 2021年12月

指导教师：Bernhard
Steffen 教授、Alnis
Murtovi 理学硕士

多特蒙德工业大学 计算机科学学院 编
程系统教席（5）
http://ls5-
www.cs.tu-dortmund.de

arXiv:2301.09381v1 [cs.LG] 23 Jan 2023

目录

1 引言与动机 **1**

1.1 本文结构. 2

2 背景与一般设定 **3**

2.1 参数化估计器、风险与损失函数.4 2.1.1 确定合理的损失函数.6 2.1.2 基于梯度的优化.7 2.2 正则化与归纳偏置.9 2.3 深度学习.12 2.3.1 一般术语与定义.12 2.3.2 全连接神经网络.13 2.4 全连接神经网络的归纳偏置.15 2.5 前馈神经网络的问题.23

3 几何深度学习引言 27

3.1 对称性与 𝔾-不变性.28 3.2 任务分离与局部性.30 3.3 特定领域神经网络设计要求.35 3.4 作为领域特定神经网络的深度集合与 GDL蓝图 36 3.4.1 GDL蓝图.38 3.5 几何深度学习总结.41

4 图神经网络作为几何深度学习实例 4 **3**

4.1 Weisfeiler-Lehman 检验.44 4.2 从 WL检验推导图神经网络.46 4.3 图神经网络与 GDL蓝图.51 4.4 能否从 GDL蓝图推导出图神经网络？.53

5 强制 𝔾-不变性对估计器性能的影响 5 **7**

5.1 刻画 𝔾-不变估计问题.58 5.2 不变性与梯度下降.59 5.3 不变性与拓扑结构.61 5.4 不变神经网络的泛化性质.63 5.5 总结.68

6 Su总结、结论与展望 **69**
6.1 结论.70 6.2 展望.71

A A附录 **73**
A.1 A. 群与对称性.73 A.1.1 轨道与等价类.74 A.2 B. 离散域中的神经网络.75 A.2.1 离散输入.
76 A.2.2 离散输出.78

图目录 83

参考 文献 **89**

Decl第 8 部分 **9**

第1章

引言与动机

过去几十年间，很少有领域能像深度学习领域那样取得如此迅速的进展。这一进步的原因之一，无疑是并行硬件日益强大，并且能很好地支持神经网络。然而，可以说，更重要的进展来自神经网络模型本身的设计。

在实践中，更优秀的架构已占据主导地位，而全连接神经网络如今基本已成为历史。例如，卷积神经网络在图像识别方面取得了最先进水平 [14]。在语言处理任务中，Transformer 占据主导地位 [60]；在基于图的任务中，图神经网络则是首选模型 [54]。

这些架构背后的强大新理念，是使用专为目标应用领域打造的神经网络。现代神经网络的先驱——全连接神经网络——原则上可以用于所有这些任务，但实践表明，利用领域专家明确的领域知识构建的神经网络，远胜于那些未将领域知识纳入考量的架构。因此，将神经网络视为通用工具的想法似乎已经过时；要达到现代水平，就需要针对特定任务精心设计模型。

然而，尽管特定领域神经网络在实践中取得了巨大成功，但为特定领域构建优秀 的神经网络仍是一项极其困难的任务。例如，对于图神经网络，从最初的理论构想到广泛应用，历经了近十年以及对模型本身的多次改进[67, 54]。原因在于，要构建优秀的特定领域神经网络，既需要精通该领域，也需要全面掌握深度学习。

因此，在创建特定领域神经网络方面仍有很多缺失。首先，似乎尚未形成关于什么才算优秀的特定领域神经网络，以及它应遵循哪些原则的概念。其次，因此也尚未有任何结构化方法来设计遵循上述原则的神经网络，更不用说真正自动化的方法了。

因此，需要回答的问题是：“优秀的特定领域神经网络是什么样的？”以及“我们如何构建它们？”

[9] 作者旨在迈出第一步，探索如何以结构化方法创建他们称为“几何深度学习”的领域特定神经网络[1]。为此，他们提出了

[1]这个名称很可能是因为这项工作的起源与几何计算机图形学任务有关而选定的。

一些他们认为是成功的领域特定神经网络所共有的标准，以及一种尝试以更结构化的方式设计领域特定神经网络的方法。

本文旨在探讨 [9] 的理念。我们希望评估，如何运用几何深度学习的理念回答本文的两个核心问题，以及这些理念会如何影响神经网络的性能。

1.1 本文结构

本文的核心问题是：“优秀的领域特定神经网络是什么样的？”以及“如何构建这类网络？”，还有几何深度学习如何在这方面提供帮助。当然，在着手回答第二个问题之前，我们必须先找到第一个问题的答案。
本文将分为四个部分。
要回答“优秀的领域特定神经网络是什么样的？”这一问题，我们首先必须明确希望应用这些网络的设定。因此，第一章旨在清晰地界定我们所研究的一般设定，并明确指出深度学习研究中通常隐含的许多假设。

在介绍完背景后，第二章将介绍几何深度学习的准则。作者认为，这些准则是优秀的领域特定神经网络所具有的特征。为此，我们将首先考察领域通用性最强的神经网络——全连接神经网络，并找出其优点（我们希望领域特定神经网络也能保留这些优点）以及其不足之处（几何深度学习旨在解决这些不足）。接着，我们将探讨深度学习的核心思想，即作者认为应当指导领域特定神经网络设计的三项准则：任务分离、局部性和不变性。之后，我们将开始讨论如何构建遵循这些原则的神经网络。为此，我们将先以深度集合为例，展示这些原则。深度集合是一种面向集合的领域特定神经网络架构。接下来，我们将以此引出几何深度学习蓝图，即几何深度学习所提出的领域特定神经网络设计方法。

在介绍了几何深度学习的理念后，我们将通过图神经网络这一例子开展案例研究。我们将在此评估图神经网络是否符合几何深度学习的原则，以及几何深度学习蓝图是否有助于推导出这类网络。本章旨在评估，构建符合几何深度学习蓝图的神经网络在实践中是否可行。图神经网络比 Deep Sets 的案例研究更进一步，因为它们复杂得多，也是一种更成熟的模型。

最后，在第五章中，我们将从理论角度探讨几何深度学习理念的影响，尤其是不变性这一理念。为了全面了解这些影响，我们将从三个不同的角度进行考察。本章旨在评估，构建符合几何深度学习蓝图的神经网络所付出的努力，是否确实能由这些神经网络带来的性能提升来证明其合理性。

本文将以总结、讨论以及对领域专用深度学习架构这一领域未来研究方向的展望作为结尾。

第2章

背景与一般设定

本文的核心问题是：“优秀的 神经网络架构是什么样的，我们又该如何构建它们？”但在回答“什么是优秀的神经网络？”之前，我们首先必须定义优秀的 和神经网络 这两个术语的确切含义，而这两者看似简单，实则都很难回答。

由于具有高度的通用性，深度学习方法可以用于多种不同场景，从经典的监督学习到无监督学习和强化学习 [24,16]。

这使得我们几乎不可能对任何深度学习架构的质量作出有意义的判断。毕竟，很难证明不存在某种情境，使得任何一种架构都可能派上用场；而且，深度学习中的大多数术语定义都过于模糊，无法进行恰当的评估。

因此，我们首先必须明确界定要解决的问题，以便判断神经网络架构在这项任务中是否表现良好。因此，本文将聚焦于神经网络最基础且最为人熟知的应用场景：监督学习，或者更简洁地说，函数逼近。

以下各节主要分别依据 [24] 和 [42] 关于深度学习和统计学概念的著作。

我们考虑的一般设定如下：
假设存在一个高维数据空间 Ω，即某个可编码为 $\mathbb{R}^i$ for i large 的数据空间、某个可能的分类结果空间 C，以及一个未知函数 $f^*: \Omega \to C$，该函数属于某个已知的函数 F 类别，有待进行逼近。请注意，深度学习通常假设 Ω 和 C 是实向量空间。如果它们不是实向量空间，则需要使用额外技术将其转换为实向量空间。附录中讨论了实现这一转换的技术。此外，设 $D = \{(x_i, y_i = f^*(x_i)\}$ 是一组已知的样本点（也称为训练集）。现在的任务是利用数据 D 找到一个与 f^* 最为相似的函数 f。根据具体语境，这可能意味着许多不同的事情，稍后我们将对此进行更深入的阐述。

在本文的语境中，深度学习被理解为函数逼近这一更一般领域的一个分支，而神经网络则更恰当地被理解为一类逼近器，它们在函数逼近这一非常一般且已有充分研究的框架中发挥作用。

Thu接下来几节中，我们将首先回顾函数逼近的基本概念 n

和统计建模，然后介绍神经网络——一种特别适合解决某些问题的函数估计器。

一般设定

我们的一般设定如下：给定一个包含 f^* 的若干样本的集合 D，估计函数 f^*：$\Omega \to C$，其中 Ω 是一个高维领域。

2.1 参数化估计器、风险与损失函数

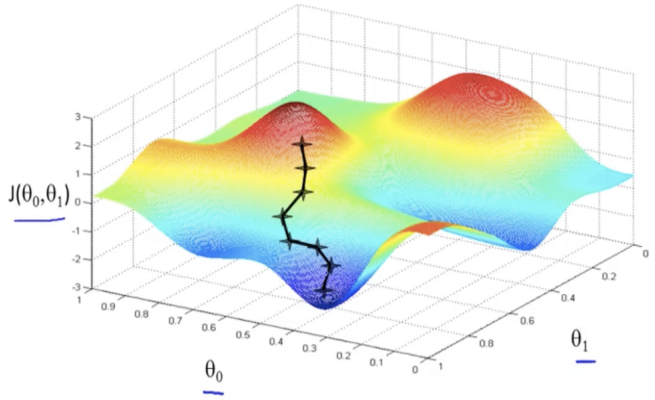


图 2.1： 梯度下降在一个简单的二维函数示例上运行。请注意基于梯度的优化所表现出的贪婪特性：它总是趋向最近的局部最小值，并沿着局部改进幅度最大的方向迈步。示例取自 [13]

.

面对上一节所述的任务，人们通常会选择一个估计器，并希望用训练数据 D对其进行拟合。直观来说，人们会预先选择一个候选函数族 F，在看到数据后，再选择一个自己认为最适合当前任务的函数 $f \in F$。例如，如果选择线性回归模型，F 对应于线性函数族 $\Omega \to C$；而在观察数据 D 后所选的函数 f，则是使样本与预测之间的平均平方距离最小的那个线性函数。有许多成熟且高效的方法可以完成这一任务，例如多项式插值或样条插值，它们可以根据样本 [17, 11]来近似一个函数。这些方法通常比神经网络更具可解释性，也更易于控制。然而，这些理论基础更为扎实的方法往往会随着维度增加而遭遇运行时间的指数级增长。例如，对于样条插值和多元多项式插值，所需项数都会随所考虑的维度数量 [28, 20]呈指数级增长。因此，我们希望找到一种方法，能够处理大量维度以及相应的输入样本，同时避免内存或时间开销呈指数级增长。实现这一目标的一种方法是使用参数化

估计器和基于梯度的训练。
对于基于梯度的估计器训练 [24], ，我们会考虑参数化估计器类。也就是说，我们假设存在一个参数空间 $\Theta = R^q$ ，使得可能的估计器空间F 可以表示为 $F = \{f_\theta \mid \theta \in \Theta\}$。这样一来，训练过程就简化了：我们只需选择合适的参数，从而得到良好的 f_θ。因此，使用参数化估计器时，训练过程中的内存使用量始终保持不变，因为我们只需将 θ 保存在内存中，而 θ 的大小不会改变。

稍微滥用一下记号，我们用 f 表示具有固定参数集合 f_θ 的具体估计器，有时也用它表示通过改变参数所诱导的估计器类别 F，例如在提及函数 f 可以近似时就是如此。这符合大多数估计器背后的直觉：首先定义一个不含参数的空模型，然后通过某种训练过程为其赋予参数。

如果上下文无法清楚地表明两者的区别，我们将用 $f_* = F$ 指代未参数化的估计器及由其产生的参数化估计器类别，并用 f_θ 指代具有一组固定参数 θ 的估计器。

作为参数化估计器的一个例子，我们来看看非常简单的线性模型。线性估计器 [42] 定义如下：

$$f_\theta(x) = \theta^T x \tag{2.1}$$

线性模型非常容易解释。线性模型的预测是其输入的加权求和，其中第 i 个输入由参数 θ_i 加权。在这种情况下，可近似的函数由 $f_* = \{f \mid f \in \mathbb{R}^n \to \mathbb{R}^m, \text{f linear}\}$ 给出，而一个具体的估计器 f_θ 就是一个单独的线性函数。

现在，我们将考虑如何使用线性估计器，根据一些样本 $D = \{(x_i, y_i = f^*(x_i)\}$，来近似函数 f^*。首先，我们必须定义理想的估计器函数应具备怎样的形式。通常，我们通过定义损失函数 $L(y, \hat{y})$ 来实现，该函数描述了当实际结果为 y 时，预测为 $\hat{y}$ 会造成多大损失。然后，我们希望估计器在所有可能的数据集上的期望损失最低；这也称为风险或泛化误差，并且需要在所有可能的估计器中达到最低。

$$R(f, f^*) = \mathbb{E}_{x \in \Omega}(L(f(x), f^*(x)) \tag{2.2}$$

如果能够找到一个函数 f，使其最小化这个风险函数，就可以得到相对于损失函数 L 的理想估计器。遗憾的是，在实践中， R 几乎从来都无法计算，因为我们无法获取 f^* 和真实的数据生成分布 $P(x)$。然而，如果假设 D 是根据真实的数据生成分布抽取的，也就是说，对于每个 $(x_i, y_i) \in D$，x_i 都是从 $P(x)$ 中抽取的，我们就可以用经验风险或训练误差来近似真实风险：

$$\hat{R}(f, f^*) = \sum_{(x_i, y_i) \in D} (L(f(x_i), y_i) \tag{2.3}$$

乍一看，人们可能会认为这是一个非常好的风险估计器 R ，因为通常来说，期望值的近似的方差与 $1/n$ 成正比，其中 n 是样本数量。然而，尽管这确实如此，要正确确定 $\arg\min_f R(f, f^*)$，我们需要一个适用于所有可能估计器 f 的估计器。当然，我们的估计器不太可能对所有函数值 f 都可靠$\hat{}$，因此优化 R 未必能很好地对应于优化 $R\hat{}$。因此，能够取得良好 R 的估计器，即在

训练数据上表现良好的估计器，未必是关于 R 的良好估计器，也就是说，它未必能在整个任务上取得良好的总体性能。构建一个能在所有可能的数据集上表现良好的估计器是一个非常困难的问题，而且不存在通用的方法来实现这一点。

2.1.1 确定一个合理的损失函数

当然，构造一个合理的损失函数供优化至关重要。如果所选的损失函数未能充分描述错误预测所带来的代价，那么即使优化过程完美运行，最终得到的估计器也必然不适合我们的任务。不过，基于最大似然估计器这一概念，有一种构造此类损失函数的方法，其原理清晰且结构严谨 [18]。

这种方法通过向估计器中添加一些随机噪声来实现。因此，估计器不再只是一个函数，而是一个概率密度函数，它给出的不只是一个估计值 $f(x)$，而是一个描述 $P(y|x)$ 的密度。其背后的思路如下：如果估计器不是随机的，那么我们对 f_θ 的猜测就只有真或假两种可能（即 $f(x) = f^*(x)$，或者不是）。这种二元分类会使学习变得困难，因为估计器发生微小变化时，结果往往不会改变。然而，对于随机估计器， $P(y|x)$ 通常不为零。因此，这为我们提供了一个关于估计器有多可能为真的判断依据，从而为优化指明方向。

回到线性分类器的例子，我们可以向分类器中添加一些正态分布噪声 $\epsilon \sim N(0, \sigma)$，从而得到

$$P(y|x,\theta) = N(\theta^T x, \sigma) \propto e^{\frac{-1}{2\sigma^2}(x-\theta^T x)^2} \tag{2.4}$$

这就是线性回归模型的标准形式。
如前所述，这不仅能让我们量化预测器在某项任务中是否出错，还能量化它与真实目标值[1]相差多远。现在，最大似然估计器是指任何满足以下条件的估计器： f 在数据由 f_θ 生成而非由f^*生成的情况下，使观测到这些数据的概率最大

$$P(D|f) = \Pi_{(x_i,y_i)\in D} P(y_i|x_i, f) \tag{2.5}$$

可以证明，如果样本数量趋于无穷大，且我们关于f^* 的假设成立，即我们假设添加到预测器中的误差ϵ_f 与数据中的噪声具有相同的分布，那么最大似然估计器将收敛到θ的最优估计器。

因此，只需最大化似然；只要噪声 ϵ 的性质适合当前任务，就有望为 θ 取得良好的值。如果要最大化似然，也可以等价地最小化其负对数

$$L(x, y) = -\log(P(y|x, f)) \tag{2.6}$$

从而得到一种实用的损失函数形式。回到线性模型的示例，我们得到以下损失函数（忽略不影响优化的加性常数）：

$$L(x, y) \propto (\theta^T x - y)^2 \tag{2.7}$$

[1]当然，我们有必要根据任务判断哪种噪声假设较为合理。通常，这取决于数据的类型：连续任务通常采用正态分布，离散任务则采用二项分布。

这就是标准的均方误差损失，常用于许多机器学习任务 [24]。

损失函数

参数化估计器解决了内存瓶颈：由于我们只考虑形式为 f_θ for $\theta \in \Theta = \mathbb{R}^k$ 的估计器，因此只需在内存中保存 θ，从而使内存使用量保持恒定。损失函数描述了估计器对数据的拟合程度。理想估计器是使期望损失最小的估计器，也称为风险。由于风险无法计算（因为我们只能访问训练集 D），因此选择使 R 最小的估计器； R 是数据集上的平均损失。损失函数通过向估计器添加随机噪声，并使用 $-log(P(y|x,\ f))$ 作为损失函数来获得。因此，如果 f^* 带有噪声，那么在数据量足够、噪声正确且优化达到全局最优解的情况下，该估计器就是一个最大似然估计器，并会收敛到 f^*。

2.1.2 基于梯度的优化

ˆ 定义了损失函数及其对应的经验风险 R之后，还需要找到使经验风险最小化的估计器 f_θ，或者等价地，找到其对应的参数向量 θ。

对于大多数非线性估计器而言，找到全局最优解是一个非常复杂的问题。例如，即使是为一个非常简单的神经网络寻找最优参数集合，也是 NP 困难 [40]。
因此，我们将讨论范围限定在能够找到局部最优解的高效算法上。梯度下降是一种实用的算法。可以使用梯度下降算法，或其众多变体之一，例如在实践中广泛使用的 ADAM [34] 。

梯度下降是一种迭代算法：它从一个随机猜测值[2] 开始，并通过多个时间步迭代地确定参数 θ_0。在梯度下降及其所有变体中，时间步 $t+1$ 处的参数下一次猜测值为：

$$\theta_{t+1} = \theta_t - \alpha \nabla_\theta \hat{R}(f_{\theta_t}, D) \tag{2.8}$$

其中学习率 α较小。梯度下降的各种变体仅在 α的选择上有所不同。ˆ 其背后的思路非常简单。梯度 $\nabla_\theta R(f_{\theta_t},\ D)$ 始终指向局部上升最快的方向。这意味着，对于无穷小的步长，梯度指向局部上使经验误差降低最快的方向

$$e(\theta) = \hat{R}(f_{\theta_t}, D) \tag{2.9}$$

对 $e(\theta)$ 进行泰勒展开，得到：

$$e(\theta + \delta\theta) \approx e(\theta) + \nabla_\theta e \delta\theta \tag{2.10}$$

[2]当然，在实践中人们不会随机初始化。已有各种精妙的初始化策略（参见： [22]），但由于它们对本文不太重要，且超出了本文的讨论范围，我们不会介绍。

并代入公式2.8中描述的更新，得到 $\delta\theta = -\alpha\nabla_{\theta_t}$

$$\begin{aligned} e(\theta_{t+1}) &= e(\theta_t + \nabla\theta_t) \approx e(\theta_t) - \nabla_{\theta_t} e \nabla_{\theta_t} e \\ &= e(\theta_t) - ||\nabla_{\theta_t} e||^2 < e(\theta_t) \end{aligned}$$

因此可以看出，如果学习率 α 足够小，使泰勒近似足够准确，那么在梯度下降的每次迭代中，误差都保证会减小。
其背后的想法相对简单：可以将函数 $e(\theta)$ 想象成一张绘制出的图。在每一步中， θ_i 都会朝着 e 下降最陡的方向移动。就像球沿着山坡滚下去一样， θ_i 的值会沿着 e的图像下降。由于 $e(\theta_i)$的值在每一步都会改善，因此该过程最终必定会找到一个局部最小值，或者会一直改善下去，直至无穷大。后一种情况在实践中并不真正发生，因为形式为 $log(P(D|f_\theta))$ 的损失函数有一个下界 0。由于人们很少要求损失恰好为零，如果过程未能收敛但仍在持续改善，就可以提前停止。

虽然梯度下降可以保证降低函数的原始误差，而且只要学习率足够小，在所有实际应用中也都能保证收敛，但它仍有一些需要注意的问题。首先，梯度下降通常不会找到全局最优解，而只会找到函数的某个局部最优解。这当然是由梯度下降的局部性质决定的：梯度下降始终只会贪心地朝着局部最优的方向移动，因此永远无法保证达到全局最优解。此外，在函数 e所有可能的局部最优解中，梯度下降往往会选择靠近其起点的局部最优解。原因很简单：梯度下降始终只采取局部步骤，因此更有可能到达附近的局部最优解。由于能力很强的估计器通常存在许多局部最优解（因为它们可以通过多种不同方式拟合整个训练数据集），所以梯度下降的这一性质非常重要，因为它揭示了梯度下降倾向于选择哪一种可能的解。

基于梯度的优化

在这项工作中，我们只考虑通过基于梯度的优化方法训练、以最小化损失函数的神经网络。基于梯度的优化方法是近似的函数最小化器，用于寻找使某个依赖于 θ 的函数达到最小值的参数 θ 。基于梯度的方法非常通用；它们只要求 f_θ 关于 θ 可微。此外，它们只需要 $O(t * d)$ 步，其中 d 是计算微分 $\frac{\partial f_\theta}{\partial d\theta}$ 所需的时间， t 是训练步数。只要学习率 α 足够小，基于梯度的优化总能找到一个局部最小值。然而，一般来说，基于梯度的优化无法找到全局最优解。 —

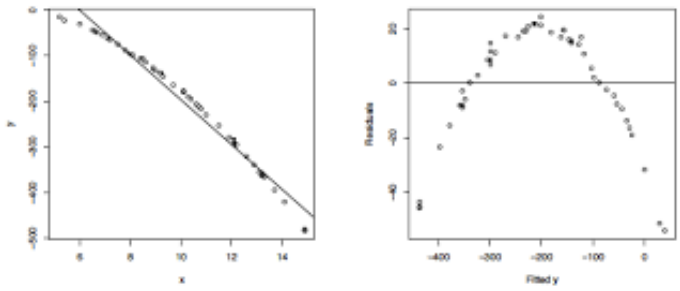


图 2.2: 线性模型估计线性函数（左）和二次函数（右）。第一个估计任务完成得很好，第二个则不然。这当然是因为模型中存在的归纳偏置假定 f^* 呈现线性特征，而实际上并非如此。

2.2 正则化和归纳偏置

在某些方面，训练一个估计器是一项棘手的任务。如果观察到一组样本 $D = \{(x_i, y_i)\}$，那么任何能够正确分类所有样本的估计器，即对于所有 i， $f(x_i) = y_i$，理论上都可能是真实函数 f^*。所有能够正确分类这些数据的函数所构成的集合

$$\{f \mid f : \Omega \to C, \forall i \; f(x_i) = y_i \} \tag{2.11}$$

通常是不可数无限的。如果没有其他信息，就必须从 $S(D)$ 中任意选择一个函数。从不可数无限集合中正确猜中的概率恰好为 0；对于几乎所有任务，猜出一个函数 f，使其与 f^* 足够接近的概率也为 0。

然而，每种机器学习算法确实都会从 $S(D)$ 中选择一个函数，而且其中大多数算法的选择方式在实践中至少有一定用处。
机器学习算法的 归纳偏置定义为编码在算法所选估计器中的规则和假设。例如，使用线性模型时，隐含的假设是函数 f^* 为线性的，因此会从 $S(D)$ 中选择一个线性函数。给定足够多的样本点后，只有一个这样的线性函数能够正确分类所有样本，因此该函数是唯一确定的。

当然，模型的归纳偏置在很大程度上依赖人类专家知识，尽管人们通常不会明确讨论这些知识。正如著名的无免费午餐定理所指出的，任何机器学习算法都不可能优于其他算法，除非它利用了人类关于领域 Ω 和待近似函数 f^* [2] 的知识。

因此，当且仅当机器学习算法的归纳偏置在其应用领域中有用时，它才是好的。请看图 2.2。可以看到，在线性情形下，线性模型效果极佳；但在近似二次函数时，它却完全不起作用，因为其归纳偏置与当前问题不匹配。

请注意，机器学习算法的归纳偏置与人们对函数 f^* 所作的假设之间存在某种二元性：归纳偏置应基于这些假设，并且往往与之相呼应。也就是说，如果我们假设 f^* 呈线性变化，通常也希望 f 呈线性变化。

For 对于大多数机器学习算法，都存在一组内置于算法本身的归纳偏置。 nt

在线性估计器中，这一假设是线性函数能够很好地估计 f^*。在傅里叶级数估计器中，这一假设是正弦函数和余弦函数的线性组合能够很好地进行估计。不过，除了这些归纳偏置之外，还有一些偏置通常可以由用户微调。

这些由用户主动引入的归纳偏置，通常称为正则化，是完整归纳偏置集合的一个子集。按照 [24], 的定义，正则化是指我们对学习算法所做的任何修改，其目的在于降低算法在整个未知数据集上的风险，而非降低训练误差。实现方式多种多样，本节将考察向损失函数中添加项这一做法。

其核心思想是：如果估计器不遵循设计者希望施加的结构，就对该估计器的预测额外增加损失。因此，训练算法需要在优化训练误差与遵循预期结构之间寻求权衡；或者，在可能的情况下，选择能够优化训练损失且最接近遵循正则化目标的函数。

为说明这一点，我们将简要回顾 L2 正则化方案。L2 正则化将通常的损失 $L\hat{}$ 替换为修改后的损失 L:

$$\hat{L}(x,t) = L(x,t) + \lambda||\theta||^2 \tag{2.12}$$

其中，将 f 的参数向量的欧几里得范数（或 L2 范数）加到损失函数 L中。

这种形式会使权重分布极不均匀且远离零的估计器产生较高的损失值，而对于参数值仅略微偏离零的估计器，损失值只会略有增加。回想一下线性估计器的例子

$$f_\theta(x) = \theta^T x \tag{2.13}$$

对参数施加较低 L2 范数约束背后的直觉相对简单：参数会被推向零，并趋于较为均匀的分布。因此，如果改变 x ， f_Θ的函数值不会发生太大变化，单个变量 x_i 对估计结果的影响也会减弱。

根据任务的不同，这可能是很好的正则化选择。例如，在进行图像识别时，单个像素发生微小变化几乎总是无关紧要的。图像中的大面积区域必须整体发生变化，才能造成有意义的差异。然而，对于棋盘来说，少了一枚棋子就可能彻底改变棋局结构，因此在这种情况下使用 L2 正则化并不恰当。总的来说，L2 正则化会产生“复杂度较低”的估计器[3] ，因此直观上并不适用于更复杂的任务。这生动地展现了归纳偏置以及正则化具有很强的领域专用性。

还有许多其他正则化方法，例如丢弃法或提前停止 [49, 3]，但它们的目标都相同：引导分类器朝着一个方向发展，而这个方向被认为是有用的；这种判断并非来自数据，而是来自估计器的设计者。
对于通过在损失中添加另一项来实现的正则化方法，可以基于贝叶斯统计给出一种非常直观的解释 [47] ，而且很容易将其推广到更一般的

[3]请注意，L2 正则化作用于参数，因此其效果与使用这种正则化的估计器类别密切相关。不过，正如我们稍后将看到的，L2 正则化对线性模型的影响也会延伸到神经网络，只是作用方式有所不同。

设定中。在贝叶斯设定下，人们会为可能的具体估计器定义一个先验概率测度 $P(f)$。然后，通常希望使用后验概率最高的估计器：

$$P(f|D) \propto P(f) * P(D|f) \tag{2.14}$$

根据贝叶斯定理，可以得到这个恒等式。当然，最大化这个量等价于最小化它的负对数：

$$-\log P(f|D) = -\log P(f) - \log P(D|f) + c \tag{2.15}$$

其中，c 是一个归一化常数，与优化本身关系不大。回到第 2.1 节中损失函数的推导，可以看到，$-\log P(D|f)$正是损失函数。因此：

$$-\log P(f|D) = -\log P(f) - L(f|D) + c \tag{2.16}$$

将其与公式 2.12 对比可以看出，在这个例子中，先验的负对数 $-\log P(f)$所起的作用与添加的正则化损失完全相同。因此，正则化可以解释为：根据先验 $P(f)$，对函数 f [4] 进行最大后验估计，其中 ρ是损失项。从这个角度看，损失函数中的正则化项编码了关于 f^* 的候选函数的先验信念（即在看到实际数据之前持有的信念），因而是一种相对软性的约束。它并未断定参数值较大的估计器不可能出现，只是降低了它们被选中的可能性。Wit

