Dong Wang

现代机器学习技术导论

2017 年 8 月 28 日

Springer

Contents

Chapter 1
机器学习概述

Chapter 2
线性模型

Chapter 3
神经模型

上节我们介绍了最简单的机器学习方法，线性模型。线性模型的基本假设是变量之间存在简单的线性关系。显然，这一假设忽略了现实应用中大量非线性关系的存在。在上节讨论中，我们提到可以用一个非线性变换将原始变量映射到一个变换空间，使得这些变量在变换空间具有更明显的线性关系。非线性变换可以通过两种方式得到：一是利用先验知识手动设计，但这种设计通常比较困难，适用性很难保证；另一种方法是通过数据进行学习，从数据中自动总结出这一变换函数。这种数据学习方法避免了人为设计的困难，同时得到的变换函数和任务直接相关，通常会得到更好的效果。总体来讲，非线性变换学习方法可分为参数学习和非参数学习两种。在参数学习中，我们预先定义一个变换的函数形式，再对该形式中的参数进行学习；在非参数学习中，模型并没有一个确定的参数集，参数的多少与训练数据相关，典型的如支持向量机（SVM）模型 [31]。本章和下一章主要关注一类重要的参数学习方法，即人工神经网络模型（ANN）。简单来说，ANN通过一系列嵌套的非线性函数来学习复杂的非线性变换，使得经过该变换后得到的变量得以通过较简单的模型（特别是线性模型）进行预测和分类。然而，历史上ANN模型具有深刻的生理学背景，被认为是描述人类神经系统、产生类人思维方式的基础模型。随着深度神经网络（DNN）的具大成功，神经网络模型已经成为机器学习领域最重要的模型之一。本章主要讨论神经网络的基本结构和基本算法，DNN方法将在下一章深入讨论。

3.1 神经网络概述

19世纪40年代，受人类神经系统结构启发，研究者提出神经网络模型。简单地说，人类神经系统的信息处理方式是一种基于同质单元的结构化处理，其中每个神经元的结构基本相同，但神经元之间的连接结构则很复杂。通过这种复杂连接，可以实现各种复杂的记忆、推理等功能。这说明在人类神经系统中，各种信息和知识表现在连接结构上，而非神经元本身。这和传统基于符号的信息处理方式存在很大差异：在符号方法中，大量信息集中在各种符号定义中，而符号间的推理规则相对简单、通用。受神经系统这种特性的启发，Warren McCulloch和Walter Pitts提出了人工神经网络的概念[97]，期望通过模拟人类神经系统的计算方式来实现类人的信息处理能力。图 3.1给出基于仿生模拟的神经网络示意图。

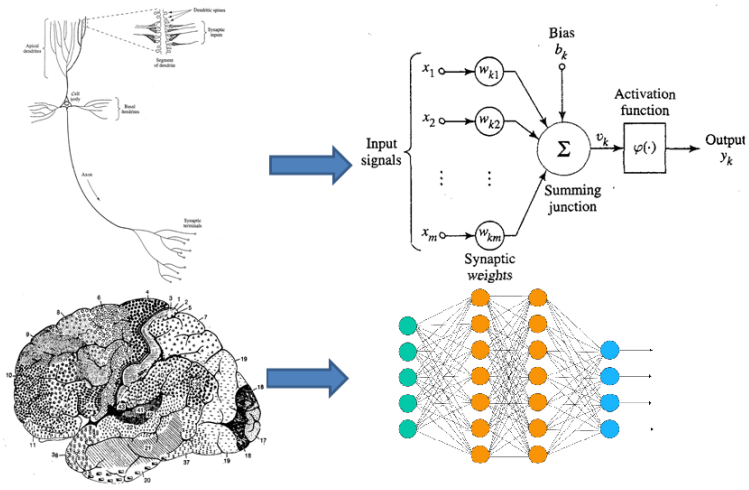


Fig. 3.1 基于仿生学的神经网络示意图。左上图表示一个独立的生物神经元，右上图是基于McCulloch-Pitts结构对该神经元的模拟。左下图表示人类神经系统，右下图为模拟该系统的神经网络。

对人工神经网络的研究可分为两个方向。一部分研究者集中研究如何描述人类大脑的实际运作方式，如激励方式、抑制机理、传导模型等，这些研究对理解人类智能的产生过程具有重要意义。基于这些研究结果，可设计相似的人工结构对其进行模仿。另一部分研究者更关注神经网络的表达能力，关注通过神经网络可实现的功能，至于该网络是否对应真实神经系统则不是

核心内容。机器学习中对神经网络的研究主要采用第二种思路，设计各种结构来提高网络对数据的建模能力和推理功能。

3.1.1 什么是人工神经网络？

关于人工神经网络，Wikipedia给出的定义是：在机器学习和认知科学中，人工神经网络（ANN）是一个受生物神经网络（动物的，特别是大脑的中枢神经系统）启发而提出的统计学习模型家族。该网络可用来估计或近似那些未知的、能够根据大量输入而产生反馈的生物神经网络的一些功能[1]。

