清 华 大 学

综 合 论 文 训 练

题目：神经网络当中的梯度爆炸现象

系　　别：致理书院

专　　业：数学与应用数学

姓　　名：谢泽钰

指导教师：倪昂修　助理教授

2024 年 6 月 22 日

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

(涉密的学位论文在解密后应遵守此规定)

签　名：_____ 导师签名：_____ 日　期：_____

中文摘要

在不稳定神经网络中，梯度爆炸和消失问题限制了反向传播算法的有效性。随着网络层数增加，梯度可能会呈指数级变化，导致训练不稳定和性能下降。本文回顾了不稳定神经网络的理论基础，包括李雅普诺夫谱和向量的概念，强调其对研究神经网络动态特性的作用。通过编程计算两个神经网络的李雅普诺夫谱，展示了 QR 方法的具体实现，并验证了“中间向量”的对偶性问题，指出固定内积可能是探究神经网络动态特性的重要参数。

关键词：神经网络；反向传播；李雅普诺夫谱

ABSTRACT

In unstable neural networks, the issues of exploding and vanishing gradients limit the effectiveness of the backpropagation algorithm. As the number of network layers increases, gradients may change exponentially, leading to instability during training and a decline in model performance. This paper reviews the theoretical foundations of unstable neural networks, including the concepts of Lyapunov exponents and vectors, emphasizing their role in studying the dynamic characteristics of neural networks. By computing the Lyapunov spectra for two neural networks using programming, the QR method's implementation is demonstrated, and the duality problem of "intermediate vectors" is verified. The fixed inner product is highlighted as a potentially important parameter for exploring the dynamic properties of neural networks.

Keywords: Neural Network; Backpropagation; Lyapunov Spectrum

目　录

主要符号表

不稳定神经网络	在文中指代具有梯度爆炸和梯度消失问题的神经网络。
梯度爆炸和梯度消失	指在反向传播算法中，梯度可能呈指数级增长或衰减的问题。
反向传播算法	指用于训练神经网络的一种常见算法，通过计算梯度来更新网络参数。
网络层数和复杂度	表示神经网络的层数和复杂程度，通常与网络的深度和参数数量相关。
数值不稳定	指在训练过程中，由于梯度爆炸或梯度消失问题导致的数值不稳定现象。
模型性能	指神经网络在任务上的表现，如准确率、收敛速度等。
李雅普诺夫谱	用于描述系统动态特性和稳定性的概念。
李雅普诺夫向量	用于描述系统特征向量的概念，与李雅普诺夫谱相关。
伴随李雅普诺夫谱	指利用李雅普诺夫谱提供的信息来调整梯度传播路径和强度的方法。
对偶性	指伴随李雅普诺夫谱方法中的概念，用于解决梯度爆炸问题。
梯度裁剪	一种传统的反向传播算法中用于解决梯度爆炸问题的方法，通过限制梯度的范围来控制其大小。
正则化技术	另一种传统的反向传播算法中用于解决梯度爆炸问题的方法，通过在损失函数中引入正则化项来限制参数的增长。
伴随噪声	本文提出的一种基于伴随李雅普诺夫谱的新反向传播方法，用于调整梯度的传播路径和强度。
核微分方法	引入核函数平滑梯度计算的方法，提高计算的稳定性和准确性。
参数更新	指在训练过程中，通过梯度下降算法更新神经网络参数的步骤。
混合优化算法	结合了伴随噪声和核微分方法的新的优化算法，在处理梯度爆炸问题时具有优势。
训练速度	表示神经网络在训练过程中的速度，通常指每个训练样本的处理时间。
收敛性	指神经网络在训练过程中是否能够达到最优解的性质。

最终模型性能	指训练完成后神经网络在测试数据上的表现。
神经网络	指一种由多个神经元组成的计算模型，用于学习和处理复杂的数据关系。

第 1 章　引言

1.1　问题背景及意义

在不稳定神经网络中，梯度爆炸问题限制了反向传播算法的有效性。随着网络层数和复杂度增加，梯度可能会指数级增长，导致训练过程中数值不稳定和模型性能下降。为了更加深刻地理解这些现象，本文首先回顾不稳定神经网络的理论基础，包括李雅普诺夫谱和的概念与计算方法，从而刻画系统的动态特性和稳定性。在计算李雅普诺夫谱的过程中我得到了一些结论，包括正向和反向传播时中间向量的“对偶性”，以及输入有无对李雅普诺夫指数收敛性的影响。

1.2　文献综述

在动态系统、深度学习和混沌理论等多个领域，李雅普诺夫指数（Lyapunov Exponents，LEs）的计算和分析一直是重要的研究课题。近年来在这一领域出现了若干关键研究成果，包括不同计算方法的效率和准确性、在神经网络训练中的应用、以及混沌系统的敏感性分析。

Geist et al.（1990）对不同离散和连续方法计算李雅普诺夫指数的效率和准确性进行了比较[1]。他们的研究表明，基于 QR 分解或奇异值分解（SVD）的方法在计算李雅普诺夫指数时表现出较高的效率和稳定性。尽管最近提出的连续方法在理论上具有一定优势，但由于其计算时间长且数值不稳定，因此不推荐使用。Geist 等人的研究为后续在动态系统中的应用奠定了基础。

Von Bremen et al.（1997）进一步提出了一种基于 QR 分解的高效计算李雅普诺夫指数的方法[2]。他们通过数值实验展示了该方法在收敛性、准确性和效率方面的优越性能，特别是在处理复杂动态系统时，显著提高了计算的稳定性和速度。这一方法的提出为大规模动态系统的研究提供了强有力的工具。

随着深度学习的快速发展，研究人员开始关注李雅普诺夫指数在神经网络训练中的应用。Pascanu et al.（2013）探讨了训练递归神经网络（RNNs）的难点，指出网络在训练过程中会经历梯度消失和爆炸的问题[3]。这种现象与李雅普诺夫指数密切相关，因为指数的大小直接反映了系统的敏感性和稳定性。

为解决这一问题，Ioffe 和 Szegedy（2015）提出了批量归一化（Batch Normalization）技术，以减少内部协变量偏移，从而加速网络训练[4]。这一方法虽然不

是直接计算李雅普诺夫指数，但通过稳定训练过程间接提升了网络的鲁棒性。

Vakilipourtakalou 和 Mou（2020）则研究了递归神经网络的混沌特性，探索了这些网络在处理时间序列数据时的行为[5]。他们发现，适当的网络参数设置可以有效控制系统的混沌程度，从而改善模型的泛化能力。

在混沌系统的敏感性分析方面，Ni 等人的研究具有重要意义。Ni 和 Talnikar（2019）提出了一种非侵入性最小二乘伴随噪声（NILSAS）方法，用于混沌动态系统的伴随灵敏度分析[6]。该方法通过减少数值误差和计算时间，提高了灵敏度分析的准确性。

同时，Ni（2019）在另一篇论文中研究了三维湍流流动的超越性、阴影方向和灵敏度分析[6]。这项研究进一步揭示了在复杂流体系统中进行灵敏度分析的挑战和方法，为工程应用提供了理论支持。

Ni（2024）提出了通过伴随噪声技术在超混沌系统中进行反向传播的方法[7]。这种方法不仅提高了计算效率，还在一定程度上解决了传统方法中的数值稳定性问题。

此外，Ni（2023）开发了一种针对随机混沌系统线性响应的无传播算法[8]。这一创新性算法通过减少计算过程中的信息传播，大大提高了处理大规模系统的效率。

近期，Storm et al.（2023）研究了深度神经网络中的有限时间李雅普诺夫指数[9]。他们发现，李雅普诺夫指数可以有效评估网络在不同训练阶段的动态特性，帮助理解和优化深度网络的训练过程。这一研究为深度学习理论提供了新的视角，并且可能会影响未来神经网络模型的设计和训练方法。

1.3 论文框架

本文第二章回顾了李雅普诺夫谱和李雅普诺夫向量，介绍了计算李雅普诺夫指数的基本方法和应用，李雅普诺夫指数是用来描述一个动力系统中轨道对初始条件的敏感性的量度。在神经网络中，李雅普诺夫指数可以帮助我们理解网络的稳定性和动态行为。为了计算这些指数，我们采用了[10] 中介绍的 QR 方法，这也是目前在计算李雅普诺夫谱中最为常用和有效的方法之一。

第三章重点分析计算了李雅普诺夫谱在神经网络的训练过程中的表现，通过对全连接神经网络和循环神经网络的计算，成功得到了随时间收敛到李雅普诺夫指数的序列，并得到了一系列中间数据，从中发现了“对偶性”。

实验结果表明，李雅普诺夫指数可以作为一种有效的指标，用于评估网络的稳定性和预测训练过程中可能出现的数值问题。通过对李雅普诺夫指数的分析，我们可以提前发现并解决网络训练中的潜在问题，避免模型在训练后期出现不稳定或发散的现象。也有助于从动态系统的角度进一步理解神经网络参数的变化规律。

第四章总结了本文的研究成果，回顾了文章中出现的理论分析和实验验证环节，归纳了理解不稳定神经网络中梯度爆炸问题的思路。

第 2 章　李雅普诺夫谱和李雅普诺夫谱的计算

2.1　李雅普诺夫谱

在这一章中，我们将重点探讨不稳定神经网络的动态特性，特别关注李雅普诺夫谱、李雅普诺夫指数和“中间向量”对偶性。所涉及的概念和方法对于分析和理解神经网络的稳定性与动态行为具有重要的参考价值。

李雅普诺夫谱用于量化动力系统中轨迹对初始误差的敏感性。它通过记录系统在各个方向上的误差指数增长率，揭示系统的混沌程度和稳定性。对于神经网络，如果我们将某部分隐藏层参数视为“状态”，并将正向传播和反向传播视为动力系统的规则，则可以从动力系统的角度理解整个训练过程。

首先，我们将回顾李雅普诺夫向量的定义及其计算方法。

2.1.1　连续时间的李雅普诺夫谱

令连续时间动力系统的状态由向量 $\mathbf{x}(t)$ 描述，其具有下面的演化方程：

$$\frac{d\mathbf{x}}{dt} = \mathbf{f}(\mathbf{x}, t) \quad t \geqslant 0. \tag{2.1}$$

李雅普诺夫指数 λ_i 可以通过对初始状态的微小扰动进行分析得到。首先，我们给定一个微小扰动 $\delta\mathbf{x}(t)$，并用下式描述其演化：

$$\frac{d(\delta\mathbf{x})}{dt} = \mathbf{J}(\mathbf{x}, t)\delta\mathbf{x}. \tag{2.2}$$

其中，系统的雅可比矩阵 $\mathbf{J}(\mathbf{x}, t)$ 定义为：

$$\mathbf{J}(\mathbf{x}, t) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}. \tag{2.3}$$

李雅普诺夫指数通过分析扰动向量 $\delta\mathbf{x}(t)$ 的指数增长率定义为：

$$\lambda_i = \lim_{t\to\infty} \frac{1}{t} \ln \frac{||\delta\mathbf{x}_i(t)||}{||\delta\mathbf{x}_i(0)||}. \tag{2.4}$$

连续时间的李雅普诺夫谱通常用于分析微分方程等连续性系统的稳定性和混沌性，本文之后将不涉及这一部分内容。

2.1.2 离散时间的李雅普诺夫谱

离散时间的李雅普诺夫谱的定义与连续时间的情况大致相同。设定离散时间动力系统的状态向量为 $\mathbf{x}_n$，其演化方程为：

$$\mathbf{x}_{n+1} = \mathbf{f}(\mathbf{x}_n, n) \quad n \geqslant 0. \tag{2.5}$$

李雅普诺夫指数 λ_i 可以通过分析系统状态的微小扰动来获得。考虑微小扰动 $\delta\mathbf{x}_n$，其演化由以下公式描述：

$$\delta\mathbf{x}_{n+1} = \mathbf{J}(\mathbf{x}_n, n)\delta\mathbf{x}_n. \tag{2.6}$$

$\mathbf{J}(\mathbf{x}_n, n)$ 是系统的雅可比矩阵，定义为：

$$\mathbf{J}(\mathbf{x}_n, n) = \left.\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}_n}. \tag{2.7}$$

离散时间的李雅普诺夫指数通过分析扰动向量 $\delta\mathbf{x}_n$ 的指数增长率定义为：

$$\lambda_i = \lim_{n\to\infty} \frac{1}{n} \ln \frac{||\delta\mathbf{x}_i(n)||}{||\delta\mathbf{x}_i(0)||}. \tag{2.8}$$

而从伴随李雅普诺夫向量（CLV）的视角来看，一个伴随李雅普诺夫向量 $\zeta(t)$ 是一个齐次切向解，其范数表现为时间的指数函数。存在 $C_1, C_2 > 0$ 和 $\lambda \in \mathbb{R}$，使得对于任意 t，

$$C_1 e^{\lambda t}\|\delta\mathbf{x}_i(0)\| \leqslant \|\delta\mathbf{x}_i(t)\| \leqslant C_2 e^{\lambda t}\|\delta\mathbf{x}_i(0)\|. \tag{2.9}$$

其中范数为 $\mathbb{R}^m$ 中的欧几里得范数，而 λ 就定义为与该向量对应的李雅普诺夫指数。指数为正称为不稳定，为负被称为稳定，而零李雅普诺夫指数的方向被称为中性。

离散时间的李雅普诺夫谱是我们分析神经网络参数所需要的，下文中提到的“李雅普诺夫指数”及“李雅普诺夫谱”等均指离散时间的情况。

2.2 计算方法

李雅普诺夫谱的计算有 Jacobi 法、QR 法等，本节我们参考 Ni（2019）[6]，总结计算李雅普诺夫向量的步骤如下：

1. 计算方程的原初解，足够长的时间使得轨迹落在吸引子上，然后设 $t = 0$，并设定原初系统的初始条件，$u_0(0)$。
2. 随机生成一个 $m \times M$ 的正交矩阵 $Q_0 = [q_{01}, \cdots, q_{0M}]$。这个 Q_0 将被用作齐次切向解的初始条件。