请记住这一点，并回想一下 L2 损失的形式。这里，正则化项由.....给出 y:

$$\rho(f_\theta) = ||\theta||^2 \tag{2.17}$$

因此，相应的先验信念为：

$$P(f) = e^{\rho(f_\theta)} = e^{-||\theta||^2} \tag{2.18}$$

除归一化常数外，这等价于估计器参数服从正态分布。

然而，还有比损失函数中添加额外项所引入的归纳偏置更强的归纳偏置。例如，可以添加严格要求，使得对于某些特定函数 f，$P(f) = 0$。当然，也可以通过将这些函数对应的损失项设为无穷大来实现，从而始终阻止它们被选为最终结果。然而，这必然会导致数值问题，并使得无法计算有意义的损失函数梯度。

因此，估计器的架构通常会内置更严格的要求。例如，使用傅里叶变换时，人们可能会假设真实函数 f^* 只包含宽广的低频模式，而数据中出现的所有高频都是噪声。因此，人们会直接舍弃所有较高频率，而不是假设较高频率不太可能存在，并将这一先验纳入数学优化。需要谨记，归纳偏置以及由此延伸出的正则化并不是

[4]请注意，从统计学角度而言，这里省略了一些技术细节。事实上，在机器学习的情境中，人们最大化的是参数的概率 θ，而非相应函数的概率 f_θ。尽管二者密切相关，但仍可能存在 θ_1, θ_2 使得 $f_{\theta_1} = f_{\theta_2}$。在这种情况下，理论上，全局最大值 for θ 可能并不对应于 f_θ的全局最小值

人们可以主动选择使用的东西。从一类能够描述给定数据的可能估计器中选择一个估计器，这一简单行为就意味着人们带有某种决定所选函数的归纳偏置。因此，选择的并非是否纳入归纳偏置，而是应选择哪种归纳偏置；对于任何机器学习算法而言，最关键的就是为当前任务纳入正确的归纳偏置。

归纳偏置与正则化

根据无免费午餐定理，在任意问题上计算时，所有机器学习方法的平均泛化误差都相同。因此，“函数逼近器是否优秀？”归根结底取决于人类的判断，无法一概而论。“归纳偏置”是一个非正式术语，用来描述估计器如何利用训练数据集进行泛化。要判断某个估计器是否有用，只能由人类评估其归纳偏置。

2.3 深度学习

本节将简要回顾本研究涉及的最重要的深度学习概念。首先定义与深度学习相关的重要术语和符号，然后介绍本研究后续部分将用到的深度学习模型的构建模块和基础知识。

2.3.1 一般术语与定义

在介绍本研究涉及的深度学习相关概念之前，先做好一些术语上的铺垫至关重要。本研究的核心关注点是不同的神经网络架构，因此有必要明确神经网络架构究竟是什么。“神经网络”一词的定义非常宽泛，涵盖了许多差异迥异、彼此几乎没有交集的模型；例如，变换器模型、全连接神经网络和玻尔兹曼机的结构就截然不同，但它们都被视为深度学习模型 [58, 24]。

为了不对神经网络的形式施加结构限制，我们在此采用一种非常激进的定义，将深度学习定义为“使用基于梯度的优化进行函数逼近”，因此将神经网络定义为“可使用基于梯度的优化器进行训练的函数”。事实上，前述所有模型都使用基于梯度的优化器进行训练。为了实现基于梯度的优化，神经网络必须由某个参数集合 θ 进行参数化，并且关于这些参数可微。因此，如果一个函数经过参数化且关于其参数可微，我们就称它为可训练。

我们将神经网络宽泛地定义为“使用基于梯度的优化进行训练的函数逼近器”，因此可以更具体地将神经网络定义为可训练的连续函数$f\colon \Omega \to C$，其中 Ω 是可能数据的空间，而 C 是可能类别的空间。

请注意，确实存在一些不使用梯度下降来训练神经网络的方法，例如遗传方法 [26]，因此这些方法不属于我们对深度学习的界定。尽管如此，我们认为这个定义是合理的，因为与本研究相关的几乎所有方法都采用基于梯度的优化，而且本研究介绍的大多数（如果不是全部）深度学习方法都是为配合基于梯度的优化而设计的，因此符合可训练函数这一称谓。

我们用 θ 表示具体的参数集合，用 Θ 表示某个神经网络所有可能参数配置构成的空间。

我们用 $\text{rep}(f) = \{f_\theta | \theta \in \Theta\}$ 表示某个神经网络 f 在给定参数集合 θ 时能够表示的所有函数所构成的空间。此外，我们将神经网络架构定义为一族神经网络 F；如果可以选择 $f \in F$，使得可表示函数 $rep(f)$ 在 S[5] 中任意稠密，那么我们就称该神经网络架构是函数空间 S 的通用逼近器。

请注意，深度学习处于计算机科学、统计学和数学的交叉领域，因此神经网络可以有许多不同的理解方式，例如基于图的模型、函数等，因而也可以采用不同于此处所给定义的方式来定义。在本文中，我们主要将神经网络视为函数逼近器，因为几何深度学习的大多数思想都受这一观点启发 [9]。因此，只要可能，我们通常会忽略具体的基于图的结构，转而关注神经网络能够逼近的函数空间。

接下来，我们将回顾一些与本研究相关的神经网络架构，并介绍它们的应用场景和关注领域。本节主要基于 [24]，但经过调整，以更好地契合 [9] 中提出的观点。

2.3.2 全连接神经网络

全连接神经网络（FC）是最早出现的深度学习模型之一，也被称为前馈神经网络或多层感知机 [24]。全连接网络可以表示为一个图 $G = (V, E)$，该图被划分为多个由顶点组成的层 $V = \coprod_i l_i$[6]（称为“神经元”），这些顶点通过带有权重的边相连$e_{i,j}$ $w_{i,j}$ 当且仅当节点属于相邻层时，它们之间才有边相连[7]。

全连接神经网络以任意向量 $u = (u_1, ..., u_n)^T \in \mathbb{R}^n$ 作为输入，并再次输出向量 $\mathbb{R}^m$，其中 n、m。全连接神经网络逐层进行计算，为每个神经元 $v \in V$ 计算所谓的“激活” $a(v)$。

在第一层中，将激活设置为与输入一致：

$$a(v_{0,i}) = u_i \tag{2.19}$$

之后，激活值逐层计算，每一层都基于前一层的结果。在第 $k+1$ 层，激活值为：

$$a(v_{k+1,j}) = \phi\Big(\sum_{i=0}^{dim(k)} a(v_i) w_{i,j} + b_j\Big) \tag{2.20}$$

[5] 例如，对于全连接神经网络，众所周知，当神经元和层的数量趋于无穷时，神经网络的可表示函数在空间 $\mathbb{R}^n \to \mathbb{R}^m$ 中任意稠密。[6] 符号 $\coprod$ 在此表示不相交集合的并集。[7] 这刻画了网络特有的全连接性

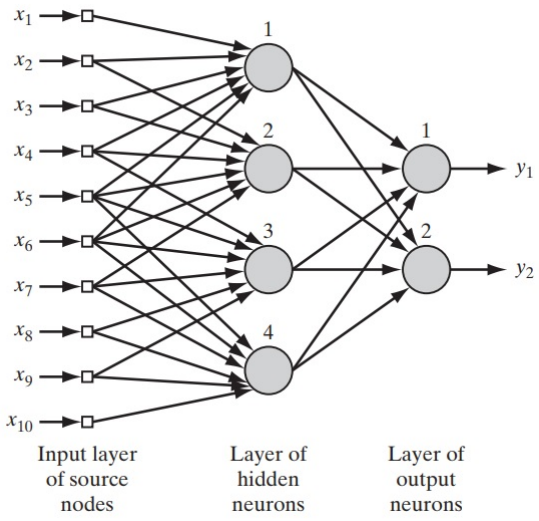


图 2.3： 全连接神经网络示例。请注意其分层架构，以及边严格朝向前方的方向。还要注意，每一层中的所有神经元都与下一层中的所有神经元相连。插图来自 [31]

.

where ϕ 是某个非线性、可微且预先确定的函数，称为激活函数 n.

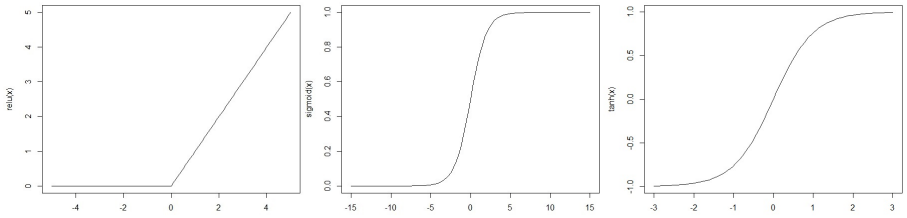


图 2.4： ϕ中几种最受欢迎的激活函数选择。请注意，原则上，只要函数可微且最好是单调的，激活函数可以选择任意函数。修正线性单元（ReLU）是撰写本文时激活函数的事实标准选择 [24]，因此本文所有示例均采用该函数。

如果省略了 ϕ，神经网络就会退化为一个简单的线性模型。这样，输入便逐层通过网络传播，最终到达最后一层。最后一层中的激活值被解释为输出；因此，可以将神经网络视为一个函数 f，使得 $f(u) = \{a(v_{L,1}, \ldots\ldots a(v_{L,n})\}$.

对于本研究而言，以向量的角度来看待这些计算或许更直观；与其单独考察神经元，不如研究每一层对其特定输入所产生的影响。当然，给定输入 u，第一层中堆叠的激活 $a(v)$ 所组成的向量 $l_u = (a_{0,1}, \ldots\ldots, a_{0,n})$ 可表示为：

$$l_0(i) = u \tag{2.21}$$

同样，后续各层可以表示为

$$l_{i+1} = \phi(W_{i+1} l_i + B_{i+1}) \tag{2.22}$$

其中， W_{i+1} 和 B_{i+1} 分别表示堆叠权重矩阵 $w_{i,\ j}$ 和堆叠偏置向量b_j ，而 ϕ 则逐元素作用于其输入参数。这个方程中包含两个不同的操作：首先，将 l_i 传入一个由 W_{i+1} 和 B_{i+1}构成的仿射变换中。随后，再应用一个非线性函数。因此，神经网络所表示的完整函数为：

$$f := \phi \circ A_L ....\phi \circ A_2 \phi \circ A_1 \tag{2.23}$$

其中， A_L 是前文所述的仿射变换， ϕ 是非线性激活函数。因此，全连接神经网络本质上由非线性激活函数 ϕ 与仿射变换 A_i交替串接而成。由于 ϕ 已预先固定，而 A_i是 f中唯一的参数化部分，因此它们是经过训练的函数； ϕ 仅用于为该函数引入足够的非线性结构。

这种结构使全连接神经网络具有极强的表达能力。正如 [32], 所示，只要提供足够多的层和/或神经元，全连接神经网络就能以任意高的精度逼近定义在 R^n 的紧子集上的任意连续函数[8]。

因此，全连接神经网络所能表示的函数空间为：

$$rep(FC) = \{g \mid g \in \mathbb{R}^n \to \mathbb{R}^m, \text{g continuous}\} \tag{2.24}$$

深度学习总结

本节介绍了深度学习。由于“深度学习”一词含义模糊，几乎不可能找到一个连贯的定义。因此，我们将所有能够通过梯度下降训练的函数都称为“神经网络”。于是，“什么样的神经网络经过基于梯度的优化训练后，能在我们的领域中取得良好效果？”这一问题就变成了：“什么样的可训练函数经过基于梯度的优化训练后，能在我们的领域中取得良好效果？”

接下来，我们介绍了全连接神经网络，将其作为神经网络的基线。全连接神经网络是最基本的神经网络形式。最有意思的是，它们能够以任意精度逼近每一个实值连续函数。

一方面，这保证了从理论上说，全连接神经网络能够逼近 $\mathbb{R}^k$ 的任意紧子集上的任何连续函数，因此，如果函数 f^*是连续的，它也能逼近该函数。另一方面，我们可能知道某个函数 g 不可能是 f^* 的合理猜测，但我们的神经网络仍可能逼近 g。如前所述，我们只能根据使用神经网络所要解决的问题来评估这种做法是否有效。此外，这也意味着仍有一些关于神经网络为何有效的问题悬而未决：如果理论上它们能够逼近任何连续函数，我们如何确定它们逼近的是有用的函数?

2.4 全连接神经网络的归纳偏置

现在，我们已经明确了要讨论的设定，可以开始回答“什么是好的领域专用神经网络”这个问题。正如本节始终所述，这个问题更具体地可以表述为：

[8]可以证明，无论是采用固定宽度、层数任意的网络，还是在单个层中设置任意数量的神经元，都足以实现通用逼近 [46, 45]

深度学习与全连接神经网络

在本文的语境中，深度学习是使用神经网络进行函数逼近。神经网络是可以通过梯度下降进行训练的函数，因此是参数化的，并且相对于其参数可微。全连接神经网络就是这样一种函数 $\mathbb{R}^k \to \mathbb{R}^l$。全连接神经网络由交替排列的仿射变换和固定的非线性激活函数构成；仿射变换具有可训练参数，而激活函数通常采用 ReLU 函数。只要层数和神经元数量足够，全连接神经网络 f 就能在 $\mathbb{R}^k$ 的紧子集上逼近每个连续函数 g，使得对于每个 $\epsilon > 0$，$\sup_{x \in \mathbb{R}^k} ||f(x) - g(x)|| \leq \epsilon$。

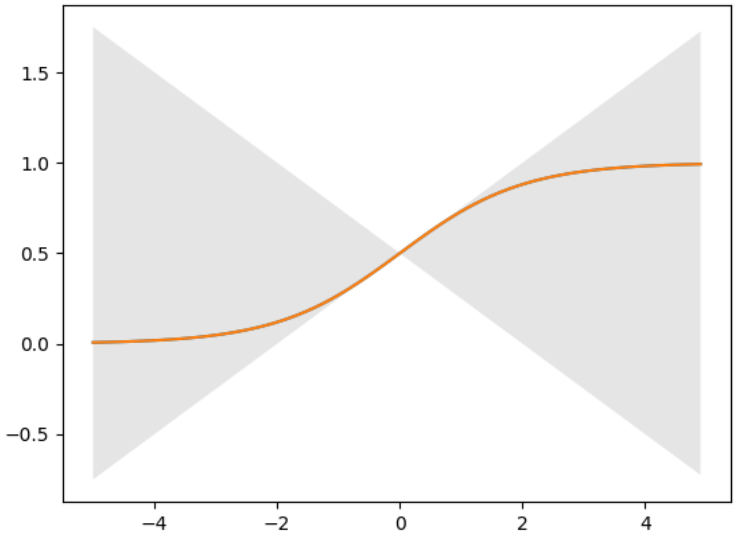
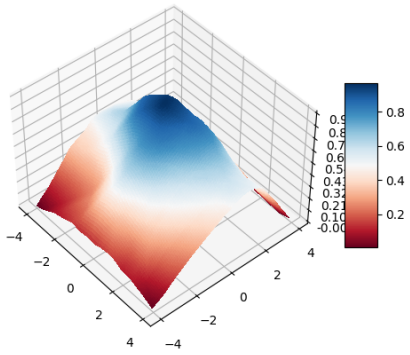


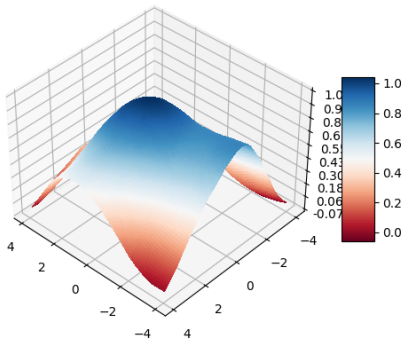
图 2.5： Sigmoid 激活函数 σ，其利普希茨常数为 $L(\sigma) = \frac{1}{4}$。灰色区域表示由 $-x = 0$ 处的利普希茨常数所限定的函数取值范围。$\frac{1}{4}$ 的利普希茨常数表明，函数的变化速率最大为 $\frac{1}{4}$。请注意，该函数始终不会超出以这种方式确定的边界。

什么样的可训练函数既能通过基于梯度的优化得到良好训练，又能使其行为符合我们的领域知识？

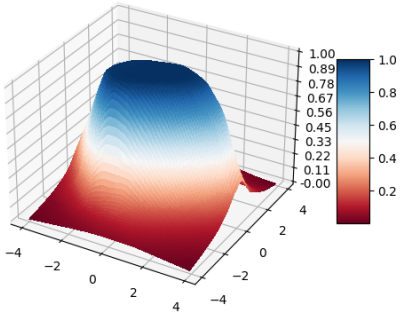
为回答这个问题，我们将按以下方式展开：首先，我们将考察全连接神经网络的表现，并以此作为基线。即使没有明确以领域知识为基础，全连接神经网络在分类任务中也表现得相当不错。因此，可以合理推断，它们具备一些使其成为实用分类器的特性，而这些特性也是我们希望在领域专用神经网络架构中保留的。我们尤其关注它们的归纳偏置，即神经网络如何泛化到未见过的数据点。随后，我们将考察全连接神经网络在特定领域中是否存在不足，以及如何利用领域知识和几何深度学习的思想来弥补这些不足。



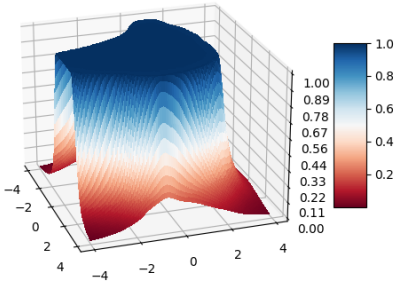
(a) 2 层



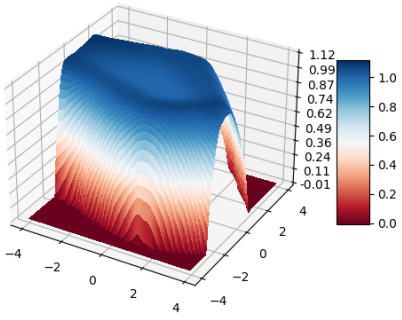
(b) 5 层



(c) 15 层



(d) 25 层



(e) 35 层

图 2.6： 层数不同的神经网络学习相同的任务。每层都接收相同的数据点。每个角上各有一个数据点（即 (x, y)，其中 $|x| = |y| = 4$

值为 0，另一个位于中间、坐标为 (0,0) 的点值为 1。每个网络都能解决这个问题，但插值方式各不相同。最显著的是，层数更多的神经网络会近似出更不平稳、利普希茨常数更高的函数。这些函数具有更陡峭的“悬崖”，与具有 2 层或 5 层的网络相比尤其如此。请注意，这种表现并无益处：所有模型的误差都为 0；之所以出现这种情况，仅仅是因为层数带来了不同的归纳偏置。

首先，神经网络引入的最重要、最一般的归纳偏置，是它们能够逼近任意连续函数。也就是说，神经网络能够学习的每个函数必然是连续的。因此，我们假设输入之间存在某种欧几里得相似性：根据连续性，如果输入 x 和 y 在欧几里得意义上相近，我们可以假设 $f(x) \approx f(y)$。这是神经网络的一个核心概念。它们（至少在一定程度上）通过假设相邻点对 f^* 具有相似的值来实现泛化。

虽然连续性本身告诉我们，神经网络会以相似的方式处理相邻点，但它并没有告诉我们这些点必须相距多 近，才能被视为足够相似。不过，根据[53]的研究，关于神经网络的利普希茨连续性，也有一些更强的结果。

A fu函数 f 被称为利普希茨连续函数，如果存在某个常数 L，使得对于所有 $x,\ y \in \Omega$

$$||f(x) - f(y)|| \leq L||x - y|| \tag{2.25}$$

使上述条件成立的最小常数 L称为 f 的利普希茨常数 $L(f)$。根据定义可知，利普希茨连续性对函数 f 的要求比连续性更严格：利普希茨常数为 $L(f)$ 的函数，在领域 Ω 中每前进单位长度，其值最多只能变化 $L(f)$。因此，利普希茨连续性可以重新表述为对梯度及偏导数的上界：

$$\sup_{x \in \Omega} ||\nabla_x f|| = L(f) \tag{2.26}$$

因此

$$\frac{\partial f(x)}{\partial x_i} \leq L(f) \tag{2.27}$$

神经网络的利普希茨常数对于理解神经网络中存在的归纳偏置非常有意义。利普希茨常数越小，彼此邻近的点就越相似。因此，如果神经网络经过训练，能够完美估计每个训练点，即$f(x_i) = y_i$ 对所有 $(x_i, y_i) \in D$ 都成立，那么很容易看出，对于满足 $||x_i - p|| = d$ 的每个点 $p \in \in \Omega$,

$$f(x_i) - dL(f) \leq f(p) \leq f(x_i) + dL(f) \tag{2.28}$$

因此，对于能够完美拟合训练数据的利普希茨连续估计器而言，训练数据点 y_i 会限制其欧几里得邻域内各点的可能取值。 [9],
要将这一点与实际的神经网络联系起来，还缺少对全连接神经网络的利普希茨常数的考察。我们将利用 [53]中的结果，说明全连接神经网络的利普希茨常数如何变化。回顾一下，[10]

$$\sup_{x \in \Omega} ||\nabla_x f|| = L(f) \tag{2.29}$$

因此，要找到利普希茨常数 $L(f)$，只需找到使梯度 $\nabla_x f$长度最大的那个 $x \in \Omega$。为了更直观地理解，假设我们要处理一个只有一个输出神经元的神经网络 f，即 f 的值域为 $\mathbb{R}$，使得梯度 $\nabla_x f$ 确实是一个

[9]虽然这并非必然，但正如 [66]所示，现代的强大神经网络在大多数应用中都能以高精度拟合任意数据集。[10]再次注意，神经网络始终是利普希茨连续的，因此每个神经网络都存在某个利普希茨常数。

向量。
如 [53], 中所述，我们可以考察网络中每个神经元的利普希茨常数。对于输入层，每个神经元的激活等同于给定的输入：

$$v_{0,i} = x_i \tag{2.30}$$

因此，梯度 $\nabla_x v_{0,\ i}$ 显然为

$$\nabla_x a_{0,i} = (0, 0, ..., 1, ..., 0)^T \tag{2.31}$$

其中，该向量的第 i 个分量等于 1，因此输入节点的利普希茨常数为 1。

对于第 k+1 层中的一个神经元，其激活表示为：

$$a(v_{k+1,j}) = \phi(\sum_{i=0}^{dim(k)} a(v_i)w_{i,j} + b_j) \tag{2.32}$$

这里可以应用链式法则，得到 $\nabla_x v_{k+1,\ j}$：

$$\nabla_x v_{k+1,j} = \nabla_x \phi(\sum_{i=0}^{dim(k)} a(v_i)w_{i,j} + b_j) \tag{2.33}$$

$$= \phi'\left(\sum_{i=0}^{dim(k)} a(v_i)w_{i,j} + b_j\right) \sum_{i=0}^{dim(k)} w_{i,j}\nabla_x a(v_{k,i}) \tag{2.34}$$

为了求出神经网络的利普希茨常数，即 $\sup_x ||\nabla_x v_{k+1,j}||$，我们需要求出 $||\nabla_x v_{k+1,j}||$ 的一个上界。根据三角不等式：

$$||\nabla_x v_{k+1,j}|| \leq \phi'\left(\sum_{i=0}^{dim(k)} a(v_i)w_{i,j} + b_j\right) \sum_{i=0}^{dim(k)} w_{i,j}||\nabla_x a(v_{k,i})|| \tag{2.35}$$

如果我们能找到该方程右侧对所有 x 的一个上界，就可以推导出利普希茨常数 $L(f)$ 的一个上界。

根据定义，很容易看出 $\nabla_x a(v_{k,i})$ 受前面各层的利普希茨常数 $L(v_{k,i})$ 所界定。类似地，量 $\phi'\left(\sum_{i=0}^{dim(k)} a(v_i)w_{i,j} + b_j\right)$ 受 ϕ 的利普希茨常数所界定；例如，对于 ReLU 激活函数，该值为 $L(\phi) = 1$ 。因此，我们得到：

$$L(v_{k+1,j}) \leq L(\phi) \sum_{i=0}^{dim(k)} |w_{i,j}|L(v_{k,i}) \tag{2.36}$$

对于所有权重都相同的全连接神经网络，可以达到这个不等式。

请观察该方程中的递归模式：层 k 中神经元的利普希茨常数上界，等于前一层利普希茨常数的加权求和。只要每个神经元的权重总和大于 1，层 $k+1$中的利普希茨常数之和就会大于前面各层。因此，每一层的平均利普希茨常数都会增加，通常会大幅增加，因为 $dim(k)$ 和 $w_{i,\ j}$ 可能很大。为了进一步简化，我们假设

$L(v_{k,\,i})$ 在前一层的所有节点中都是一致的，因此 $L(v_{k,\,i}) = L(v)$，并且还假设$|w_{i,\,j}|$ 有一个上界 w[11]。于是，我们得到：

$$L(v_{k+1,j}) \leq L(\phi)\dim(k)\bar{w}\overline{L(v)} \tag{2.37}$$

这表明，每个神经元的利普希茨常数上界会随着层数增加而呈指数增长，并会在单个层中添加更多神经元时呈线性增长[23]。这与关于全连接神经网络能够近似的函数复杂度的类似结果相吻合，例如 [6]。请注意，正如 [53] 所示，本节的发现可以推广到几乎任意的分层神经网络架构。

这为理解全连接神经网络架构的归纳偏置如何受层数和神经元数量影响，提供了一种十分直观的方式。根据层数和神经元数量[12]，，由此产生的神经网络往往具有较大或较小的利普希茨常数，对非常相似的输入产生的变化也相应较大或较小。

因此，较大的网络能够学习变化幅度大得多的函数，并且从信号理论的角度来看，能够捕捉数据中存在的更高频率。反之，较小的神经网络只能学习平稳得多、变化缓慢且频率较低的函数。如果数据充足，较大的网络因而更可取，因为较小的网络可能无法拟合所有训练样本。相反，如果数据较少，较小的网络更可取，因为每个数据点对神经网络输出的影响范围更广，能够外推到更多可能的取值。因此，通过调整神经元/层的数量，也可以改变神经网络中的归纳偏置，从而允许更高或更低的利普希茨常数。

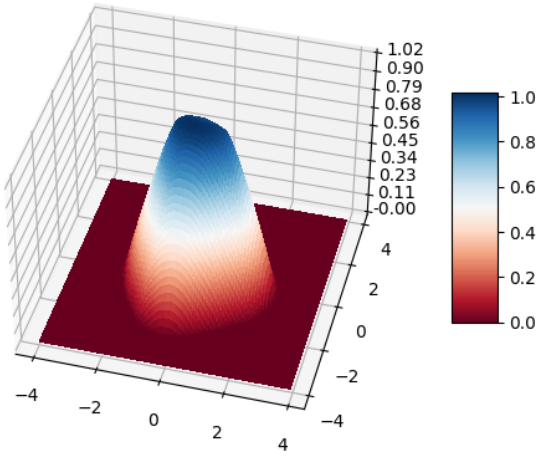
最后，这也为层本身的概念提供了依据。由于层可能使神经网络的利普希茨常数呈指数级增长，因此在构建强大的架构时，它们大有裨益：这类架构必须拟合大型数据集，同时只需线性增加神经网络的规模。

通过改变神经网络中神经元和层的数量，可以改变网络中的归纳偏置；根据所使用的层数，利普希茨常数可以更高，也可以更低。请注意，这不仅是理论结果，在实践中也会出现，如 [53] 开展的实验所示；本研究开展的实验也得出了相同结果。同样， [6] 在额外层的作用方面也取得了非常相似的结果，这进一步支持了这样一种观点：额外的层能够近似复杂度呈指数级增长的任务[13]。

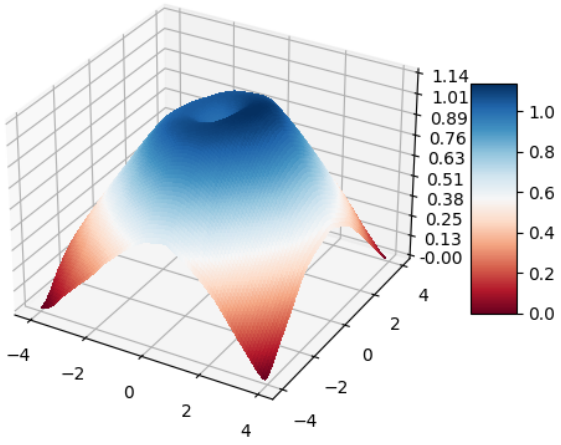
在本文余下部分中，我们将这些属性视为神经网络架构应具备的归纳偏置，只有如此，该架构才有用。相近点彼此相似这一观点适用于许多不同的任务，也是梯度下降和神经网络本身的核心。此外，神经网络归纳偏置的可调性也很重要。与其他机器学习算法相比，神经网络最大的优势之一或许是其极强的灵活性。无需改变神经网络的基本架构，就能轻松改变其归纳偏置。可调节的参数例如每层的神经元数量、