Simon Haykin 给出的定义更加工程化[62]：神经网络是由简单处理单元构成的大规模并行分布式处理器，天然地具有存储经验知识并对其进行运用的能力。神经网络在两方面对人脑进行模拟：（1）知识通过学习从环境中获得；（2）知识被存储在神经元之间的连接权重中。

综合上述定义，神经网络的主要特性应包含如下三点：

- 同质性：神经网络中的处理单元（神经元）是简单的、同质的，不同单元不论从信息接收、信息处理、激发模式等方面都具有高度一致性；
- 连接性：神经网络中的神经元之间是互联的，通过组成网络来存储知识和模拟推理过程；
- 可学习性：神经网络是可学习的，通过改变神经元之间连接的权重，实现网络学习，适应外来数据。

基于其强大的学习能力，人工神经网络可以近似人脑的各种功能，包括记忆、归纳（抽象）和演绎（预测）等功能。这些功能基于不同网络结构，如预测功能一般基于前向网络，而记忆功能更多基于递归网络。我们将神经网络结构分成如下几种：基于映射的神经网络、基于记忆的神经网络、基于过程的神经网络以及模拟人类大脑的神经图灵机。我们将这些模型统称神经模型。本章后续几节将对这些模型做逐一介绍。

[1] https://en.wikipedia.org/w/index.php?title=Artificial_neural_network&oldid=666866254

3.1.2 神经模型与其它方法

如第 1章所述，神经模型的提出对机器学习乃至人工智能的发展起到了重要作用。早期的人工智能多基于符号方法（Symbolic AI）[2]，该方法将人工智能问题归结为依据某种计算规则的符号演算系统。符号方法是一个白箱系统，对问题进行精确定义，包括每一个概念的内涵和外延、概念与概念间的关系和操作。这一方法是人类逻辑思维的形式化，具有清晰的问题描述和严格的形式化推理，很少具有不确定性。符号方法是安全的、可理解、可信任的，但在实际应用中存在极大限制。首先，概念符号化具有任务特殊性，需要对每个任务单独设计，缺少通用性；其次，概念与概念间的关系缺少概率意义，对实际应用中的不确定性缺乏泛化能力；另外，如果有新的数据（经验）出现，很难对现有系统进行修正和增强。为解决这些困难，研究者提出了连接主义方法[3]，用简单同质的神经单元组成网络来描述复杂的映射关系，即神经网络方法。在这种方法里，只关注对问题的解决而非解决过程，因此不必对每个概念进行精确定义，从而避免了概念符号化的困难；同时，推理过程被形式化为网络结构及其参数，具有很强的学习性，当新的知识出现时，可以实现自适应；最后，这一学习依赖大量数据，从中抽取具有全局性、显著性的特征和规律，忽略个例和特例，因此对数据中的不确定性有较强的抵抗能力。

然而，神经模型处理传统人工智能中的符号问题还有一定困难，因为符号表达是离散的，不适合神经模型对目标函数求梯度的要求。相对来说，基于统计概率模型的符号方法更适合对这种离散数据建模。最近研究者提出Embedding概念，将概念表达成低维向量，再基于该向量构造神经网络。这一方法极大扩展了神经模型的应用范围，使其得以处理离散数据和符号任务。该方法在自然语言处理领域得到广泛应用，如语言模型、词性标注、句法分析、机器翻译、语言理解等 [13, 30]。

仅管如此，神经模型用于符号任务依然存在困难，这是因为该模型基于一个参数高度共享的神经网络来表示概念和规则，该网络可有效学习通用的、典型的信息和模式，忽略数据中的特例和噪声，因而可增强模型的泛化能力。然而，在很多符号任务中，很多小概率的特例是有价值的，有时甚至要比普适规律更重要。例如在机器翻译中，大量专有名词出现的概率很低，但这些词包含重要信息，对翻译的质量有直接影响。基于神经模型，这些小

[2] https://en.wikipedia.org/wiki/Symbolic_artificial_intelligence

[3] https://en.wikipedia.org/wiki/Connectionism

概率专有名词很有可能被当作噪声忽略，代之以一些语法或语义上类似但完全不正确的翻译。一种可行的思路是将神经模型和基于概率的符号方法接合起来，让神经模型和符号模型分别处理高频词和低频词，可有效提高翻译系统的性能 [159, 42]。

3.2 基于映射的神经模型

基于映射的神经网络是最常见到的神经模型之一。在这种网络中，输入一个特征向量，输出为基于该输入的预测目标。这一目标既可以是回归任务中的一个预测值，也可以是分类任务中的后验概率。这一模型是上一章所述线性预测模型的非线性扩展。

3.2.1 从线性模型开始

线性预测模型回顾

一个简单的线性回归模型可表示为:

$$y = \sum_{i=0}^{D} w_i x_i = w^T x$$

其中$\{x_i : i = 1,2,...,D\}$为D维自变量（又称特征向量），y为一维输出变量，$\{w_i : i = 1,2,...,D\}$为相应的回归参数。图 3.2给出该模型的图形化表示。这一表示可以认为是一个简单的、不包括隐藏层的神经网络。如果输出变量是多维的，则该网络结构如图 3.3所示。第 2章讨论过，该模型的最小平方误差估计等价于假设目标变量的观察值t为以y为中心的高斯分布时的最大似然估计。