3. 对于 $i = 0$ 到 $K - 1$，在第 i 段时间内，$t \in [t_i, t_{i+1}]$，进行以下步骤：
   (a) 从 t_i 到 t_{i+1} 计算原初解 $u_i(t)$。
   (b) 计算齐次切向解 $W_i(t) = [w_{i1}(t), \cdots, w_{iM}(t)]$。
      i. 对于每个齐次切向解 w_{ij}，$j = 1, \cdots, M$，从初始条件 $w_{ij}(t_i) = q_{ij}$ 开始，积分方程 (3.2) 从 t_i 到 t_{i+1}。
      ii. 进行 QR 分解：$W_i(t_{i+1}) = Q_{i+1}R_{i+1}$，其中 $Q_{i+1} = [q_{i+1,1}, \cdots, q_{i+1,M}]$。
4. 第 j 大的李雅普诺夫指数（LE），λ_j，通过下式近似：

$$\lambda_j = \frac{1}{K\Delta T} \sum_{i=1}^{K} \log |D_{ij}|. \tag{2.10}$$

   其中 D_{ij} 是 R_i 中的第 j 个对角元素。随着 T 变大，这个公式中的 λ_j 收敛到真实值。
5. 我们定义一个 $m \times M$ 的矩阵 $V(t)$ 为：

$$V(t) = W_i(t)R_{i+1}^{-1} \cdots R_K^{-1}, \quad t \in [t_i, t_{i+1}]. \tag{2.11}$$

   当 t 和 $T - t$ 都变大时，$V(t)$ 的第 j 列收敛到第 j 个共轭李雅普诺夫向量（CLV）的方向。注意，虽然 $V(t)$ 在不同的时间段上有不同的表达式，但它的各列在所有时间段上是连续的。

通过以上步骤，我们可以用数值方法得到一个动力系统的李雅普诺夫谱。因为，粗略地说，QR 分解中的 Q 矩阵总是单位体积的高位正方体，所以 R 矩阵的对角元素记录了高维立方体体积的增长速率。

虽然理论上该算法也能够计算李雅普诺夫向量，但本文中不会涉及此概念。

第 3 章　不稳定神经网络

本章将重点讨论不稳定神经网络，并结合文献中的相关研究，探讨若干已经提出的解决方法，包括权重初始化方法、伴随噪声法和核微分方法。

3.1　梯度爆炸

在神经网络的训练过程中，梯度爆炸问题是影响网络训练效率和效果的主要障碍之一。梯度爆炸较多发生于深层神经网络，特别是进行反向传播时，梯度值可能会因为连续的链式法则计算而指数增长，导致数值不稳定和训练失败。

梯度爆炸问题可以通过一个简单的深层神经网络训练过程来说明：设一个多层感知器（MLP），其损失函数为 L，网络的权重为 $\mathbf{W}$，每层的输出为 $\mathbf{a}_l$：

$$\mathbf{a}_{l+1} = \sigma(\mathbf{W}_l \mathbf{a}_l + \mathbf{b}_l). \tag{3.1}$$

其中，σ 是激活函数，$\mathbf{b}_l$ 是第 l 层的偏置。

在反向传播过程中，需要计算损失函数 L 对权重 $\mathbf{W}_l$ 的梯度：

$$\frac{\partial L}{\partial \mathbf{W}_l} = \delta_{l+1} \mathbf{a}_l^T. \tag{3.2}$$

其中，δ_{l+1} 是误差项，定义为：

$$\delta_{l+1} = \frac{\partial L}{\partial \mathbf{a}_{l+1}} \odot \sigma'(\mathbf{z}_{l+1}). \tag{3.3}$$

通过链式法则，误差项 δ_l 的更新为：

$$\delta_l = (\mathbf{W}_l^T \delta_{l+1}) \odot \sigma'(\mathbf{z}_l). \tag{3.4}$$

如果有正的李雅普诺夫指数，则说明随着该神经网络的层数加深，上述过程会导致梯度的累积乘积，其中每一项都将会放大误差，最终使得梯度在反向传播过程中指数增长，导致梯度爆炸。

3.2　权重初始化

为了解决梯度爆炸的问题，人们提出了一系列方法，其中权重初始化方法最为常用。这些手段旨在通过合适的初始化策略，使得梯度在反向传播过程中保持稳定，避免梯度爆炸的发生。

下面介绍 Xavier 初始化和 He 初始化两种常见的权重初始化方法，并简述其原理和应用。

3.2.1 Xavier 初始化

Xavier 初始化（也称为 Glorot 初始化）是由 Xavier Glorot 和 Yoshua Bengio 在 2010 年提出的一种权重初始化方法。该方法旨在使网络层的输入和输出的方差保持一致，从而在前向传播和反向传播过程中，信号能够有效传递。

在 Xavier 初始化中，权重 W_l 的初始化遵循以下分布：

$$W_l \sim \mathcal{U}\left(-\sqrt{\frac{6}{n_{l-1}+n_l}}, \sqrt{\frac{6}{n_{l-1}+n_l}}\right). \tag{3.5}$$

或

$$W_l \sim \mathcal{N}\left(0, \frac{2}{n_{l-1}+n_l}\right). \tag{3.6}$$

其中，n_{l-1} 是第 $l-1$ 层的神经元数量，n_l 是第 l 层的神经元数量。前者使用均匀分布，后者使用正态分布。

Xavier 初始化的基本思想是通过选择合适的初始权重范围，使得每层输出的方差接近输入的方差，从而在训练的初始阶段避免信号的过度放大或缩小。

3.2.2 He 初始化

He 初始化（也称为 Kaiming 初始化）是由 Kaiming He 等人在 2015 年提出的一种改进的权重初始化方法，主要针对使用 ReLU 激活函数的神经网络。在 ReLU 激活函数下，输出的方差会受到输入方差的影响，因此需要更大的初始权重范围。

在 He 初始化中，权重 W_l 的初始化遵循以下分布：

$$W_l \sim \mathcal{N}\left(0, \frac{2}{n_{l-1}}\right). \tag{3.7}$$

或

$$W_l \sim \mathcal{U}\left(-\sqrt{\frac{6}{n_{l-1}}}, \sqrt{\frac{6}{n_{l-1}}}\right). \tag{3.8}$$

其中，n_{l-1} 是第 $l-1$ 层的神经元数量。与 Xavier 初始化相比，He 初始化在方差上增加了一倍，从而更加适应 ReLU 激活函数的特性。

3.3 伴随噪声法

除去权重初始化方法，伴随噪声法是另一种用于解决梯度爆炸问题的有效方法。

伴随噪声法出现的时间更晚，它通过引入伴随变量和李雅普诺夫分析来稳定反向传播过程，从而降低梯度爆炸的发生概率。该方法在复杂动态系统的控制中已有广泛应用，最近被引入到神经网络的训练中，以应对深层网络中的梯度爆炸问题。

3.3.1 理论背景

神经网络的反向传播过程中，梯度的计算依赖于复合函数求导的链式法则，具体现为层与层之间的梯度乘积。这种乘积会导致梯度的指数级别增长或减小，从而引发梯度爆炸或梯度消失问题。

为了解决这一问题，尝试引入伴随变量和李雅普诺夫分析，通过调整梯度计算过程间接控制梯度增长。

3.3.2 伴随变量的引入

设伴随变量 $\mathbf{u}_l$ 是通过以下李雅普诺夫方程定义的：

$$\mathbf{u}_l = \mathbf{Q}_l + \mathbf{A}_l \mathbf{u}_{l+1} \mathbf{A}_l^T. \tag{3.9}$$

其中，$\mathbf{Q}_l$ 是对称正定矩阵，$\mathbf{A}_l$ 是系统矩阵。伴随变量 $\mathbf{u}_l$ 捕捉了系统在反向传播过程中积累的数值不稳定性。

3.3.3 梯度计算的调整

在每一步反向传播中，利用伴随变量来调整梯度计算。具体而言，传统的梯度计算公式为：

$$\frac{\partial L}{\partial \mathbf{W}_l} = \delta_{l+1} \mathbf{a}_l^T. \tag{3.10}$$

其中，δ_{l+1} 是第 $l+1$ 层的误差项，$\mathbf{a}_l$ 是第 l 层的激活输出。在伴随噪声法中，通过伴随变量 $\mathbf{u}_l$ 来修正梯度计算公式：

$$\frac{\partial L}{\partial \mathbf{W}_l} = \mathbf{u}_l (\delta_{l+1} \mathbf{a}_l^T). \tag{3.11}$$

该修正公式通过伴随变量调整梯度的增长，使得梯度在反向传播过程中得到有效控制。

3.3.4 李雅普诺夫方程的求解

李雅普诺夫方程在动态系统中用于分析系统的稳定性，其形式为：

$$\mathbf{AP} + \mathbf{PA}^T + \mathbf{Q} = 0. \tag{3.12}$$

对于李雅普诺夫方程，本节仅仅是简单介绍，与第四章中的李雅普诺夫谱的计算并无直接关联。

3.4 核微分方法

3.4.1 方法简介

另一种解决梯度爆炸问题的方法为赫微分方法。

核微分方法通过将梯度计算问题转换为核函数的操作，从而平滑梯度并减少梯度爆炸的风险。

设核函数 $k(\mathbf{x}, \mathbf{y})$ 满足 Mercer 定理，即满足正定性和对称性。我们通过构造核矩阵 $\mathbf{K}$ 来替代直接的梯度计算：

$$\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j). \tag{3.13}$$

在反向传播过程中，利用核矩阵来平滑梯度：

$$\frac{\partial L}{\partial \mathbf{W}_l} = \mathbf{Kg}. \tag{3.14}$$

其中，$\mathbf{g}$ 是传统方法计算得到的梯度。

核微分方法的关键在于选择合适的核函数 $k(\mathbf{x}, \mathbf{y})$，例如高斯核或多项式核。通过核函数的平滑作用，能够减小梯度的波动，从而减轻或消除梯度爆炸问题。

3.4.2 核函数选取

1. 高斯核：

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{||\mathbf{x} - \mathbf{y}||^2}{2\sigma^2}\right). \tag{3.15}$$

其中，σ 是核的宽度参数。

2. 多项式核：

$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T\mathbf{y} + c)^d. \tag{3.16}$$

其中，c 是常数，d 是多项式的度数。

总结起来，核微分方法通过引入核函数，平滑了梯度变化，使得梯度在反向

传播过程中不易发生爆炸。同时，这种方法也能够保持梯度信息，从而提高训练效率和效果。核微分方法是一种较为优秀的梯度控制方法。

第 4 章　神经网络中的李雅普诺夫谱

4.1　符号约定

在分析神经网络的李雅普诺夫谱时，我们使用以下符号约定，如表 4.1 所示.

表 4.1　符号约定

符号	含义
x_l	第 l 层的状态向量
$\mathbf{J}_l$	第 l 层的雅可比矩阵
$\mathbf{R}_l$	第 l 层的上三角矩阵
$\mathbf{Q}_l$	第 l 层的正交矩阵
$\lambda_{forward,i}$	正向传播的第 i 个李雅普诺夫指数
$\lambda_{backward,i}$	反向传播的第 i 个李雅普诺夫指数
q_{ij}^{+}	正向传播中第 i 层的第 j 个偏置向量
q_{ij}^{-}	反向传播中第 i 层的第 j 个偏置向量

4.2　理论分析

伴随李雅普诺夫谱是指在系统反向传播过程中计算得到的李雅普诺夫指数。理论上，正向传播和反向传播的李雅普诺夫谱应该具有一定的对偶性，即相同方向对应的正向和反向传播的两个向量内积不随时间变化.

$$\langle q_{ij}^{+}, q_{ij}^{-} \rangle = \text{常数}. \tag{4.1}$$

4.2.1　研究方案

为了验证这一结论，本文进行了以下研究：

1. 正向传播计算：按照前述步骤，计算神经网络在正向传播过程中的李雅普诺夫谱 $\lambda_{forward,i}$.
2. 反向传播计算：在反向传播过程中，同样计算出相应的李雅普诺夫谱 $\lambda_{backward,i}$.
3. 对偶性验证：计算正向和反向的李雅普诺夫谱的同时，会得到每一步的中

间向量 q_{ij}^{+} 和 q_{ij}^{-}. 我们发现这两个向量具有“对偶性”。

4.2.2 对偶性的定义及理论证明

设正向传播中第 i 层的第 j 个中间向量为 q_{ij}^{+}，反向传播中第 i 层的第 j 个中间向量为 q_{ij}^{-}，则对偶性定义为：

$$\langle q_{ij}^{+}, q_{ij}^{-} \rangle = \text{常数}. \tag{4.2}$$

证明如下：

$$\begin{aligned} \langle q_{ij}^{+}, q_{ij}^{-} \rangle &= (\boldsymbol{J}^{T} \cdot q_{ij}^{-})^{T} \cdot q_{ij}^{+T} \\ &= q_{ij}^{-T} \cdot (\boldsymbol{J} \cdot q_{ij}^{+}) \\ &= q_{ij}^{-T} \cdot q_{ij+1}^{+} \\ &= \langle q_{ij+1}^{+}, q_{ij+1}^{-} \rangle. \end{aligned} \tag{4.3}$$

我们将在 4.3.3 一节验证这一现象，但下一节将首先介绍前述算法在全连接神经网络中和循环神经网络中的应用。

4.3 实验分析

本章的最终目的在于使用算法 2.2 验证 4.2，首先展示该算法在全连接神经网络中的表现：

4.3.1 全连接神经网络

4.3.1.1 网络结构

选取一个三层全连接神经网络，针对 28×28 的灰度图像数据集（例如 MNIST 手写数字数据集）进行训练。网络参数和结构示意图如下所示：

1. 输入层：维度为 28x28=784.
2. 隐藏层：一个，维度为 50，激活函数为 ReLU.
3. 输出层：维度为 10，激活函数为 Softmax.