[11]这有两个合理的原因：首先，反向传播往往会在初始化附近（通常接近 0）得到局部极小值。其次，正则化通常会使 $w_{i\,j}$, 收缩。当然，如果神经网络中的权重非常大，即使是浅层网络也能具有很高的利普希茨性。不过，正如脚注 4 所指出的，出于一些原因，这种情况通常并不会发生。[12]

[13]有趣的是，这对于多种不同的“复杂度”定义都成立。正如 [6]，所示，使用 f^* 的拓扑复杂度时这一点成立；而正如 [53] 所示，使用基于利普希茨的复杂度定义时，这一点也成立。



(a) $\lambda = .001$



(b) $\lambda = .002$

图 2.7： 两个具有 4 层的神经网络，使用不同的 L2 损失权重值，在相同任务上进行了训练。 λ 的值越高，边权重就越低，因此利普希茨常数也越低。这些网络经过训练后，会将所示区域的所有角点映射为 0，将中点映射为 1，即 (0，0)。两者都很好地拟合了该任务，但采用了不同的方式。网络 a（ λ 的值较低）生成了一个平坦的表面，但在 (0，0) 周围有一个尖峰，其利普希茨常数非常高。网络 b 则选择了更平坦、更连续的版本。当然， b 的利普希茨常数要低得多。

层数以及正则化损失函数的选择。
另请注意，这并非全连接神经网络中所有归纳偏置的完整列表。例如，参数初始化是一种主动引入偏置的方法，并在神经网络如何学习方面发挥着巨大作用 [22]。此外，仅凭神经网络的利普希茨性很难对其进行完整描述。

然而，在本文余下部分中，我们专注于推导新的架构及其期望属性。本节介绍的隐式偏置是神经网络中一些最基本的隐式偏置，也可能是其成功的关键。因此，我们希望每种新架构都能解决全连接网络中的某个领域专用问题，同时保留以下偏置和属性，并满足神经网络始终需要具备的可训练性：

神经网络的归纳偏置

全连接神经网络是利普希茨连续的，且具有某个利普希茨常数 L。因此，

$$f(x+v) \in [f(x) - L||v||, f(x) + L||v||] \tag{2.38}$$

因此，在欧几里得意义上彼此接近的点，在神经网络下具有相似的函数值。利普希茨常数越大，可近似的函数就越多、越复杂。神经网络的利普希茨常数取决于其结构和权重。它会乘以 $O((w_{\max} * b_{\max})^l$，其中 $w_{\max}$ 是神经网络中的最大权重，$b_{\max}$ 是任意一层中神经元数量的最大值，l 是层数。w_{max} 会受到正则化技术和初始化的显著影响。然而，无法对 w_{max} 作出任何保证。

- “连续性”

  – 神经网络可表示的每个函数 f 都是连续的– 此外，f 是利普希茨连续的，其利普希茨常数为 $L(f)$– 因此，按照欧几里得度量彼此接近的点，在 f方面也相似；它们相似的程度由 f的利普希茨常数描述

- “可调性”

  – “邻近点”的相似程度，以及由此决定的利普希茨常数 $L(f)$，应当能够（至少间接地）进行调节– 为了良好地扩展到复杂任务，需要存在一种机制，使其能够随任务复杂度的增加呈指数级（或至少超线性）扩展。实现这一点最常见的方式是采用某种分层架构。

- “完备性”

  – 我们的神经网络架构应能够近似所有目标函数。

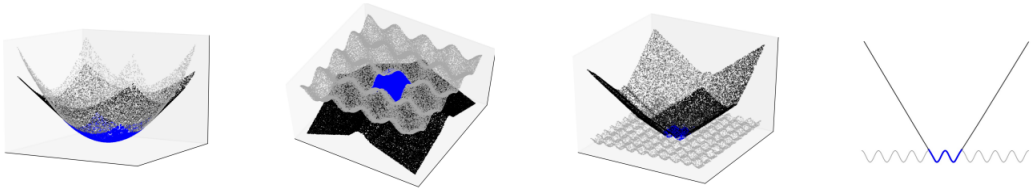


图 2.8： [61] 全连接神经网络无法对非线性函数进行外推的一些示例。神经网络的预测以黑色显示，实际函数以灰色显示，训练数据集以蓝色显示。请注意，第一个任务本质上是二次函数，因此在训练数据范围之外，网络的预测开始偏低。第二个任务是周期函数，神经网络完全无法对其进行近似，而是沿着一个周期训练数据的趋势线性延伸。后两个任务也有类似发现。

- “可训练性”
  - 作为基线，神经网络需要是可训练的，即需要参数化，并且关于其参数可微

2.5 前馈神经网络的问题

如前一节所述，前馈神经网络在函数逼近方面具有一些非常实用的特性：它们可以通过次线性的努力进行调整，以近似复杂程度各异的函数，并且假设彼此接近的点在待逼近函数方面也相似。在没有任何额外信息的情况下，这似乎是合理的。在大多数现实世界任务中，相似的输入至少在某种程度上会产生相似的结果[14]。然而，全连接神经网络存在一些问题，因此它们目前确实并非大多数关注任务中的最先进水平 [1]，人们更倾向于采用领域专用架构，例如图神经网络 [54]。

全连接神经网络的第一个问题在于其对连续性的假设。虽然我们在上一节中指出，对于许多领域而言，这是一项有用的性质，但确实存在其他领域，其中领域或陪域并非连续。神经网络本质上是针对某些 $\mathbb{R}^k \to \mathbb{R}^j$ 的函数，其中 k，j。神经网络常用于分类。在这种情况下，陪域不是欧几里得空间，而是由不同标签构成的集合。这与全连接神经网络的陪域不相符，也会给基于梯度下降的优化带来问题，因为这类优化要求梯度有定义。由于无法在标签集合上定义梯度，因此梯度下降无法使用。尽管如此，神经网络仍常被用于这些领域，人们会采用各种变通方法，但这些问题并未得到解决。因此，我们关注的不是离散域，而是具有额外结构的连续域。关于离散情境下神经网络的深入讨论，请参阅附录。

目前，我们考虑那些可表示为实值函数逼近任务的领域中的全连接神经网络。尽管全连接神经网络在这些场景中适用且表现出色，但它们也存在一些问题：即使应用于连续域，大多数任务中的全连接神经网络也无法很好地外推。这意味着，如果一个神经

[14]对于不属于这种情况的任务，不应采用神经网络，或者应选择不同的输入表示。

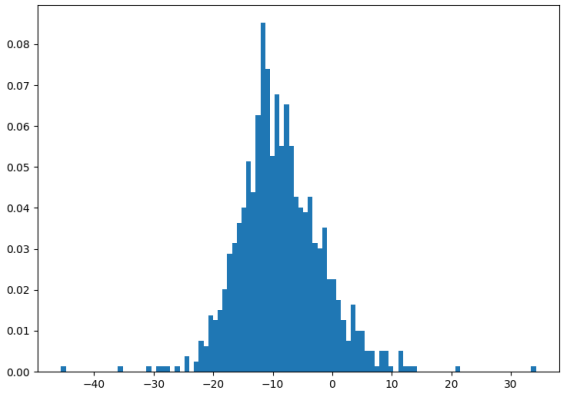


图 2.9： 直方图展示了神经网络线性外推的稳定性。对同一个简单示例问题，使用同一个神经网络进行 1000 次训练；在该问题中，给定 $f^*((0,0)) = 1$ 和 $f^*(\pm 4, \pm 4) = 0$，最终在远离所有训练数据点的点 (50,50) 处进行查询。请注意，$f(50,50)$ 的分布以 12.5 为中心，这正是线性外推所预期的值。所得曲线有些向右倾斜，这很可能是初始化偏差导致的。

网络从一组数据点 D 中学习，那么对于距离 D 中所有数据点都很远的点，它就无法很好地学习。如 [61] 中的理论分析以及许多其他实践研究所示，[4, 29],采用 ReLU 激活（即迄今最常见的激活函数）的全连接神经网络，在远离训练数据 D 时会很快变成线性，因此很难外推非线性函数 f^*。

这清晰地展示了神经网络在某个数据集 D上训练后通常会有怎样的表现。在数据点密集的区域，神经网络可以利用其可能很高的利普希茨性，非常精确地逼近这些点，从而得到复杂度很高的函数。在这些数据点之外，神经网络会继续沿线性趋势变化，并且相对较快地呈现出这种线性特征，如 [61] 所示。

具体来说，他们证明了：采用梯度下降训练、具有 2 层和 ReLU 激活函数的任何神经网络，只要没有数据点位于由 $x + hv$ 确定的直线附近，就会呈现这种行为；其中 x 和 v 是向量。

$$|\frac{||f(x) + hv||}{h} - \beta_v| < \epsilon \tag{2.39}$$

其中 $\epsilon \in O(\frac{1}{h})$。$\beta$ 是一个与 h 无关的线性系数。这意味着，在任何延伸到数据集所在区域之外的直线上，神经网络都会迅速趋近于线性行为。

不应将此视为全连接神经网络固有的弱点。如果缺乏各种领域知识，那么认为观察到的模式会线性延续，是一种非常合理的选择。然而，如果确实具备领域知识，就应该能够改进这种线性延续。这或许是几何深度学习最重要的理念之一：没有领域知识，通常就无法进行外推。

例如，考虑图 2.8 中呈现的最后一个问题。神经网络通过线性延续已观察到的模式来进行外推，而不是以周期函数的方式延续；对于这个问题，

周期函数的方式会更合适。因此，它对实际的 f^* 产生了严重的误估。如果事先知道 f^* 的周期性，并能强制神经网络以周期函数的方式延续已观察到的模式，就能更好地近似 f^*。

这一论述带来了两个核心挑战。首先，需要借助人类领域知识，识别出 f^* 的哪些属性可用于函数逼近，而无需实际了解 f^*。其次，需要构建一种神经网络架构，使其能够根据 f^* 的属性进行外推；这通常意味着让 f 模仿 f^* 的预期行为。

当然，要普遍做到这一点是一项极其艰巨的任务，因为神经网络应用的领域多种多样、数量众多。因此，对于 f^* 所假定的一些函数属性，可能很难在 f 中加以约束。几何深度学习因此聚焦于两种领域专用属性：它们既存在于许多不同的深度学习领域（因而具有普遍用途），又相对容易加以约束。这些属性是 f^* 对某些变换群（称为对称性）的不变性，以及将 f^* 分解为不同尺度上局部作用函数的可分性。

在接下来的章节中，我们将介绍这些性质，说明它们确实存在于许多不同的领域中，然后再通过一个示例展示如何在估计器 f 中强制实现这些性质。

神经网络的外推行为

尽管神经网络能够逼近任意函数，但使用基于梯度的优化技术训练的神经网络，会在输入数据之外沿着线性方向延续它们所学到的模式，从而进行外推。 [61] 的作者表明，使用梯度下降和平方误差损失训练的两层神经网络，在数据集之外会收敛到任意接近线性函数的结果：

$$|\frac{||f(x)+hv||}{h}-\beta_v|<\epsilon \tag{2.40}$$

其中 $\epsilon \in O\left(\frac{1}{h}\right)$。此外，对于层数多于两层的网络，我们的实验和其他经验结果也支持这一结论 [4]。参见图 2.8。

第 3 章

几何深度学习简介

既然我们已经为希望应用神经网络的一般设定确立了基线，也对那些明确并非 以领域知识为构建依据的全连接神经网络的行为有了一些了解，现在就可以开始探讨“优秀的领域特定神经网络应该是什么样的”这一问题。

正如我们之前所见，全连接神经网络具有一些非常好的性质，无论我们打算将它们用于哪个领域，这些性质几乎都同样有用。因此，如果我们想设计实用的特定领域神经网络，那么任何优秀的领域特定神经网络理应保留这些性质，因为它们看起来几乎不受具体情境影响，始终具有实用价值。

1. **可训练性**。 全连接神经网络是可训练的，这是将其用于基于梯度的优化所必需的，也是我们框架得以应用的必要条件。

2. **连续性**。 全连接神经网络假设 f^* 是连续的，而且是利普希茨连续的，其利普希茨常数不会任意大。因此，如果 x 和 y 彼此相对接近，那么 $f^*(x) \approx f^*(y)$也成立。这一性质对许多现实世界任务都很有用，因为通常在欧几里得意义下彼此接近的输入，在当前任务方面也相似。

3. **可调性**。 通过增加层数或采用不同的初始化和正则化方案，可以控制利普希茨常数，从而控制我们能够假设邻近点相似的程度。

4. **完备性**。 如果资源充足，全连接神经网络就能够以任意精度逼近每一个连续函数 $\mathbb{R}^n \to \mathbb{R}^m$ ，

然而，我们也观察到全连接神经网络存在一些问题，这些问题源于全连接神经网络缺乏内置的领域知识。我们观察到两个主要问题：首先，如果神经网络观察到某个 x 的 $f^*(x)$，就会假设 x 的邻近点 y 具有相似的值 $f^*(y)$。这种假设虽然合理，但在大多数领域中，借助领域知识，我们可以将学习扩展到更多点 y 。例如，如果将一幅图像镜像翻转，原始图像与镜像版本之间的欧几里得距离会

非常大。尽管如此，它们表示的是相同的对象，因此学习应当从一方扩展到另一方。其次，神经网络通过对观察到的模式进行线性延伸来外推。在缺乏领域知识时，这种做法是合理的；但如果我们对当前任务了解更多，它就会失效。例如，神经网络因此往往无法很好地泛化到周期性任务。

因此，我们必须找出希望纳入神经网络的领域属性。此时，我们转向几何深度学习。它指出，许多我们感兴趣的领域似乎具有两种共同属性；作者认为，领域特定神经网络应当遵循这些属性 [9]。

3.1 对称性与 $\mathbb{G}$-不变性

如前所述，全连接神经网络包含这样的归纳偏置：对于 f^*，彼此相近的点具有相似的值。对于许多任务而言，这是一种非常直观且有用的偏置。例如，如果任务是识别图片中的物体，那么即使将每个像素稍微调暗，物体的分类也只应发生很小变化。原始图片与调暗后的图片在欧几里得意义上彼此非常接近，因此，全连接神经网络会正确地假定它们所表示的物体也相似。

然而，尽管这种偏置在许多情况下都很有用，但在许多问题中还存在更强的领域专用结构，函数逼近时应当加以利用。
首先，我们来考虑固定尺寸图像的领域，其中图像 x 被形式化为一个函数 $x\colon\ P \to \mathbb{R}^3$，其中 $P = \{1, 2, ..., j\} \times \{1, 2, ..., k\}$ 是像素的集合，而 x 为每个像素指定其 RGB 值向量。或者，由于 x 的定义域是离散的，我们也可以将 x 的函数表写成矩阵形式，并根据上下文在这两种表示之间切换。

对于大多数计算机视觉任务，这些图像具有一些不应影响神经网络输出的对称性。例如，图像的线性平移

$$tx(p) = x(p + c) \tag{3.1}$$

其中 c 是某个常向量，不会影响图像本身。即使图像发生了固定幅度的平移（对于边界，假设添加或移除一些空白区域以允许图像移动，并假设空白区域不会影响预测），在大多数计算机视觉任务中，它仍然是同一幅图像。

因此，我们可以放心地假设，几乎所有目标函数 f^* 在线性平移下都保持不变：

$$f^*(tx) = f^*(x) \tag{3.2}$$

再举一个例子，考虑图 $G = (V, E, l)$，其中 $l\colon\ V \to \mathbb{R}^d$ 表示每个节点的标签。要将其输入神经网络，自然可以用其函数表表示为矩阵 $\bar{l} \in \mathbb{R}^{|V|, d}$[1]，并将边的集合 E 表示为邻接矩阵 $\bar{E} \in \mathbb{R}^{|V|, |V|}$。当然，这种表示方式高度依赖于节点的排序。如果改变节点的排序，矩阵 V 就会发生置换

$$\bar{l}' = \hat{l}P \tag{3.3}$$

[1]我们用 $\bar{x}$ 表示某个对象 x的基于矩阵的表示。

其中 P 是一个置换矩阵，它会交换与之相乘的矩阵的列。邻接矩阵也会发生类似的变换：

$$\bar{V}' = P^T \bar{V} P \tag{3.4}$$

ˆ ˆ 直观地说，这表示交换 l 中的行，并相应地改变 E ，不会破坏图的结构。对于几乎所有任务，节点的顺序都无关紧要，因为底层图在结构上保持不变（而且如果将其绘制出来，看起来也可能完全相同）。因此，可以放心地假设，几乎所有函数 f^* 在这类置换下都是不变的。

$$f^*(\bar{l}, \bar{E}) = f^*(\bar{l}P, P^T \bar{E} P) \tag{3.5}$$

这种数据中普遍存在的对称模式，常见于传统上会采用深度学习技术的领域，例如前文提到的计算机视觉和图处理。这种模式在许多类型的数据中自然出现；对于许多问题领域，都存在只改变对象表示形式而不改变对象本身的变换。例如，旋转后的图像本质上仍是同一幅图像，因此对于大多数计算机视觉任务而言是等价的。同样，在几何建模任务中处理三角网格时，无论模型对象位于三维空间的何处，或如何旋转[2]，它仍然是同一个对象。

此外，这也是使用深度学习所产生的技术难题的后果。要使一个函数能够通过基于梯度的优化进行训练，它必须是可微的，因此其领域和陪域都必须是实值的。因此，神经网络的输入必须是一个矩阵，其输出也必须如此。

考虑由向量构成的有限集领域 $\Omega = \{x \mid x \subset R^n, \text{x finite}\}$。将这样的集合转换为矩阵的直观方法，就是简单地将集合中的向量“堆叠”$x = \{x_1, ..., x_n\}$ 成一个矩阵：

$$\bar{x} = (x_1 \;\; x_2 \;\; ... \;\; x_n) \tag{3.6}$$

这种表示忠实地包含了原始集合 x的所有信息，但并不唯一。要得到这种表示，必须任意选择 x 中元素的一个排列，而集合本身并不具备这种结构。如果选择不同的排列，就会得到上例的一个置换 π 作为表示：

$$\pi(\bar{x}) = \left(x_{\pi(1)} \;\; x_{\pi(2)} \;\; ... \;\; x_{\pi(n)}\right) \tag{3.7}$$

由于两者都是同一个集合的表示，我们知道

$$f^*(\bar{x}) = f^*(\pi(\bar{x})) \tag{3.8}$$

在这种情况下，对称性并非存在于数据本身，而是源于神经网络要求输入具有顺序，因此在数据上强加了额外的结构；这种结构实际上并不存在于数据中，因此不应影响我们的估计。尽管如此，总体思路相同：存在某种形式的变换，不应影响估计任务。

[2]一般而言，这可以推广到等距变换，即网格之间保持所有角度和面积不变的变换。网格的平移和旋转就是这类变换的例子。因此，这些领域的深度学习模型非常关注对等距变换的不变性 [10]。

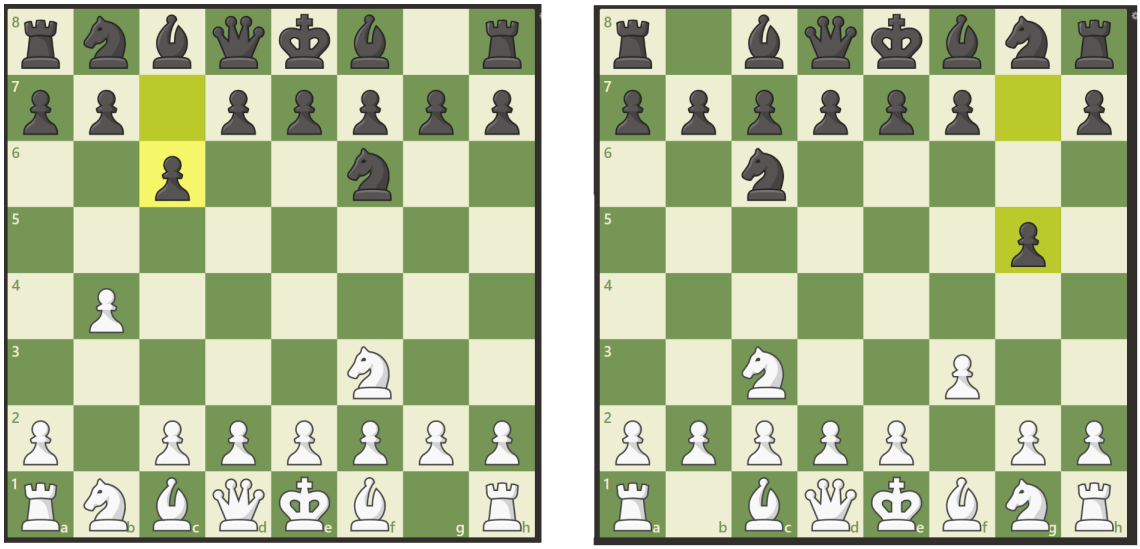


图3.1： 领域中的对称性示例。在国际象棋中，我们的目标是逼近这样一个函数：它将国际象棋棋盘状态映射为当前局面下的最优走法。 f^* 当然是未知的，但根据国际象棋领域知识，我们很容易看出，最左侧图像中黑方的最优走法，就是最右侧图像中白方的最优走法，因为国际象棋关于颜色是对称的（轮到哪一方走棋这一点除外）。

在所有这些例子中，仅凭对领域的基本了解[3]，我们就能立即知道，存在一些变换 T，目标函数 f^* 对这些变换是不变的。这些变换称为对称性，非常有用，因为它们通常是数据本身固有的，因此即使不了解 f^*，也能知道它们。

如果使用全连接神经网络，最终得到的估计器可能不具备这些变换下的不变性，也就是说，存在某个 $x \in \Omega$、 $t \in T$，使得 $f(x) \neq f(tx)$。回想一下全连接神经网络的外推偏差，这种情况很可能发生，因为线性外推在大多数情况下会违背这一条件。

因此，我们希望在架构中纳入的第一种额外归纳偏置，是对一组称为对称性的变换保持不变。这些变换通常遵循群结构，且通常构成置换群。因此，领域特定神经网络应遵循以下额外性质：

$\mathbb{G}$-不变性。如果存在作用于 Ω 中元素的某个变换群 $\mathbb{G}$，并且假设 f^* 在这些变换下保持不变，那么 f 必须是 $\mathbb{G}$-不变的，即对于所有 $g \in \mathbb{G}$， $f(x) = f(gx)$。

有关群的更多信息，尤其是如何将我们感兴趣的变换表示为置换群，请参阅附录。

3.2 任务分离与局部性

几何深度学习的第二个核心思想源于这样一种观察：许多深度学习任务可以拆分为许多不同的、处于不同分辨率尺度上的任务。这一思想的依据是，在许多领域中都存在某种局部性的概念，使得局部

[3]这一点非常重要，因为神经网络架构通常是为某个领域而非特定任务设计的。因此，关于该领域的先验知识是最值得利用的知识。

$\mathbb{G}$-不变性

几何深度学习提出，如果 f^* 被假定为 $\mathbb{G}$-不变，那么领域特定神经网络应当是 $\mathbb{G}$-不变的：如果存在一个由变换 $\Omega \to \Omega$ 构成的群 $\mathbb{G}$，使得我们可以假定对于所有 $g \in \mathbb{G}$，$f^*(x) = f^*(gx)$，那么对于任意领域特定神经网络 f，对于所有 g，也应满足 $f(x) = f(gx)$。

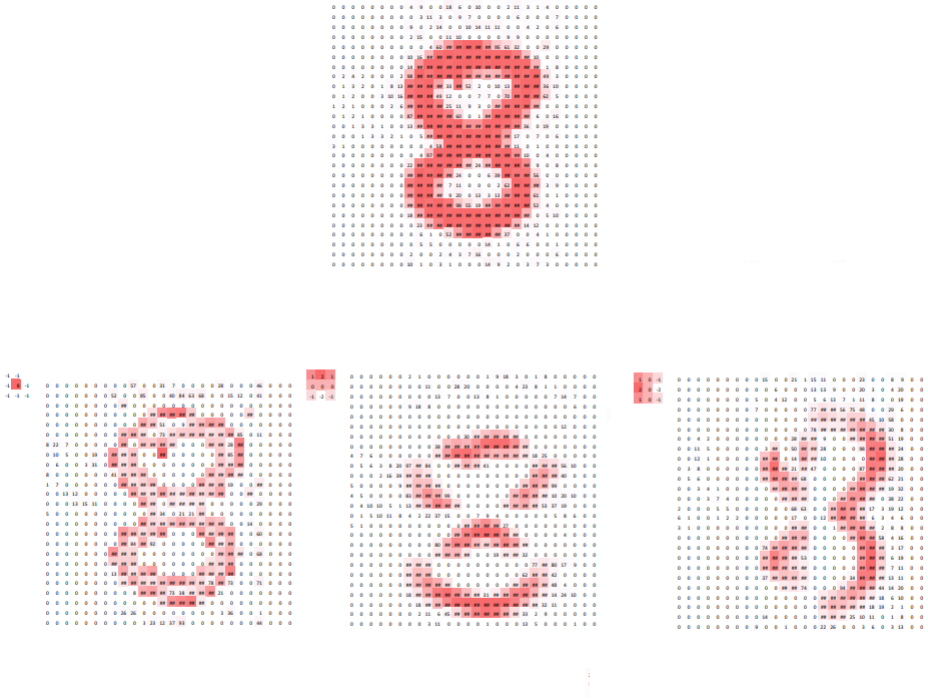


图 3.2： 卷积神经网络作用于一张“8”的图像时的第一层（为清晰起见，省略了非线性）。卷积神经网络从输入中生成三张特征图：左侧的一张大致捕捉边界，第二张大致捕捉水平线，第三张捕捉垂直线。这与标准图像识别中的方法相符，也展示了卷积神经网络的构建方式能够轻松近似标准图像识别实践。图片来自 [62]。

在不丢失太多全局信息的情况下，可以对结构进行概括。卷积神经网络在计算机视觉领域中的一个著名应用就是如此 [37]。在计算机视觉中，通常并非单个像素决定图像所表示的物体，而是该物体的一组典型特征，例如人脸的眼睛和嘴唇，或汽车的车轮。所有这些特征都由欧几里得意义上彼此相近的像素构成。

类似地，在许多基于图的任务中，图都存在一些局部子结构，可以对其进行概括，同时不会严重丢失整个图的信息。因此，可以概括图的局部子结构，从而降低近似问题的维数，同时不丢失正确近似 f^* 所需的信息。

This 由此引出了神经网络中的尺度分离这一概念。直观来说，有些任 ks

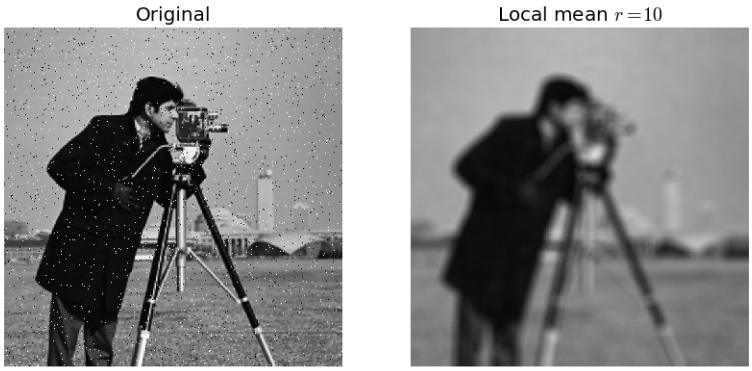


图 3.3： 均值滤波器的应用示例。此处，均值滤波器直接应用于原始图像x，而非函数 $\phi(x)$ 的输出；在神经网络中通常会采用后者，这里这样处理是为了更清楚地展示其工作原理。此处的补丁大小为 10，也就是说，每一步都会对 10*10 个像素的邻域进行聚合。图像来自 [59]。

务需要精细的解决方案。要对图像中的小型局部模式进行分类，需要用到图像块中的每一个像素。然而，要将这些小模式组合成更大的模式，就不需要逐像素的信息；如果已经找到了较小的模式，便可以依靠较粗的分辨率。例如，在对汽车图像进行分类时，第一步可能是寻找边和曲线等非常小的模式。这一步可以直接在图像像素上轻松完成。接下来，可能需要寻找车轮（由曲线构成）和车牌（由曲线和直线构成）。然后，只需询问“这张图像中是否有四个车轮，位于带有车牌的车身下方？”，便能轻松完成“这张图像是否呈现一辆汽车？”这一最终分类任务。这样就将图像分类任务（其本身相当困难）拆解成了若干个简单得多、模块化的任务。