相应的，在二分类问题中的Logistic回归模型如下式所示：

$$y = \sigma(\sum_{i=0}^{d} w_i x_i) = \sigma(w^T x); \quad \sigma(a) = \frac{1}{1+\exp(-a)}.$$

与线性回归模型相比，Logistic回归模型在线性模型基础上增加了一个Logistic变换。前一章讨论过，如果假设观察值（分类标记）是以y为参

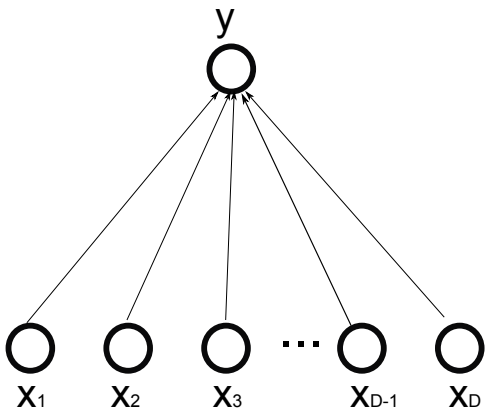


Fig. 3.2 线性回归模型可表示为不包含隐藏层的神经网络。

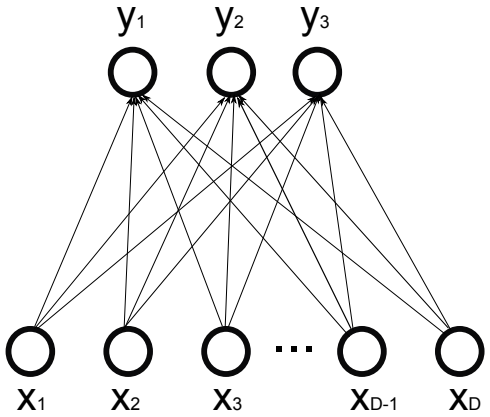


Fig. 3.3 多输出线性回归模型可表示为不包含隐藏层的神经网络。

数的伯努利分布，则该模型的最大似然估计等价于一个以交叉熵为准则的优化问题。

如果将上述Logistic函数替换为如下阶跃函数：

$$g(a) = \begin{cases} -1 & \text{if} \quad a < 0 \\ +1 & \text{if} \quad a \geq 0 \end{cases} \tag{3.1}$$

则对二分类问题的预测为:

$$y = g(\sum_{i=0}^{D} w_i x_i) = g(w^T x),$$

其中$y \in \{-1,+1\}$代表预测结果。这一模型即是神经网络发展早期著名的感知器模型。Logistic 回归模型和感知器模型本身是非线性模型，但和线性模型很接近，可称为“近线性模型”。

模型优化方法

线性回归模型存在闭式（Closed-form）解法，通过数学推导可直接求出参数w的最优值。对于近线性或更复杂的模型，一般不存在闭式解，这时通常采用数值解法，通过迭代逐渐逼近最优解。梯度下降法（Gradient Descent, GD）是最常用的数值解法，通过求目标函数（最小平方误差或最大交叉熵）对参数的梯度，选择合适的步长将参数沿梯度方向做有限变动。这一过程迭代进行，当步长选择合理时，可得到局部最优解。更复杂的数值优化方法将在第章 11介绍。

梯度下降法要求目标函数已知且连续可导。对感知器模型，其目标函数中包含阶跃函数，因此不能直接利用梯度下降法进行优化。一种解决方法是不考虑阶跃函数，仅考虑线性预测$w^T x$部分，使其与目标变量t尽可能符号相同。由此可得到如下误差函数：

$$L(w) = -\sum_{n \in \mathscr{M}} w^t x^{(n)} t^{(n)},$$

其中其中w^t表示在第t次更新后的模型参数，$x^{(n)}$为第n个训练样本，$t^{(n)} \in \{-1,1\}$为该样本的分类，$\mathscr{M}$为错误预测（即$y^{(n)}$与$t^{(n)}$的符号相反）的样本集合。注意上述目标函数依然是不连续的，因为$\mathscr{M}$会随着w的更新发生改变。仅管如此，我们仍然可以假定$\mathscr{M}$在w的某个邻域内保持不变，由此求出$L(w)$在该处的梯度。通过简单计算可得到如下参数更新公式：

$$w^{t+1} = w^t - \alpha \nabla L(w) = w^t + \alpha \sum_{n \in \mathscr{M}} x^{(n)} t^{(n)},$$

注意经过上述参数更新后，$\mathscr{M}$可能会发生变化，因此$L(w)$可能会发生阶跃性变化，收敛性并不直观。经过研究，人们发现上述迭代过程对于任何一个线性可分的数据集都可确保通过有限步迭代后收敛到一个对该数据集完美可分的分类器。这一结论称为感知器收敛定理（Perceptron Convergence Theorem）[103, 19]。