首先，我们选定网络每层的状态：

1. 输入层：接受 28×28 的灰度图像作为输入，展平成 784 维的向量：

$$\mathbf{x} \in \mathbb{R}^{784}. \tag{4.4}$$

2. 隐藏层：一个全连接层，包含 50 个神经元，激活函数为 ReLU：

$$\mathbf{h} = \text{ReLU}(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1). \tag{4.5}$$

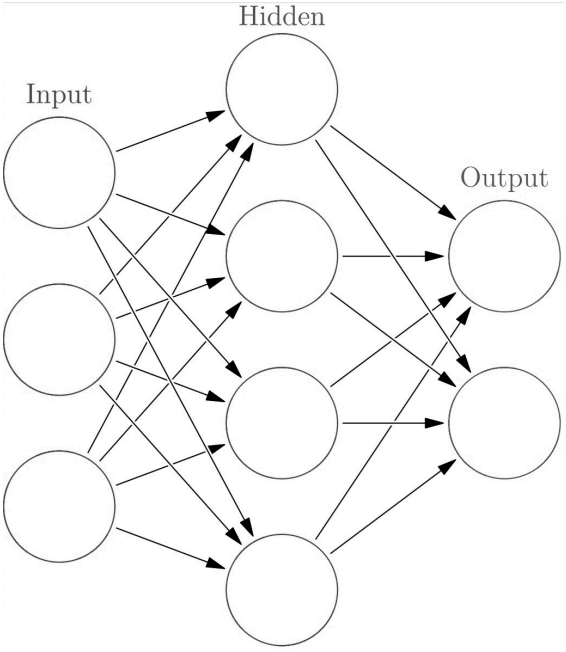

图 4.1　全连接神经网络

其中，$\mathbf{W}_1 \in \mathbb{R}^{50\times784}$ 为权重矩阵，$\mathbf{b}_1 \in \mathbb{R}^{50}$ 为偏置向量。

3. 输出层：一个全连接层，包含 10 个神经元，激活函数为 Softmax：

$$\mathbf{y} = \text{Softmax}(\mathbf{W}_2\mathbf{h} + \mathbf{b}_2). \tag{4.6}$$

其中，$\mathbf{W}_2 \in \mathbb{R}^{10\times50}$ 为权重矩阵，$\mathbf{b}_2 \in \mathbb{R}^{10}$ 为偏置向量。

约定这个神经网络中的符号如下表所示：

表 4.2　神经网络参数符号

符号	含义
$\mathbf{x}$	输入向量
$\mathbf{h}$	隐藏层输出
$\mathbf{y}$	输出层输出
$\mathbf{W}_1$	隐藏层权重矩阵
$\mathbf{b}_1$	隐藏层偏置向量
$\mathbf{W}_2$	输出层权重矩阵
$\mathbf{b}_2$	输出层偏置向量

4.3.1.2 训练过程

对上一节规定好的全连接神经网络进行训练，中间使用交叉熵损失函数和随机梯度下降法（SGD）进行优化。训练的具体步骤如下：

1. 前向传播：计算网络的输出 $\mathbf{y}$：

$$\mathbf{y} = \text{Softmax}(\mathbf{W}_2\mathbf{h} + \mathbf{b}_2). \tag{4.7}$$

2. 计算损失：使用交叉熵损失函数 $\mathcal{L}$：

$$\mathcal{L} = -\sum_i y_i \log \hat{y}_i. \tag{4.8}$$

3. 反向传播：计算每层的梯度，并更新权重：

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_1}, \frac{\partial \mathcal{L}}{\partial \mathbf{b}_1}, \frac{\partial \mathcal{L}}{\partial \mathbf{W}_2}, \frac{\partial \mathcal{L}}{\partial \mathbf{b}_2}. \tag{4.9}$$

4.3.1.3 李雅普诺夫谱计算

考虑该训练过程作为动力系统，隐藏层的 50 个神经元作为状态向量，使用算法 2.2 计算隐藏层的李雅普诺夫谱。

首先，选取总运行时间步 $T = 300$，执行算法，计算出正、反各 50 个维度的李雅普诺夫指数。

4.3.1.4 运行结果

实验结果显示，隐藏层的 50 个李雅普诺夫指数的计算结果如下：（$T = 300$ 步时的收敛结果，并非准确值）

1. 正向传播：

0.2491	0.25	0.2493	0.2499	0.2489
0.25	0.25	0.2486	0.2498	0.2494
0.2493	0.25	0.2495	0.25	0.2496
0.2499	0.25	0.2499	0.2497	0.2494
0.249	0.2475	0.2497	0.2499	0.25
0.25	0.2492	0.249	0.2496	0.25
0.25	0.2497	0.2499	0.2479	0.2474
0.25	0.2497	0.25	0.2482	0.25
0.25	0.249	0.25	0.2499	0.2493
0.25	0.2494	0.2499	0.2483	0.2494

2. 反向传播：

0.2491	0.25	0.2493	0.2499	0.2489
0.25	0.25	0.2486	0.2498	0.2494
0.2493	0.25	0.2495	0.25	0.2496
0.2499	0.25	0.2499	0.2497	0.2494
0.249	0.2475	0.2497	0.2499	0.25
0.25	0.2492	0.249	0.2496	0.25
0.25	0.2497	0.2499	0.2479	0.2474
0.25	0.2497	0.25	0.2482	0.25
0.25	0.249	0.25	0.2499	0.2493
0.25	0.2494	0.2499	0.2483	0.2494

将其画成如下两幅散点图：

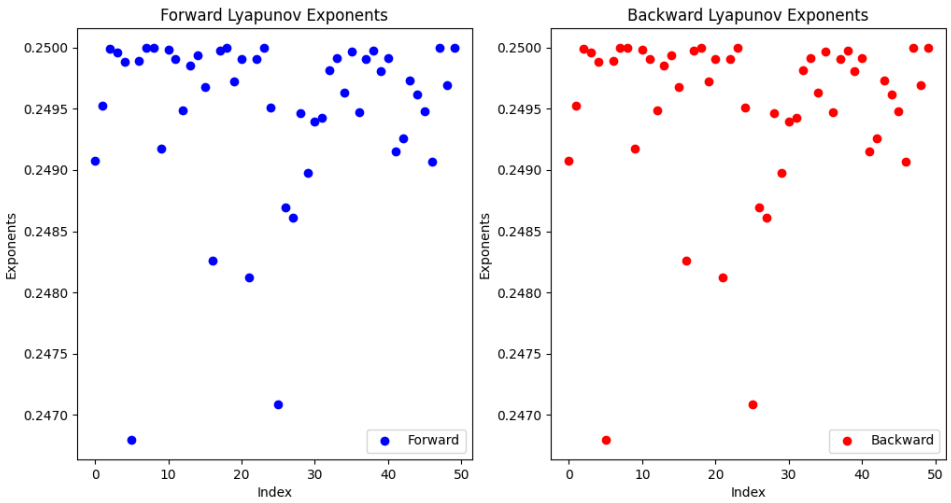

图 4.2　隐藏层李雅普诺夫指数分布

根据计算结果和图 4.2，可以得到如下的明显结论：

1. 50 个李雅普诺夫指数均小于 0.25，但越靠近 0.25 指数分布越密集，反之越稀疏。
2. 最小的指数也大于 0.248，说明 50 个指数的分布范围较小，分布区间及其相似。
3. 该网络正向和反向传播的李雅普诺夫谱相同，推测可能与全连接神经网络的结构有关。

至此已经完成了算法 2.2 的运行，但考虑到该全连接神经网络中间层的维数

偏多，若验证对偶性将花费过长时间，因此我们将在下一节中考虑隐藏层维数较小的循环神经网络（RNN）。

4.3.2 循环神经网络

4.3.2.1 网络结构

RNN 在处理序列数据方面具有显著优势，能够捕捉时间步长上的依赖关系。在本实验中，我们选取一个循环神经网络（RNN），分别针对无输入、有输入的情形进行计算。选取任意时间步长 T，如下图示设计了一个简单的 RNN 模型，然后使用李雅普诺夫谱分析其动态行为。

和全连接神经网络类似，我们先尝试运行算法 2.2，首先确定神经网络的参数规模和结构：

1. 输入层：每个时间步的输入维度为 2
2. 隐藏层：一个，隐藏状态维度为 3，激活函数为 tanh
3. 输出层：维度为 2，激活函数为 Softmax

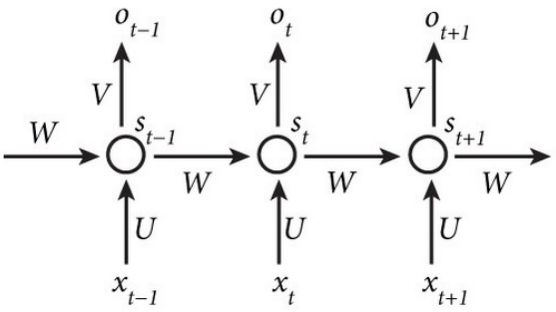

图 4.3　循环神经网络

其次确定激活函数、权重矩阵和偏置向量：

1. 输入层：每个时间步的输入维度为 2，时间步可任意长（在程序可计算的范围内）

$$\mathbf{x}_t \in \mathbb{R}^3. \tag{4.10}$$

2. 隐藏层：一个 RNN 层，隐藏状态维度为 3，激活函数为 tanh：

$$\mathbf{h}_t = \tanh(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{xh}\mathbf{x}_t + \mathbf{b}_h). \tag{4.11}$$

其中，$\mathbf{W}_{hh} \in \mathbb{R}^{3\times3}$ 和 $\mathbf{W}_{xh} \in \mathbb{R}^{3\times3}$ 分别为隐藏状态和输入的权重矩阵，$\mathbf{b}_h \in \mathbb{R}^3$ 为偏置向量。

3. 输出层：维度为 3，激活函数为 Softmax：

$$\mathbf{y}_t = \text{Softmax}(\mathbf{W}_{hy}\mathbf{h}_t + \mathbf{b}_y). \tag{4.12}$$

其中，$\mathbf{W}_{hy} \in \mathbb{R}^{3\times3}$ 为权重矩阵，$\mathbf{b}_y \in \mathbb{R}^3$ 为偏置向量。

4.3.2.2 训练过程

训练过程开始，我们简单地使用差值作为损失函数，并通过随机梯度下降法（SGD）对神经网络进行训练。具体步骤如下：

1. 前向传播：计算每个时间步的隐藏状态和最终输出 $\mathbf{y}$.
2. 计算损失：使用差值作为损失函数 $\mathcal{L}$.
3. 反向传播：计算每层的梯度，并更新权重。

为了分析网络的动态行为，我们在训练过程中记录每一层的雅可比矩阵。假设 $\mathbf{J}_t$ 是第 t 个时间步的雅可比矩阵，则其定义为：

$$\mathbf{J}_t = \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}}. \tag{4.13}$$

在训练的每个时间步，我们需要参考算法 2.2，记录所有的中间变量 $q_{i,1}, q_{i,2}, q_{i,3}$，它们来源于 QR 分解的结果：

$$Q_i = [q_{i,1}, q_{i,2}, q_{i,3}]. \tag{4.14}$$

4.3.2.3 李雅普诺夫谱计算

计算正向传播的李雅普诺夫谱的具体算法如下：

算法 4.1 计算 RNN 正向传播的 Lyapunov 谱

初始化 RNN，设定初始条件 h_0 和输入序列 $x = \{x_t\}_{t=1}^{T}$.
随机生成 $m \times M$ 的正交矩阵 Q_0 作为齐次切向解的初始条件。
for $i = 0$ to $K - 1$ **do**
    计算 RNN 的隐藏状态 h_i 和输出 y_i，$t \in [t_i, t_{i+1}]$.
    计算齐次切向解 $W_i(t)$：
    for $j = 1$ to M **do**
        从 $w_{ij}(t_i) = q_{ij}$ 积分方程到 t_{i+1}，该方程基于 RNN 的雅可比矩阵 $\frac{\partial h_{t+1}}{\partial h_t}$.
    end for
    QR 分解：$W_i(t_{i+1}) = Q_{i+1}R_{i+1}$.
end for
计算李雅普诺夫指数 λ_j：

$$\lambda_j = \frac{1}{K\Delta T}\sum_{i=1}^{K}\log|D_{ij}|. \tag{4.15}$$

其中 D_{ij} 是 R_i 的第 j 个对角元素。

类似地，可以用如下算法计算反向传播的李雅普诺夫谱：

算法 4.2 计算 RNN 反向传播的 Lyapunov 谱

初始化 RNN，设定初始条件 h_0 和输入序列 $x = \{x_t\}_{t=1}^{T}$.
进行正向传播，计算每个时间步的隐藏状态 h_t 和输出 y_t.
随机生成 $m \times M$ 的正交矩阵 Q_T 作为齐次切向解的初始条件。
for $i = T$ to 1 **do**
    计算 RNN 的反向传播梯度 δ_i，$t \in [t_i, t_{i-1}]$.
    计算齐次切向解 $W_i(t)$：
    for $j = 1$ to M **do**
        从 $w_{ij}(t_i) = q_{ij}$ 积分方程到 t_{i-1}，该方程基于 RNN 的雅可比矩阵 $\frac{\partial h_t}{\partial h_{t+1}}$.
    end for
    QR 分解：$W_i(t_{i-1}) = Q_{i-1}R_{i-1}$.
end for
计算李雅普诺夫指数 λ_j：

$$\lambda_j = \frac{1}{K\Delta T}\sum_{i=1}^{K}\log|D_{ij}|. \tag{4.16}$$

其中 D_{ij} 是 R_i 的第 j 个对角元素。

4.3.2.4 运行结果 - 无输入情形

首先，简便起见，令输入序列 $\{x_t\}_{t=1}^{T}$ 全部为 0，即神经网络正向和反向的传播均不受到输入的影响。在此基础上尝试不同的 T 值，以观察李雅普诺夫指数的收敛性。