[9] 的作者将其形式化表述如下。设 $\Omega = \{x \mid x \in H \to \mathbb{R}, H \in S\}$ 为一个函数空间，这些函数将实数值赋给属于某个网格空间集合 S 的网格空间 H 中的元素。以计算机视觉任务为例，此处的网格空间 H 确实是大小不一的规则二维网格，

$$S = \{H \mid H = \{0, 1, ..., i\} \times \{0, 1, ..., k\}\} \tag{3.9}$$

用于表示图像中的像素。函数 $x \in \Omega : H \to \mathbb{R}$ 为 H 中的每个像素赋予其灰度值[4]。即使在与计算机视觉无关的领域中，我们也将用“像素”一词来指代 H 中的元素，但请注意，此时“像素”的具体实现可能与实际像素大相径庭。

在图分类中，网格空间 H 是相应图的节点集，而函数

[4]要将其扩展到彩色图像，只需为每个像素引入三个通道。

$x \in \Omega$ 为每个节点指定一个实数值标签。为了也在此框架中对边进行建模，我们选择 $H = V \cup V \times V$，其中 x 为每个 $v \in V$ 指定标签，并且 $x((v_1, v_2)) = 1$ 当且仅当 v_1 和 v_2 之间存在一条边时。
某些 $x : H \to \mathbb{R}$ 的分辨率由 H的基数给出。例如，在图像中，这意味着像素更多的图像分辨率更高，而像素较少的图像分辨率较低。在图中，节点更多意味着分辨率更高，而节点较少则意味着分辨率较低。

我们将粗化函数 $\rho : \Omega \to \Omega$ 定义为一种降低其输入分辨率的函数，即若 $\rho(x) = y$，则 x 的网格空间是 y的网格空间的超集。在图像语境中，这对应于实际的粗化：新图像的像素更少，因此更粗糙，也必然会丢失原始图像的信息。虽然这个定义并未明确指出这一点，因为若不对 ρ 施加过多限制，就很难将其形式化，但我们也假设 $\rho(x)$尽可能接近原始 x 。计算机视觉中一个简单的例子是所谓的均值池化。均值池化使用一个像素来表示一个像素邻域，并以这些像素的均值作为其值

$$\rho(x)(p) = \frac{1}{|N(d)|} \sum_{d' \in N(d)} x(d') \tag{3.10}$$

接下来，我们将局部运算函数 $\psi : \Omega \to \Omega$ 定义为一种仅在每个像素的某个邻域内进行运算的函数。具体而言，我们假设 ψ的输入和输出都作用于同一网格空间 H，且每个像素的值仅受其邻域影响：

$$\psi(x)(p) = \psi'\left(x(p_1), ..., x(p_w)\right) \tag{3.11}$$

其中， p的邻域定义为： $N(p) = \{x(p_1)..... x(p_w)\}$。
因此，我们定义了两种不同类型的函数：一种用于降低图像的分辨率，另一种则在图像的小图像块上进行局部运算。将它们串联起来，即可将一个函数分解为许多在不同尺度上执行的较小任务。

我们假设 f^* 遵循这样的结构：

$$f^* = \psi_{i+1} \circ \rho_i \circ \psi_i \circ ... \circ \rho_1 \circ \phi_1 \tag{3.12}$$

这种结构与本节开头所阐述的直觉非常相似。首先，函数 ψ_1 作用于像素的局部邻域。然后，结果被粗化到更粗的域 H_2 中，另一个函数 ψ_2 则作用于H_2中像素的局部邻域。由于 H_2 更粗，邻域的含义也随之扩展。例如，如果使用均值池化来聚合四个相邻像素的邻域，就会得到一幅分辨率仅为原图四分之一的图像。现在，新图像中的每个点代表原图中的 4 个点。如果 ψ_2 现在作用于点 p的四个直接邻居，那么这些点中的每一个都代表原图中的 4 个点。因此， ψ_2 取决于原图中 16 个点构成的邻域；而在粗化之前，它只取决于由 4 个像素构成的邻域。因此，随着不断粗化，处理的原图区域会越来越大。然而，在更粗的尺度上，无法观察到像素级尺度上的直接交互。例如，ψ_2 取决于原图中四组各含四个点的均值。由于粗化过程中信息已丢失， ψ_2 不可能取决于邻域内每个像素的确切位置。构成一个均值的这些像素各自的具体位置无法

确定，只能确定它们属于给定的邻域。
这就是尺度分离假设的核心：细粒度的交互（例如两个像素之间的交互）取决于像素的确切位置，并且只发生在直接相邻的节点之间；而图像中相距较远的部分只在较大尺度上发生交互。距离越远，交互的粒度就越粗，对像素实际位置的依赖也越小。这一思想同样适用于图像识别以外的其他领域。尺度分离假设比不变性假设具体得多。首先，它只适用于具有此处所述 Ω 结构的领域，即数据由映射 $H \to R$ 构成；它不仅高度依赖于 f^* 能以此处所述的方式分离，还要求人们知道相对于 f^* 的正确邻域概念。尽管如此，它仍适用于许多最常见的深度学习领域，例如计算机视觉（其中邻域概念很简单）；而集合等其他领域则并未真正包含这种结构[9, 64]。

因此，我们将其拆分为两个不同且要求较低的假设，并将以下归纳偏置视为理想特性

局部性。如果 Ω 中存在有意义的"局部性"概念，并且合理地认为对 f^* 产生重大影响的直接交互具有局部性，那么估计器 f 就应由局部运算函数构成。

and

任务分离。如果合理地认为 f^* 可以分解为多个不同的简单函数，那么 f 就应遵循具有类似分离结构的架构。

这两个概念在深度学习中都至关重要。例如，表现出色的卷积神经网络 [37] 就高度依赖这些概念。遵循这些归纳偏置的架构具有一大优势：它们能够将一个维度极高的近似任务转化为维度较低的任务。这一点非常有用，但目前对于这些归纳偏置对实际可近似的函数空间所产生影响的理论认识似乎仍然有限，因此本文仅对其作简要介绍。

然而，有趣的是，至少在某些领域中，多尺度结构与某种形式的近似形变不变性相关，正如 [9] 中所述。对于卷积神经网络， [8] 的作者利用上述思路，成功证明卷积神经网络在微分同胚作用下相对稳定。

具体而言，他们的工作涉及卷积神经网络的实值扩展，即在连续像素空间上处理图像的卷积神经网络，使得 $H \subseteq \mathbb{R}^2$。他们考虑位移映射 $\tau\colon \mathbb{R}^2 \to \mathbb{R}^2$，它会移动图像中的像素，其中 $\tau(p)$ 表示像素 p 移动的距离。这会对图像产生如下变换：

$$\tau(x)(p) = x(p - \tau(p)) \tag{3.13}$$

对于这些可微位移映射，他们证明，对于所有卷积神经网络 f 以及所有微分同胚τ，均有

$$||f(\tau x) - f(x)|| \leq (C_1 \sup_{p \in H} ||\nabla \tau(p)|| + C_2 \sup_{p \in H} ||\tau(p)||)||x|| \tag{3.14}$$

直观地说，这个界限由两个独立部分组成：一部分取决于位移本身的大小，$\sup_{p \in H} ||\tau(p)||$；另一部分取决于位移的梯度，

$\sup_{p\in H}||\nabla\tau(p)||$。后者在此尤为重要，因为它涉及形变，即超越单纯平移的非线性变换。正如 [8] 的作者所示，相应因子 C_1 由因子 $O(k^{l+1})$决定，其中 l 是网络中的层数， k 是所用的池化尺寸，即每一步池化所聚合的像素数量。量 $O(k^{l+1})$决定了最后几层神经元的感受野大小，也就是说，它决定了最终分类层之前的最后一层中，每个神经元会受到原始图像中多少像素的影响。因此，层数较少且池化滤波器非常小的卷积神经网络，对小幅形变 τ近似不变。

不过请注意，在实践中，这种不变性对现代卷积神经网络的适用程度很低，因为这类网络通常包含大量的层。如 [51], 所示，卷积神经网络的近似形变不变性极其复杂，目前尚未得到充分理解。

任务分离与局部性

几何深度学习提出，特定领域神经网络应遵循 f^*的尺度分离。这意味着，任何优秀的特定领域神经网络都应遵循 f^*的任务分离与局部性。任务分离意味着，如果假设 f^*存在某种组合结构，那么优秀的特定领域神经网络 f 就应遵循这一结构。局部性意味着，如果 Ω 上存在局部性的概念，也就是说，如果整个输入 x中存在局部子结构，那么这些子结构就应与特定领域神经网络紧密连接。

3.3 领域特定神经网络的期望

据此，我们可以将几何深度学习中的领域专用理念，即任务分离、局部性和 $\mathbb{G}$-不变性，加入我们认为全连接神经网络应具备的属性中。由此，我们得到一组期望属性，即领域特定神经网络应具备的一些属性。这组属性结合了全连接神经网络的有用属性，以及几何深度学习提出的领域专用属性。这些期望属性如下：

1. **可训练性**。 函数需要是可训练的，即用参数向量 θ 进行参数化，并且相对于这些参数可微。

2. **连续性**。 我们假设 f^* 是连续的，此外还是利普希茨连续的，且其利普希茨常数不会大得离谱。因此，如果 x 和 y 彼此足够接近，我们假设 $f^*(x) \approx f^*(y)$ 成立。

3. **可调性**。 由于某个领域中的不同问题对利普希茨连续性的强度有不同要求，我们希望这种属性能够以某种方式进行调节。尤其重要的是，应当存在某种方法，能够以指数级方式提高最大利普希茨常数，从而以亚线性工作量适应高度复杂的任务。

4. **完备性**。 如果资源充足，该架构类别应能够以任意精度近似所有不违反其他约束的函数，即 $rep(f) = S$其中 S是 f^*所有合理候选项的集合。

5. $\mathbb{G}$**不变性**。如果存在某个作用于 Ω 的群 $\mathbb{G}$，并且假设 f^* 对该群作用具有不变性，那么 f 必须具有 $\mathbb{G}$ 不变性，即对于所有 $g \in \mathbb{G}$， $f(x) = f(gx)$。

6. **局部性**。如果 Ω 中存在有意义的“局部性”概念，并且合理的假设是：对 f^* 有重大影响的直接交互都是局部的，那么估计器 f 应由局部运算函数构成。

7. **任务分离**。如果合理地假设 f^* 可以分解为多个更简单的函数，那么 f 应遵循类似分解方式的结构。

运用几何深度学习的思想，我们可以这样回答“好的领域特定神经网络是什么样的？”这个问题。现在，我们将开始尝试回答：“如何构建这样的神经网络？”为此，我们首先将从这组期望条件推导出 [64] 。起初，我们会以一种非结构化的方式进行推导，然后以此为基础阐述几何深度学习蓝图 [9] ，因为我们认为，这是一个能非常直观地体现该蓝图思想的应用领域。

3.4 DeepSets：作为领域特定神经网络的 DeepSets 与 GDL蓝图

集合是最基础、最重要的数学对象之一，因此，拥有适用于处理集合的神经网络架构至关重要。如前文所述，神经网络只能接收一定数量的实数作为输入，并且基线假设是这些输入具有顺序。因此，全连接神经网络实际上依赖于输入的顺序；而集合中的元素天然没有顺序，这在处理集合时会带来问题。

我们考虑实数的有限集，这是神经网络最基本的应用场景。请注意，本节介绍的所有内容不仅适用于集合，也适用于多重集，即其中的值可以重复出现的集合。尽管如此，本节仍以集合为应用场景，介绍专为集合设计的深度集合 [64]。我们考虑 $\Omega = \{x \mid x \subset \mathbb{R}, x \text{ 有限}\}$。如第 3.3 节所述，我们将集合 $x = \{x_1, ..., x_n\} \in \Omega$ 表示为一个向量

$$\bar{x} = (x_1 \;\; x_2 \;\; ... \;\; x_n) \tag{3.15}$$

这是必要的，因为神经网络需要有序的实数值作为输入。当然，这种排序是任意的，因此我们需要定义一个可训练函数（即神经网络），并从架构上保证该函数对这种排序的置换保持不变。

我们将从特定领域神经网络设计要求的性质 4 开始，也就是前面提到的集合元素对置换的不变性。我们希望构造 f，使节点的顺序无关紧要。一个简单的做法是将所有数值 x_i 相加。若定义

$$f(x) = g(\sum_{i=1}^{n} x_i) \tag{3.16}$$

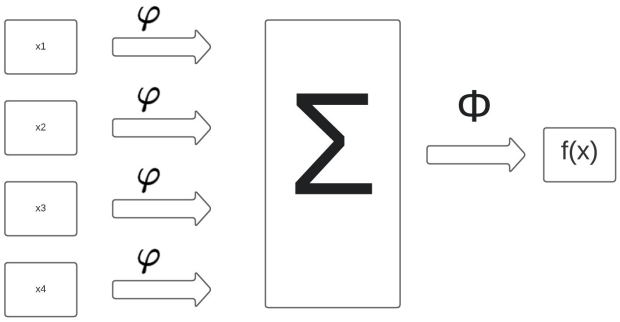


图 3.4： 深度集合架构作用于一个包含四个元素的集合 $x = \{x_1, x_2, x_3, x_4\}$ 的结构。对每个元素应用 ϕ，然后将结果求和，以得到最终结果 $f(x)$。请注意，如果在此架构中交换 x_2 和 x_3，最终结果仍将保持不变，因为

函数 f 可确保对输入的置换保持不变。如果 g 是可训练、连续且可调的，我们也就立即满足了所需的性质 1 和 2。当然，这种方法也存在一些显而易见的问题。使用 f 的这种表达方式，具有相同总和的输入值无法彼此区分。例如，$f(\{4,3\}) = g(7) = f(\{2,5\})$，这违反了性质 6。因此，函数需要更加复杂。

我们可以利用柯尔莫哥洛夫-阿诺德表示定理[36]继续推导。根据柯尔莫哥洛夫-阿诺德表示定理，每个连续多元函数 f^* 都可以表示为求和与一元函数的复合：

$$f^*(x_1, x_2, ..., x_n) = \sum_{q=0}^{2n} \Phi_q(\sum_{p=1}^{n} \phi_{q,p}(x_p)) \tag{3.17}$$

请注意这一构造非常特别：存在若干一元函数 ϕ，将它们应用于单个输入 x_p，然后求和，再传递给 Φ。

这至少在一定程度上类似于公式 3.16，而且还可以进一步简化。上述内容是任意函数的柯尔莫哥洛夫-阿诺德表示；但我们只对置换不变函数感兴趣。对于这类函数，ϕ 可以相同，因为所有 x_i 的行为方式都相同，因此可以省略外层求和 [64]：

$$f^*(x_1, x_2, ..., x_n) = \Phi(\sum_{p=1}^{n} \phi(x_p)) \tag{3.18}$$

其中 $\Phi : \mathbb{R}^n \to C$ 和 $\phi : \mathbb{R} \to \mathbb{R}^n$ 都是连续的[5]。与公式 3.16 的区别在于，x_i 不会立即求和，从而避免丢失信息，而是预先映射到一个n维向量空间中。直观地说，向量 $\sum_{p=1}^{n} \phi(x_p)$ 包含的信息量与原始值一样多，因此，由 $d(x) = \sum_{p=1}^{n} \phi(x_p)$ 诱导的映射可以是连续且可逆的（置换除外）。如果选择 $\Phi = f^* \circ d^{-1'}$，就可以用这种方式表示每个连续的置换不变函数 f^*。

因此，我们以这种方式构建网络架构：

$$f(x_1, x_2, ..., x_n) = \Phi(\sum_{i=1}^{n} \phi(x_p)) \tag{3.19}$$

[5]这种表示方式也可以有更简单的版本，但连续性对神经网络至关重要。

这样我们就能确保满足完备性和 $\mathbb{G}$-不变性。当然，我们必须为 Φ 和 ϕ 找到合适的函数，以确保最终函数具有连续性、可训练性和可调性，并且我们确实能够近似 Φ 和 ϕ，从而使每个函数 f^* 都可被近似。幸运的是，这相对容易。 Φ 和 ϕ 都是任意函数，我们对它们没有任何先验知识。因此，我们可以对Φ 和 ϕ都使用简单的全连接神经网络。这样，全连接神经网络所有值得保留的性质都得以维持。可训练性、可调性和连续性都通过复合与求和得以维持。同样，完备性和 $\mathbb{G}$-不变性也得以维持：前者如 [64] 所示，后者则可直接由构造得出。

接下来需要检验局部性和任务分离。由于集合的本质，所有元素的行为都是对称的。因此，要么所有元素彼此局部相关，要么每个元素只与自身局部相关。不难看出，这种表述采用了尺度分离，因此同时满足任务分离与局部性。 ϕ 作用于单个元素，而 Φ 作用于所有元素。

深度集合之所以有趣，原因有二：首先，集合需要处理时，深度集合本身就很有价值，可作为其他神经网络的构建模块。其次，它们有助于理解如何构建满足所有期望的几何深度学习属性的神经网络架构。

Deep Sets

深度集合是一种遵循几何深度学习理念、用于处理（多重）集合的领域特定神经网络架构。（多重）集合包含多个无序元素。深度集合是可训练函数 $f: \mathbb{R}^i \to \mathbb{R}^o$，对其输入的顺序保持不变。深度集合 f 可表示为

$$f(x_1, x_2, ..., x_n) = \Phi(\sum_{p=1}^{n} \phi(x_p)) \tag{3.20}$$

其中， Φ 和 ϕ 是全连接神经网络。Deep Sets 是所有对输入置换不变的连续函数 $\mathbb{R}^n \to \mathbb{R}^k$ 的通用逼近器。

3.4.1 几何深度学习蓝图

在了解了如何从特定领域神经网络设计要求推导出深度集合，以及它们如何满足这些要求之后，我们想进一步探讨一种更具结构化的方法。为此，几何深度学习提出了几何深度学习蓝图；我们将在此通过深度集合的例子来阐释这一蓝图。

领域特定神经网络设计要求中的前三项属性，即连续性、可训练性和可调性，通过在需要替换任意未知函数的地方加入全连接神经网络，相对容易实现。同样，只需按照所提出的方案构建网络，就能轻松实现局部性和任务分离

结构。
更难实现的是 $\mathbb{G}$-不变性和完备性。从深度集合的示例可以看出，实现不变性本身通常并不困难。对于有限群 $\mathbb{G}$，总可以通过对 x[6]的轨道取平均来实现不变性

$$f(x) = f'(\sum_{g\in\mathbb{G}} gx) \tag{3.21}$$

由于轨道构成一个等价关系，很容易验证，这种表述确实对变换 $g \in \mathbb{G}$ 不变。这种形式保证能够产生对 $\mathbb{G}$ 不变的函数。现在，人们可能会认为，只需应用上述构造，并为 f' 选择合适的函数逼近器（例如全连接神经网络），创建一个对 $\mathbb{G}$ 不变的神经网络就轻而易举了。

然而，实际情况并没有那么简单。以 DeepSets 为例，此时 $\mathbb{G}$ 不变意味着对置换不变。因此，如果我们有 $x = (x_1, x_2, ..., x_n)$，那么这个求和就变成：

$$f(x) = f'(\sum_{\pi\in\mathbb{G}} \pi(x)) \tag{3.22}$$

对 x 的所有可能置换求和，其结果是一个向量。考虑 $a = \sum_{\pi\in\mathbb{G}} \pi(x)$，即 f' 的输入。对于它的第一个分量，有：

$$a_1 = x_1 + \pi_1(x_1) + \pi_2(x_2)... \tag{3.23}$$

现在，考虑这样一个事实：存在 $n!$ 个置换 π，可将 x_1 映射到 x_1；也存在 $n!$ 个置换，可将 x_1 映射到 x_2，依此类推。因此，这就变成了：

$$a_1 = n!x_1 + n!x_2 + ... + n!x_n = n! * \bar{x} \tag{3.24}$$

$a_2, ..., a_n$ 也是如此，因此 f' 只取决于 x的平均值。这不仅适用于 $\mathbb{G}$ 是对称群的情况，也同样适用于它的所有子群。当然，这样的估计器不可能是完备的，例如，它无法区分输入 (1，2) 和 (0，3)，尽管我们的假设并未要求它必须做到这一点。因此，这违背了我们所期望的完备性。由此，我们需要先对这种聚合应用某个非线性函数。如果我们写成

$$f(x) = f'(\sum_{g\in\mathbb{G}} gf''(x)) \tag{3.25}$$

其中 f'' 是一个精心选取的连续非线性函数 $\Omega \to \Omega$，那么，通过对柯尔莫哥洛夫表示定理进行改造，我们实际上可以表示所有期望的函数，前提是选取了合适的 f'。然而，同样由于柯尔莫哥洛夫表示定理，这种表示也允许 $f(x)$ 不是 $\mathbb{G}$-不变的，从而违背了初衷。为避免这一点，我们需要对 f''加以限制。

考虑 $f''(x) = (x_1^2, x_2)$ 的例子。于是，$f''((2, 1)) = (4, 1)$，且 $f''((1, 2) = (1, 2)$。那么，对于 (2，1)：

$$\sum_{g\in\mathbb{G}} gf''(x)) = (4, 1) + (1, 4) = (5, 5) \tag{3.26}$$

而对于 $\pi(2, 1) = (1, 2)$：

$$\sum_{g\in\mathbb{G}} gf''(x)) = (2, 1) + (1, 2) = (3, 3) \tag{3.27}$$

[6]有关群和轨道的更多信息，请参阅附录。

因此， f' 无法保证对置换保持不变。直观原因在于， f'' 可以根据输入的顺序对其进行编码，因此也会受到置换的影响。不过，我们仍然需要非线性的 f'' 来确保完备性。

因此， f'' 需要是一个非$\mathbb{G}$ 不变函数，它虽然是非线性的，但也不能受到群作用 $g \in \mathbb{G}$ 的显著影响，否则最终函数将不再是 $\mathbb{G}$不变的。因此，嵌入需要与 $\mathbb{G}$正交，即:

$$d(gx) = gd(x) \tag{3.28}$$

也就是说， d 的结果不受 g 影响，并保持 g 不变。这一性质称为$\mathbb{G}$等变性。例如，在深度集合中， d 由函数 ϕ给出。 ϕ分别作用于每个节点。显然，将 ϕ 应用于堆叠的 x_i向量，会得到一个由堆叠的ϕ组成的向量（x_i）。接着可以看到，先应用 ϕ 再对行进行置换，等价于先对行进行置换再应用 ϕ，从而得到前文所述的 $\mathbb{G}$不变性。这就形成了确保 $\mathbb{G}$不变性的几何深度学习蓝图。

根据几何深度学习蓝图，几何深度学习模型应构建为:

$$f = \Phi \circ \phi_n \circ \phi_{n-1} .... \circ \phi_1 \tag{3.29}$$

其中所有 ϕ_i 都是 $\mathbb{G}$等变的，而 Φ 是 $\mathbb{G}$ 不变的。通过增加 ϕ_i的数量，这种方法能够轻松地堆叠层，从而实现可调性，并且往往能够达到完备性。同样，由于该网络采用组合式设计，单个函数的连续性和可训练性可以扩展到它们的复合。因此，若将 Φ 和所有 ϕ_i 限制为可训练的利普希茨连续函数，则 f也保留这些性质。此外，我们需要确保这种构造能够近似任意函数。为此，ϕ_i 不丢失原始输入中的相关信息至关重要。因此，我们要求每个 ϕ_i 在给定训练过程中获得的正确权重时，都可以是一个双射函数 $\Omega \to \Omega$。

关于 $\mathbb{G}$不变性，一个简单的归纳论证表明，以这种方式构造的任何 f 都是 $\mathbb{G}$不变的:

$$\begin{aligned}
f(gx) &= (\Phi \circ \phi_n \circ \phi_{n-1} .... \circ \phi_1)(gx) \\
&= (\Phi \circ \phi_n \circ \phi_{n-1} .... \circ \phi_2)(\phi_1(gx)) \\
&= (\Phi \circ \phi_n \circ \phi_{n-1} .... \circ \phi_2)(g\phi_1(x)) \\
&... \\
&= (\Phi(g(\phi_n \circ \phi_{n-1} .... \circ \phi_2 \circ \phi_1)(x)) \\
&= (\Phi \circ \phi_n \circ \phi_{n-1} .... \circ \phi_2 \circ \phi_1)(x) \\
&= f(x)
\end{aligned}$$

其中用到了 ϕ_i 的 $\mathbb{G}$等变性以及 Φ 的 $\mathbb{G}$不变性。很容易验证，深度集合遵循这一通用蓝图。

综合来看，几何深度学习蓝图将创建一个$\mathbb{G}$-不变神经网络的问题简化为定义两个函数

1. 特征表示函数 $\phi: \Omega \to \Omega$。该函数应当可训练、利普希茨连续且 $\mathbb{G}$等变。

2. 分类函数 $\Phi : \mathbb{R}^k \to C$，用于执行最终分类。该函数应当可训练、利普希茨连续且 $\mathbb{G}$不变。它可以构造为 $f(x) = f'(\sum_{g\in\mathbb{G}} gx)$，但并非必须如此。

这一蓝图由 [9]的作者提出，旨在构建满足领域特定深度学习属性（主要是 $\mathbb{G}$不变性）的神经网络。由于局部性和任务分离涉及映照 f^*的假定结构，因此通常很容易看出应如何实现它们。

然而，对于其余领域专用神经网络设计要求，这一蓝图是否必然能够满足这些要求并不明确。例如，无法保证这一蓝图必然能构建出完备架构，确实可以逼近所有目标函数。

3.5 几何深度学习总结

本章借助几何深度学习的思想，尝试回答“优秀的领域专用神经网络是什么样的？”以及“我们该如何构建它们？”这两个问题。
针对第一个问题，我们列出了一系列我们认为领域专用神经网络应具备的属性。这份清单包含一些全连接神经网络的属性，它们几乎 不受上下文影响，依然有用。这些属性包括可训练性（确保可以使用基于梯度的优化）、完备性（能够以任意精度逼近任意目标函数），以及估计器的利普希茨性，这确保欧几里得邻域中的点会被我们的估计器以相似方式处理。在这些普遍有用的属性之上，我们又加入了三个源自几何深度学习思想的领域特定属性。这些属性包括任务分离，即如果假定待逼近的函数具有某种组合结构，那么我们的估计器也应遵循相同的结构；局部性，即如果 Ω 上存在局部性的概念，那么我们的估计器应将局部邻域视为一个整体；最后是 $\mathbb{G}$不变性，即如果存在某个变换群，使得 f^* 在这些变换下保持不变，那么 f也应如此。

这份期望特性清单未必完整，而且很难断言它是完整的。不过，清单中的所有非领域专用特性都是最优估计器所必需的。例如，无法逼近所有目标函数的不完备估计器，必然会在某些任务上失败。至于从几何深度学习中借鉴的非领域专用特性，很难评估它们对神经网络成功有多关键。以 $\mathbb{G}$-不变性为例，我们将在后续章节讨论其影响。

对于“如何构建这类架构？”这个问题，我们首先展示了如何从这些设计要求推导出深度集合。深度集合是一种成功处理集合的神经网络架构。事实上，在最初的介绍中，深度集合的提出者明确依赖于 $\mathbb{G}$ 不变性和完备性 [64] 的理念。不过，我们发现这一过程需要大量手动工作和领域专业知识。因此，我们介绍了几何深度学习蓝图——作为几何深度学习 [9] 的一部分提出的一种设计此类神经网络的方法。该蓝图只需定义两个函数，一个是 $\mathbb{G}$ 不变的，另一个是 $\mathbb{G}$ 等变的。如果二者都可训练且利普希茨连续，我们就能保证得到一个 $\mathbb{G}$ 不变、连续且可训练的函数。然而，其他期望的性质还需要额外工作，无法得到保证。

现在，我们已经借助几何深度学习的理念回答了最初提出的问题，接下来将尝试评估这些理念的影响。为此，我们将以图神经网络为案例研究来考察它们。我们将观察图神经网络如何被设计为满足领域专用神经网络设计要求，以及这些理念如何体现在其中。之后，我们将尝试应用几何深度学习蓝图，并探究是否能够切实地利用该蓝图创建图神经网络。

在本案例研究之后，我们将尝试评估几何深度学习所提出的领域特定神经网络属性的理论影响。由于任务分离与局部性较难量化，且超出了本文的研究范围，我们将聚焦于 $\mathbb{G}$-不变性，并展示一些能够量化 $\mathbb{G}$-不变性对我们的估计器性能影响的理论结果。

第4章

作为几何深度学习实例的图神经网络

在上一章中，我们阐述了几何深度学习如何回答“优秀的领域特定神经网络应当是什么样的，以及我们应如何构建它们？”这一问题。在本节中，我们将以图神经网络作为案例研究，尝试了解这些思想是如何在图神经网络这一实际的、处于最先进水平的领域特定神经网络架构中得到实现的。