非线性扩展

感知器收敛定理给研究者很大信心，推动了神经网络技术的早期发展。然而，现实生活中绝大部分问题是线性不可分的，对这些问题感知器模型不仅缺乏区分能力（这一模型本质上是线性的），而且收敛性亦不能保证。一个经典例子是如下异或运算问题:

$$y(x_1,x_2) = \begin{cases} 1 & (x_1 = 0 \,\&\, x_2 = 0)||(x_1 = 1 \,\&\, x_2 = 1) \\ 0 & (x_1 = 0 \,\&\, x_2 = 1)||(x_1 = 1 \,\&\, x_2 = 0) \end{cases}$$

其中$x_i \in \{0,1\}$ 为输入变量，$y \in \{0,1\}$代表类别标签。如图 3.4所示，无论怎样设计，都无法找到一条直线将黑色的点和白色的点按类分开。换句话说，线性模型无法模拟异或运算。

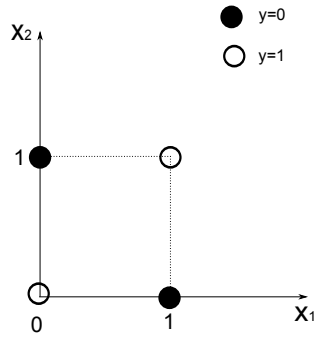


Fig. 3.4 异或运算表示图。x_1和x_2是两个输入变量，$y = x_1 \ xor \ x_2$。

经典一层感知器遇到线性不可分问题时，训练不能收敛，这使得该方法在实际应用中有很大局限性，这也是导致早期神经模型走向低谷的原因之一 [99]。为解决这一困难，可以将传统感知器模型的阶跃输出函数（式 3.1）修改为连续输出函数（如Sigmoid），并通过调整学习率保证训练收敛，即Logistic回归或Softmax回归。这一方法对于因噪声导致的线性不可分问题具有良好效果，但当数据本身具有很强的非线性时，包括一层感知器在内的各种线性模型都很难适用。

为解决线性不可分问题，研究者对传统线性模型（包括感知器）进行了非线性扩展，将输入变量通过非线性变换映射到变换空间，并在变换空间建立线性模型，即：

$$y(x) = w^T \phi(x),$$

其中$\phi(x)$是对x的非线性变换函数。依变换函数的形式不同，可得到不同的非线性模型。几种典型的非线性变换和其对应的模型如下：

$$\begin{aligned} \phi_j^n(x) &= \sum_i w_{ij}\phi_i^{n-1}(x) && \text{多层感知器} \\ \phi_j(x) &= \phi_j(\|x - v_j\|) && \text{径向基函数} \\ K(x,y) &= \phi(x)^T\phi(y) && \text{核函数} \end{aligned}$$

本章将介绍多层感知器和径向基函数，关于核函数方法将在第 5章讨论。注意，在第 2章介绍线性模型时，我们同样提到对x 进行非线性变换，但该变换是固定的，不需要学习，因此模型依然是线性的。本章所讨论的非线性变换需要对变换函数进行学习，因而模型是非线性的。

3.2.2 多层感知器

将传统一层感知器模型扩展到多层即得到多层感知器（MLP）模型。除了结构上的扩展，当前标准MLP模型不再采用阶跃函数作为输出函数，而是采用线性或Logistic函数，这些函数是连续可导的，因此可基于梯度下降算法进行优化；同时，这些输出函数对应不同数据分布假设，可分别对回归任务和分类任务进行建模。因此，与其说MLP是一层感知器的扩展，不如说是对线性回归和Logisitc/Softmax回归的扩展。本节将介绍多层感知器的基本结构和训练方法。

3.2.2.1 模型结构

多层感知器是对线性回归模型和线性分类模型的扩展。从结构上看，MLP将线性模型的一层网络扩展到多层，每一层输出经过一个非线性变换后作为后一层的输入，由此得到一个信息逐层传导的前向网络（Feed-Forward Network）。图 3.5给出一个包含一个隐藏层的MLP结构。该模型的计算过程可形式化如下：

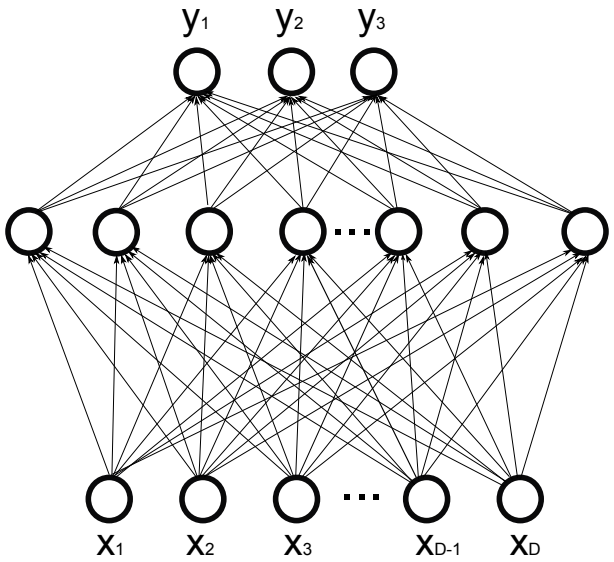


Fig. 3.5 多层感知器结构图。

$$
\begin{aligned}
a_j &= \sum_{i=0}^{D} w_{ij}^{(1)} x_i \\
z_j &= g(a_j) \\
a_k &= \sum_{j=0}^{M} w_{jk}^{(2)} z_j \\
y_k &= \widetilde{g}(a_k),
\end{aligned} \tag{3.2}
$$