三个方向的李雅普诺夫指数随 T 的变化而变化，下图展示了迭代次数从 1 至 300 的正向李雅普诺夫指数变化规律：

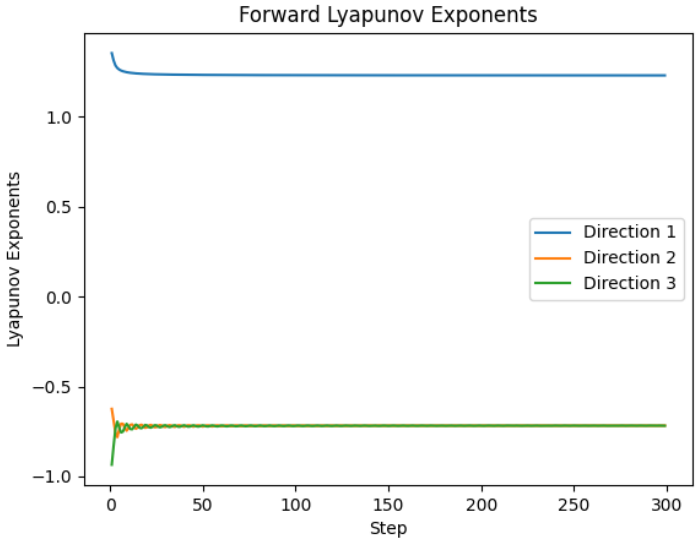

图 4.4 正向传播收敛情况

由于不考虑输入的影响，反向的李雅普诺夫指数与正向相同：

在上面的算法中，我们取 $T = 300$，运行程序。

实验结果显示，隐藏层的 3 个李雅普诺夫指数分别为：

$$1.22852421 \quad -0.71748841 \quad -0.71789691 \tag{4.17}$$

根据运行结果显示，正向传播和反向传播的李雅普诺夫指数各自有一个大于 0，两个小于 0，其最大李雅普诺夫指数大于 0，从动力系统的角度分析，其最大李雅普诺夫指数大于 0，表明正向传播和反向传播的梯度不稳定，步数增多可能会导致梯度爆炸。

4.3.2.5 运行结果 - 有输入情形

下面我们进一步验证了有输入情形下的李雅普诺夫谱。我们将输入序列 $\{x_t\}_{t=1}^{T}$ 设置为随机生成的序列（numpy 的随机数种子设置为 42），以观察李雅普诺夫指数的变化。

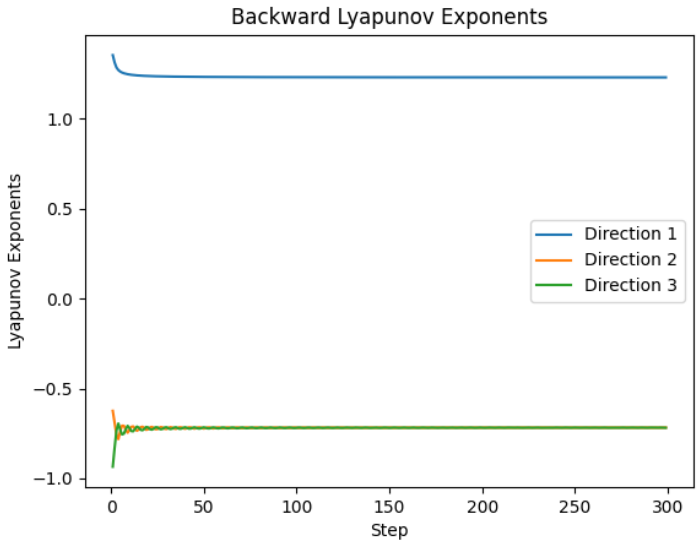

图 4.5　反向传播收敛情况

运行 300 个时间步后得到的李雅普诺夫指数分别为：

1. 正向：

$$0.00887475 \quad -0.00384957 \quad -0.00613681 \tag{4.18}$$

2. 反向：

$$0.00351433 \quad -0.0024369 \quad -0.00389064 \tag{4.19}$$

正向传播的 300 步的收敛过程如下图所示：

反向传播的收敛过程与正向传播类似，且由于李雅普诺夫指数均十分接近 0，图像趋势没有太大区别。同时可以看到，随着 T 的增加，正向和反向传播的李雅普诺夫指数均收敛，这表明前述算法对与一般情形的 RNN 同样有效。

经过对比，加入随机输入因素的前后有如下区别：

1. 不考虑输入时，正反的李雅普诺夫谱数值相同，但有输入时，尽管两个李雅普诺夫谱均更接近 0，但二者不再相同。
2. 加入随机输入后李雅普诺夫指数的绝对值变小，本文暂未找到明确的理论解释。
3. 加入随机输入后李雅普诺夫指数随 T 的收敛速度更快，对此同样暂没有明确的解释。

虽然有种种不同，但下一节将验证，中间向量的对偶性依旧成立。

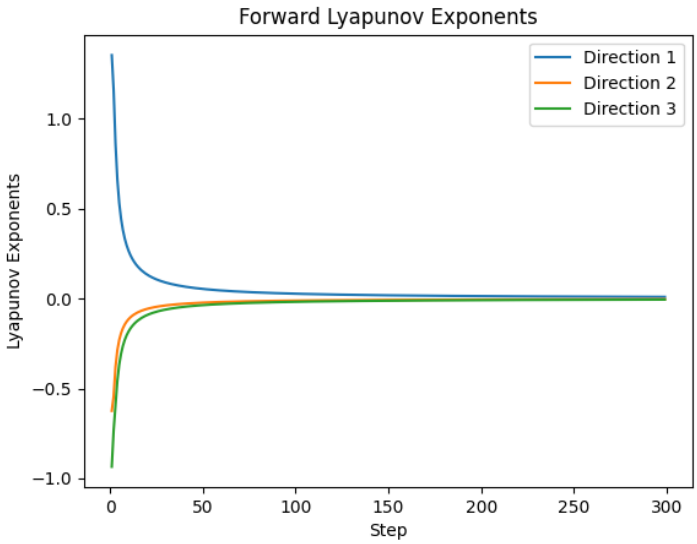

图 4.6　有输入情形下的李雅普诺夫指数

4.3.3　对偶性的验证

对偶性的验证用到 4.3.2.2 中记录的中间向量，由算法 2.2 的原理，它们可以在计算李雅普诺夫谱的同时得到。将这些中间向量输出到日志文件中，然后读取并计算它们的内积，以验证对偶性。

下图形象展示了正向和反向传播的中间向量的内积，横轴表示时间步，可以发现，除去最边缘的极少数时间步，内积图像为一条直线，这表明正向和反向传播的中间向量的内积是一个常数，三条线的纵坐标即为三个方向对应的内积值。

由此我们便验证了正向传播和反向传播的中间向量 q_{ij} 的有对偶性（不但内积固定，而且三个方向内积相同），但由于提前约定了输入为 0，仅能代表特殊情况，我们还需要进一步验证有输入情形下的对偶性。

使用相同方法，有输入时的中间向量内积如下图所示：

有输入的情形下正反两边的李雅普诺夫谱不再完全相同，但对偶性得到了保留，说明该性质与输入无关，是神经网络本身的特性，由其结构决定。

4.4　结论

本章深入探讨了不稳定神经网络的李雅普诺夫谱和中间向量的对偶性。通过详细的理论分析和实验验证，我们发现李雅普诺夫谱能够有效地揭示神经网络的动态特性，为网络的设计和优化提供了重要的理论支持，值得在未来的研究中进

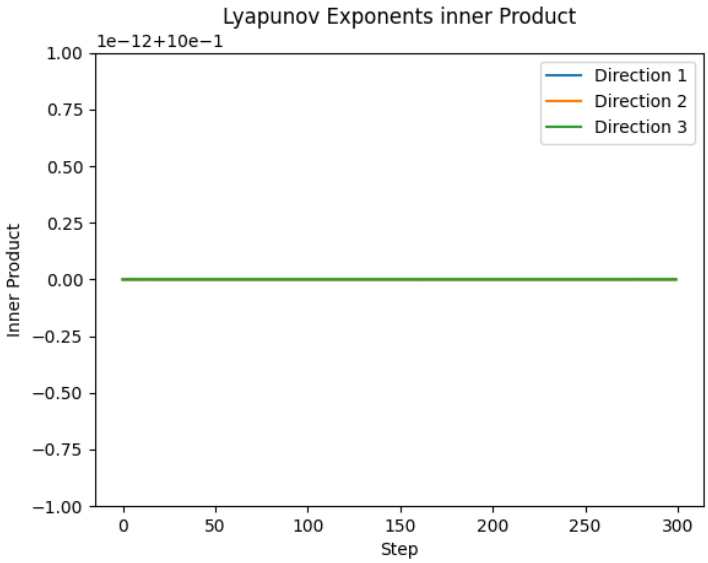

图 4.7　无输入情形下的中间向量 q_{ij} 内积

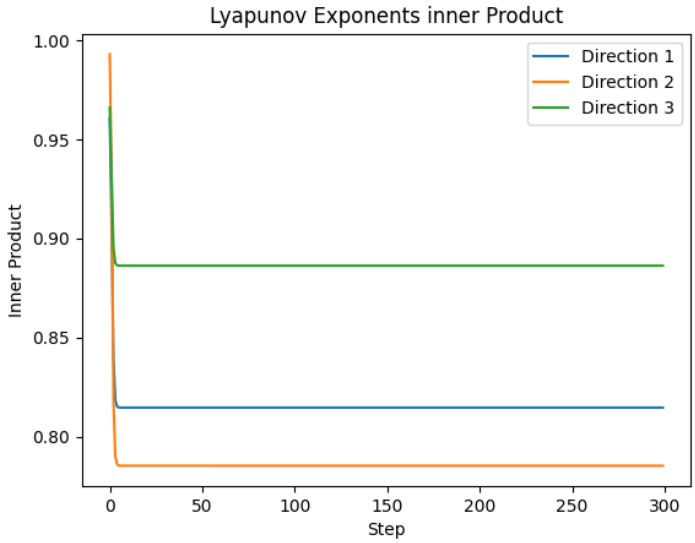

图 4.8　有输入情形下的中间向量 q_{ij} 内积

一步应用，此外，对偶性的验证也为神经网络的理论研究提供了新的视角，固定的内积值对于神经网络是否具有更多数学意义，值得进一步探讨。

第 5 章　总结

本文回顾了李雅普诺夫谱和李雅普诺夫指数，并介绍了计算李雅普诺夫指数的基本方法和应用。李雅普诺夫指数是用来描述一个动力系统中轨道对初始条件的敏感性的量度。在神经网络中，李雅普诺夫指数可以帮助我们理解网络的稳定性和动态行为。为了计算这些指数，本文采用了 QR 分解法，这是目前在计算李雅普诺夫谱中最为常用和有效的方法之一。

我们重点分析了在神经网络的训练过程中计算李雅普诺夫谱的表现。实验结果表明，李雅普诺夫指数可以作为一种有效的指标，用于评估网络的稳定性和预测训练过程中可能出现的数值问题。通过对李雅普诺夫指数的分析，我们可以提前发现并解决网络训练中的潜在问题，避免模型在训练后期出现不稳定或发散的现象。

插图索引

表格索引

参考文献

[1] Geist K, Parlitz U, Lauterborn W. Comparison of Different Methods for Computing Lyapunov Exponents[J/OL]. Progress of Theoretical Physics, 1990, 83(5): 875-893. https://doi.org/10.1143/PTP.83.875.

[2] von Bremen H F, Udwadia F E, Proskurowski W. An efficient qr based method for the computation of lyapunov exponents[J/OL]. Physica D: Nonlinear Phenomena, 1997, 101 (1): 1-16. https://www.sciencedirect.com/science/article/pii/S0167278996002163. DOI: https://doi.org/10.1016/S0167-2789(96)00216-3.

[3] Pascanu R, Mikolov T, Bengio Y. On the difficulty of training recurrent neural networks[A]. 2013. arXiv: 1211.5063.

[4] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift[A]. 2015. arXiv: 1502.03167.

[5] Vakilipourtakalou P, Mou L. How chaotic are recurrent neural networks?[A]. 2020. arXiv: 2004.13838.

[6] Ni A. Hyperbolicity, shadowing directions and sensitivity analysis of a turbulent three-dimensional flow[J/OL]. Journal of Fluid Mechanics, 2019, 863: 644–669. http://dx.doi.org/10.1017/jfm.2018.986.

[7] Ni A. Adjoint shadowing for backpropagation in hyperbolic chaos[J]. Nonlinearity, 2024, 37 (035009).

[8] Ni A. No-propagate algorithm for linear responses of random chaotic systems[A]. 2023. arXiv: 2308.07841.

[9] Storm L, Linander H, Bec J, et al. Finite-time lyapunov exponents of deep neural networks [A]. 2023. arXiv: 2306.12548.

[10] Ni A, Talnikar C. Adjoint sensitivity analysis on chaotic dynamical systems by non-intrusive least squares adjoint shadowing (nilsas)[J/OL]. Journal of Computational Physics, 2019, 395: 690–709. http://dx.doi.org/10.1016/j.jcp.2019.06.035.