此处以 Scarselli 等人提出的图神经网络 [54], 作为案例研究，因为它们是最杰出的领域特定神经网络之一，并且在过去十年中，在深度学习传统上难以取得良好成果的领域（例如 SAT 求解 [55] 和旅行商问题 [48], ）中取得了有趣的成果，展现了领域特定神经网络的潜力。

此外，正如 [9] 中所述，它们非常严格地遵循几何深度学习的设计理念，因此非常适合作为案例研究，借此观察几何深度学习的思想如何体现在最先进水平的神经网络架构中。

图神经网络是一种用于处理图的领域特定神经网络架构。本节将按以下顺序展开：首先介绍图神经网络。介绍时将先讲解 Weisfeiler-Lehman 检验，这是一种利用图着色来近似判断两个图是否同构的测试。该测试是图神经网络的基础，因为图神经网络本质上是这一测试的一种变体，其中某些部分被全连接神经网络取代。

介绍完这项测试后，我们将用它根据领域专用深度学习设计要求推导出图神经网络，并展示图神经网络如何遵循这些原则。完成这一步后，我们将以图神经网络为例评估几何深度学习蓝图。正如本章所述，图神经网络是根据 Weisfeiler-Lehman 测试推导出来的，因此建立在深厚的专家领域知识之上。几何深度学习蓝图有望缓解这一问题。如果它确实有用，那么就应该能够在不需要太多额外领域知识的情况下，从中推导出图神经网络。因此，我们将尝试这样做，并评估几何深度学习蓝图是否足以推导出图神经网络。

4.1 Weisfeiler-Lehman 测试

两个图 $G_1 = (V_1, E_1, l)$ 和 $G_2 = (V_2, E_2, l)$ 之间的图同构是一个双射函数 $\phi: V_1 \to V_2$，它保持边和标签不变，即

$$l_1(v) = l_2(\phi(v)) \forall v \in V_1 \tag{4.1}$$

$$(u, v) \in E_1 \leftrightarrow (\phi(u), \phi(v)) \in E_2 \tag{4.2}$$

直观来说，同构图，即存在图同构关系的图，在结构上是等价的，只是节点的命名不同。对于几何深度学习而言，这是一种非常值得关注的对称性，因为图上几乎所有有意义的任务都只关心图的结构，因此可以安全地假定它们对于同构是不变的。

如果两个图之间存在同构映射 ϕ，使得 $u = \phi(v)$，我们就称 u 和 v 为同构节点。此外，稍微宽泛地使用这一记号，我们还将令同构映射 ϕ 直接作用于图 $G = (V, E)$，使得 $\phi(G) = G'$，其中 $G' = (V', E', c')$，且 $V = \{\phi(v_1) \ \phi(v_2) \ ...\}$，$E' = \{ (\phi(v) \ \phi(u)) \ | \ (v, u) \in E\}$，并且 $l'(\phi(v)) = l(v)$。根据上下文需要，我们会在两种表示之间切换：一种是将两个图的节点相互映射的同构，另一种是将图映射到图的同构。

虽然图神经网络应当具有图同构不变性，但要实现精确的图同构不变性并不容易。更具体地说，很难构建既能快速评估、又能区分所有非同构图，同时还具有同构不变性的神经网络。这是因为图同构本身就是一个难以判定的问题。虽然尚不清楚它是否属于 NP完全问题，目前也没有能够解决该问题的多项式时间算法 [19]。因此，能够精确保持同构不变性的神经网络架构可能需要指数运行时间，这对于大多数使用神经网络的任务而言都不切实际。

不过，可以使用近似同构检验，它们对真实的同构关系进行过度近似或不足近似。一个例子是 Weisfeiler-Lehman 检验：这是一种基于迭代抽象细化、通过图着色实现的简易而实用的近似方法[43]。WL检验与图神经网络密切相关， [43] 证明了图神经网络的能力与该检验完全相同。

在继续讨论图神经网络之前，我们将简要回顾 WL 检验，因为它展示了图神经网络关于图同构的核心思想，同时无需涉及机器学习细节即可使其可训练。

WL 检验会为图计算一个规范签名，可用于检验它们是否同构。如果两个图的签名相同，它们可能同构。如果两个图的签名不同，则可以确定它们不同构。

为便于记述，本节将考虑无向无标号图；不过，WL 检验以及由此延伸出的图神经网络，都可以很容易地推广到各种不同的设定。
考虑一个图 $G = (V, E)$。WL 检验为每个节点 v 赋予一个颜色 $c(v)$，并执行多个细化步骤来更新这些颜色。直观来说，如果对于两个节点，有$c_t(v) = c_t(u)$，那么在时间步 t，WL 检验无法否定这两个节点可能同构

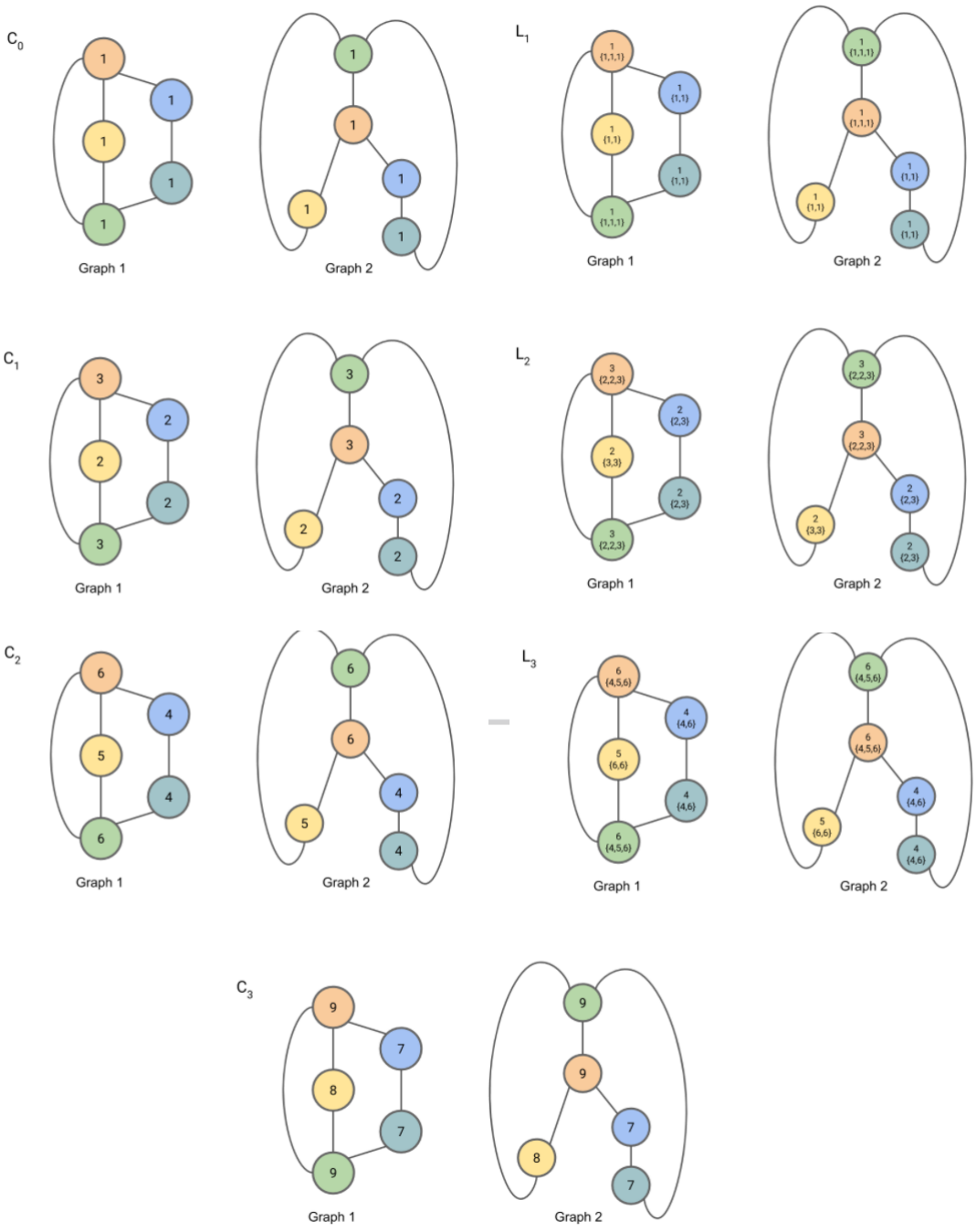


图 4.1： 在两个同构图上运行 Weisfeiler-Lehman 测试。由于这两个图同构，因此它们得出的签名相同，均为“9,9,8,7,7”。对于两个图，左侧展示了每一步的着色，右侧则展示了表示各节点邻域的多重集。请注意，在第一次迭代中，所有节点的颜色都相同；在第二次迭代中，各节点则根据其出度得以区分。示例摘自 [7]

彼此映射。先验地说，所有节点都有可能彼此同构，因此 WL 检验会为每个节点分配相同的颜色：

$$c_0(v) = 0 \tag{4.3}$$

对于带标签图，只有标签相同的节点才能彼此同构映射。因此，在带标签图中：

$$c_0(v) = l(v) \tag{4.4}$$

然后，WL 检验根据以下观察迭代地细化这些颜色：只有当两个节点u 和 v 拥有相同数量的各颜色邻居时，它们才可能彼此同构。这显然源于这样一个事实：任何将 u 映射到 v的同构 ϕ，都需要将 u 的每个邻居映射到 v 的恰好一个邻居，并且不同颜色的节点无法彼此映射。因此，在每一步中，WL 检验都会根据节点 v 邻居的颜色为其分配一种新颜色：

$$c(v) = h(c(u_1), c(u_2), ..., c(u_n)) \tag{4.5}$$

其中，$N(v) = \{u_1, ..., u_n\}$和 h是一个单射哈希函数，它会将每种邻域颜色配置分配为一种新的、不同的颜色，并且不依赖于其输入的顺序。重复这一过程，直至收敛。通过归纳法很容易证明，在每个时间步，同构节点都具有相同的颜色。

因此，在 WL 检验结束时，图 G 的签名是其节点颜色构成的无序列表。

$$s(G) = [c(v_1), c(v_2), ..., c(v_n)] \tag{4.6}$$

显然，只有签名相同的图才可能同构。若图 G_1 与 G_2之间存在同构，则 G_1 中的每个节点都需要映射到 G_2中的一个节点。对于每个同构 ϕ 和所有节点 v，都有 $c(v) = c(\phi(v))$，因此每种颜色在两个图中出现的次数必须相同，签名也就必然等价。有关 Weisfeiler Lehman算法的一次运行示例，请参见图 4.1。

请注意，有些图虽然具有相同的 WL签名，却并不同构。一个非常简单的例子是图 4.2 所示的图。由此可见，WL 检验受其纯粹局部性质的限制，无法区分长度分别为三和六的环。

4.2 从 WL检验 推导图神经网络

考虑以下情形：需要估计一个从图空间映射到目标空间的函数 f^*，并希望构建一种新的神经网络架构来实现这一点。在此例中，我们考虑顶点带标签的图，图表示为 $G = (V, E, l)$，其中 $l: V \to \mathbb{R}^n$ 是一个为顶点分配标签的函数。图以邻接矩阵和标签列表的形式给出。我们假设函数 f^* 关于顶点标签是连续的。更具体地说，对于在节点和边的结构上相同的图 $G_1 = (V, E, l_1)$和 $G_2 = (V, E, l_2)$，我们假设有 [1]

$$l_1 \approx l_2 \implies f^*(x) \approx f^*(y) \tag{4.7}$$

[1]请注意，此处使用的相似性概念是欧几里得相似性概念。这一点很重要，因为它是基于梯度的优化的核心假设之一。

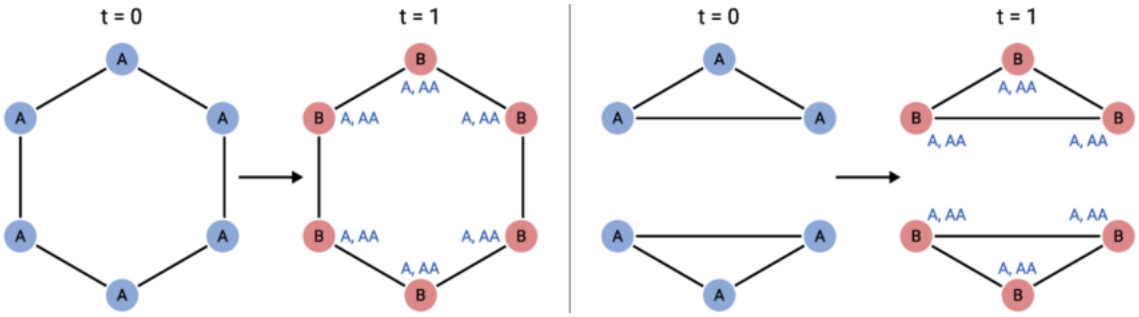


图 4.2： WL 检验无法区分的两个图示例。请注意，这两个图本质上都是循环的，但由于 WL 检验具有局部性，因此无法区分长度为 3 的圆和长度为 6 的圆。示例取自 [15]。

.

Weisfeiler-Lehman 测试

Weisfeiler-Lehman 测试会为图 G 计算一个签名 $sig(G)$，并可用它近似判断图同构。如果两个图具有相同的签名，它们可能同构；如果签名不同，它们则不同构。签名通过迭代过程计算，在每一步中为每个节点分配一种颜色。第一步中，所有颜色都相同。此后，根据节点邻域的颜色是否不同，为节点分配颜色，以区分这些节点。颜色仅用于描述节点的子集，除此之外没有其他特定含义。

在许多实际情境中，这是一个合理的假设，而且在某种程度上，它也被许多深度学习应用隐含地采用。例如，在以图表示的三角网格中，标签可以表示每个顶点的几何位置。可以认为，如果两个三角网格具有相同的连通性，且顶点位置只有细微差异，那么它们表示的几乎是同一个对象，因此预测结果也应该只发生很小的变化。

此外，正如前文所述，我们假设 f^* 在图同构下保持不变，这几乎适用于所有实际应用。

当然，也可以直接采用一个全连接神经网络，将标签和邻接矩阵作为输入。不过，这样做并未利用前文提出的诸多假设；按照几何深度学习的理念，我们希望构建一种从设计上就满足这些假设的神经网络架构。

采用第 3 章中提出的特定领域神经网络设计要求，并代入我们对估计问题所作的假设，可得出图神经网络的以下目标：[2]

1. **可训练性**。我们的网络 f 需要是一个可训练函数（即关于其参数 Θ 既参数化又可微，以便进行基于梯度的优化）。

[2]请注意，此处没有涉及任务分离和局部性的概念，因为它们并非我们先前提出的 f^* 假设中不可或缺的部分。如果所处理的函数 f^* 被假定满足这些性质，则应将它们纳入其中。不过值得注意的是，图神经网络天然就具备局部性，因此这里对此不必过多关注。

2. $\mathbb{G}$**-不变性**。由于我们假设 f^* 在图同构下保持不变，因此我们也希望 f 在图同构下保持不变，从而缩小可近似函数的空间。具体而言，这意味着如果图 x和 y之间存在同构，则 $f(x) = f(y)$。

3. **连续性与可调性**。我们希望确保，当 x 和 y 仅在标签上有所不同，且它们的标签在欧几里得意义上相似时，$f(x)$ 与 $\approx f(y)$ 也相似，因为根据假设，f^* 具备这一性质。此外，为了能够针对特定任务进行微调，我们希望这一性质的程度（即函数的利普希茨性）可以轻松调节。

4. **完备性**。每个关于节点标签连续且对图同构不变的函数 f^* 都应能由我们的神经网络近似。由于图同构难以判定，要求这一性质完全成立并不合理。因此，我们要求尽可能多的连续且对图同构不变的函数 f^* 能够得到近似。

现在，我们将根据这些目标推导出图神经网络架构。
请注意，在深度学习领域，除第二个目标外，所有成功的深度学习模型通常都能满足其余目标。因此，我们似乎可以通过调整标准深度学习工具箱中的一些工具来实现大多数目标。为此，我们将先尝试解决目标（2），然后以此为基础，继续解决其他目标。构建一个在图同构下保持不变的函数，其核心思路是运用 Weisfeiler-Lehman 检验的思想。如上一节所述，WL 检验会为每个图计算一个签名 $sig(x)$，并确保两个图 x 和 y 同构时，它们的签名相同。因此，如果分类函数仅取决于图的签名，那么它在同构下就会自动保持不变。于是，令 ψ 为任意分类函数，我们可以如下定义 f_Θ，并确保它在图同构下保持不变：

$$\mathrm{f}_\Theta(G) = (\psi \circ sig)(G) \tag{4.8}$$

这立即解决了如何构造对图同构不变的神经网络架构这一问题[3]。如果使用全连接神经网络处理图的签名，所得函数也将是可微且参数化的。然而，我们的目标之一尚未达成。WL检验计算出的签名由任意数字组成。请记住，计算签名时，会根据每个顶点周围节点的签名多重集，反复为每个顶点分配哈希值；但具体使用什么哈希函数则完全不受限制。因此，即使只对图的标签做了微小改动，图的签名也可能发生剧烈变化。

因此，对于顶点标签的微小变化，我们无法实现所期望的连续性，因而尚未达成目标（3）。

为了实现目标（3），WL 检验的颜色需要以连续的方式依赖于原始顶点标签。为此，我们来回顾一下最初如何计算图颜色。回想一下 WL 检验的更新方程，其形式如下：

$$c(v) = h(c(u_1), c(u_2), ..., c(u_n)) \tag{4.9}$$

[3]请注意，如果将图的 WL签名作为神经网络的输入，它只能区分 WL 检验能够区分的图，因此其不变性所涵盖的函数范围会比仅针对同构的情况更广一些。不过，在大多数实际应用中，这似乎并不是什么大问题。

节点的颜色会根据其邻居的颜色进行更新，并由某个函数 h 确定用于表示邻居颜色特定配置的具体颜色。为了使 WL 检验能够正常工作， h 只需是一个任意的单射函数，且与其参数的顺序无关。对于估计，我们还要求 h 关于其输入连续。回想一下，WL 检验的迭代过程从颜色等同于原始标签开始，并反复使用 h 更新颜色。如果 h 连续，那么最终标签也会以连续的方式依赖于标签 l ，因为最终签名是通过反复应用 h 构造的，而连续函数的复合仍然连续。

此外，出于估计的目的，我们可以不要求 h 是单射。虽然在区分非同构图这一任务中这项要求很重要，但在我们的应用场景中，可能存在一些图，它们彼此不同构，却相对于 f^* 仍然等价，因此这项要求并非必需。只有当具体的估计任务有此要求时， h 才需要是单射。

总而言之，为了得到一个函数 f_Θ ，使其具有参数化特性、关于其参数可微、对图同构不变，并且关于节点标签连续，我们可以按如下方式实现 f_Θ ：

给定一个输入图 G，使用连续更新函数 h 计算 WL签名 $sig_h(G)$。然后，以 WL签名为最终分类的依据，并计算 $\mathrm{f}_\Theta(G) = \psi(sig_h(G)$，其中 ψ 是一个任意函数，它关于其参数连续，且关于其输入可微。

这种结构保证我们能够满足所有目标值。不过，我们仍需恰当地选择 h 和 ψ ，才能真正实现这一结构。

构建 h 和 ψ

回顾一下对 h施加的限制。它是一个多变量函数[4]，，取决于邻域中的所有顶点颜色；它关于其输入是连续的，并且具有交换性[5]。除此之外， h 基本不受限制。我们并不完全清楚，在满足这些约束所诱导的函数集合中，哪一个 h 是最优的。因此，似乎有必要让 h 也成为可训练函数。幸运的是，我们已经可以引入一种非常适合这项任务的神经网络架构：深度集合。第 3 章介绍的深度集合，专门用于逼近对其输入置换不变的任意函数。因此，根据深度集合架构，我们定义

$$h(v) = \Phi_{update}\left(\sum_{u \in N(v)} \phi_{encode}(c(u))\right) \tag{4.10}$$

作为用于计算 WL签名的更新函数。这里， Φ_{update} 和 ϕ_{encode} 由全连接神经网络实现。

直观而言，某个顶点 v 的 $h(v)$ 按以下方式计算：对于每个相邻顶点 u，

[4]请注意，我们甚至不知道 h 的具体输入维度，因为它取决于顶点的邻近颜色。不同顶点的邻域大小不同，因此我们需要 h 的输入维度可变。[5]我们将用“交换函数”一词来指称不依赖于输入顺序的函数。

计算 $\phi_{encode}(c(u))$，对 $c(u)$ 中包含的信息进行编码，并将计算结果求和。随后，对这些结果应用 Φ，以更新 h 的状态。这直接引出了常用于描述图神经网络[6][21, 9]的消息传递类比。

对 h 这一具体实现有两种解读方式。第一种是直观地看待 h。如果 h 按这种方式实现，我们可以将这一过程想象为：顶点迭代地向彼此发送消息，这些消息由编码全连接网络 ϕ_{encode} 编码，再通过求和进行聚合，最后借助 Φ_{update} 创建新的顶点颜色。

理解以这种方式构建的 h 的第二种思路是：h 的构造使其能够近似任意交换多元函数。正如[52]，所证明的，如果 Φ_{update} 和 ϕ_{encode} 足够强大（也就是说，层数和神经元数量足够多），h 就能以任意精度近似任意交换连续函数。

最后需要指出的是，本节相对轻描淡写地忽略了一个问题：h 的维数是可变的，因为 h 的输入是某个顶点邻域内的所有颜色。由于顶点的度各不相同，这个函数的维数事先并不确定。通常这并不是问题，因为无论输入数量多少，h 都会使用求和来聚合传入的信息。不过，如果我们希望 h 能够近似所有具有 n 个输入的函数，根据公式 3.17，实现 Φ 的网络中最后一个非线性层必须拥有 $2n$ 个神经元。如果我们的问题中出现出度大于 n 的顶点，h 就无法再近似任意函数。不过，这似乎并不是问题，因为大多数问题领域都不包含度极大的顶点，而且神经网络通常会构建大量神经元。回顾一下，为了构建完整的函数 f_Θ，我们将函数拆分为两部分：

$$\mathrm{f}_\Theta(G) = \psi(sig_h(G)) \tag{4.11}$$

第一个模块使用学习得到的连续哈希函数 h计算图 G 的 WL签名。第二个模块 ψ则根据先前计算出的签名，对图进行实际分类。幸运的是，使用构建 h时采用的方法，构建 ψ 相对容易。与 h类似，我们无需对 ψ 施加任何先验限制，只需确保它是连续、多变量、可训练、可交换的[7] 函数，且其确切维度不一定已知。因此，我们使用与 h 相同的结构来构建 ψ，以确保能够构建任意多变量函数。

按照这一构造，ψ 定义如下：

$$\psi(G) = \Phi_{final}(\sum_{v \in V} \phi_{vote}(c(u)) \tag{4.12}$$

对于每个节点，经过 sig(G) 计算后，都会得到一个最终颜色。投票网络 ϕ_{vote}计算每个节点对最终预测的贡献。将这些贡献求和聚合，再通过 Φ_{final}形成最终预测；其中 ϕ_{vote} 和 Φ_{final} 均由简单的全连接神经网络实现。

再次提醒，需注意这种实现基本意味着，只要提供足够多的层/节点，Φ_{final} 和 ϕ_{vote} 就能近似任意交换函数。

[6]虽然消息传递图神经网络只是图神经网络的一种，但这里会介绍它们，因为它们是最通用的一类图神经网络 [9]。在我们的推导中，没有对更新函数 h施加任何领域专用约束。如果对 h施加限制，就会得到不同版本的图神经网络，例如基于注意力的图神经网络或图卷积神经网络 [35, 56] 。这是因为 WL 签名本质上具有置换不变性。

请注意，由于 ψ 可以近似任意交换函数，通用逼近定理几乎可以不费吹灰之力地实现：对于满足我们假设的每个函数 f^* ，只要添加以下条件，就能任意精确地近似它：

$$sig(x) = sig(y) \implies f^*(x) = f^*(y) \tag{4.13}$$

对于任意图 x 和 y[8]。这是因为只要参数设置正确， h 就可以是单射。如果h 是单射，那么图神经网络的两个图签名等价，当且仅当对同一组图进行 WL 检验所计算出的签名等价。

最后， ψ 可以近似从图签名空间到可能类别空间的任意函数，因此普适近似性质也就成立。

至于图神经网络的训练，请注意，图神经网络仅由可训练函数的串接构成。因此，图神经网络可以采用所有基于梯度的优化方法进行训练。

图神经网络总结

本节介绍了图神经网络（GNN），它是针对图这一领域的领域特定神经网络，并符合第 3 章提出的特定领域神经网络设计要求。最重要的是，图神经网络对图同构保持不变，这是特定领域神经网络设计要求所要求的。

为此，图神经网络被构建为 WL 算法的一种变体：其中，颜色更新函数由 Deep Sets（置换不变神经网络）实现，最终的颜色集合则输入另一个 Deep Sets，用于分类。由于图神经网络实现了 WL 算法，因此具有该算法的许多性质。最重要的是，对于图神经网络 f ，只要 WL 检验无法区分 x 和 y，就有 $f(x) = f(y)$。因此，图神经网络对图同构具有不变性。然而，也存在一些图神经网络无法区分的非同构图。因此，它们不满足完备性。

尽管如此，它们仍满足其他所有特定领域神经网络设计要求；对于许多类别的图，WL 检验确实足以将其区分开来。
本节展示了第 3 章提出的特定领域神经网络设计要求如何体现在图神经网络中。本节的核心观点有两点：第一，图神经网络确实遵循第 3 章提出的设计要求。第二，它们通过改造图领域中一种已有且已获充分理解的算法来实现这些要求，并在可以选择函数的地方引入神经网络。这一思路也见于 Deep Sets [64]的推导中。

在下一节中，我们将探讨几何深度学习蓝图能否在此提供捷径，并取代领域知识，成为构建神经网络的工具。

4.3 图神经网络与几何深度学习蓝图

本节将说明 GNN 如何契合几何深度学习蓝图，以及除了隐含使用的 GDL 设计要求之外，GNN 特有的理念如何体现在其设计中

[8] sig 此处指由实际的 WL 检验计算出的签名，而非由采用可学习哈希函数 h 的图神经网络计算出的签名。

图神经网络

图神经网络是针对图领域的领域特定神经网络。图神经网络是作用于图的函数 $\to C$，以图 $G = (V, E)$ 作为输入，这些图由节点标签矩阵 $\hat{V} \in \mathbb{R}^{d\times|V|}$ 和邻接矩阵 $\hat{} E \in \mathbb{B}^{|V|\times|V|}$表示。图神经网络使用 Deep Sets 作为颜色更新函数 h，计算图的 WL签名。然后将此签名传递给另一个 Deep Sets，以得出最终分类结果。对于所有图神经网络 f，对于所有无法通过 WL 检验区分的图 x, y，均有 $f(x) = f(y)$。因此，图神经网络对图同构保持不变。

在本节前面的推导中。为此，首先回顾几何深度学习蓝图的结构。如第 3 章所述，几何深度学习蓝图规定，估计器 f 的构造应满足

$$f = \Phi \circ \phi_n \circ \phi_{n-1} .... \circ \phi_1 \tag{4.14}$$

wher其中 ϕ_i 是可训练的利普希茨连续函数，并且对于 $\mathbb{G}$ 的作用具有等变性 :

$$\phi_i(gx) = g\phi_i(x) \tag{4.15}$$

而 Φ 是可训练的利普希茨连续函数，并且对于 $\mathbb{G}$ 的作用也具有不变性

$$\Phi(gx) = \Phi(x) \tag{4.16}$$

回想一下，在图神经网络的语境中，我们关注的作用 $\mathbb{G}$ 包括图同构，即对节点集 V 进行置换，并相应地改变边。

我们将从某个函数 ϕ[9] 开始，其中需要满足条件 $\phi(gx) = g\phi(x)$。首先，对于所有 $\mathbb{G}$-等变函数而言，要能够将 g 应用于 $\phi(x)$，必须要求 $\phi(x)$ 与 $\phi(x)$ 属于同一类对象。因此，$\phi(x)$ 需要是一个将带标签图映射到带标签图的函数，并且其映射方式与节点顺序的置换正交。在图神经网络的情况下，图神经网络所实现的改进的WL算法中的每个 WL步骤都构成这样的函数ϕ。首先，很容易验证，一个 WL步骤会将一个带标签图映射到另一个带标签图，因为它只会为该图计算新的标签，更具体地说，是新的颜色。此外，我们还可以通过观察 WL步骤（或消息传递步骤）对图 $G = (V, E, c)$ 以及由 G 通过某个同构/置换 π 得到的一个同构图 $G' = (V', E', c')$ 的影响，发现每个 WL步骤确实满足等变条件。

回想一下，每个 WL 步骤都会根据以下函数为 G 计算新的颜色，从而得到新的图 $G_{new} = h(G) = (V, E, c_{new})$：

$$c_{new}(v) = \Phi_{update}\left(\sum_{u\in N(v)} \phi_{encode}(c(u))\right) \tag{4.17}$$

对于一个节点 $v \in V$，其邻域为 $N(v) = \{u_1, ..., u_n\}$，其中 Φ_{update}和 ϕ_{encode}均由全连接神经网络实现。现在考虑同构图中相同的更新步骤

[9]为简洁起见，我们将省略下标，但请注意，这适用于蓝图中的所有函数 ϕ_i 。

图 G'，具体来说，是与 v相对应的节点 $v' = \pi(v)$。由于π是同构， v' 的邻域为 $N(v')$ $= \{u'_1 = \pi(u_1) ..., u'_n = \pi(u_n)\}$。由于 G和 G'同构，因此必然有

$$c(u) = c'(\pi(u)) \tag{4.18}$$

对所有 $u \in V$均成立。因此，我们可以立即看出

$$\begin{aligned} c_{new}(v) &= \Phi_{update}\left(\sum_{u \in N(v)} \phi_{encode}(c(u))\right) \\ &= \Phi_{update}\left(\sum_{u' \in N(\pi(v))} \phi_{encode}(c'(u'))\right) \quad \text{(as } \pi \text{ is an isomorphism)} \\ &= c_{new}(\pi(v)) \end{aligned}$$

因此，我们可以立即看出，在经过这一步消息传递后， π 仍然是图 G_{new}与G'_{new} 之间的一个同构。因此，这一步消息传递 h 保留了它们之间的同构，并且

$$\pi(h(G)) = h(\pi(G)) \tag{4.19}$$

因此，更新函数 h 满足等变性。
最后，通过训练好的 WL 算法计算出节点颜色后，图神经网络会按如下方式计算最终结果:

$$\psi(G) = \Phi_{final}(\sum_{v \in V} \phi_{vote}(c(u)) \tag{4.20}$$

当然，这种表述对图同构是不变的。原因在于， ϕ_{vote} 分别应用于每个节点，不受图结构影响，并且所有节点嵌入都通过求和进行聚合。如果对节点顺序进行置换，通过求和进行聚合可确保最终结果不变。因此，这个最终分类步骤取代了几何深度学习蓝图中的不变最终函数，由此可知，图神经网络遵循几何深度学习蓝图；这一点也已在 [9]中阐述。

4.4 能否从 GDL蓝图推导出图神经网络?