其中$g(\cdot)$和$\widetilde{g}(\cdot)$为第一层和第二层的激发函数（Activation Function），z_j为隐藏层的激发值。注意该网络第二层相当于以隐藏层激发值为输入的线性或近线性模型，因此在回归任务中，激发函数$\widetilde{g}$一般取线性，即$\widetilde{g}(x) = x$，在分类任务中，一般取Logistic或Softmax函数，即$\widetilde{g}(x) = \sigma(x)$。

通过上述前向网络，输入x经过逐层非线性变换，直到倒数第二层，得到非线性映射$\phi(x)$，再由最后一层线性或近线性模型完成回归或分类任务。简单计算可得到变换$\phi(x)$如下：

$$
\phi_j(x) = g\left(\sum_{i=0}^{D} w_{ij}^{(1)} x_i \right),
$$

输出层的激发值为:

$$y_k = \widetilde{g}\left(\sum_{j=0}^{M} w_{jk}^{(2)} g\left(\sum_{i=0}^{D} w_{ij}^{(1)} x_i\right)\right).$$

研究表明，如果激发函数选择适当（如Sigmoid，Tanh，Relu等），当隐藏层的神经元个数足够多时，包含一个隐藏层的MLP可以有效模拟一切连续函数。这一结论称为一般近似定理（Universal Approximation Theorem）[72, 61, 19, 32]。

3.2.2.2 训练方法

随机梯度下降与BP算法

MLP的训练和线性模型训练的基本原则是相似的：定义好一个目标函数，通过调整模型参数使得目标函数最大化（或误差函数最小化）。和线性模型不同的是，多层结构和非线性激发函数使得MLP的目标函数变得非常复杂，一般不能得到解析解，因此通常采用数值解法。梯度下降（Gradient Descent）是最常用的数值优化算法，其参数更新如下：

$$w^{t+1} = w^t - \eta(t)\nabla L(w^t),$$

其中w^t为第t次迭代的模型参数，$\nabla L(w^t)$为目标函数在第t次迭代的梯度，η为学习率。GD算法每次迭代在整个数据集上计算梯度，效率较低。Stochastic Gradient Descent （SGD）是最常用的MLP优化算法。SGD和梯度下降法（GD）类似，都是求目标函数对参数的梯度，并依梯度方向对参数进行迭代调整。不同的是，SGD每次迭代仅随机选择一部分训练数据，基于该数据计算梯度并调整参数。这一随机选择的数据集通常称为一个Mini-Batch。基于Mini-Batch，SGD的随机性更强，可部分消除不合理的参数初始化带来的影响；同时，因为每个Mini-Batch后都对参数进行更新，更新后的参数被用于下一个Mini-Batch的梯度计算，因此收敛速度更快。不论GD还是SGD，都只能达到局部最优。

一个大规模MLP参数可能达数百万，对如此大量的参数进行优化，即使基于SGD算法也依然效率低下。Rumelhar和Hinton等人 [125]在1986年提出反向传递算法（Backpropagation, BP）很大程度上解决了这一问题。BP算法利用了MLP的层次结构，基于导数的链式法则和动态规划算法对参数进行顺序求导，避免了重复计算。我们用一个不严格的例子来说明BP算法的原理。设一个K层MLP，每一层只有一个参数，这一结构代表的映射函数如下：

$$f_{MLP} = f^1_{w_1} \odot f^2_{w_2} ... \odot f^K_{w_K},$$

其中$\odot$表示函数嵌套，$f^k_{w_k}$为倒数第k层代表的映射函数，w_k为该函数的参数。目标函数$L(w_1, w_2, ..., w_K)$对w_k的导数可写成如下形式：

$$\frac{\partial L}{\partial w_k} = \frac{\partial L}{\partial f^1} \frac{\partial f^1}{\partial f^2} ... \frac{\partial f^{k-1}}{\partial f^k} \frac{\partial f^k}{\partial w_k}.$$

上式说明，要求L对w_k的导数，需要对第k层之后的所有层求映射函数对输入的梯度。如果对每个w_k单独计算导数，显然会造成重复计算。一种解决方法是利用如下递推公式，由$\frac{\partial L}{\partial f^1}$开始顺序计算$\frac{\partial L}{\partial w^k}, k = 1, 2, 3, ...$：

$$\frac{\partial L}{\partial f^k} = \frac{\partial L}{\partial f^{k-1}} \frac{\partial f^{k-1}}{\partial f^k}$$

$$\frac{\partial L}{\partial w_k} = \frac{\partial L}{\partial f^k} \frac{\partial f^k}{\partial w_k}.$$