致　　谢

总觉得未来还在远方，却未曾料到时光如梭。当我开始动笔写下致谢时，才意识到四年的大学生活即将结束，告别的时刻已经来临。这四年中，所有的遇见和经历都成为了最珍贵的回忆。愿我们离开校园后，都能成为更好的自己。

在这次论文创作过程中，我特别要感谢我的指导老师倪昂修。他不仅是我的良师，更是朋友。从选题到中期再到最终定稿，他一直耐心地指导我，对我提出的疑问及时解答，还关心我们的生活和工作，提供帮助和引导。他的尽职尽责让我深感铭记，感谢他帮助我顺利完成毕业设计和论文。我也非常感谢参与答辩工作的老师们，感谢你们抽出时间给予指导意见，让答辩对我意义非凡。

此外，我要感谢我的家人。虽然我们家不富裕，但父母都是勤奋的教师，二十年来对我的教育更多的是包容和理解，而非苛责和否定。在我心中，他们是世界上最伟大的人，给予我生命，教会我成长，尊重我的选择，是我最坚强的后盾。希望父母平安喜乐。

感谢我的朋友们，感谢你们在我写论文和毕业设计时给予的帮助。是你们陪我度过了这四年的大学时光，让平淡的生活充满乐趣。特别感谢夏斐然同学，四年来，你给我的生活带来了无尽的欢乐。愿大家前程似锦，在各自的领域发光发热。

最后，我想感谢自己。感谢过去努力的自己，这一路虽不易，但我始终坚持做自己。我们都应该活成自己喜欢的样子，做自己喜欢的事，与喜欢的人交往，接受平凡和不完美的自己。

踏入社会后，希望自己能保持初心，自由独立、自信勇敢，不必羡慕他人，也不依附他人，做一个心中有光的人。宇宙山河烂漫，人间点滴温暖都值得我们继续前行。

至此落笔，回头看却不走回头路，追风赶月莫停留，前方仍有荣光在，愿我们前路漫漫，前程似锦。

声　　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：谢泽旋　　日　期：2024年6月13日

附录 A　文献翻译 - 关于训练循环神经网络的困难性

①

A.1　关于训练循环神经网络的困难性

A.1.1　基本信息

A.1.1.1　作者

姓名	单位	电子邮件
Razvan Pascanu	Universite de Montreal	pascanur@iro.umontreal.ca
Tomas Mikolov	Brno University	t.mikolov@gmail.com
Yoshua Bengio	Universite de Montreal	yoshua.bengio@umontreal.ca

A.1.1.2　引用

项目	文本
arXiv 标识符	arXiv:1211.5063v2 [cs.LG] 16 Feb 2013

A.1.2　摘要

在正确训练循环神经网络时会面临两个众所周知问题，即 Bengio 等人（1994 年）详细介绍的梯度消失问题和梯度爆炸问题。在本文中，我们试图通过从分析学、几何学和动力系统的角度探讨这些问题来提高对潜在问题的理解。我们的分析被用来验证一个简单而有效的解决方案。本文提出了一种处理梯度爆炸的范数剪切策略和处理梯度消失问题的软约束方法，并在实验部分验证了我们的假设和提出的解决方案。

A.1.3　介绍

循环神经网络（RNN，如图 A.1）是 80 年代提出的一种为时间序列建模而生的神经网络模型（Rumelhart 等人，1986 年；Elman，1990 年；Werbos，1988 年）。

① 此章节内容为开题文献翻译，图片均为外文文献原图。

该网络的结构类似于标准的多层感知器，区别在于我们允许与时间延迟相关联的隐藏层之间的连接。通过这些连接，该模型可以保留关于过去输入的信息，使其能够发现数据中可能彼此相距遥远的事件之间的时间相关性（这是正确学习时间序列的一个关键属性）。虽然原则上循环神经网络是一个简单而强大的模型，但实践中很难进行正确的训练。这个模型显得笨拙的主要原因之一是梯度消失以及梯度爆炸问题（Bengio 等人，1994）。

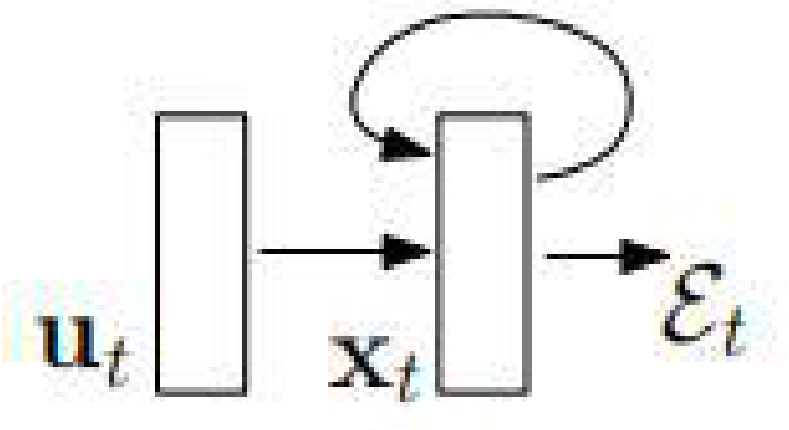

图 A.1　一个循环神经网络的示意图。隐藏层的循环连接允许信息从一个输入持续保存到另一个输入。

A.1.3.1　训练神经网络

考虑一个生成式循环神经网络，输入为 u_t，在时间阶段 t 时的状态为 x_t（如方程（A.1））. 在本文中的理论部分，为了提供条件更精准、更符合直觉的日常用例，我们有时会使用方程（A.11） 给出的特殊参数化。

$$x_t = F(x_{t-1}, u_t, \theta). \tag{A.1}$$

$$x_t = W_{rec}\sigma(x_{t-1}) + W_{in}u_t + b. \tag{A.2}$$

模型参数由权重矩阵 W_{rec}、偏差值 b 和输入矩阵 W_{in} 给出，其中 W_{in} 一般整理为 θ. x_0 由用户给出，设置为 0 或由学习得到，σ 是一个逐元素操作函数（通常为 *tanh* 或 *sigmoid* 函数）。代价 ϵ 衡量网络在特定任务上的表现，可以被拆分为每一步独立的代价 $\epsilon = \sum_{1\leqslant t\leqslant T} \epsilon_t$，其中 $\epsilon_t = \mathcal{L}(x_t)$.

一种计算必要梯度的方法为时域反向传播（BPTT），具体来说，循环模型表示为一个深的多层结构（并包含一个无范围限制的数目层），反向传播应用于展开模型上（如图 A.2）

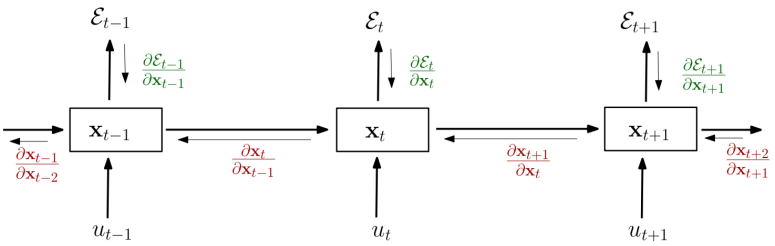

图 A.2　通过为每个时间步长创建一个模型的副本，及时展开递归神经网络。我们使用 x_t 表示网络在 t 时刻的隐藏状态，用 u_t 表示时间 t 时网络的输入，用 ϵ_t 表示截止 t 时刻的累计误差。

在这一点上我们将偏离经典的 BPTT 方程，并重写梯度（见方程（A.3）、（A.4）和（A.5））以更好地突出梯度爆炸问题。这些方程通过将梯度以元素之和的形式写出而得到。

$$\frac{\partial \epsilon}{\partial \theta} = \sum_{1 \leqslant t \leqslant T} \frac{\partial \epsilon_t}{\partial \theta}. \tag{A.3}$$

$$\frac{\partial \epsilon_t}{\partial \theta} = \sum_{1 \leqslant k \leqslant t} \left(\frac{\partial \epsilon_t}{\partial x_t} \frac{\partial x_t}{\partial x_k} \frac{\partial^+ x_k}{\partial \theta}\right). \tag{A.4}$$

$$\frac{\partial x_t}{\partial x_k} = \prod_{t \geqslant i > k} \frac{\partial x_i}{\partial x_{i-1}} = \prod_{t \geqslant i > k} W_{rec}^T diag(\sigma'(x_{i-1})). \tag{A.5}$$

$\frac{\partial^+ x_k}{\partial \theta}$ 表示状态 x_k 关于 θ “立刻的”偏微分，换句话说，x_{k-1} 被当作关于 θ 的常函数。特别地，考虑方程（A.2），任何第 i 行的矩阵 $\frac{\partial^+ x_k}{\partial W_{rec}}$ 仅仅是 $\sigma(x_{k-1})$. 方程（A.5）也为方程（A.11）的参数化提供了 Jacobi 矩阵 $\frac{\partial x_i}{\partial x_{i-1}}$ 的形式，σ' 以逐元素的形式计算了 σ 的微分。

注意到（A.3）中的每一个 $\frac{\partial \epsilon_t}{\partial \theta}$ 都是形式相同的和式，这些单独项的行为决定了和式的行为。因此之后我们主要关注这样的一个通用术语，在不会引起歧义的情况下将其简单地称为“梯度”。

任何梯度元素 $\frac{\partial \epsilon_t}{\partial \theta}$ 也是一个和式（如方程（A.4）），我们将其称为时间贡献或时间分量。可以看到，每一个这样的时间贡献 $\frac{\partial \epsilon_t}{\partial x_t} \frac{\partial x_t}{\partial x_k} \frac{\partial^+ x_k}{\partial \theta}$ 测量了 θ 在第 k 步如何影响之后的第 $t > k$ 步。因子 $\frac{\partial x_t}{\partial x_k}$（方程（A.5））及时传递了 t 时刻第误差至第 k 步。我们可以进一步宽松地区分长期和短期的贡献值，其中“长期”指 $k \ll t$，“短期”指其它。

A.1.4 梯度爆炸和梯度消失

如 Bengio 所介绍（1994 年），“梯度爆炸”的问题指的是训练时梯度的范数过大增长。这样的问题由长期的元素指数级别大小“爆炸”导致。“梯度消失”则是指相反的情况——元素长期以指数级别速度趋近于零，从而导致模型几乎无法学习时间跨度遥远的事件之间的相关性。

A.1.4.1 机制

为了理解这一现象，我们需要观察每个时间分量的形式，特别是以 $t-k$ Jacobi 矩阵乘积形式存在的矩阵因子 $\frac{\partial x_t}{\partial x_k}$（见方程（A.5））. 同样地，$t-k$ 实数的乘积可以收缩到零或爆炸到无穷大，这个矩阵也会。

接下来我们需要尝试将这些直觉形式化（扩展 Bengio 等人在论文中所做的类似推导（1994 年），在论文中他们只考虑了一元情况）

如果我们考虑这个模型的一个线性版本（如方程（A.11） 中将 σ 代入特征方程），可以使用幂迭代法正式分析 Jacobi 矩阵的乘积，得到梯度爆炸的严格条件（这些条件的详细推导见补充材料）。对于循环权重矩阵的最大特征值 λ_1 而言，只要其小于 1，长期分量就会消失（随着 $t \to \inf$），而要使梯度爆炸，必须确保它大于 1.

我们可以将这些结果推广到非线性函数 σ，其中 σ 的绝对值 $\sigma'(x)$ 是有界的（不妨设 $\gamma \in \mathbb{R}$ 为上确界），从而有 $\|diag(\theta'(x_k))\| \leqslant \gamma$.

我们首先证明 $\lambda_1 < \frac{1}{\gamma}$ 是充分的条件，事实上，这时候 λ_1 是能够出现梯度消失问题的循环权重矩阵 W_{rec} 最大的特征值。注意到我们假设了（A.11） 的参数化。Jacobi 矩阵 $\frac{\partial x_{k+1}}{\partial x_k}$ 由 $W_{rec}^T diag(\sigma'(x_k))$ 给出。这个 Jacobi 矩阵的 2-范数范围被两个矩阵的范数限制（方程（A.6））. 由于我们的假设，这推出它比 1 小。

$$\forall k, \|\frac{\partial x_{k+1}}{\partial x_k}\| \leqslant \|W_{rec}^T\| \|diag(\sigma'(x_k))\| < \frac{1}{\gamma}\gamma < 1. \tag{A.6}$$

令 $\eta \in \mathbb{R}$ 使得 $\forall k, \|\frac{\partial x_{k+1}}{\partial x_k}\| \leqslant \eta < 1$，$\eta$ 的存在性由方程（A.6） 给出，通过变换 i 的值，我们发现

$$\frac{\partial \epsilon_t}{\partial x_t}(\prod_{i=k}^{t-1} \frac{\partial x_{i+1}}{\partial x_i}) \leqslant \eta^{t-k} \frac{\partial \epsilon_t}{\partial x_t}. \tag{A.7}$$

因为 $\eta < 1$，因此根据（A.7），长期贡献（即 $t-k$ 较大）以指数级别趋近于零。

通过引入这个证明，我们得到了梯度爆炸的必要条件，也就是最大特征值 λ_1

比 $\frac{1}{\gamma}$ 大（否则长期数值不会爆炸，而是逐渐消失）。对于 $tanh$ 函数，我们有 $\gamma = 1$；对于 $sigmoid$ 函数，我们有 $\gamma = \frac{1}{4}$.