正如刚才所示，图神经网络遵循几何深度学习蓝图。然而，最有趣的问题并不是图神经网络是否遵循几何深度学习蓝图，因为几何深度学习蓝图本就是基于图神经网络的理念构建的。相反，既然我们关注的是将几何深度学习应用于新领域这一理念，那么问题在于，能否以可行的方式从这份蓝图推导出图神经网络。这个问题当然很难事后回答，因为图神经网络已经存在，其理念也早已为人所知。此外，能否从几何深度学习蓝图推导出图神经网络，正如前文所说，由于这份蓝图给使用者留下了不容忽视的工作空间，因此在很大程度上取决于主观判断，也受使用者的技能和想法影响。因此，本节更多是一种非正式的论述，本身并不能完全令人信服。尽管如此，我们仍将尝试评估这份

几何深度学习蓝图留给使用者多少工作，以及在假设由几何深度学习蓝图指导图神经网络设计的情况下，它能提供多大帮助。
回顾一下，几何深度学习蓝图要求包含三个不同的函数：

1. 特征表示函数 $\phi : \Omega \to \Omega$。该函数应当可训练、利普希茨连续且满足 $\mathbb{G}$-等变性，即 $f(gx) = gf(x)$，并且能够无损，也就是说，存在一组参数 θ，使得 ϕ_θ 是双射。2. 不变性保证聚合函数 $\Sigma : \Omega \to \mathbb{R}^k$，它是一个固定的 $\mathbb{G}$-不变函数，通常实现为简单求和或类似的聚合函数。
3. 分类函数 $\Phi : \mathbb{R}^k \to C$，用于完成最终分类。该函数应当可训练且利普希茨连续。

我们先从最容易定义的函数开始。确保 $\mathbb{G}$-不变性的函数 Σ 相对简单，就是对所有节点嵌入求和。以这种方式求和会彻底舍弃图结构的所有概念，因此对同构（以及许多其他变换）保持不变。同样，函数 Φ 只需满足可训练和利普希茨连续这两个条件，因此可以通过简单的全连接神经网络来实现。

这两种选择都与图神经网络中的做法一致，而且正如我们将要论证的那样，它们是合理的。使用求和来确保某个对象集合的置换不变性，这一思路见于 Deep Sets，也隐含在所有使用均值池化的领域中。同样，正如我们在本文前文所确立的，每当需要使用利普希茨连续的可训练函数进行估计时，全连接神经网络都是深度学习中的标准选择。因此，我们认为，如果采用几何深度学习蓝图来设计适用于图的神经网络架构，就有可能得出图神经网络中使用的函数 Σ 和 Φ。

然而，对于等变函数 ϕ，情况就困难得多，因为有许多函数都满足约束 $f(gx) = gf(x)$。例如，可以像 Deep Sets 中那样，分别将神经网络应用于每个节点标签，以此实现 ϕ。这符合 $\mathbb{G}$ 等变性以及对 ϕ 的所有其他限制，但这种做法很难令人满意，因为它完全忽略了边，因此不可能使整个构造成为所需领域的通用逼近器。同样，也可以通过沿着那些不相连的节点进行消息传递来实现 $\mathbb{G}$ 不变性，但这种做法本身也不尽如人意。此外，即使不选择上述任何一种不尽如人意的思路，也没有明显的结构性方法能够指出，通常在图神经网络中实现的消息传递就是正确的选择。甚至有可能想到以某种方式实现 ϕ，使其改变输入图的边结构，从而产生无限多种可行但很可能毫无用处的 ϕ 版本。

这极具问题性，因为在实践中，几乎所有特定领域神经网络的实例中，特征表示函数 ϕ 都构成了神经网络架构中迄今为止最大的部分（可与 [37, 24]中的 CNN架构相比较）。在此处选择错误的函数可能会导致神经网络性能严重下降，可以说，其影响远大于在另外两个函数 Σ 和 Φ中出错。事实上，至少在 CNN 的语境下，存在一些有用的架构完全不使用后面提到的函数，却仍能取得不错的结果 [50]„，这表明真正发挥主要作用的是特征表示函数 ϕ，它对应于

图神经网络中的 WL 步骤／消息传递。
因此，我们认为，几何深度学习蓝图在最关键的领域缺乏具体性，也就是最初几层：这些层负责为输入计算嵌入，供后续更简单的分类函数使用。事实上，在这一领域，它几乎没有提供任何传统深度学习直觉 [24]未涵盖的内容。我们认为，不能指望这一蓝图能够在传统方法之外，显著助力特定领域的架构创建；它似乎只是提出了关于实际架构蓝图应有何种形式的初步构想，而非一种在实践中真正有用的方法。

第5章

强制施加$\mathbb{G}$-不变性对估计器性能的影响

上一章中，我们以图神经网络为案例研究，探讨了基于几何深度学习的特定领域神经网络设计理念，如何体现在最先进水平的神经网络架构中。然而，这只能说明这些理念在实践中得到了应用，并不能充分证明几何深度学习的理念对于构建优秀的特定领域神经网络架构是必不可少的。简言之，上一章表明，借助几何深度学习的理念可以构建出优秀的神经网络架构，但我们无法证明这些理念正是该架构取得成功的原因。

本章将尝试回答这个问题：几何深度学习的理念究竟有何影响？回顾一下，几何深度学习应用于特定领域神经网络的核心理念是任务分离、局部性和 $\mathbb{G}$-不变性。全面探讨这三者将超出本文的范围；而且，任务分离和局部性更难以恰当地刻画，因此我们将聚焦于 $\mathbb{G}$-不变性这一关注属性。因此，本章将尝试回答以下问题：“在已知 f^* 是 $\mathbb{G}$-不变的情况下，在估计器中强制施加 $\mathbb{G}$-不变性会产生什么影响？”

首先需要注意的是，要对一大类神经网络给出理论保证（从而评估 $\mathbb{G}$-不变性的影响）通常是一个非常困难的问题。正如 C. Zhang 在深度学习领域两篇极具影响力的论文中指出的， [66, 65], 深度学习模型的表达能力实在过于强大，许多标准分析方法难以应对。正如 [66],所示，经过大量正则化的标准神经网络甚至可以轻易拟合最荒谬的训练数据：在毫无结构的随机任务上，通常也能达到几乎为 0 的训练误差[1]。

要衡量神经网络的成功程度，我们可以考察两点：第一，训练误差，也就是说，这个神经网络能否正确拟合训练数据？正如刚才所说，对于现代神经网络而言，这几乎是必然的，因为现代神经网络可以利用极其大量的层和神经元，正确拟合训练数据轻而易举

[1] [65] 开展的实验与本研究直接相关。在他们的研究中，他们采用了数据增强；与本研究使用的方法相比，这是一种较弱的融入 $\mathbb{G}$-不变性的方式。他们表明，这种方法不足以对神经网络进行充分正则化。相反，他们发现，即使将这些网络训练为具有 $\mathbb{G}$-不变性，它们在非 $\mathbb{G}$-不变任务上仍然可以达到几乎为 0 的训练误差

的网络 [65]。因此，这个问题在实践中并不值得探讨。第二，泛化误差，也就是说，在训练误差较低的情况下，我们有多大把握认为模型对未曾观测到的数据点也能表现良好？这个问题对于实际应用而言有趣得多，但也难以回答得多。

传统统计观点通常认为，能力较弱的估计器，也就是只能拟合有限数量函数的估计器，具有更好的泛化能力。其思路是：如果函数 f^* 不属于我的估计器能够近似的那一小部分函数，那么我们随机抽取到的样本 f^* 很可能无法被该估计器恰当地拟合。因此，如果训练误差较低，我们就有理由相信，真实误差不会偏离太远。广泛使用的 Vapnik-Chervonenkis 维数[57]界就是这种理念的一个例子。

正如 [66], 所指出的，这些结果并不适用于现代神经网络，因为神经网络在可近似的范围上并不受限。因此，传统的神经网络泛化统计界并不起作用。

尽管如此，神经网络在实践中往往具有良好的泛化能力，尤其是在经常使用它们的领域。因此，我们必须采用 [66]所要求的新方法。这些方法大多基于这样一种思路：将领域的特定属性纳入其中。如果我们假设函数 f^* 具有某些属性，就可以给出泛化性能的界，并更好地说明 𝔾-不变性如何帮助学习。

本章将尝试提供一些见解，并尽可能量化 𝔾-不变性如何塑造学习过程。我们将特别关注一些技术，这些技术在 [66], 中被列为分析神经网络时可能的答案，即贝叶斯 PAC 界，以及基于梯度的优化所引入的偏置。

本章首先将形式化对 𝔾-不变函数施加限制所产生的影响，并从三个不同角度探讨 𝔾-不变性对神经网络的影响：一是基于梯度的优化及这种训练方法所引入的偏置，二是分类空间的拓扑刻画，最后是对泛化性质的贝叶斯 PAC 分析。

5.1 刻画 𝔾-不变估计问题

首先，我们需要了解 𝔾-不变性如何改变估计问题。在本章余下部分，我们假设有一个待近似的函数 f^* ，它实际上对于某个变换群 𝔾是不变的。

我们所要解决的原始、未受约束的估计问题表述如下：

$$\hat{f} = \arg\min_{f \in F} \int_{x \in \Omega} l(t(x), f(x)) d\mu(x) \tag{5.1}$$

其中，积分尤其是在所有可能输入构成的完整空间 Ω 上进行，通常是向量空间R^n, $t : \Omega \to C$ 表示 x的真实标签， l 是某个损失函数， μ 是 Ω 上的某种分布。通过引入关于 𝔾不变性的条件：

$$\forall x \in \Omega \forall g \in \mathbb{G}\, f(x) = f(gx) \tag{5.2}$$

只需估计一个函数轨道(Ω) $\to C$，其中轨道(Ω) $= \{$轨道$(x) \mid x \in \Omega\}$，领域便从 Ω 简化为由 $\mathbb{G}$ 诱导、由轨道[2]表示的划分。
这可能使问题大幅简化。例如，考虑平移三单位所对应的不变变换群作用于 $\mathbb{R}$，即对所有 x，$f(x) = f(x+3)$，其轨道为

$$\text{orbit}(x) = \{...x-3, x, x+3, ...\} \tag{5.3}$$

容易看出，这个轨道集合与空间 $(0, 3) \subset \mathbb{R}$ 同构，而后者本身并非向量空间。虽然估计问题 $(0, 3) \to \mathbb{R}$ 确实可能很困难，但它们应该比相应的估计问题 $\mathbb{R} \to \mathbb{R}$ 更容易。神经网络中的具体表现，我们将在接下来的章节中探讨。

$\mathbb{G}$不变估计问题

引入 $\mathbb{G}$-不变性可以简化估计问题。我们不再估计函数 $\Omega \to C$，而是利用 $\mathbb{G}$-不变性来近似函数轨道(Ω) $\to C$，其中轨道(Ω) 是由 $\mathbb{G}$ 诱导的轨道集合。在我们讨论的函数示例中，该函数对 3 的倍数加法保持不变，因此复杂度大幅降低。

5.2 不变性与梯度下降

神经网络首先是机器学习模型，因此人们常从统计视角将其视为随机采样和训练过程 [24] 的最终结果。举例来说，人们常将神经网络视为最大似然估计器，如 [24] 中所述。这种视角非常通用，因为它抽象掉了具体的优化过程。这有助于论证损失函数以及优化该损失函数这一具体思路的合理性，但忽略了这样一个事实：神经网络通常并未得到完全优化，也就是说，通常无法找到损失函数的全局最小值，而且优化器可能收敛到的局部最小值不止一个。因此，我们有必要关注优化函数的具体行为，以了解最终到达了哪个局部最小值

正如 [66], 所指出的，理解基于梯度的优化过程是解释神经网络如何学习以及如何泛化的一种途径。标准统计界往往不适用于神经网络，而优化过程实际上并非随机；训练过程引入的偏置或许能够解释神经网络为何表现良好。

在各种基于梯度的优化器中，例如梯度下降 [24] 或 ADAM, [34], 更新的形式为

$$\Theta_{new} \leftarrow \Theta_{old} + \alpha \nabla_{\Theta} l_x(\Theta) \tag{5.4}$$

其中 x 是给定的样本。对于无穷小的学习率 α，可以很容易地证明，这总能改善误差。

我们不只检查误差，还将考察函数值本身如何随着基于梯度的更新而变化，因为我们关注学习过程及其所受的影响

[2]关于群和轨道的更多内容，请参阅附录。

以及网络的 𝔾-不变性。进行泰勒级数展开并代入上述更新后，可得：

$$f_x(\Theta_{new}) \approx f_x(\Theta_{old}) + \nabla_\Theta f_x(\Theta)\alpha * \nabla_\Theta l_x(\Theta) \tag{5.5}$$

$$= f_x(\Theta_{old}) + \alpha\nabla_\Theta f_x(\Theta)\frac{\partial l}{\partial f}\nabla_\Theta f_x(\Theta) \tag{5.6}$$

$$= f_x(\Theta_{old}) + \alpha\frac{\partial l}{\partial f}||\nabla_\Theta l_x(\Theta)||^2 \tag{5.7}$$

$$\tag{5.8}$$

由于梯度连续，因此在 x的（非常接近的）邻域内，梯度相似；由此可进一步将结论扩展到 x 的直接邻域中的点。

$$f_y(\Theta_{new}) \approx f_y(\Theta_{old}) + \nabla_\Theta f_y(\Theta)\alpha\frac{\partial l}{\partial f}\nabla_\Theta f_y(\Theta) \approx f_x(\Theta_{old}) + \alpha\frac{\partial l}{\partial f}||\nabla_\Theta l_x(\Theta)||^2 \tag{5.9}$$

这对基于梯度的优化算法给出了非常直观的描述，有助于考察 𝔾-不变性对学习过程的影响。根本而言，给定一个样本 (x,$label$)，函数值会按比例发生变化，其变化量与 $\alpha||\nabla_\Theta l_x(\Theta)||^2$ 成正比，而项 $\frac{\partial l}{\partial f}$ 隐式依赖于 $label$，表示变化的方向。这很直观，因为项 $\alpha||\nabla_\Theta l_x(\Theta)||^2$ 由点 x处梯度的平方长度构成，衡量了 f 对 Θ 变化的敏感程度以及学习率，因此描述了当 Θ 发生变化时，点 x 处的 f 如何变化。相较之下，项 $\frac{\partial l}{\partial f}$ 描述了 f变化时损失如何变化，因此指明了 f期望的变化方向。正如公式5.9所示，对于梯度 $\nabla_\Theta f(x)$的方向与 x处梯度方向相似的点y ，其变化也相似。对于梯度方向相反的所有点，则会发生相反的变化。如前所述，这尤其适用于点 x非常近的直接欧几里得邻域中的点，因为梯度是连续的。然而，对于其他所有点，在一般设定下，很难了解它们各自的梯度 ∇_Θ ，因而难以预测优化将如何进行。

这一点直接说明了为什么标准神经网络架构难以很好地泛化到非欧几里得数据。优化过程假设：在欧几里得意义下彼此接近的点是相似的。显然，这对于非欧几里得数据来说并不合理；在欧几里得度量下彼此接近的点可能截然不同，而在欧几里得度量下并不接近的点也可能几乎是同一个数据点。例如，考虑仅移动了一个像素的图像。显然，这样移动后的图像仍与原图非常相似，但这两幅图像之间的欧几里得距离可能非常大。

随着加入 𝔾-不变性，情况发生了变化。事实上，将 𝔾-不变性引入网络后，对于所有 g、 x 和 Θ，恒等式 $f_\Theta(x) = f_\Theta(gx)$均成立。因此，我们知道 $\nabla_\Theta f_\Omega(x) = \nabla_\Theta f_\Omega(gx)$。由此，发生在点 x 处的同一变化，也会发生在所有移位点 gx 处，其中 $g \in \mathbb{G}$均成立。此外，公式5.9表明， gx 的直接邻域也会经历类似的变化。这说明，不变性为学习过程增添了额外的结构。我们可以将其理解为在学习过程中引入了一种新的邻域概念：

$$dist(x, y) = \min_g ||y - gx|| \tag{5.10}$$

而不是在不受约束的情形下基于梯度的优化所遵循的标准欧几里得距离。

这直接关系到几何深度学习的核心目标之一，即将深度学习的成功拓展到非欧几里得域 [9], ，使其适用于相关研究中涉及的三角网格和图等领域。然而值得注意的是，公式 5.10 中上述距离的表达仍然将欧几里得距离作为一个因素。对于研究中考察的几乎所有用例，上述表达都非常合理，因为它们几乎都具有类似图的结构，其中有若干由实数编码的对象，处于某种有序结构中。图像表示为像素网格（每个像素都由一些实数表示），图由节点组成（每个节点都由一个特征向量表示），而三角网格由顶点组成，每个顶点都表示为三维空间中的点。在这些情形中引入不变性，涉及的是这些欧几里得表示的排序；但表示本身在本质上仍是欧几里得的。例如，考虑图像旋转的情形以及公式 5.10。这种相似性概念将图像视为相似，是指它们在欧几里得意义上与彼此旋转后的版本相似。

对于这种用例而言，这是一个非常合理的邻域概念，而且通常也适用于 [9], 中介绍的所有用例；但似乎不能据此声称，引入不变性就能涵盖完全非欧几里得域、离散输入、非连续输入等情况。目前看来，仅引入不变性还不足以做到这一点。相反，它很适合某一类领域，即 5.10 所表示的距离概念合理的领域。

对 $\mathbb{G}$-不变估计器进行梯度下降

假设学习率 α 是无穷小的。给定一个样本 $(x,\ f^*(x))$，梯度下降会使 $f(x)$ 更接近 $f^*(x)$。由于 f 的连续性， x 的欧几里得邻域也会经历类似的变化。对于 $\mathbb{G}$不变网络，情况则有所不同。此时，不再是 x 的欧几里得邻域经历相同的变化，而是由以下邻域取代：

$$dist(x,y) = \min_g ||y - gx|| \tag{5.11}$$

5.3 不变性与拓扑结构

神经网络能够模拟的函数空间的拓扑复杂度，一直是深度学习文献中反复关注的主题。在他们的研究中， [6] 和 [27], 的作者分别给出了神经网络能够在其原像中近似的拓扑洞数量的上界。

考虑一个由神经网络负责处理的二分类问题。因此，存在两个类别， $C^+ \subset \Omega$ 和 $C^- \subset \Omega$，需要通过一个分类函数加以区分。由于我们关注的是二元分类，将所有 $x \in \Omega$ 正确分类到各自类别中的任务可以很容易地表述如下：

$$f(x) > 0 \Leftrightarrow x \in C^+ \tag{5.12}$$

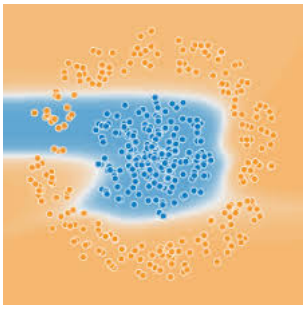


图5.1： 二分类问题中的若干点 [33]。蓝色点属于类别C^+，黄色点属于 C^-。对于神经网络而言，这个空间在拓扑上相对容易学习，因为子空间 $C^+ \subset \Omega$和子集 $C^- \subset \Omega$各自只有一个拓扑孔。

因此，我们的神经网络分类器应生成一个函数，使其恰好在 C^+ 的区域内为正，在 C^- 的区域内为负。如 [6] 所示，只有当子空间 C^+ 和 C^- 中的拓扑洞数量有限时，这才有可能实现。神经网络仍可近似的拓扑洞确切数量取决于其架构、激活函数和规模，但这一数量必然有限； [6] 和 [27] 均表明，要近似具有更多拓扑洞的空间，需要更大、能力更强的神经网络以及更多数据点。

请注意，这一论证与优化算法无关，而仅涉及神经网络架构的表达能力，但它引发的问题与第5.2节中提出的问题类似；一般的神经网络假设某种形式的连续性，因此难以处理缺乏这一性质的问题。

从原像的拓扑结构来看，很容易看出，神经网络缺乏 𝔾-不变性可能会造成灾难性影响。例如，考虑要由神经网络 f 近似的函数 $f^*(x) = x \bmod 3 > 1$。表面上看，这个函数极其简单，但实际上神经网络很难学会它。在我们的实验中，采用不同激活函数、最多有 8 层的全连接神经网络都未能完整地正确学习这个函数。不过，如果考虑空间

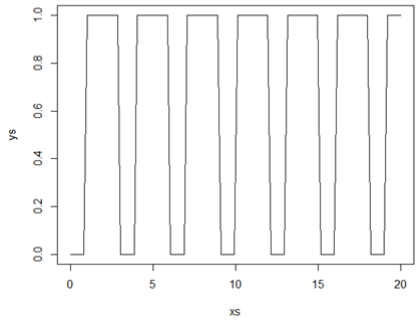


图5.2： $f^*(x) = x \mod 3 > 1$ 的图像。请注意，在 $f^*(x) = 1$的区域之间存在无限多个间隙。为了更容易地绘制这个函数的图像，我们用1表示布尔值“真”，用0表示“假”。

$C^+ = f^{-1}(1)$，其定义如下：

$$...(-2,0) \cup (1,3) \cup (4,6) \cup .. \tag{5.13}$$

并且包含无限多个孔洞。因此，无论我们的神经网络有多大，都无法充分近似 f^* 这个无限大的结构。当然，随着增加层和神经元，神经网络可以近似 f^* 中越来越多的部分，如 [6]，所示；但它永远无法真正达到这个无限结构，而小型网络与实际问题相差甚远。

相较之下，考虑 f^* 的平移不变性时，问题会容易得多。正如第 5.1 节所述，考虑 $\mathbb{G}$ 不变性后，近似的领域从 Ω 缩小为 orbit(Ω)。因此，我们无需对所有 x 近似 $f(x)$，只需近似 f(orbit(x)) 即可。在这个具体例子中，由于 f^* 中的模 3，我们知道 f^* 在平移 3 个单位后保持不变，即对于所有 x，$f^*(x) = f^*(x+3)$。因此，x 的轨道为：

$$\text{orbit}(x) = \{...x-3, x, x+3, x+6, ...\} \tag{5.14}$$

所有轨道的集合因此同构于任意长度为 3 的开区间，尤其是区间 (0，3)。因此，我们的近似任务等价于在区间 (0，3) 上近似 f^*，该区间只包含一个拓扑孔（参见图5.2），这大幅降低了该问题的复杂度。
这显然是一个极端的例子；由于网络的确切行为取决于不变性的实现方式，因此很难以一般方式量化拓扑孔在 $\mathbb{G}$-不变性下的变化。当然，也可以通过某种方式实现 $\mathbb{G}$-不变性，从而得到另一个复杂度很高的领域。尽管如此，强制施加 $\mathbb{G}$-不变性应始终能大幅降低拓扑复杂度。

$\mathbb{G}$-不变性与拓扑复杂度

神经网络 $\mathbb{R}^k \to \mathbb{R}$ 通过 $C^+ = \{x \mid f(x) > 0\}$ 和 $C^- = \{x \mid f(x) \leq 0\}$在其领域 Ω 上诱导出两个拓扑空间。对于标准的神经网络架构，这些空间中的拓扑孔数量有限，其数量取决于激活函数和层数。对于 ReLU 激活函数，拓扑孔的数量为 $O(2^l)$，其中 l 是层数 [6]。$\mathbb{G}$-不变性会将领域从 Ω 切换为 orbit(Ω)。后者的拓扑孔数量从不多于前者，而且可能远少于前者。

5.4 不变神经网络的泛化性质

前两个部分讨论了 $\mathbb{G}$-不变神经网络的表达能力和学习过程，也就是具体的数学优化；现在，我们将探讨它们的统计泛化性质。正如本节开头所述，传统统计分析难以证明神经网络的泛化性能。因此，在本节中

我们将采用贝叶斯 PAC 技术。与传统统计方法类似，贝叶斯 PAC 技术旨在根据训练集上实现的风险，对整个数据集上的实际风险给出一个界。贝叶斯 PAC 技术的优势在于，它允许用户使用“定制”的界，并在拥有可利用的领域知识时对其进行改进。

本节介绍 [39]取得的结果。
首先需要指出的是， [39] 的结果是在统计学视角下得出的，因此将估计问题重新表述如下：

给定输入 X 及标签 T 的数据样本 $d \in X \times T \sim D$，其中 D 是生成数据的分布，求属于某个估计器类 F 的（概率）估计器 f，使风险 $R(f, D) = \mathbb{E}_{d\sim D}(l(f, d))$ 最小。

请注意，这假设神经网络以某种方式对概率分布进行建模。尽管这与计算机科学领域通常介绍神经网络的方式不同，而且确实对神经网络的函数施加了限制，但这种做法适用于几乎所有分类架构；事实上，神经网络使用的大多数损失函数都源自这一视角，并且可以直接让我们将神经网络视为概率分布。还需注意，由于 D 是数据样本及其对应标签的联合分布，因此它也隐式编码了样本 $x \in X$ 的密度，即边际似然

$$D(x) = \int_{t\in T} D(x, t)dt \tag{5.15}$$

以及在给定 x 的条件下 t 的条件密度

$$D(y|x) = \frac{D(x, y)}{D(x)} \tag{5.16}$$

当然，尽管我们希望最小化实际风险 $R(f, D)$，但它几乎从未被实际获知。因此，我们转而优化训练集 D 上的经验风险：

$$\hat{R}(f, D) = \frac{1}{n}\Sigma_{(x,t)\in\hat{D}} l(x, t) \tag{5.17}$$

[57]。例如，考虑包含所有可能函数 $X \to T$ 的函数空间。当然，如果给定一个合理的训练集，我们可以定义一个概率分布 $f(x, y)$，使得 $p(y|x) = 1$[3]$\leftrightarrow (x, y) \in D$，从而轻而易举地达到 $R(f, D) = 0$，而实际风险仍可能任意高。当然，这是因为函数类 F 完全不受限制。

由于 𝔾-不变性对估计器所属的函数类 F 施加了限制，因此从直觉上看，这种限制似乎有望降低泛化误差。
为得到他们的结果， [39] 的作者考虑了某个非 𝔾-不变估计器 f 的对称化估计器 f°。这里，f° 定义为：

$$f^\circ(x) = \int_{g\in\mathbb{G}} f(gx)d\lambda(g) = \mathbb{E}_{g\ \lambda}[f(gx)] \tag{5.18}$$

直观而言， f°是对原始估计器 f进行变换后，再对每个元素的轨道求平均所得的版本。这确保了 𝔾- 不变性（请检查上述方程中的积分是否等价于

[3]请注意此处 f 与 p 的区别。 f 指的是（连续或离散）分布，而 p 指的是实际概率。在离散分布中， $p(x) = 1$ 意味着 $f(x) = 1$；但在连续分布中， $p(x) = 1$意味着 $f(x) = \infty$，因为连续测度与离散测度不同。