从神经网络角度看，这一过程可形象表示为梯度由最后一层向前逐层传导，因此称为反向传递算法，即BP算法。以回归任务为例，其目标函数为如下最小平方误差:

$$L = \sum_{n=1}^{N} \left(f_{MLP}(x^{(n)}) - t^{(n)} \right)^2,$$

起始梯度计算如下：

$$\frac{\partial L}{\partial f^1} = 2 \sum_{n=1}^{N} \{ f_{MLP}(x^{(n)}) - t^{(n)} \},$$

注意上述梯度为预测值与观察值的误差和，因此BP算法也可以看作是误差的反向传递。

训练技巧

BP算法在原理上是清晰的，但实际实现时还需要仔细设计。这是因为当层数增加以后，受到非线性函数嵌套的影响，误差向前传递变得越来越困难，可能发生消失或爆炸。另一方面，当模型参数增加时，过拟合问题越来越严重，导致在测试集上性能下降。最后，因为无法得到全局最优，模型质量受参数初始化影响较大。研究者总结了一些神经模型训练中常用的经验，这些经验可以有效提高训练的稳定性和收敛速度。需要说明的是，这些经验在训练其它神经模型时也经常用到。

- **输入/输出正规化和特征变换(Feature Normalization and Transfer)。** 如果输入和输出在取值上过于分散，训练难度将大为增加。一方面，神经网络的连接权重一般初始化在零点附近，如果输入或输出过大或过小，则需要更多轮迭代才能使模型适应数据的值域。另一方面，有些非线性函数在0附近较为敏感，过大或过小的输入会进入非线性函数的饱和区，导致梯度传导效率下降。因此，将输入特征和输出目标变量进行正规化是提高神经网络训练效率的重要方法。常用的正规化方法包括最大-最小值归一（将一个Mini-Batch里的值归一到0到1或-1到1之间）、均值-方差归一（对一个Mini-Batch里的值减均值除标准差）、高斯化（将一个Mini-Batch里的数据变换为高斯分布）。最近提出的Batch Norm方法不仅对输入层进行正规化，对隐藏层也进行正规化，有效提高了模型训练效率 [74]。除了正规化外，一些变换方法也可促进模型训练，特别是各种降维方法，如PCA、LDA等。这些降维方法可预先去除一些与任务无关的噪声，从而降低神经网络的训练难度。
- **选择合适的激发函数(Appropriate Activation Function)。** 对于不同任务，应考虑选择不同的激发函数。可选的激发函数有Sigmoid，Tanh，ReLU，PNorm，Max-out等。实验表明，某些激发函数（如Sigmoid）容易在训练初始阶段陷入饱和区，导致训练困难 [49]；有些激发函数（如ReLU）具有无界性，容易导致训练发散 [50]。因此，应依据不同任务、不同数据分布情况对激发函数做认真选择。在深度神经网络中，研究者普遍倾向使用分段线性函数（如ReLu，Max-out）作为激发函数，这些函数的线性属性使得梯度传导更加容易 [50]。
- **连接权重初始化(Weight Initialization)。** 权重初始化往往会影响最终训练效果。不同任务、不同参数（如连接权重和偏移量）可能需要采用不同的初始化方法。通常建议对k层前向连接进行初始化时，其权重可从一个方差为$\frac{1}{n_{k-1}}$的随机变量进行采样得到，其中n_{k-1}为前一层结点数。这一方法使得每层次隐藏结点的方差在前向计算过程中保持为1。最近研究表明，同时考虑前向计算的方差和反向梯度传递的方差可得到更好的初始模型，如可采用如下均匀分布对第k层的前向连接权重进行采样：$\left[-\frac{6}{\sqrt{n_{k-1}+n_{k+1}}},+\frac{6}{\sqrt{n_{k-1}+n_{k+1}}}\right]$，其中$n_{k-1}$为前一层结点数，$n_{k+1}$为下一层节点数 [49]。
- **二阶信息(Second Order Information)。** SGD方法是一阶方法，只考虑目标函数的梯度，不考虑目标函数的曲率。这种方法的一个缺陷在于对所有参数使用相同学习率，这显然是不合适的，因为曲率越大，学习率应该越小，否则容易引起震荡；反之，曲率越小，学习率应该越大，否则会降低

收敛速度。牛顿-拉弗森方法（Newton-Raphson method）可依曲率对学习率进行调整，因而学习效率更高。该方法可形式化如下：

$$w^{t+1} = w^t - H^{-1}(w)\nabla L(w^t),$$

其中$H(w)$为目标函数的Hessian 矩阵，$\nabla L(w^t)$为t时刻目标函数的梯度。由上式可知，引入二阶信息相当于对各个参数自动设置学习率。为直观起见，我们只考虑Hessian 矩阵上的对角值，即沿某一参数方向的曲率。可以看到，对那些曲率大的参数方向，目标函数变化较大，参数的学习率自然调低，使学习不致于过度激进引起震荡；对那些曲率较小的参数方向，目标函数取值变化不大，可放心学习，该参数的学习率自然增加，加快学习步伐。对小规模网络，二阶方法效率很高，但对较大网络，直接计算Hessian矩阵非常困难，这时一般采用一阶信息的统计量来获取近似二阶信息，如Hessian Free [96]，AdaGrad [39]，AdaDelta [157]，Adam [38]，Natural SGD [123, 3, 109, 113] 等方法。