A.1.4.2 绘制动态系统的相似性

我们可以通过动态系统的视角来进一步理解梯度爆炸和梯度消失问题，就像之前 Doya（1993 年），Bengio 等人（1993 年）所做的工作。

我们建议阅读 Strogatz（1994 年）对动力系统理论的正式且详细的处理。对于任何参数分配，取决于初始状态 x_0 和状态 x_t，在映射 F 的重复应用下，自治动态系统收敛到几种可能吸引子状态之一（如点吸引子，尽管存在其他类型的吸引子）。该模型也可以在混沌状态下运行，在这种情况下，下面的一些观察结果可能不成立，但在这里没有深入讨论。吸引子描述了模型的渐近行为。状态空间被划分为吸引盆地，每个吸引子一个。如果模型在一个吸引盆地中启动，则随着 t 的增长，模型将收敛到相应的吸引子。

动力系统理论告诉我们，随着变化的缓慢，渐近行为几乎在任何地方都平滑地变化，除了某些发生剧烈变化的关键点（新的渐近行为在拓扑上不再与旧的等价）。这些点被称为分岔边界，它们是由出现、消失或改变形状的吸引子引起的。

Doya（1993 年）假设这种分叉交叉可能会导致梯度爆炸。我们想将这一观察结果扩展到梯度爆炸的一个合理条件，因此，我们将重复使用 Doya（1993 年）的单隐藏单元模型（如图 A.3）

x 轴表示参数 b，y 轴表示渐近状态 $x \to \text{inf}$. 粗体线跟随着点吸引子 $x \to \text{inf}$ 移动，随 b 变化。在 b_1 我们有一个分岔边界，其中一个新的吸引子出现（当 b 从 1 减小），而在 b_2 我们还有另一个方法，它会导致这两个吸引子之一的消失。在间隔 (b_1, b_2) 我们处于一个丰富的状态，其中有两个吸引子，它们之间的边界位置的变化，当 b 变化，用虚线追踪出来。向量场（灰色虚线箭头）描述了在该区域初始化网络时状态 x 的演化。

我们证明了有两种类型的事件可能导致 x_t 随着 $t \to \text{inf}$ 的大变化。一个是跨越吸引盆地之间的边界（用未填充的圆表示），而另一个是跨越分叉边界（未填充的圆）。对于大的 t，x_t 即使 b 的变化很小（因为 b 的变化也会被不同的吸引子吸引），这也会导致很大的梯度。然而，要使梯度爆炸，既没有必要也不可能跨越一个分岔，因为分岔是不可能在局部没有缺陷的全局事件。学习在参数状态空间中追踪出一条路径。如果我们在一个分岔边界，但模型的状态是这样的，它在一个吸引子的吸引盆地，当分叉交叉时不会改变形状或消失，那么这个分岔将不

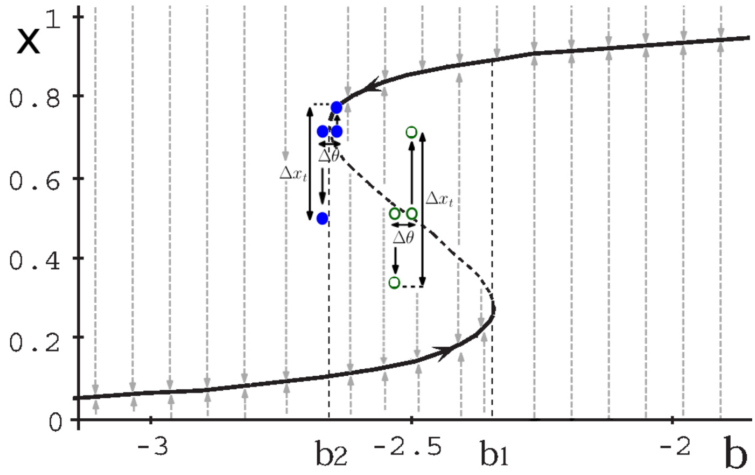

图 A.3　单个隐藏单元 RNN 的分叉图（混合循环权重为 5.0，可调偏差 b，为 Doya（1993 年）介绍的例子），参见文本

会影响学习。

跨越吸引盆地之间的边界是一个局部事件，而梯度的爆炸是必然的。如果我们假设跨越到一个新兴吸引子或从一个消失（由于分支）符合穿越一些吸引子之间的边界，我们可以制定一个可靠的条件梯度爆炸封装的观察 Doya（1993 年），扩展也正常跨越不同盆地之间的边界的吸引力。注意，在图 A.4 中，只有两个具有分叉的 b 值，但有一个整个范围的值可以有一个边界交叉。先前分析的另一个局限性是，它们只考虑自治系统，并假设观察结果适用于输入驱动的模型。在 Bengio 等人（1994 年）中通过假设输入是有界噪声来处理它。这种方法的缺点是，它限制了人们对输入的推理方式。在实践中，输入应该驱动动力系统，能够使模型处于某种吸引子状态，或者在某些触发模式出现时，将其踢出吸引盆地。

我们建议通过将输入折叠到地图中，将我们的分析扩展到输入驱动模型。我们考虑映射 F 的族 t，其中我们应用了一个不同的 F_t 在每一步。直观地说，为了使梯度爆炸，我们需要与之前相同的行为，其中（至少在某个方向上）映射 $F_1, F_2, ..., F_t$ 同意并改变方向。图 A.4 描述了这种行为。

对于方程(A.11) 提供的特殊参数化，我们可以通过分解映射 F 为修正映射 $\widetilde{F}$ 和随时间变化的量 U_t 来进一步进行类比。$F(x) = W_{rec}\sigma(x) + b$ 表示一个无输入的循环神经网络，$U_t(x) = x + W_{in}u_t$ 描述了输入造成的影响。这由图 A.5 展示。既然 U_t 随着时间改变，它不能用于分析标准动力系统的工具，但 $\widetilde{F}$ 可以。这说明当一个盆地之间的边界被 $\widetilde{F}$ 跨越时，状态会向着一个不同的吸引子前进，使得较大的 t 可能导致 x_t 的较大差异（除非输入的映射 U_t 与之相反）。因此研究 $\widetilde{F}$ 的渐进行为可以提供关于此事件发生地点的有用信息。

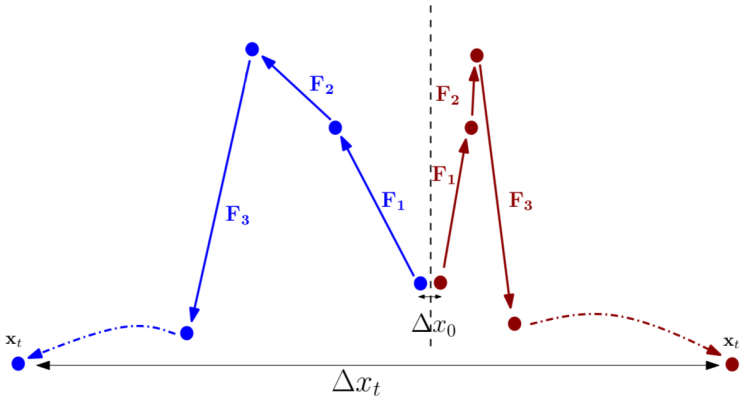

图 A.4　这个折线图预测了 x_t 的变化，即 Δx_t 可能会产生于一个值较小的 Δx_0. 蓝色与红色（左边和右边）的对比由相同的映射 $F_1, F_2, ...$ 产生，不同之处仅在于初始状态。

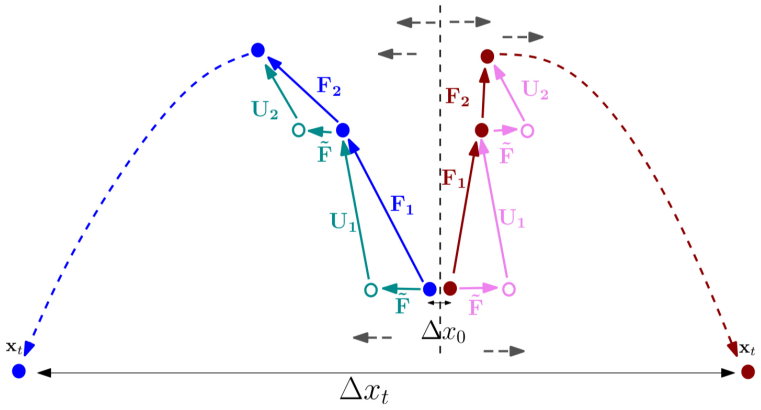

图 A.5　描述如何分解 $F_1, ..., F_t$ 为一个常值映射 $\widetilde{F}$ 和映射 $U_1, U_2, ..., U_t$. 这个虚竖线代表盆地喜好之间的边界，直虚箭头在边界上指向 $\widetilde{F}$ 映射的方向，这个图像是图 A.4 的扩展。

从动力学系统的角度来看，关于消失的梯度的一个有趣的观察结果如下。如果因子 $\frac{\partial x_1}{\partial x_k}$ 趋于零（对于 $t-k$ 很大），这意味着 x_t 不依赖于 x_k （如果我们改变 x_k 一些的 Δ，x_t 保持不变）。这就转化为 x_t 处的模型接近收敛到某个吸引子（它将从 x_k 附近的任何地方到达）。

A.1.4.3 几何解释

让我们考虑一个简单的单隐单元模型（方程(A.8)），其中我们提供了一个初始状态 x_0 并训练模型在 50 步后得到一个特定的目标值。请注意，为了简单起见，我们假设没有输入。

$$x_t = \omega\sigma(x_{x-1}) + b. \tag{A.8}$$

图 A.6 展示了表面误差 $\epsilon_{50} = (\sigma(x_{50}) - 0.7)^2$，其中 $x_0 = .5$ 且 σ 是 *sigmoid* 函数。

我们可以通过将其简化为 $b = 0$ 的线性形式（σ 是特征函数）以更加轻松地分析这个模型。$x_t = x_0\omega^t$ 因而可以根据 $\frac{\partial x_t}{\partial \omega} = tx_0\omega^{t-1}$ 和 $\frac{\partial^2 x_t}{\partial \omega^2} = t(t-1)x_0\omega^{t-2}$ 推导出首次、第二次及之后的微分梯度爆炸的时间。

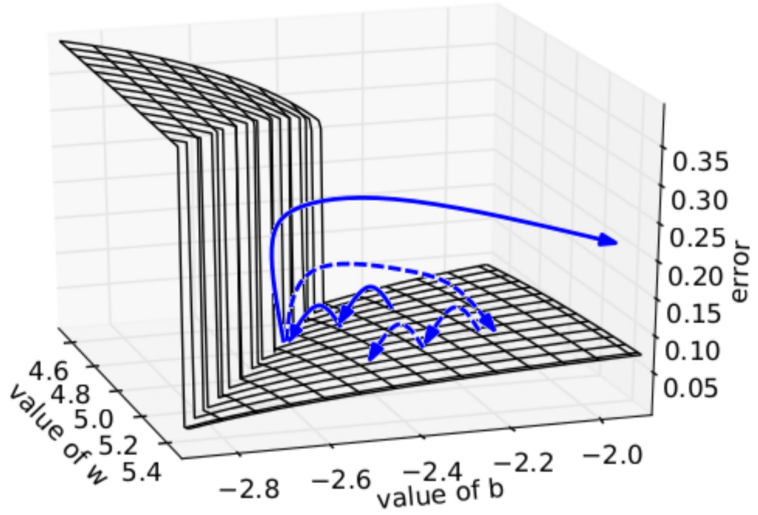

图 A.6　我们绘制了单个隐单元循环网络的误差面，突出了高曲率壁的存在。实线描述了梯度下降可能遵循的标准轨迹。使用虚线箭头，图表显示了当梯度的范数超过一个阈值时，如果梯度被重新缩放到一个混合的大小会发生什么。

一般地，梯度爆炸单独发生于部分方向如 v，也就是说，在这些情形下，存在一个向量 v 使得 $\frac{\partial \epsilon_t}{\partial \theta} v \geqslant C\alpha^t$，其中 $C, \alpha \in \mathbb{R}$ 且 $\alpha > 1$. 对于线性情形（σ 为特征函数），v 是 W_{rec} 的最大特征值对应的特征向量。如果边界是紧的，我们假设对于一般情况，当梯度爆炸同时 v 也弯曲，从而产生一个误差表面的边界，如图

A.6 所示。

如果这成立，那么它给了我们一个简单的梯度爆炸问题的解决方案，如图 A.6.

如果曲率的梯度和前导特征向量都与爆炸方向 v 对齐，则误差面有一个垂直于 v 的陡峭壁（因此也垂直于梯度）。这意味着，当随机梯度下降（SGD）到达墙壁并进行梯度下降步骤时，它将被迫以垂直于陡峭墙壁的方式移动穿过山谷，可能离开山谷并破坏学习过程。

图 A.6 中的虚线箭头对应于忽略这个大步骤的规范，确保模型保持靠近墙壁。关键的见解是，当梯度爆炸时所采取的所有步骤都与 v 对齐，而忽略了其他下降方向（换句话说，该模型垂直于墙壁运动）。因此在墙上，沿梯度方向的小标准步只是把我们推回到墙之外更平滑的低曲率区域，而规则的梯度步会把我们带得非常远，从而减缓或阻止进一步的训练。相反，通过一个有界的步骤，我们回到了在墙附近的平滑区域，在那里 SGD 可以自由地探索其他下降方向。

在这种情况下，对经典的高曲率谷的重要补充是，我们假设谷是宽的，因为我们在墙周围有一个很大的区域，如果我们着陆，我们可以依靠一阶方法向局部极小值移动。这就是为什么仅仅裁剪梯度可能是合理的，而不需要使用二阶方法。注意，该算法在梯度增长率和曲线的增长率不同时依旧可以运行（在这种情况下，由于梯度和曲率之间的比率依旧会爆炸，二阶方法就会爆炸）。

我们的假设也有助于理解与 Hessian-Free 方法相比，其他二阶方法最近的成功。Hessian-Free 和其他排序算法之间有两个关键的区别。首先，它使用了完整的 Hesse 矩阵，因此可以处理不一定是轴对齐的爆炸方向。其次，它在每个更新步骤之前计算 Hesse 矩阵的新估计，并可以考虑曲率的突变（如我们的假设提出的），而大多数其他方法使用平滑假设，也即取许多第二排名的信号求平均。