对每个 $y \in$ 轨道（x）进行求和），而将范围限制为 f° 形式的函数，则意味着只考虑 F 中的 $\mathbb{G}$-不变函数，假设 $f^\circ \in F$。另请注意，这类似于 GDL蓝图中可能采用的不变最终层。事实上，该工作提出了 [9] 以这种方式构建最终层。

随后，作者推导出了一些有趣的结果，首先涉及风险 $\hat{R}$ 及其经验对应量 R。首先，他们证明，对于所有函数 $f \in F$ 和所有凸损失函数 l，都有

$$\hat{R}(f^\circ, D^N) \leq R(f, D^N) \tag{5.19}$$

这表明，对于每个不受限制的函数 f，都可以通过对称化来降低经验风险。这本身很有趣，但对神经网络而言帮助不大。鉴于现代神经网络的规模 ˆ，通常可以将 R 降至任意接近零，无论是否具有$\mathbb{G}$-不变性。请注意，所有这些结果都假设 l 是凸的。这是这些定理的局限性，但仍适用于几乎所有神经网络架构，因为深度学习损失的设计目标是使其凸 [24]。

ˆ 更有趣的是，他们证明，假设 R 的方差有限，这个不等式也适用于其方差：

$$Var[\hat{R}(f^\circ, D^N)] \leq Var[R(f, D^N)] \tag{5.20}$$

这意味着对 f 进行对称化能更好地估计风险。这有两个主要影响。首先，由于方差降低，对测试集上的误差估计会更加准确。然而，更重要的是，在学习过程中，当 f 尚未完全拟合训练集时，梯度 $\nabla_\Theta R$的估计也会得到改善，因为 ˆ$R(f^\circ,\ D^N)$ 更接近 $R(f^\circ,\ D^N)$ 的真实值。直观而言，这意味着 $\mathbb{G}$-不变函数的学习过程所采取的步骤，应当更贴近实际全局风险的改善，而不只是经验风险的改善。

不过值得注意的是，这些不等式都不是严格不等式，而且几乎没有说明估计究竟改善了多少。相反，我们只能证明，对称化之后某些特定指标不会变差。

可以证明，更准确的估计会改善估计器的 PAC-贝叶斯界。PAC-贝叶斯界是对可能近似准确（PAC）的估计器实际风险的一个界，即一个界 $R(f,\ D) \leq$ˆ$R(f,\ D^N) + K$；随着 K增大，该界以任意高的概率成立。请注意，PAC-贝叶斯界只是指一类界，并非某个特定的界。还需注意，PAC-贝叶斯方法指的是吉布斯估计机制 [12]。吉布斯估计假设训练的并非单个分类器，而是对可能分类器的类别 F 上的一个分布 Q 进行训练。这并不能直接应用于神经网络，因为神经网络通常编码的是单个分类器；不过，可以通过向其参数添加随机噪声，或者将 $Q(f)$ 设为选择初始参数 θ_0 的概率来轻松扩展，使得 f 是从 θ_0开始，经基于梯度的优化所得到的估计器。吉布斯估计在理论上有时更容易处理，也能考虑神经网络训练中普遍存在的随机性，这种随机性通常由初始化或提前停止引起（可将其理解为添加到最终权重中的随机噪声）。

作者使用 [12] 的 PAC 界，其内容如下：

如果 D^n 是根据某个数据生成分布 P_D 独立同分布抽取的， Q 是可能的估计器 $f \in F$ 上的后验分布，那么对于所有先验分布 P 以及所有 $\delta \in (0,1)$，并且 $\beta > 0$，以至少 $1-\delta$ 的概率，对所有可能的样本都有

$$R(Q) \leq \frac{1 - e^{-\beta \hat{R}(Q,D^N) - \frac{1}{n}(KL(Q||P) + \log \frac{1}{\delta}}}{1 - e^{-\beta}} \tag{5.21}$$

$KL(Q||P)$表示Kullback-Leibler散度，这是一种衡量两个概率分布 Q 和 P之间距离的函数。

贝叶斯 PAC 界向来难以直观理解，因此我们简要回顾这一界，并对其合理性提供一些直观解释。

首先，请注意，上述界更容易被理解为一项统计实验。如果选择任意一个 P、 β、 δ，并从 P_d 中抽取 D^n 个样本，随后训练估计量分布 Q并计算此界，那么该界准确的概率为 $1-\delta$。

需要特别注意的是，由于这一界具有统计性质，因此对各个组成部分的定义顺序非常敏感。因此，不允许先针对某个估计器类计算此界，然后再选择 P来使此界最小；必须在不了解其他任何信息的情况下选定它。否则，只需选择 $P = Q$，令 $KL(P||Q) = 0$即可。当然，这是不允许的，因为 Q 取决于自身。

接下来请注意，先验分布 P 在这个界中表现得十分有趣。它只出现在项 $KL(Q||P)$ 中，而最终分布 Q 实际上并不依赖于它，这与通常由“先验”一词所表示的含义不同。相反，它更像是定义在 F 上的一种通用测度，因此可以与所有可能的分布互换。直观来说， P 充当参考分布，它是在训练 Q之前选定的。这使得 PAC 贝叶斯界能够利用领域知识，绕过传统统计方法的局限。 P 可以由了解相关领域的领域专家用户来定义。虽然必须在训练 Q 之前选定 P，但如果有人知道对于给定任务而言，哪些估计器是合理的，就可以选择合适的 P，从而为 $R(Q)$ 得到一个紧致的界。

从本质上说，传统方法背后的思路是：“如果分类器只能拟合有限数量的函数，并且 f^* 不属于这些函数，那么我们抽取到分类器能够拟合的样本的可能性就很低。”然而，对于贝叶斯 PAC 方法，其思路是：“如果我们在数据上训练一个估计器，那么经验风险就无法估计真实风险，因为估计器依赖于数据。不过，如果我们能在看到数据之前正确猜出最终估计量，那么这种依赖就不存在，因此经验风险就能很好地估计真实风险。

在这一表述中，人类专业知识参与了对数据训练后得到的最终估计量的猜测。因此，PAC 贝叶斯方法可以利用领域知识，为神经网络估计器提供可用的上界。

从分数的分子部分可以看出，上界随 KL项和经验风险的增加而增大。 β 决定了经验风险对该上界的影响程度，而 KL 项的影响则由因子 $\frac{1}{n}$ 决定，这印证了这样一种观点：观测到更大的 n 时，泛化会更加可靠。最后，上界随δ 以对数方式减小，这很直观，因为 δ 增加也会降低该上界成立的概率。

正如 [39] 的作者所强调的，也如 [66], 神经网络所展示的那样，即便经过正则化，神经网络也能近乎完美地拟合最荒谬的训练数据集合。因此，经验风险通常为（接近）0，因而可以忽略不计。所以，这个方程中有意思的部分是 KL项

它直接编码了相对于 P所假设的泛化误差。
具体来说，通过对 Q 和 P [4]进行 $\mathbb{G}$对称化，可以改进这一项。虽然 [39] 给出的实际证明较为繁琐，但背后的直觉相对简单。
对分布 Q进行对称化，意味着将 Q 从定义在 F 上的分布转换为定义在 F°上的分布Q° ，即对称化估计器，其中 Q° 定义如下：

$$Q^{\circ}(g) = \int_{f \in F, f^{\circ} = g} Q(f) df \tag{5.22}$$

也就是说， Q° 对在 $\mathbb{G}$对称化下相同的所有 f 的概率求和。直观地说，这样可以减小 Q° 和 F°之间的差异，因为 P和 Q 在 $\mathbb{G}$等价估计器上的分歧被积分消去了。

此外， [39] 的作者成功给出了一个具体界限，用以刻画对称化前后的KL项之差。这个所谓的对称化间隙定义如下：

$$\Delta(Q||P) = KL(Q||P) - KL(Q^{\circ}||P^{\circ}) = \mathbb{E}_{f \sim Q}[\log \frac{q(f)}{q^{\circ}(f^{\circ})}] \tag{5.23}$$

请注意，这印证了上述KL约简背后的直觉：KL项之差由 $q(f)$和$q^{\circ}(f^{\circ})$中对数差的期望决定，它表示对 $g \in$轨道(f)和$q(f)$中的所有 $q(gf)$的积分概率之差。

这一结果清楚地表明了 $\mathbb{G}$-不变性如何提升神经网络性能。5.24 中界限的泛化误差降低源于 KL 项的减小。KL 项的减小是通过缩小估计量空间实现的：从具体的估计器转向其对称化变体，从而改善了编码估计器表达能力的 KL 项。

$\mathbb{G}$-不变性与统计泛化性质

如果在看到数据之前，选择一个关于可能的估计器 f 的分布 P，选择标量值 $\delta \in [0, 1]$、$\beta > 0$ ，然后通过训练得到一个关于估计器 P的分布，那么以 $1 - \delta$的概率：

$$R(Q) \leq \frac{1 - e^{-\beta \hat{R}(Q, D^N) - \frac{1}{n}(KL(Q||P) + \log \frac{1}{\delta}}}{1 - e^{-\beta}} \tag{5.24}$$

如果事先知道最终估计量 f 和要近似的函数 f^* 都是 $\mathbb{G}$-不变的，那么通过对分布 Q 和 P 进行对称化（即对所有在对称化后等价的估计器取平均），可以改进这一界限。这种改进体现在 KL 项上，其值为：

$$\Delta(Q||P) = KL(Q||P) - KL(Q^{\circ}||P^{\circ}) = \mathbb{E}_{f \sim Q}[\log \frac{q(f)}{q^{\circ}(f^{\circ})}] \tag{5.25}$$

这适用于所有估计器；如果能对 Q 作出一个与最终估计量分布 P高度吻合的良好猜测，这些界限就可能相当紧。

[4]请注意，在任何有意义且假定具有$\mathbb{G}$-不变性的情境中， P可能早就应该对称地选定，因为可以在选择 P时利用这些领域知识。

5.5 总结

本节旨在以 𝔾-不变性为例，考察几何深度学习的思想对领域专用网络的影响。

有趣的是，可以证明，至少有三种不同的视角可以用来评估 𝔾不变性的影响，而每种视角对神经网络如何受益于融入 𝔾不变性都有不同的解释。

首先，可以证明， 𝔾不变性会改变基于梯度的优化所带来的训练过程。由于基于梯度的优化器的特性，梯度$\nabla_\theta f(x)$相似的点会以等效的方式发生变化。对于不受约束的神经网络，这意味着欧几里得邻居被认为具有相似的输出。对于 𝔾不变神经网络，优化过程则有所不同，它会考虑点的轨道之间的最小距离。对于大多数问题而言，这比单纯的欧几里得距离更能合理地衡量距离，也支持了这样一种观点：不变网络在大多数关心的任务上更容易训练，并且泛化能力更强。

这与全连接神经网络中的连续性概念有关：在全连接网络中，如果数据点 x 和 y 在欧几里得意义上彼此接近，那么它们的取值 $f(x) \approx f(y)$也相近。在 𝔾-不变网络中，只要对于任意 $g \in \mathbb{G}$，x 都接近 gy，这一点便成立。这表明， 𝔾-不变神经网络利用领域结构来拓展学习范围。

接下来从拓扑视角探讨神经网络，考察由网络分类函数所诱导的空间的拓扑性质。研究表明，只有当属于各个类别的数据点构成的拓扑空间较为简单、拓扑洞较少时，神经网络才能很好地近似这些函数；而要近似包含更多拓扑洞的复杂空间，则需要能力更强的网络 [6, 27]。虽然很难给出 𝔾-不变性如何改变拓扑复杂度的一般表达式，但强制施加 𝔾-不变性不会增加拓扑洞的数量，而且示例表明，复杂度有可能大幅降低。

最后，统计视角关注神经网络训练与评估过程中所做估计的统计性质。正如 [39], 的研究所示，风险估计可以改善神经网络所达到的泛化误差理论界，而不受所使用先验的影响。

这些结果表明，无论从什么角度看待深度学习，融入领域专用结构，尤其是不变性，几乎总能以可量化的方式提升泛化能力，这也佐证了将不变性融入神经网络的必要性。此外，其影响可能极其显著，例如学习一个涉及取模运算符的函数时便是如此。如果不在神经网络中强制施加不变性，那么即使是像 $x \bmod 3 > 1$ 这样简单的函数，对于大型神经网络而言也无法学习。因此，问题似乎并不在于将领域知识融入神经网络只是改进学习过程的一种方式；相反，对于几乎所有足够复杂且具有结构的领域而言，不具备领域知识的学习似乎注定会失败。

第 6 章

总结、结论与展望

在这项工作中，我们着手评估几何深度学习的理念。这是首次尝试从领域专用的视角，对深度学习架构进行结构化分类。我们试图借助几何深度学习的理念回答以下问题：“领域特定神经网络应该是什么样的？我们如何构建它们？它们会产生什么影响？”

我们首先界定了本研究的范围，将其限定为使用梯度下降作为训练算法的函数逼近问题。随后，我们考察了全连接神经网络的性质，也就是神经网络最初的非领域专用版本。我们发现，全连接网络具备若干理想估计器应有的便利性质：完备性，即在神经元和层数足够的前提下，能够以任意精度逼近所有目标函数；连续性与可调性，即估计器满足利普希茨连续性，且可通过增加层数或神经元来调节其连续程度。

不过，我们也指出，全连接神经网络无法利用领域中的附加结构，因此可以通过加入更多领域知识来改进。
在此基础上，我们借鉴几何深度学习的理念，进一步完善了领域特定神经网络的“愿望清单”。几何深度学习提出，领域特定神经网络应具备三项性质：第一，如果领域中存在一组不会改变分类结果的变换，那么最终估计量应对这些变换保持不变。例如，图同构可以改变图的表示形式，却不改变其基本性质。第二，如果领域中存在局部性的概念，那么局部结构在网络架构中应当联系得更紧密。第三，若假设真实函数 f^* 具有组合结构，那么我们的估计器也应遵循这一结构。

我们将这些想法与全连接神经网络的有用特性结合起来，得出了一份适用于特定领域神经网络的理想特性清单。随后，我们利用这份清单推导出 Deep Sets——一种针对集合领域的非常简单的神经网络架构。Deep Sets 的例子展示了如何利用这些特性，为一种已有且广泛使用的领域专用神经网络提供理论依据；同时也表明，即使知道领域专用神经网络应具备哪些特性，实际构建这样的网络仍然很困难。

基于深度集合，我们提出了几何深度学习蓝图。这是一种创建领域特定神经网络的方法，由几何深度学习提出。

随后，我们以图神经网络作为案例研究，应用这些理念，展示这些理念如何体现在更复杂、现代的领域特定神经网络中。我们再次证明，图神经网络符合我们对领域专用网络的要求。然而，在尝试将几何深度学习蓝图应用于图神经网络时，我们发现它有所欠缺，因为它留下了许多尚未解答的自由度。

最后，我们评估了对估计器施加关于一群 $\mathbb{G}$ 变换的不变性所产生的影响。由此可见，在适当条件下，强制估计器具有 $\mathbb{G}$-不变性能够显著提升其性能，尤其是在处理极其复杂的任务时。

6.1 结论

正如本文前文所述，领域专用神经网络架构是现代深度学习最重要的发展之一，似乎也是其取得成功的重要原因。以 $\mathbb{G}$-不变性为例，我们可以非常清楚地观察到这一效果。如第4章所示，在具有 $\mathbb{G}$-不变性这一性质的领域中不施加该不变性，甚至会使简单问题变得极其复杂；有理由认为，其他领域专用性质也可能产生类似的显著影响。

尽管领域专用神经网络的有效性毋庸置疑，但即使借助几何深度学习的思想，构建这类网络仍然似乎很困难。
将几何深度学习中的领域专用思想与全连接神经网络的便利特性相结合，可以得出一种出人意料地实用的领域专用神经网络刻画。本文表明，Deep Sets 和图神经网络不仅遵循几何深度学习的思想（ [9]中也对此进行了说明），还保留了全连接神经网络的优势。因此，它们完全满足了我们对领域专用神经网络的设计要求[1]。几何深度学习的思想在这里尤为有用：对于 Deep Sets 和图神经网络而言，不变性和局部性都是其架构十分显著的特征。

然而，就这类架构的实际构建而言，几何深度学习蓝图——即几何深度学习的构建方法——似乎存在很大不足。正如第 3 章所述，对于图神经网络而言，它给设计者留下了太多自由度；要构建一个真正有用的网络架构，仍然需要同样多的专业知识，和依据几何深度学习提出的要求手动构建相比并无二致。

因此，几何深度学习似乎准确地描述了一些有用的领域属性，而领域特定神经网络架构可以利用这些属性；它也说明了估计器的哪些属性能够用于利用这些属性。然而，在领域特定神经网络的构建方面，显然仍有许多工作要做。

此外，值得注意的是，几何深度学习所采用的领域属性，主要针对传统上属于几何深度学习范畴的领域。尤其是，它们隐含地重点关注类似于图像识别的领域，即输入是由实数构成的向量 $x \in \mathbb{R}^k$ ，且向量各个元素之间存在某种关系，

[1]对于图神经网络，它们确实违反了完备性，因为它们无法逼近所有目标函数。然而，考虑到判定图同构的复杂度，这几乎是可以预料的。

如上文所述。对于这些想法，几何深度学习应该能发挥良好作用。不过，对于不同的领域，这些想法可能并不适用。例如，在没有实值输入的离散域中，深度学习仍依赖变通方法，并使用连续模型来近似离散过程，这当然并非最优。因此，尽管几何深度学习做出了一些非常有用的贡献，但其应用仍局限于深度学习较为常见的用例，因此未必全面。

这当然是很自然的，因为几何深度学习只是朝着结构化设计领域专用神经网络架构这一方向迈出的第一步，目前不能指望它回答所有可能的问题。

6.2 展望

尽管几何深度学习的理念被证明是创建领域专用神经网络的良好开端，但仍有大量研究工作有待开展。

首先，几何深度学习并不能解决神经网络所面临的所有问题。事实上，几何深度学习仅适用于具有实值域且具备某些额外结构的领域，例如图像和带有实值节点标签的图。在实践中，深度学习应用于许多完全离散的领域。例如，在语言识别中，单词是离散对象。正如第5章所示，几何深度学习无法改变这样一个事实：神经网络是在欧几里得空间上运行的，因此难以处理离散任务。要以概念上合理的方式将神经网络扩展到完全离散的领域，还需要开展进一步研究。

其次，尽管对于几何深度学习适用的领域而言，它确实提供了一套有用的标准，用于界定何为优秀的领域专用神经网络架构，但这些神经网络架构的实际构建仍然十分困难，而几何深度学习蓝图也未能对此提供充分的结构化指导。因此，我们需要进一步研究更具结构化的方法，以构建领域专用神经网络架构。

附录 A

附录

A.1 A.群与对称性

[9]中最重要的思想之一，涉及某些数据形式中存在的对称性，即对数据对象进行变换，同时保持该数据的重要特征不变。例如，在图中，如果重新排列某个图的节点，那么对于大多数感兴趣的任务，所得的图仍具有等价的表现。类似地，在图像处理中，如果将图像平移一定数量的像素（假设图像周围有足够的空白区域），那么图像所表示的对象应保持不变。

这些变换可以用群来进行数学描述，更具体地说，可以用对称群[5],的子群来描述，从而提供必要的数学泛化。
形式上，群定义为一个元组 $\langle \mathbb{G}, \cdot \rangle$ ，其中包含一个集合 $\mathbb{G}$，以及与之关联的一个二元运算 $\cdot: \ \mathbb{G} \times \mathbb{G} \to \mathbb{G}$ 满足以下三个性质 [1]。首先， $\mathbb{G}$ 必须满足结合律:

$$a \cdot (b \cdot c) = (a \cdot b) \cdot c \tag{A.1}$$

结合律是代数结构中经常假定的一种性质，它允许省略括号。此外，更有意思的是，群必须具有一个幺元 $id \in \mathbb{G}$ ，使其他元素保持不变:

$$id \cdot x = x \cdot id = x \tag{A.2}$$

请注意，单位元既是左单位元，也是右单位元；也就是说，无论将其用于运算的左侧还是右侧，其作用都同样是单位元。这是单位元的特殊之处，因为群通常不满足交换律（满足交换律的群称为阿贝尔群）。最后，群中的每个元素都必须有逆元。对于每个 $g \in \mathbb{G}$ ，都存在 g^{-1} ，使得

$$g \cdot g^{-1} = g^{-1} \cdot g = id \tag{A.3}$$

由此定义可直接推出， g^{-1} 的逆元仍是 g ，因此$(g^{-1})^{-1} = g$。此外，如果 $\langle \mathbb{G}, \cdot \rangle$ 是一个群，那么任何同样构成群 $\langle G, \cdot \rangle$ 且运算为 $\cdot$的子集 $G \subseteq \mathbb{G}$，都称为 $\mathbb{G}$的子群。

As p如前所述，在本文的语境中，最重要的群是对称群 p,

[1]提到群 $\langle \mathbb{G}, \cdot \rangle$ 时，我们通常只会提及集合 $\mathbb{G}$ ，并假定相应的算子可根据上下文明确判断。

$S(\Omega)$ 及其子群，即某个集合 Ω 上的置换群。对称群定义为

$$S(X) := \langle \{\pi \mid \pi : \Omega \to \Omega, \pi \text{ bijective}\}, \circ \rangle \tag{A.4}$$

由 X 到其自身的所有可能双射（或置换）构成的集合，配备函数复合这一二元运算。容易验证，这满足群的条件：两个双射的复合仍是双射（封闭性）；恒等函数 $f(x) = x$ 是双射（单位元）；而双射函数本质上都可逆（逆元）。对称群一直是许多既有研究的对象；它们，或者更具体地说，其子群——置换群——涵盖了本研究所关注的几乎所有群，并且还具有一些有趣的性质。

首先，对于有限集合 $\Omega = \{x_1, x_2, ..., x_n\}$，对称群由相邻对换构成的集合生成。相邻对换 $\sigma_i : \Omega \to \Omega$ 是一种双射，仅交换 Ω 中第 i 个和第 i+1 个元素，其定义如下：

$$\sigma_i(x_i) = x_{i+1} \tag{A.5}$$

$$\sigma_i(x_{i+1}) = x_i \tag{A.6}$$

$$\sigma_i(x) = x \text{ if } x \neq x_i, x \neq x_{i+1} \tag{A.7}$$

因此，对于 $S(\Omega)$ 上的任意同胚，我们只需考察它如何作用于 $S(X)$ 的生成集，就能推导出其完整行为。
最后，让我们考察一下前文提到的数据的一些对称性，并确认它们对应于 $S(X)$ 的某个子群。

首先，让我们考察平移若干像素后的图像。我们将图像表示为像素的集合 $P = \{1, 2, ..., n\}^2$。向右平移则由这样一个置换生成：它将每个像素向右移动一个像素[2]

$$\pi_r((i, j)) = (i + 1 \mod n, j) \tag{A.8}$$

其他方向的情况也等价。于是，所有平移都由每个方向上的单像素平移生成。很容易验证，这个集合满足群的条件，因此是 $S(P)$ 的一个子群，也就是一个置换群。

其次，考虑图 $G = (V, E)$ 及其节点重新排序的情况。这里考虑该集合的所有置换，因此我们关注的子群实际上是 $S(V)$。这两者都是 [9] 中所讨论的对称性的典型例子。

A.1.1 轨道与等价类

介绍了对称群后，我们现在来研究它及其子群在集合 Ω^3 上所诱导的结构。

如前所述，我们非常关注对称性的概念，即保持感兴趣的性质不变的变换，这些变换被视为某个置换群的元素，也就是对称群的一个子群。因此，我们假设以某种方式

[2]为确保双射性，我们假设像素呈周期性排列，即第 n+1 个像素再次与第一个像素等价。[3]请注意，本节的大部分内容都适用于作用于某个领域上的任意群，但我们主要关注对称群及其子群。

集合 Ω 与其图像 $p(X)$ 之间存在某种等价关系。这被形式化为 x 的轨道：

$$\text{orbit}(x) = \{gx \mid g \in \mathbb{G}\} \tag{A.9}$$

x 的轨道包含所有由 $\mathbb{G}$变换得到的 x。例如，考虑作用于二维图像的旋转群。此时，一幅图像的轨道由它所有可能的旋转版本组成。在这个例子中也很容易看出，对于一个满足 $y \in$orbit(x)的元素，有orbit(y) = orbit(x），因为一幅图像 y 的旋转版本（而该图像本身是由 x 旋转得到的）同样也是 x的旋转版本。这普遍成立，因为：

$$y \in \text{orbit}(x) \implies y = gx \Leftrightarrow g^{-1}y = x \tag{A.10}$$

因此，每个 $z \in$orbit(x)都可以写成

$$z = g'x = (g'g^{-1})y \in \text{orbit}(y) \tag{A.11}$$

逆命题可以用类似的方式证明。
将轨道视为一种关系，上述内容清楚地表明，轨道关系

$$x \cong_{\mathbb{G}} y \Leftrightarrow \text{orbit}(x) = \text{orbit}(y) \tag{A.12}$$

是对称的。类似于上述证明，可以证明它具有传递性和自反性，从而表明这些轨道确实构成等价关系。因此，在本文余下部分，同一轨道内的元素将称为 $\mathbb{G}$等价。

A.2 B.离散域中的神经网络

神经网络，即使是 [9], 中所描述的领域专用网络，也都是为本质上连续的领域而设计的。在其中所讨论的所有领域中，估计问题都可以表述为：

估计连续函数 f^*： $\mathbb{R}^k \to \mathbb{R}^j$，其中某个 k, $j \in \mathbb{N}$。此外，假设存在一些已知性质 P，且 f^* 满足这些性质。

估计问题始终是一个实值估计问题，其中 f^* 的定义域和值域都是实数集。这里的领域专用性由假定 f^* 所具有的性质 P 来刻画。然而，在深度学习的许多应用场景中，这还不够。

以图像识别为例。通常，在图像识别中，神经网络的任务是为每幅图像分配一个标签。图像可以表示为实数向量，因此与上述估计问题兼容。然而，图像的标签是离散且无序的。因此，它们无法用实数值表示；如果这样表示，就会给标签附加实际上并不存在的距离和顺序概念。这个估计问题具有离散的陪域，因此不能直接用于全连接神经网络和梯度下降。例如，假设将标签“狗”表示为 1，将“猫”表示为 2，将“汽车”表示为 3。再假设一幅图像的标签是“汽车”，而神经网络输出的值为 0.9，于是预测结果为“狗”。此时，梯度下降以无穷小的学习率将网络的预测结果向 3 微幅调整

。如果学习率足够小，神经网络现在对同一幅图像的输出就会变成 1。这样一来，它的输出反而更接近“狗”，而“狗”正是最初错误估计的结果。因此，梯度下降并不适用于对输出进行简单的实数值编码。

考虑语言处理任务。在语言处理中，输入是单词。单词本质上也是离散对象，因此需要估计一个关于单词的函数 $\to C$。这使得学习变得困难。本书中介绍的所有神经网络都基于连续性假设进行学习。这意味着，它们假设如果某个数据点 x 具有输出 o，那么它的欧几里得邻域内也会有相似的输出。当然，对于单词而言，并不存在可直接用于学习的欧几里得距离。

我们刚才介绍了两个棘手的估计问题示例，一个涉及离散输入，另一个涉及离散输出。语言处理和图像处理都是深度学习的重要应用场景 [44, 63]。本节将概述在输入或输出为离散值时，使用深度学习进行函数逼近所面临的困难，以及深度学习中通常用来克服这些困难的启发式方法。

A.2.1 离散输入

假设我们要估计一个函数 $f^* : W \to \mathbb{R}$，其中 W 是某个离散值集合。本节将以单词作为 W 的示例，但请注意，本节中的所有内容都可以应用于其他场景。如前所述，按照神经网络通常的学习方式，这会使学习变得非常困难。因此，要使用深度学习，必须解决两个挑战：首先，输入必须表示为实数向量。从根本上说，只有满足这一点，才能使用基于梯度的优化。其次，这种表示必须适用于神经网络学习。作为基线，我们要求：当且仅当我们假设从 x 学到的知识能够迁移到 y 时，输入 x 和 y 在欧几里得意义上才应彼此接近。

因此，对于离散输入，要有意义地使用深度学习进行函数逼近，唯一的方法是构造表示函数 $r : W \to \mathbb{R}^d$，使其满足某个 d，并附加约束 $r(w) \approx r(v)$，当且仅当我们假设 $f^*(w) \approx r(v)$。这些表示称为 嵌入，并用于各种机器学习应用，以解决离散输入问题 [38, 25]。

有多种方法可以做到这一点。对于较小的集合 W，可以手动决定哪些输入单词应该相似，哪些不应该相似。给定一组应当相似的单词元组：$S = \{(w_1, v_1) ..., (w_n, v_n)\}$，只需解决以下优化问题：

$$\sum_{(w,v) \in S)} ||r(w), r(v)|| \to \max \tag{A.13}$$

以获得每个单词的合适表示。有了这些表示后，就可以创建如下形式的估计器

$$f(w) = f'(r(w)) \tag{A.14}$$

其中 f' 是一个神经网络 $\mathbb{R}^d \to \mathbb{R}$，并使用基于梯度的优化。
这是词嵌入最简单的版本，只适用于较小的单词集合。例如，如果要涵盖整个英语，似乎不可能手动