A.1.5 梯度爆炸和梯度消失的处理

A.1.5.1 先前的方案

对循环权值使用 L1 或 L2 惩罚可以帮助解决梯度爆炸。假设参数初始化的值较小，则 W_{rec} 的光谱半径可能小于 1，由此可以得出梯度不会爆炸（见第 2.1 节中发现的必要条件）。正则化项可以确保在训练过程中，光谱半径永远不会超过 1。这种方法将模型限制在一个简单的范围内（在原点有一个单点吸引子），其中插入到模型中的任何信息都必须在时间上以指数速度消失。在这种情况下，我们不能训练一个发电机网络，也不能表现出长期的记忆痕迹。

Doya（1993 年）提出对模型进行预编程（在正确的制度下初始化模型）或使用教师强制使用。第一个建议假设，如果模型从一开始就表现出与目标所要求的相同的渐近行为，那么就不需要跨越一个分岔边界。缺点是，人们不可能总是知道所需的渐近行为，而且，即使这些信息是已知的，在这个特殊的情况下初始化一个模型也不是微不足道的。我们还应该注意到，这种初始化并不能阻止跨越吸引盆地之间的边界，如图所示，即使没有跨越分岔边界，也可能发生。

教师强迫是一个更有趣的方法，但又不是一个很容易理解的解决方案。它可以被看作是在正确的体系和正确空间范围下初始化模型的一种方式。研究表明，在实践中，它可以减少梯度爆炸的机会，甚至允许训练生成器模型或使用无限内存的模型（Pascanu 和 Jaeger，2011 年；Doya 和 Yoshizawa，1991 年）。一个重要的缺点是，它需要在每个时间步长中都确定一个目标。

在 Hochreiter 和 Schmidhuber（1997 年）；Graves 等人（2009 年）提出了一个解决梯度消失的方法，其中模型的结构可变。具体地说，它引入了一组特殊的单元，称为 LSTM 单元，它们是线性的，且与自身有一个固定为 1 的循环连接。进入单元和流出单元的信息流由输入和输出门保护（它们的行为被学习）。这个基本结构有几个变体，但构造这个解决方案并没有明确地解决爆炸性梯度的问题。

Sutskever 等人（2011 年）结合 Hessian-Free 优化与结构阻尼，这是 Hesse 矩阵的一种特殊的阻尼策略。这种方法似乎能很好地处理梯度消失，尽管仍然缺少更详细的分析。据推测，这种方法之所以有效，是因为在高维空间中，长期分量与短期分量正交的可能性很高。这将允许 Hesse 矩阵独立地重新调整这些分量。在实践中，我们不能保证这个性质是否成立。如第 2.3 节所述，这种方法也能够处理梯度爆炸。结构阻尼是一种增强，当参数变化 $\Delta\theta$ 较小时，迫使状态变化很小。这就需要 Jacobi 矩阵 $\frac{\partial x_t}{\partial \theta}$ 范数较小，从而进一步帮助解决爆炸性的梯度。事实证明它在长序列上训练循环神经模型效果很好，这表明虽然曲率可能与梯度同时爆炸，但它可能不会以相同的速度增长，因此不适合处理梯度爆炸。

回波状态网络（Lukoševičius 和 Jaeger，2009 年）通过不学习循环权值和输入权值来避免梯度爆炸和梯度消失问题。它们是从手工制作的分布图中取样的。由于通常循环权值的最大特征值小于 1，因此输入模型的信息必须以指数速度消失。这意味着这些模型不能轻易地处理长期的依赖关系，即使其原因与消失的梯度问题略有不同。对经典模型的扩展是由泄漏积分单元表示的（Jaeger 等人，2007 年），其中 $x_k = \alpha x_{k-1} + (1-\alpha)\sigma(W_{rec}x_{k-1} + W_{in}u_k + b)$.

虽然这些单元可以用于解决 Hochreiter 和 Schmidhuber（1997 年）提出的学

习长期依赖的标准基准（见（Jaeger，2012 年)，但它们更适合处理低频信息，因为它们充当低通滤波器。因为大部分的重量都是随机分布的。目前还不清楚人们需要什么样的模型来解决复杂的现实世界的任务。

我们将特别注意到 Tomas Mikolov 在他的博士论文（Mikolov，2012 年）中提出的方法（并在语言建模的最新结果中隐含地使用（Mikolov 等人, 2011 年))。它涉及到按元素方式剪切梯度的时间分量（当它超过绝对值的混合阈值时，剪切一个项)。剪切在实践中已经被证明做得很好，它构成了我们的方法的支柱。

A.1.5.2 梯度下降

正如 2.3 节中介绍的一样，处理梯度范数突然增加的一个简单机制是，当它们超过一个阈值时重新调整它们（参见算法 1)。

算法 1（伪代码)，用于在任何爆炸时刻范数剪辑梯度

$\hat{g} \leftarrow \frac{\partial \epsilon}{\partial \theta}$

如果 $\|\hat{g}\| \geqslant threshold$ 则:

$\hat{g} \leftarrow \frac{threehold}{\|\hat{g}\|}\hat{g}$

结束循环

这个算法非常类似于托马斯 · 米科洛夫提出的，我们只是偏离最初的提议，试图提供一个更好的理论基础（确保我们总是在下降方向对当前小批)，尽管在实践中两个变体行为相似。

所提出的剪切易于实现和计算效果显著，但它确实引入了一个额外的超参数，即阈值。设置这个阈值的一个很好的启发式方法是查看在通常的大量更新中的平均范数的统计数据。在我们的实验中，我们注意到，对于给定的任务和模型大小，训练对这个超参数不是很敏感，即使在相当小的阈值下，算法也表现得很好。该算法也可以被认为是基于梯度的范数来适应学习速率。与其他学习速率自适应策略相比，后者侧重于通过收集梯度上的统计数据来提高收敛性（例如 Duchi 等人 2011 年)、或 Moreira 和 Fiesler，1995 年的概述)，我们依赖于瞬时梯度。这意味着我们可以处理规范中非常突然的变化，而其他方法将不能这样做。

A.1.5.3 梯度消失的正规化

为了解决消失的梯度问题，我们选择使用一个正则化项，该项表示对参数值的偏好，以使反向传播的梯度既不会显著增加也不会减少。我们的直觉是，增加 t 时刻 $\frac{\partial x_t}{\partial x_k}$ 范数的误差对所有输入 $u_t, u_{t+1}, ..., u_k$ 更为敏感（其中 $\frac{\partial x_t}{\partial x_k}$ 是 $\frac{\partial \epsilon_t}{\partial u_k}$ 的一个因子)。实际上，其中一些输入在实践中与 t 时刻的预测无关，表现为网络需要

学习忽略的噪声。然而，网络不能学会忽略这些不相关的输入，除非有一个错误信号。这两个问题似乎不能同时解决，因此我们自然地期望需要迫使网络增加规范的代价更大的错误（由于不相关的输入条目），然后等待它学习忽略这些不相关的输入。这提示我们在遵循误差 E 的下降方向时（例如，尝试使用二阶方法），我们需要通过正则化项来执行这一操作。我们在下文提出的正则化器更倾向于在返回时间时保持误差信号的范数。

我们提出的正则化算子更倾向于在返回时间时保持误差信号的范数：

$$\Omega = \sum_k \Omega_k = \sum_k \left(\frac{\left| \frac{\partial \epsilon}{\partial x_{k+1}} \frac{\partial\, x_{k+1}}{\partial x_k} \right|}{\left| \frac{\partial \epsilon}{\partial x_{k+1}} \right|} - 1 \right)^2 . \tag{A.9}$$

为了在计算上更有效，我们仅使用 Ω 关于 W_{rec} 的“直接”偏导数（在计算偏导数时，假设在计算 Ω_k 时，x_k 和 $\frac{\partial \epsilon}{\partial x_{k+1}}$ 分别是关于 W_{rec} 的常数），见方程（A.10）。请注意我们使用的是参数化的（A.11），因为我们可以高效地从 BPTT 中获得 $\frac{\partial \epsilon}{\partial x_k}$ 的值。我们使用 Theano 来计算梯度（Bergstra 等人，2010 年；Bastien 等人，2012 年）：

$$\frac{\partial^+ \Omega}{\partial W_{rec}} = \sum_k \frac{\partial^+ \Omega_k}{\partial W_{rec}} = \sum_k \frac{\partial \left(\frac{\left| \frac{\partial \epsilon}{\partial x_{k+1}} W_{rec}^T diag(\sigma'(x_k)) \right|}{\left| \frac{\partial \epsilon}{\partial x_{k+1}} \right|} - 1 \right)^2}{\partial W_{rec}}. \tag{A.10}$$

请注意，我们的正则化项只强制使用 Jacobi 矩阵 $\frac{\partial x_{k+1}}{\partial x_k}$ 以保证相关方向的误差项 $\frac{\partial \epsilon}{\partial x_{k+1}}$ 的范数不变，而不是所有方向的误差项不变（换句话说，我们不要求所有特征值接近于 1）。第二个观察是我们使用了一个软性约束条件，因此我们不能保证误差信号的范数被保留下来。如果这些 Jacobi 矩阵的范数爆炸（随着 $t-k$ 的增加），可能导致梯度的问题，我们需要处理，正如 3.2 节所述。从动力系统的角度看，这可以解释为防止梯度消失，我们将模型推离吸引子（使其不收敛，从而避免梯度消失），并使其更接近边界，从而更有可能出现梯度爆炸。

A.1.6 实验与结果

A.1.6.1 病态合成问题

正如 Martens 和 Sutskever（2011 年）所做的那样，我们解决了 Horchreiter 和 Schmidhuber（1997 年）提出的病理问题，这些问题需要学习长期的相关性。我们建议读者参考这篇原始论文对任务的详细描述，并参考补充材料来对实验设置的完整描述。

时间序列问题　我们将时间顺序问题作为典型的病理问题，并将我们的结果扩展到其他提出的任务。输入是一长串离散的符号。在两个时间点（在序列的开始和中间）发出 $\{A, B\}$ 中的一个符号。该任务包括对序列末尾的顺序（可能的选择有 AA, AB, BA, BB）。

图 A.7 展示了标准 SGD、SGD-C（带裁剪策略的 SGD 增强）和 SGD-CR（带裁剪策略和正则化项的 SGD）的成功率。请注意，对于长度超过 20 的序列，消失梯度问题确保了 SGD 和 SGD-C 算法都不能解决该任务。x 轴是基于对数尺度的。

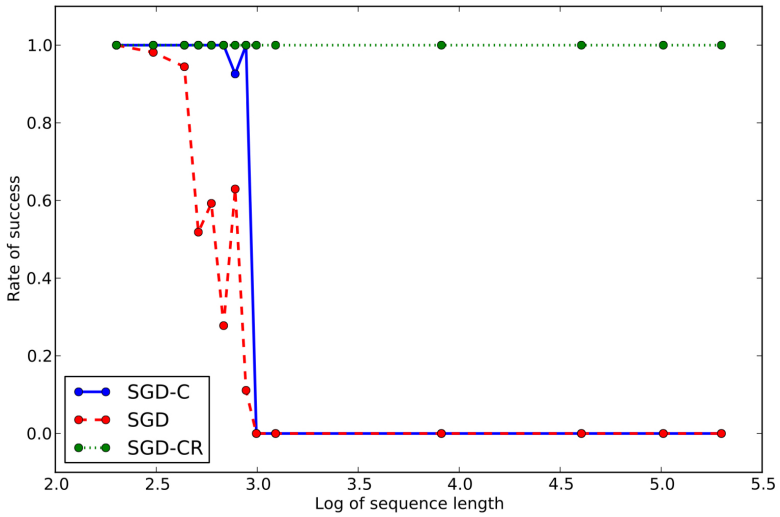

图 A.7　解决时间顺序的成功率问题与序列长度的关系图。详情参见文本。

这项任务的研究为我们提供了证据，表明爆炸性梯度与需要长时间记忆轨迹的任务密切相关。我们了解到，初始模型运行在单吸引子制度（即 $\lambda_1 < 1$），其中内存量由 λ_1 控制。更多的内存表示更大的光谱半径，当这个值超过某个阈值时，模型就进入了可能发生梯度爆炸的丰富状态。从图 A.7 中可以看到，只要不涉及梯度消失问题，解决梯度爆炸问题就能确保更高的成功率。

有趣的是，经过训练的模型能够推广到新的序列，其长度可以是训练期间所见序列长度的两倍。

其它病理性任务　SGD-CR 在解决其他病理性任务方面也表现出色，以下列出的任务中，几乎所有都取得了 100% 成功，仅在其中一个任务中出现了部分成功。这些任务包括 Hochreiter 和 Schmidhuber（1997 年）提出的添加问题、乘法问题、3 位时间顺序问题、随机排列问题以及二元无噪声记忆问题（其中模式需要记忆 5 位长度，且包含超过 20 位信息；详见 Martens 和 Sutskever（2011 年））。对于

前四个问题，我们使用了一个长度高达 200 的单一模型，而对于无噪声记忆问题，我们为每个序列长度（50、100、150 和 200）使用了一个不同的模型。在这些任务中，最具挑战性的问题是随机排列问题，其中只有一条路径取得了成功。值得注意的是，在所有情况下，我们观察到成功地泛化到比训练序列更长的序列。在大多数情况下，这些结果在成功率方面优于 Martens 和 Sutskever（2011 年）的结果，尤其是在处理比 Hochreiter 和 Schmidhuber（1997 年）更长序列时，与 Jaeger（2012 年）相比，它们也可以被推广到更长的序列。

Table 1. Results on polyphonic music prediction in negative log likelihood per time step. Lower is better.

DATA SET	DATA FOLD	SGD	SGD+C	SGD+CR
PIANO-MIDI.DE	TRAIN	6.87	**6.81**	7.01
	TEST	7.56	7.53	**7.46**
NOTTINGHAM	TRAIN	3.67	**3.21**	3.24
	TEST	3.80	3.48	**3.46**
MUSEDATA	TRAIN	8.25	6.54	**6.51**
	TEST	7.11	7.00	**6.99**

Table 2. Results on the next character prediction task in entropy (bits/character)

DATA SET	DATA FOLD	SGD	SGD+C	SGD+CR
1 STEP	TRAIN	1.46	**1.34**	1.36
	TEST	1.50	1.42	**1.41**
5 STEPS	TRAIN	N/A	3.76	**3.70**
	TEST	N/A	3.89	**3.74**

A.1.6.2 自然问题

我们致力于解决复调音乐预测任务，使用了来自 Piano-midi.de、Nottingham 和 MuseData 数据集（Boulanger-Lewandowski 等人，2012 年）以及在宾夕法尼亚大学 Treebank 数据集上的字符级别语言建模（Mikolov 等人，2012 年）。此外，我们还研究了任务的一种修改版本，其中模型需要预测未来第 5 个字符，而不是下一个字符。我们的假设是，在解决这个修改后的任务时，长期相关性比短期相关性更为重要，因此我们的正则化术语可能更有助于性能提升。

表 1 中所报告的训练和测试分数是每个时间步的平均负对数似然值。在三次运行中，我们对除了正则化因子和剪切阈值之外的所有超参数进行了调整。SGD-CR 在所有复调音乐预测任务中表现出改进，除了在博物馆数据上，此处我们获得了与 Bengio 等人相同的性能水平。尽管使用了不同的架构，表 2 展示了语言

建模任务的结果（每个字母的位）。

这些结果表明，剪切梯度不仅解决了一个优化问题，而且在训练误差和测试误差都普遍提高的情况下，其行为更符合正则化器的特性。我们的研究结果与 Mikolov 等人（2012 年）在宾夕法尼亚 Treebank 数据库上的最新水平相匹配，尽管我们使用了类似但不同的剪切算法，从而为两者的相似性提供了证据。正则化模型在性能上与 Hessian-Free 训练模型一样优秀。

值得注意的是，通过引入我们提出的正则化项，我们甚至能够改进测试错误，即使在任务中短期相关性占主导地位。

A.1.7 5. 总结和结论

我们从不同的角度探讨了梯度爆炸和消失问题，通过这些视角能够更全面地理解这两个挑战。为了应对梯度爆炸的问题，我们提出了一种解决方案，其中包括对梯度爆炸进行裁剪，以限制其范数在过大时。该算法的设计理念是基于梯度爆炸时，曲率和高阶导数也可能呈爆炸趋势，使得误差面出现一种特殊的模式，即具有陡峭壁的山谷。而针对梯度消失问题，我们引入了一个正则化项，以确保误差信号在随时间推移时不会完全消失。这个正则化项迫使 Jacobi 矩阵 $\frac{\partial x_i}{\partial x_{i-1}}$ 在相关方向上保持范数。在实际应用中，我们通过实验证明了这些解决方案在我们考虑的病理合成数据集、复调音乐预测以及语言建模任务上均提升了性能。

A.1.8 致谢

我们也要感谢 Theano 开发团队（特别是 Frederic Bastien、Pascal Lamblin 和 James Bergstra）的帮助。我们感谢 NSERC，FQRNT，CIFAR，RQCHP 和计算加拿大提供的资源。

A.1.9 参考文献

（本部分略）

A.1.10 从分析的角度分析梯度爆炸和梯度消失

$$x_t = W_{rec}\sigma(x_{t-1}) + W_{in}u_t + b. \tag{A.11}$$

让我们考虑为方程（A.11） 中的参数线性化版本而生的术语 $g_k^T = \frac{\partial \epsilon_t}{\partial x_t}\frac{\partial x_t}{\partial x_k}\frac{\partial^+ x_k}{\partial \theta}$

（也即将 σ 加入特征方程）并考虑 t 趋于无穷的情况并设 $l = t - k$。从而我们有：

$$\frac{\partial x_t}{\partial x_k} = (W_{rec}^T)^l. \tag{A.12}$$

通过基于一般幂迭代方法可以看到，特定条件下，$\frac{\partial \epsilon_t}{\partial x_k}(W_{rec}^T)^l$ 以指数级别增长。

证明设 W_{rec} 的特征值为 $\lambda_1, \lambda_2, ..., \lambda_n$，其中 $|\lambda_1| > |\lambda_2| > ... > |\lambda_n|$ 并且相应的特征向量 $q_1, q_2, ..., q_n$ 是一组基向量。可以将向量 $\frac{\partial \epsilon_t}{\partial x_t}$ 写为这个形式：$\frac{\partial \epsilon_t}{\partial x_t} = \sum_{i=1}^{N} c_i q_i^T$.

如果 j 使得 $c_j \neq 0$ 且对任意的 $j' < j$，$c_{j'} = 0$，由 $q_i(W_{rec}^T)^l = \lambda_i^l q_i^T$ 可得：

$$\frac{\partial \epsilon_t}{\partial x_t}\frac{\partial x_t}{\partial x_k} = c_j \lambda_j^l q_j^T + \lambda_j^l \sum_{i=j+1}^{n} c_i \frac{\lambda_i^l}{\lambda_j^l} q_i^T \approx c_j \lambda_j^l q_j^T. \tag{A.13}$$

由 $|\frac{\lambda_i}{\lambda_j}| < 1$ $(i > j)$，即得 $\lim_{l \to \inf} |\frac{\lambda_i}{\lambda_j}|^l = 0$. 若 $|\lambda_j| > 1$，则 $\frac{\partial x_t}{\partial x_k}$ 随着 l 的增长指数级增长，且增长方向沿着 q_j.

该证明为简单起见，假设了 W_{rec} 是对角矩阵，即便使用 W_{rec} 的 Jordan 标准型可以通过考虑最大特征值以外的特征值对应的特征向量以扩展此证明。

此结果提供了一个梯度爆炸的必要条件，即 W_{rec} 的谱半径（最大特征值的模）要大于 1。

如果 q_j 不在 $\frac{\partial^+ x_k}{\partial \theta}$ 零空间中，则整个时间组件随 l 呈指数增长。这可以轻易地超过整个梯度。如果我们用 W 的特征值分解重写它，将得到：

$$\frac{\partial \epsilon_t}{\partial \theta} = \sum_{j=1}^{n}(\sum_{i=k}^{t} c_j \lambda_j^{t-k} q_j^T \frac{\partial^+ x_k}{\partial \theta}). \tag{A.14}$$

此时，我们可以选择 j 和 k，使得在取最大的 $|\lambda_j|$ 的过程中，$c_j q_j^T \frac{\partial^+ x_k}{\partial \theta}$ 的范数始终不为零。如果对于选中的某个 j，满足 $|\lambda_j| > 1$，那么 $\lambda_j^{t-k} c_j q_j^T \frac{\partial^+ x_k}{\partial \theta}$ 将成为和式。由于这种形式随着 t 的增长呈指数级增长，相同的情况将在和式上发生。

A.1.11 实验准备

请注意，我们所有超参数的选择都基于它们在验证集上的性能，经过了网格搜索的调优过程。

A.1.11.1 病态合成任务

在所有任务中，我们采用了相似的成功标准，即模型在一批包含 10,000 个测试样本的数据集上的错误率不应超过 1%. 对于离散符号，我们使用了一热编码表

示法，而在回归情况下，对于给定序列的预测被视为成功，只要误差小于 0.04。

求和问题 输入是由一系列随机数组成的，其中两个随机位置（一个在开始，一个在序列的中间）被标记。该模型需要在看到整个序列后，预测这两个随机数的总和。对于每个生成的序列，我们从 $[T, \frac{11}{10}T]$ 中采样一个样本 T'. 在文中，我们将 T 称为序列的长度，为了清晰起见。第一个位置从 $[1, \frac{T'}{10}]$ 中采样，而第二个位置从 $[\frac{T'}{10}, \frac{T'}{2}]$ 中采样。这些位置 i, j 被标记在不同的输入通道中，除了两个采样位置为 1 在哪里都是 0。该模型需要预测在采样位置 i, j 处发现的随机数之和除以 2。

为了解决这个问题，我们采用了一个包含 50 个隐藏单元的模型，具有一个 *tanh* 激活函数，学习速率设置为 0.01，正则化项前的因子 α 为 0.5。我们在梯度的范数上使用了裁剪阈值为 6 的裁剪。权值初始化服从均值为 0，标准差为 1 的正态分布。

该模型在长度 T 在 50 到 200 之间的序列上进行了训练。我们成功地获得了解决这个任务的 100% 的成功率，这超过了 Martens 和 Sutskever（2011 年）（使用 Hessian-Free 算法）的结果。值得注意的是，随着序列长度接近 200，成功率下降。Horchreither 和 Schmidhuber（1997 年）只考虑了最多 100 步的序列。Jaeger（2012 年）也以 100% 的成功率解决了这个任务，尽管该解决方案似乎不能很好地推广，因为它依赖于非常大的输出权值，对于 ESN 来说，这通常是不稳定的标志。我们使用一个单一的模型来处理所有长度的序列 $(50, 100, 150, 200)$，并且训练后的模型可以成功地推广到新序列长度增加到 400 步的情况下（而误差仍然低于 1%）.

乘积问题 这个任务类似于上面的问题，只是预测值是随机数的乘积，而不是总和。我们使用了与前面的情况相同的超参数，并得到了非常相似的结果。

时间序列问题 针对长度为 T 的时间序列，我们使用了两个符号 $\{A, B\}$ 和 4 个干扰符号 $\{c, d, e, f\}$. 序列条目来自两个随机位置，其中第一个位置取自 $[\frac{T}{10}, \frac{2T}{10}]$，而第二个位置来自 $[\frac{4T}{10}, \frac{5T}{10}]$. 任务的目标是预测非干扰物符号的提供顺序，即 $\{AA, AB, BA, BB\}$ 中的一个。

我们采用了一个包含 50 个隐藏单元的模型，学习速率设为 0.001，α 为初始值为 2 的正则化系数。剪切梯度范数的阈值保持在 6。在处理另外两个任务时，我们通过训练一个单一模型来处理序列长度在 50 到 200 步之间，取得了 100% 的

成功率。由于这一成功率超过了以前的技术水平，而且单一模型还可以推广到更长的序列（最多 400 步）。

3 位时间顺序问题 与前一个类似，不同之处在于我们有 3 个随机位置。第一个样本取自 $[\frac{T}{10}, \frac{2T}{10}]$，第二个取自 $[\frac{4T}{10}, \frac{5T}{10}]$，最后一个取自 $[\frac{6T}{10}, \frac{7T}{10}]$.

我们使用了与上述类似的超参数，但将隐藏层的大小增加到 100 个隐藏单元。与之前相同，我们在训练一个能够泛化到新序列长度的单一模型时，取得了超过最先进水平的性能。

随机排列问题 在这种情况下，我们的字典包含 100 个符号。除了从 $\{1,2\}$ 中采样的具有相同值的第一个和最后一个位置外，其它条目都从 $[3,100]$ 中随机选取。任务的目标是进行下一次的特征预测，尽管只有最后一个特征是可预测的。

我们采用了 100 个隐藏单元，学习率为 0.001，α 作为正则参数，初始值设为 1。切割阈值被设定为 6。经过检验，这个任务被证明更加难以学习，8 次实验中仅有 1 次成功。与之前一样，我们使用单独的模型来处理 T 的多元问题（从 50 到 200 个单元）。

去噪声记忆问题 对于无噪声记忆，我们获得了一个长度为 5 的二进制模式，然后经过常值的 T 步之后，模型需要生成先前看到的模式。我们也考虑了由 Martens 和 Sutskever（2011 年）提出的该问题的扩展：当模式的长度为 10 时，符号集的基数为 5 而不是 2。

在这些任务中，我们取得了 100% 的成功率，尽管我们为五个不同长度的序列训练了不同的模型。

A.1.11.2 自然任务

复调音乐预测 在训练模型的过程中，我们采用了一个包含 200 个序列长度的 *sigmoid* 单位 RNN。在所有情况下，我们将共切阈值设置为相同的值，即 8（请注意，在计算梯度时必须取序列长度的平均值）。

对于 Piano-midi.de 数据集，我们选用了 300 个隐藏单元，并将初始学习率设为 1.0。在每个时期内，如果误差增加而不是减少，我们将学习率减半。对于正则化模型，我们设定正则化系数 α 的初始值为 0.5，其中 α 遵循一个 $\frac{1}{t}$ 的时间表，即 $\alpha_t = \frac{1}{2t}$（其中 t 为测量时期的数量）。

对于 Nottingham 数据集，我们使用了完全相同的设置。而对于 MuseData 数据集，我们将隐藏层增加到 400 个隐藏单元，学习率下降到 0.5。对于正则化模型，α 的初始值为 0.1，而 $\alpha_t = \frac{1}{t}$.

我们观察到，对于自然任务，采用一个减小正规化项的时间表似乎是有益的。我们假设正则化项迫使模型关注长期相关性，代价是忽略短期相关性。因此，采用衰减因子可能有助于模型更好地利用短期信息。

语言建模 在语言建模任务中，我们采用了一个具有 500 个不带偏差的 sigmoid 层的模型（Mikolov，2012 年）。该模型通过 200 个序列训练步骤，其中隐藏状态从一个步骤延续到下一个步骤。

我们在所有实验中使用了阈值 45 进行切割（尽管我们考虑了序列长度的成本总和）。对于下一个字符的预测，当我们使用没有正则化项的切割时，学习率为 0.01；当添加正则化项时，学习率为 0.05；而当我们不使用切割时，学习率为 0.001。在预测未来的第五个字符时，我们采用学习率为 0.005 的伴随正则项和学习率为 0.1 的非伴随正则项。

我们将用于下一个预测的正则因子 α 设置为 0.01 并保持不变。然而，对于正规化的任务，我们采用了初值为 0.05 和 $\frac{1}{t}$ 计划。