创建一份详尽清单，列出所有彼此相似的单词。
对于更大的输入集合 W，需要采用另一种方式来定义哪些单词应当彼此相似。一个广泛使用的例子是 word2vec [41]。
与前面的词嵌入示例不同，word2vec 专门用于文本中的单词。由于这种方法的复杂度较高，我们将只概述其背后的思想。
我们首先为每个单词 w 分配任意表示 $r(w)$，然后通过迭代不断改进这些表示。假设我们拥有大量示例文本。接着，我们从示例文本中随机选取固定的奇数长度 k 的单词串。设这样的单词串为

$$w_1 \quad w_2 \quad w_3 ... w_{mid} ... \quad w_n \tag{A.15}$$

其中 w_{mid} 位于该单词串的中心。然后，我们最大化 w_{mid} 的表示与其所有周围单词之间的余弦相似度：

$$\sum_{i=1}^{n} \omega_i \frac{r(w_i)^T r(w_{mid})}{||r(w_i)^T r(w_{mid})||} \rightarrow \max \tag{A.16}$$

其中，权重 ω_i 通常基于单词 w_i 与 w_{mid}之间的距离确定。同样地，一个

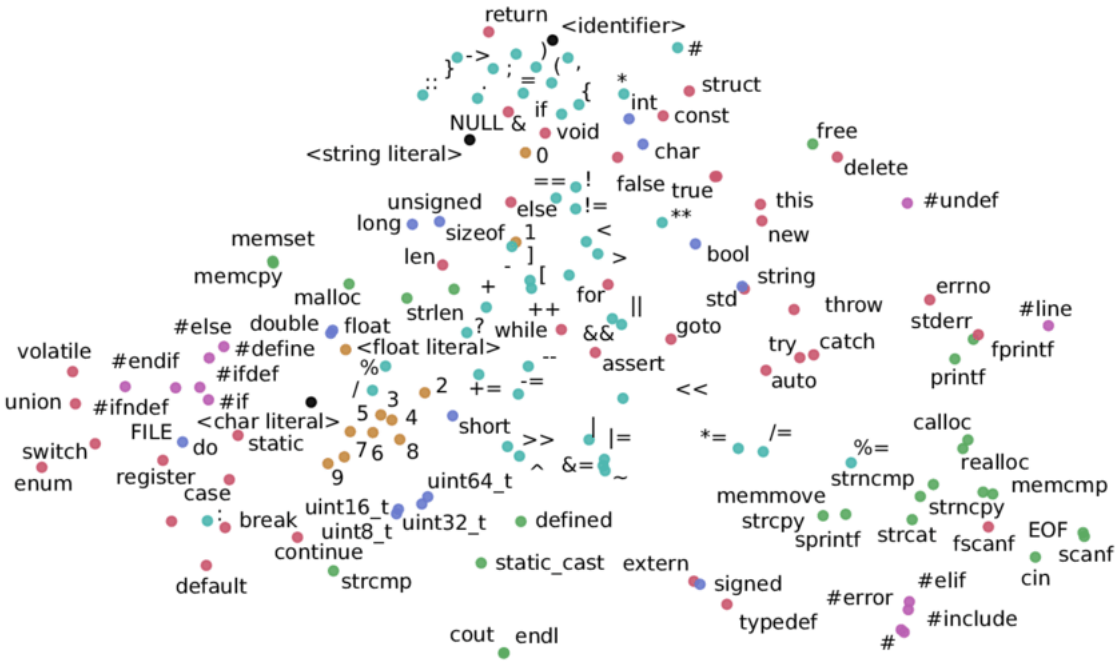


图 A.1： 语言 C++ 的词嵌入示例，由 word2vec 计算并投影到二维空间，图像取自 [30]。单词是程序语句。请特别注意哪些语句彼此接近：例如，”return”和”标识符”经常一起书写；而加法、减法和比较运算符等操作数则各自聚集在一起。

可以添加额外的项，对 $r(w_i)$ 与所有不在其邻域中的其他单词之间的相似度进行惩罚。这个过程会反复迭代。word2vec 实现了两件事：首先，经常同时出现的单词具有相似的表示。这由上面的方程直接强制实现。其次，出现在相似句子中的单词也具有相似的表示。这通过相似度的传递性来实现。如果我们认为经常

同时出现且可以相互替换的单词在 f^* 方面相似，那么这种表示就是合理的。当然，word2vec 算法也可以通过添加额外的约束来扩展。

这种方法在词嵌入中非常流行，但无法直接扩展到其他领域。因此，尽管嵌入方法通常能为深度学习领域中的离散输入提供良好的解决方案，但创建良好的嵌入仍然颇具挑战，而且良好的嵌入很大程度上是领域专用的。

A.2.2 离散输出

现在，假设我们需要估计一个函数 $f^*: \mathbb{R}^k \to L$ where $L = \{l_1, l_2, ..., l_n\}$，其中 $L = \{l_1, l_2, ..., l_n\}$ 是某个有限标签集合。在这里，同样的问题依然存在：若要使用神经网络和基于梯度的优化，输出领域必须是某个实值空间 $\mathbb{R}^d$。因此，我们需要构建一个能够输出实数值的模型，并找到一种将这些输出转换为 L 中标签的方法。与离散输入的情况类似，这需要以一种适用于连续模型的方式来构建；也就是说，我们可以合理地假设，输入中的微小欧几里得变化与输出中的微小欧几里得变化相关联。

实现这一点最常用的方法，是估计 f^* 的概率版本。在这里，我们不直接估计 f^*，而是估计概率分布 $P(f^*(x) = l_i)$，它表示数据点 x 具有标签 l的概率。我们的估计器 f 将数据点 x 作为输入，并为每个标签 $l \in L$返回一个实数值：

$$f : \mathbb{R}^k \to \mathbb{R}^n \tag{A.17}$$

其中 $n = |L|$。我们将 $f(x)$ 的第 i 个分量解释为 x 具有标签 l_i的概率：

$$P(f^*(x) = l_i) := f(x)_i \tag{A.18}$$

在我们的设定中， f^* 是一个确定性函数。因此，这符合概率的贝叶斯解释，描述的并非真正不确定的事件，而是对 $f^*(x)$ 的值所持的主观置信程度。

当然，这意味着 $f(x)$ 必须表示一个合法的概率分布：

$$\sum_{i=1}^{n} f(x)_i = 1 \tag{A.19}$$

$$f(x) > 0 \tag{A.20}$$

对所有 x 都成立。大多数估计器默认情况下并不满足这些性质。常见的解决方案是将 softmax 函数作为最后一个操作来构造 f [24]。softmax 函数 $\text{softmax} : \mathbb{R}^d \to \mathbb{R}^d$ 定义如下：

$$\text{softmax}(a_1, ..., a_n)_i = \frac{e^{a_i}}{\sum_{j=1}^{n} e^{a_j}} \tag{A.21}$$

通过应用指数函数，确保结果向量的每个分量均为正；再对其进行归一化，使其求和为一。因此，如果构建一个如下形式的估计器

$$f(x) = \text{softmax}(f'(x)) \tag{A.22}$$

使用 $f': \mathbb{R}^k \to \mathbb{R}^n$，可以得到一个估计器，它会输出 x 标签上的规范概率分布。此外，softmax 函数作用于 x 的指数，因此 f'

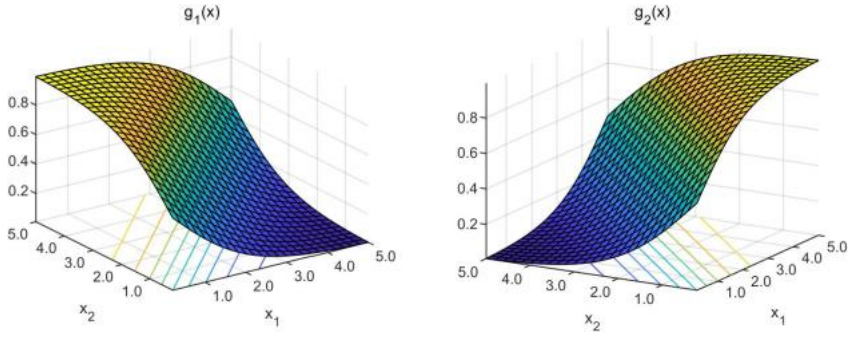


图 A.2： 针对两个输入绘制的 softmax 函数。

必须输出对数标签概率。这很方便，因为对数空间允许在概率之间定义非常自然的距离。

当然，这也意味着需要一个新的损失函数。如第2章所述，我们将模型 f 的负似然用作损失函数。对于数据集 $D = \{(x_1, y_1).....,(x_n, y_n)\}$， f 的对数似然为

$$\begin{aligned}\log(P(D|f)) &= \sum_{i=1}^{n} \log(P(f^*(x_i) = y_i)) \\ &= \sum_{i=1}^{n} \log(f^*(x_i)_{y_i})\end{aligned}$$

其中， $f^*(x_i)_{y_i}$ 是 $f^*(x_i)$ 中与 y_i 对应的分量。这个损失函数是带有确定性标签的交叉熵损失，是分类任务中最常用的损失函数 [24]。

总结：为处理离散输出，神经网络不再用于逼近 $f^*(x)$，而是用于逼近 $f^*(x)$ 上的概率分布。这是通过构建一个估计器 $\mathbb{R}^k \to \mathbb{R}^n$ 并对输出应用 softmax 函数来实现的。这样一来，估计问题就变成了连续问题，并且可以使用基于梯度的优化算法。最终的概率分布可用于实际分类，方法是选取概率最高的标签。

这种构造存在一些缺陷。首先，它用概率来编码关于一个本身并非概率性的函数的不确定性。这对应于概率的贝叶斯解释，而这种解释本身就存在争议。其次，这种方法仅适用于有限数量的标签。任何无限离散空间，例如全体整数，都无法用这种方式处理，而这些空间很难用于神经网络。第三，概率 $P(f^*(x))$ 实际上并不存在，完全是主观的。因此，待估计的实际函数并未得到明确定义，这使得我们很难判断 $P(f^*(x))$ 是否适合用神经网络进行估计，以及对其进行估计能否很好地泛化。

图目录

2.1 梯度下降在一个简单的二维函数示例上运行。请注意基于梯度的优化所具有的贪婪特性：它总是趋向最近的局部最小值，并沿着局部改善幅度最大的方向迈步。示例取自 [13] .4

2.2 线性模型估计线性函数（左）和二次函数（右）。第一个估计任务完成得很好，第二个则不然。当然，这是因为模型中存在的归纳偏置假设 f^* 呈线性变化，而实际上并非如此.

9

2.3 全连接神经网络示例。请注意其分层架构，以及边严格朝向前方的方向。还可以看到，每个神经元都与后续层中的所有神经元相连。插图取自 [31] 。 . . . . . .

14

2.4 ϕ的一些最常用激活函数选择。请注意，原则上，只要激活函数可微且最好是单调的，就可以选择任意函数。撰写本文时，修正线性单元（ReLU）是激活函数的事实标准选择 [24]，因此本文所有示例均采用该函数 .

14

2.5 Sigmoid 激活函数 σ 是利普希茨连续的，其 $L(\sigma) = \frac{1}{4}$ 。灰色区域表示在$x = 0$处由利普希茨常数决定的函数取值界限。1 4 的利普希茨常数规定，函数的变化速率最多只能达到1 4。请注意，该函数始终不会超出以这种方式确定的边界.

16

2.6 使用不同层数的神经网络学习相同任务。每层都使用相同的数据点。每个角上各有一个数据点（即（x， y），其中$|x| = |y| = 4$.

17

2.7 两个具有 4 层的神经网络，它们在同一任务上接受训练，但使用了不同的 L2 损失权重值。λ 的值越高，边权重就越低，因此利普希茨常数也越低。这些网络经过训练，将所示区域的所有角点都映射为 0，并将中点映射为 1，即 (0，0)。两个网络都很好地完成了任务，但方式不同。网络 a（ λ 的值较低）生成了一个平坦表面，并在 (0，0) 周围形成一个尖峰，其利普希茨常数非常高。网络 b 则选择了更平坦、更连续的版本。当然， b 的利普希茨常数要低得多。2

1

2.8 [61]中的一些示例，展示了全连接神经网络在外推非线性函数时的失败。神经网络的预测以黑色表示，实际函数以灰色表示，训练数据集以蓝色表示。请注意，第一个任务本质上是二次函数，因此在训练数据范围之外，网络的预测开始偏低。第二个任务是周期函数，神经网络完全无法对其进行近似，而是将训练数据中一个周期的趋势线性延续下去。后两个任务也有类似发现。. 23

2.9 展示神经网络线性外推稳定性的直方图。对同一个简单示例问题，使用同一个神经网络进行了 1000 次训练，其中给定了$f^*((0，0)) = 1$和 $f^*(\pm 4，\pm 4) = 0$，最终查询了远离所有训练数据点的点 (50，50)。请注意， f(50，50)的分布以 12.5 为中心，这正是线性外推所预期的值。得到的曲线略微偏向右侧，这很可能是初始化偏差造成的.2 4

3.1 领域中的对称性示例。在国际象棋中，我们的目标是近似这样一个函数：它将国际象棋棋盘状态映射到该局面下的最佳走法。 f^* 当然是未知的，但根据国际象棋领域知识可以清楚地看出，最左侧图像中黑方的最佳走法，就是最右侧图像中白方的最佳走法，因为国际象棋关于颜色是对称的（轮到哪一方走棋除外）。. . . 30

3.2 卷积神经网络作用于数字“8”图像时的第一层（为清晰起见，省略了非线性变换）。卷积神经网络从输入中生成三个特征图：左侧的特征图大致捕捉边界，第二个大致捕捉水平线，第三个则捕捉垂直线。这与标准图像识别中的方法相符，也展示了卷积神经网络的构建方式能够轻松近似标准图像识别实践。图片来源： [62].31

3.3 均值滤波器的应用示例。此处的均值滤波器应用于原始图像 x ，而非神经网络中通常使用的某个函数 $\phi(x)$ 的输出，以便更清楚地展示其工作原理。此处的补丁大小为 10，即每一步都会汇总 10*10 像素的邻域。图片来源：[59]. 32

3.4 作用于含四个元素的集合上的深度集合架构 $x = \{x_1, x_2, x_3, x_4\}$。对每个元素应用 ϕ ，然后将结果求和，得到最终结果 $f(x)$。注意，如果在该架构中交换 x_2 和 x_3 ，最终结果将保持不变，如此类推.....37

4.1 在两个同构图上运行 Weisfeiler-Lehman 测试。由于两个图同构，它们得到相同的签名“9,9,8,7,7”。对于这两个图，左侧展示了每一步的着色，右侧展示了表示各节点邻域的多重集。请注意，在第一次迭代中，所有节点的颜色都相同；在第二次迭代中，则根据出度区分节点。示例摘自 [7] . 45

4.2 WL 检验中两个不可区分的图的示例。注意，这两个图本质上都是循环的，但由于 WL 检验具有局部性，无法区分长度为三的圆和长度为六的圆。示例摘自 [15]。..... 47

5.1 二分类问题中的若干点 [33]。蓝色点属于类别 C^+，黄色点属于 C^-。对于神经网络而言，这个空间在拓扑上相对容易学习，因为子空间 $C^+ \subset \Omega$ 和子集 $C^- \subset \Omega$ 都只有一个拓扑孔. 62 5.2 $f^*(x) = x \bmod 3 > 1$ 的图像。请注意，在 $f^*(x) = 1$ 的区域之间存在无限多个间隙。为了更轻松地绘制这个函数的图像，我们用 1 表示布尔值“true”，用 0 表示“false”.62 A.1 C++语言的词嵌入示例，通过 word2vec 计算并投影到二维，图像摘自 [30]。单词为程序语句。特别注意彼此接近的语句：例如，“return”和“identifier”经常一起出现；而加法、减法和比较运算符等操作数则各自聚为一簇。 77 A.2 两个输入对应的 softmax 函数图像.79

参考文献

[1] *Alom, Md Zahangir, Tarek M Taha, Chris Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Mahmudul Hasan, Brian C Van Essen, Abdul AS Awwal* 和 *Vijayan K Asari*：深度学习理论与架构的前沿综述。*Electronics，8(3):292，2019*年。[2] *Alpaydin, Ethem*：机器学习。*De Gruyter Oldenbourg，2019*年。[3] *Baldi, Pierre* 和 *Peter J Sadowski*：理解丢弃法。《神经信息处理系统进展》，*26:2814–2822，2013*年。[4] *Barnard, Etienne* 和 *LFA Wessels*：神经网络分类器中的外推与插值。*IEEE* 控制系统杂志，*12(5):50–53，1992*年。[5] *Baumslag, Benjamin*、*Bruce Chandler* 和 *Schaum's outline Series*：群论：理论与问题。*McGraw-Hill* 纽约，*1968*年。[6] *Bianchini, Monica* 和 *Franco Scarselli*：论神经网络分类器的复杂度：浅层架构与深层架构的比较。*IEEE* 神经网络与学习系统汇刊，*25(8):1553–1565，2014*年。[7] *Bieber, David*：*Weisfeiler-Lehman* 同构测试。*https://davidbieber.com/post/2019-05-10-weisfeiler-lehman-isomorphism-test/*。访问日期：*2021*年*8*月*16*日。[8] *Bietti, Alberto* 和 *Julien Mairal*：群不变性、变形稳定性与深度卷积表示的复杂度。《机器学习研究杂志》，*20(1):876–924，2019*年。[9] *Bronstein, Michael M, Joan Bruna, Taco Cohen* 和 *Petar Veličković*：几何深度学习：网格、群、图、测地线与规范。*arXiv* 预印本 *arXiv:2104.13478，2021*年。[10] *Bronstein, Michael M, Joan Bruna, Yann LeCun, Arthur Szlam* 和 *Pierre Van-dergheynst*：几何深度学习：超越欧几里得数据。*IEEE* 信号处理杂志，*34(4):18–42，2017*年。[11] *Carslaw, Horatio Scott*：傅里叶级数与积分理论导论，第*1*卷。*Macmillan，1921*年。[12] *Catoni, Olivier*：*PAC-*贝叶斯监督分类：统计学习的热力学。*arXiv* 预印本 *arXiv:0712.0248，2007*年。

[13]*Cerigo, Daniel*：论为何仍需要梯度下降。*https://medium.com/@DBCerigo/on-why-gradient-descent-is-even-needed-25160197a635*。访问日期：*2021-08-29*。[14] *Dai, Zihang, Hanxiao Liu, Quoc V Le* 和 *Mingxing Tan*：*CoAtNet*：结合卷积与注意力，适用于各种规模的数据。*arXiv* 预印本 *arXiv:2106.04803*，*2021*年。[15] *Damke, Clemens, Vitalik Melnikov* 和 *Eyke Hüllermeier*：一种新颖的高阶 *Weisfeiler-Lehman* 图卷积。载于《亚洲机器学习会议》，第*49–64*页。*PMLR*，*2020*年。[16] *Dayan, Peter* 和 *Yael Niv*：强化学习：好的一面、坏的一面与丑陋的一面。《神经生物学当前观点》，*18(2):185–196*，*2008*年。[17] *De Boor, Carl* 和 *Amos Ron*：论多元多项式插值。《构造性逼近》，*6(3):287–302*，*1990*年。[18] *Eliason, Scott R*：极大似然估计：原理与实践。第*96*号。*Sage*，*1993*年。[19] *Fortin, Scott*：图同构问题。*1996*年。[20] *Gasca, Mariano* 和 *Thomas Sauer*：多变量多项式插值。《计算数学进展》，*12(4):377–410*，*2000*。[21]*Gilmer, Justin, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals* 和 *George E Dahl*：用于量子化学的神经消息传递。载于《国际机器学习会议》，第*1263–1272*页。*PMLR*，*2017*年。[22] *Glorot, Xavier* 和 *Yoshua Bengio*：理解深层前馈神经网络训练的困难。载于《第十三届国际人工智能与统计会议论文集》，第*249–256*页。*JMLR* 研讨会与会议论文集，*2010*年。[23] *Glorot, Xavier* 和 *Yoshua Bengio*：理解深层前馈神经网络训练的困难。载于 *Teh, Yee Whye* 和 *Mike Titterington*（编）：《第十三届国际人工智能与统计会议论文集》，《机器学习研究论文集》第*9*卷，第*249–256*页，奇亚拉古纳度假村，意大利撒丁岛，*2010*年*5*月*13*日至*15*日。*PMLR*。[24] *Goodfellow, Ian, Yoshua Bengio* 和 *Aaron Courville*：深度学习。麻省理工学院出版社，*2016*年。*http://www.deeplearningbook.org*。[25] *Goyal, Palash* 和 *Emilio Ferrara*：图嵌入技术、应用与性能：综述。《基于知识的系统》，*151:78–94*，*2018*。[26] *Gupta, Jatinder ND* 和 *Randall S Sexton*：比较反向传播与遗传算法在神经网络训练中的效果。《欧米伽》，*27(6):679–684*，*1999*年。[27] *Guss, William H* 和 *Ruslan Salakhutdinov*：论利用代数拓扑刻画神经网络的容量。*arXiv* 预印本 *arXiv:1802.04443*，*2018*。

[28] *Habermann, Christian* 和 *Fabian Kindermann*：多维样条插值：理论与应用。《计算经济学》，*30(2):153–169*，*2007*年。[29] *Haley, Pamela J* 和 *Donald Soloway*：多层前馈神经网络的外推局限性。载于 *[*《 *1992] IJCNN*国际神经网络联合会议论文集》，第*4*卷，第*25–30*页。*IEEE*，*1992*年。[30] *Harer, Jacob A, Louis Y Kim, Rebecca L Russell, Onur Ozdemir, Leonard R Kosta, Akshay Rangamani, Lei H Hamilton, Gabriel I Centeno, Jonathan R Key, Paul M Ellingwood* 等：利用机器学习进行自动化软件漏洞检测。*arXiv*预印本 *arXiv:1803.04497*，*2018*。[31] *Haykin, Simon*：《神经网络与学习机器》，第*3*版。培生印度教育，*2010*年。[32] *Hornik, Kurt, Maxwell Stinchcombe* 和 *Halbert White*：多层前馈网络是通用逼近器。《神经网络》，*2(5):359–366*，*1989*年。[33]*Johnson, Jesse*：深层窄型神经网络并非通用逼近器。*arXiv*预印本 *arXiv:1810.00393*，*2018*。[34] *Kingma, Diederik P* 和 *Jimmy Ba*：*Adam*：一种随机优化方法。*arXiv*预印本 *arXiv:1412.6980*，*2014*年。[35] *Kipf, Thomas N* 和 *Max Welling*：利用图卷积网络进行半监督分类。*arXiv*预印本 *arXiv:1609.02907*，*2016*年。[36] *Kolmogorov, Andre Nikolaevich*：关于通过较少变量的连续函数的复合表示多个变量的连续函数。美国数学会，*1961*年。[37] *LeCun, Yann, Yoshua Bengio* 等：用于图像、语音和时间序列的卷积网络。《脑理论与神经网络手册》，*3361(10):1995*，*1995*年。[38] *Li, Yang* 和 *Tao Yang*：用于理解自然语言的词嵌入：综述。载于《大数据应用指南》，第*83–104*页。*Springer*，*2018*。[39]*Lyle, Clare, Mark van der Wilk, Marta Kwiatkowska, Yarin Gal* 和 *Benjamin Bloem-Reddy*：神经网络中不变性的益处。*arXiv*预印本 *arXiv:2005.00178*，*2020*年。[40] *Manurangsi, Pasin* 和 *Daniel Reichman*：训练*ReLU(s)*的计算复杂性。*arXiv*预印本 *arXiv:1810.04207*，*2018*。[41] *Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S Corrado* 和 *Jeff Dean*：单词和短语的分布式表示及其组合性。载于《神经信息处理系统进展》，第 *3111–3119*页，*2013*年。

[42] Minka ，Thomas：贝 y叶斯线性回 g归。技术报告 port，Citeseer ，2000。

[43]Morris, Christopher, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan 和Martin Grohe：*Weisfeiler* 和 *Leman* 的神经化：高阶图神经网络。载于*AAAI*人工智能会议论文集，第 33 卷，页码 4602–4609，2019年。

[44] Pak, Myeongsuk和 Sanghoon Kim： 图像识别中的深度学习综述。载于*2017*年第四届计算机应用与信息处理技术国际会议（*CAIPT*），第 1–3 页。IEEE，2017年。

[45] Park, Sejun, Chulhee Yun, Jaeho Lee 和 Jinwoo Shin：通用逼近的最小宽度。arXiv 预印本 arXiv:2006.08859，2020年。

[46] Pinkus, Allan： 神经网络中多层感知机模型的逼近理论。Acta Numerica，8:143–195，1999年。

[47] Polson, Nicholas G和 James G Scott： 全局收缩，局部作用：稀疏贝叶斯正则化与预测。贝叶斯统计学，9(501-538):105，2010年。

[48] Prates, Marcelo，Pedro HC Avelar，Henrique Lemos，Luis C Lamb和 Moshe YVardi： 学习求解*NP*完全问题：用于判定型旅行商问题的图神经网络。载于*AAAI*人工智能会议论文集，第33卷，第4731–4738页，2019年。

[49] Prechelt, Lutz： 提前停止——但何时停止？载于《神经网络：技巧与诀窍》，第 55–69 页。Springer，1998年。

[50] Ronneberger, Olaf， Philipp Fischer和 Thomas Brox： 用于生物医学图像分割的 *U-Net* 卷积网络。载于国际医学图像计算与计算机辅助干预会议，第 234–241 页。Springer，2015年。

[51] Ruderman, Avraham， Neil C Rabinowitz， Ari S Morcos 和Daniel Zoran： 池化对于卷积神经网络中的适当形变稳定性既非必要条件，也非充分条件。arXiv 预印本 arXiv:1804.04438，2018。

[52]Sannai, Akiyoshi， Yuuki Takai 和 Matthieu Cordonnier： 深度神经网络对置换不变/等变函数的通用逼近。arXiv 预印本 arXiv:1903.01939，2019年。

[53] Scaman, Kevin 和 Aladin Virmaux：深度神经网络的利普希茨正则性：分析与高效估计。arXiv 预印本 arXiv:1805.10965，2018。[54]Scarselli, Franco、Marco Gori、Ah Chung Tsoi、Markus Hagenbuchner 和 Gabriele Monfardini：图神经网络模型。IEEE神经网络汇刊，20(1):61–80，2008年。

[55]Selsam, Daniel、Matthew Lamm、Benedikt Bünz、Percy Liang、Leonardo de Moura 和 David L Dill：从单比特监督中学习 *SAT* 求解器。arXiv 预印本 arXiv:1802.03685，2018。

[56] Thekumparampil, Kiran K，Chong Wang，Sewoong Oh和Li-Jia Li：用于半监督学习的基于注意力的图神经网络。arXiv 预印本 arXiv:1803.03735，2018。

[57] Vapnik, Vladimir：统计学习理论的本质。施普林格科学与商业媒体，2013年。[58] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser 和 Illia Polosukhin：注意力就是你所需要的一切。载于神经信息处理系统进展，第 5998–6008 页，2017年。[59] Walt, Stefan Van der, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart 和 Tony Yu：*scikit-image*：*Python* 图像处理。PeerJ，2:e453，2014年。[60] Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement De- langue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz 等：*Hugging Face* 的 *Transformers*：最先进的自然语言处理。arXiv 预印本 arXiv:1910.03771，2019年。[61] Xu, Keyulu, Mozhi Zhang, Jingling Li, Simon S Du, Ken-ichi Kawarabayashi 和 Stefanie Jegelka：理解神经网络如何外推：从前馈网络到图神经网络。arXiv 预印本 arXiv:2009.11848，2020年。[62] Yamashita, Rikiya, Mizuho Nishio, Richard Kinh Gian Do 和 Kaori Togashi：卷积神经网络：概述及其在放射学中的应用。影像学见解，9(4):611–629，2018。[63] Young, Tom, Devamanyu Hazarika, Soujanya Poria 和 Erik Cambria：基于深度学习的自然语言处理最新趋势。IEEE 计算智能杂志，13(3):55–75，2018。[64] Zaheer, Manzil, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan Salakhutdinov 和 Alexander Smola：深度集合。arXiv 预印本 arXiv:1703.06114，2017年。[65] Zhang, Chiyuan, Samy Bengio, Moritz Hardt, Benjamin Recht 和 Oriol Vinyals：理解深度学习需要重新思考泛化。CoRR, abs/1611.03530，2016。[66] Zhang, Chiyuan, Samy Bengio, Moritz Hardt, Benjamin Recht 和 Oriol Vinyals：理解深度学习（仍然）需要重新思考泛化。美国计算机协会通讯，64(3):107–115，2021年。[67] Zhou, Jie, Ganin Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li 和 Maosong Sun：图神经网络：方法与应用综述。AI Open，1:57–81，2020年。