- **使用动量（Using Momentum）。** 动量是指在更新当前参数时，不仅考虑当前梯度，也考虑上一个Mini-Batch的梯度。形式化如下：

$$w^{t+1} = w^t - \alpha[\beta\nabla L(w^{t-1}) + (1-\beta)\nabla L(w^t)],$$

其中β是动量参数，$\nabla L(w^t)$为在t时刻目标函数的梯度。当目标函数在不同方向曲率相差较大时，动量方法可有效补偿不同曲率对学习率的不同要求，因此增加训练稳定性和训练效率 [112]。动量方法可认为是一种以较低计算代价获得二阶信息的方法。

- **课程学习（Curriculum Learning）** Bengio等 [14]研究表明，在神经网络训练中，可以将训练样本进行分组，先学习比较容易的样本再学习比较困难的样本，可提高学习效率。这可类比对学生的授课过程，一般先让学生学习些较容易的知识，再学习较深入的知识，学生的学习效率更高。

- **迁移学习（Transfer Learning）** 神经网络训练需要大量数据，但很多时候领域数据的获取和标注难度很大。一种解决方法是利用已有网络作为基础，用其中所包含的知识帮助新任务学习，这一方法称为迁移学习。研究表明 [16, 10]，如果我们已经有一个较好的模型，可以用各种方法将其中的知识迁移到新的学习任务中。最简单的方法是将原始模型的前面几层直接拿来用作新任务模型的特征提取层，再基于新任务的领域数据进行继续学习。另一种方法是用原始模型对新任务进行指导，使得新任务的预测与原模型的预测近似 [69, 148, 139]。事实证明，这些迁移学习方法可极大提

高新模型的训练效果，特别是当新任务的训练数据较少时，这一方法的效果更加明显。

- **正则化（Regularization）** 神经网络模型缺少先验知识，完全基于数据驱动来优化参数，因此容易陷入过拟合。在训练过程中加入正则化可有效防止过拟合，提高模型的可扩展性。一种正则化方法是在目标函数中加入对参数或神经元的正则化因子，如l_1 [92]和l_2 [85, 129]约束。另一种常用的正则化方法是Eearly stop，该方法基于一个验证集，当在验证集上的性能开始下降时即停止训练 [114, 115, 24]。除此之外，我们可能需要对参数更新过程进行控制，如对梯度或参数本身的取值范围进行限制，防止数值上的不稳定性 [110]。带噪训练也可认为是一种正则化方法，通过在训练数据中随机加入一些噪声，可以使神经网络关注更有价值的数据模式和规律 [154]。最近提出的Dropout方法 [133]也可认为一是加噪训练，只不过噪声不是加在数据上，而是加在隐藏结点。研究表明，这一方法可有效防止参数间的协同训练问题，使得每个神经元更有代表性。最后，各种模型剪裁方法也可认为是一种引入结构稀疏性的正则化方法 [33, 119, 155, 93, 92]。

3.2.3 径向基函数网络

MLP基于函数嵌套（Function Composition）设计非线性变换$\phi(x)$，每一层变换函数是简单的线性映射附加一个非线性激发函数。径向基函数（Radio Basis Function, RBF）网络基于另一种思路实现这一线性变换。该方法在映射空间设计一系列标识点（Anchor Points）$\{v_j\}$，基于这些标识点，可以将每一个采样点x用该点到v_j之间的距离表示出来，实现了由原始空间到变换空间的非线性变换$\phi(x)$。每个标识点v_j代表变换空间中的一个基（Basis），基于v_j的距离函数称为一个径向基函数，即RBF。RBF网络在函数近似、时序预测、分类任务以及系统控制中有广泛应用 [22, 128, 94]。

3.2.3.1 RBF 网络与训练方法

一个包含M个RBF的网络如图3.6所示，其中第二层为映射层，每个ϕ_j对应一个RBF，定义如下：

$$\phi_j(x) = \phi_j(\|x - v_j\|),$$

其中ϕ是任意一个变换函数，$\|\cdot\|$表示向量间的某种距离测度（如欧氏距离）。输出层计算与MLP类似：

$$y_k(x) = \sum_{j=0}^{M} w_{jk}\phi_j(x). \tag{3.3}$$

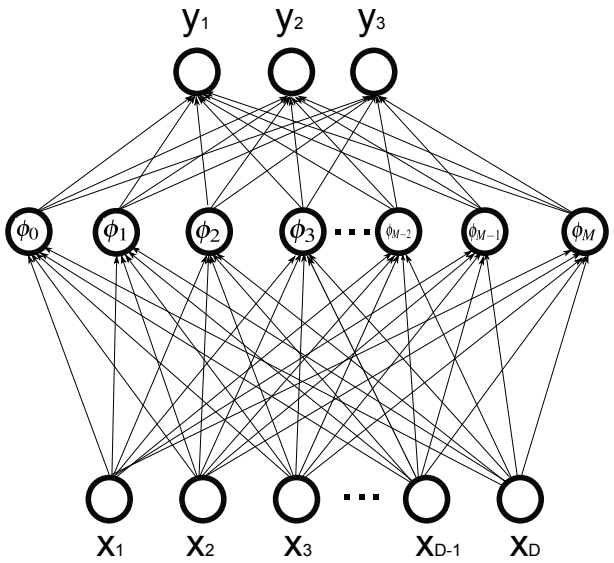


Fig. 3.6 包含M个RBF的径向基函数网络。

RBF中的ϕ_j形式可以是多样的，最常用是如下高斯形式:

$$\phi_j(x) = \exp\left(-\frac{\|x-\mu_j\|^2}{2\sigma_j^2}\right), \tag{3.4}$$

其中$\{\mu_j\}$即为前面讨论的标识点集，σ_j是高斯分布的方差。注意上述高斯形式是距离$||x-\mu_j||$的单变量函数，而非以x为变量的多变量高斯分布。Hartman等人证明 [60]，当隐藏结点足够多时，式 3.4所示的高斯RBF组成的网络可以近似一切连续函数。Park和Sandberg等人推广了这一结论，他们发现很多RBF核函数只需满足一定的限制条件都可以近似所有连续函数 [108]。

RBF网络的核心思想是用映射空间里一些具有代表性的点（标识点）作为基来实现原始数据的非线性变换，因此这些标识点的选择至关重要。通常用无监督学习方法来选择标识点，如高斯混合模型，不仅可以确定v_j，还可用来学习每个$\phi_j(x)$的参数，如高斯RBF中的σ_j。确定好RBF之后，可利用线

性模型的学习方法学习第二层映射的参数。另一种模型训练方法是将RBF和线性模型看作整体，基于BP算法进行统一学习。这种方法得到的RBF对当前任务更为优化，但得到的RBF可能表征性比较差（每个RBF的中心可能不再代表一个有效的标识点），从而降低模型泛化能力。

3.2.3.2 RBF网络的意义

RBF网络的意义可以从插值分析、核回归和分类任务等几个方面理解，本节对这些思路做简要介绍。

从插值分析到RBF

在插值分析任务中，给定一组训练数据$\{(x^{(n)},t^{(n)}):n=1,2,..,N\}$，对一个新的输入$x$，估计合适的输出$t$。给定一个映射函数$\phi(\cdot)$，插值方法定义一个输入$x$对应的输出$t$由下式计算：

$$f(x)=\sum_{n=1}^{N}w_n\phi(\|x-x^{(n)}\|). \tag{3.5}$$

取训练集中任何一个样本$(x^{(n)},t^{(n)})$，令$f(x^{(n)})=t^{(n)}$，可得一个包含N个等式的方程组。注意该方程组的参数$\{w_n\}$一共有N个，因此当其系数矩阵满秩时可得到唯一确定解。对于不在训练集上的点，式 3.5提供了一种预测方法，该方法以x到每个训练样本$x^{(n)}$的距离$\phi(\|x-x^{(n)}\|)$为权重，加权平均每个训练样本$x^{(n)}$对应的预测w_n即可得到t。

式 3.5所示的插值方法与式 3.3所示的RBF网络非常相似，唯一不同的是插值公式 3.5中的$\phi(\cdot)$是由训练集确定好的，而RBF网络中的$\phi_j(\cdot)$是训练出来的。如果我们对该插值法进行扩展，允许每个$\phi(\cdot)$的中心可以灵活取值，且数目不受训练数据N的限制，则得到一个受限的RBF网络，即：

$$f(x)=\sum_{j=0}^{M}w_j\phi(\|x-v_j\|).$$

注意当M小于样本数时，不存在一组参数$\{w_j\}$可以保证上式对训练集中的样本都成立。这时需通过上一章讨论的线性模型方法对$\{w_j\}$做最大似然估计，这事实上正是对RBF网络第二层进行训练的过程。

从核回归理解RBF

给定训练数据$\{(x^{(n)},t^{(n)}):n=1,2,...,N\}$，假设$(x,t)$服从如下分布：

$$p(x,t)=\frac{1}{N}\sum_{n=1}^{N}f(x-x^{(n)},t-t^{(n)}),$$

其中$f(x-x^{(n)},t-t^{(n)})$为以某一个训练样本$(x^{(n)},t^{(n)})$为中心的联合概率分布。如果$f(\cdot,\cdot)$取高斯分布，则上式为一个先验概率为$\frac{1}{N}$的高斯混合模型。基于上述联合概率，给定一个输入x，可以计算对t的预测：

$$y(x)=E[t|x]=\int tp(t|x)dt=\frac{\sum_n\int tf(x-x^{(n)},t-t^{(n)})dt}{\sum_n\int f(x-x^{(n)},t-t^{(n)})dt}.$$

令$g(x)=\int f(x,t)dt$，通过变量代换，可得:

$$y(x)=\frac{\sum_n g(x-x^{(n)})t^{(n)}}{\sum_n g(x-x^{(n)})}=\sum_n k(x,x^{(n)})t^{(n)},$$

这一方法称为核回归（Kernel Regression），其中核函数$k(x,x^{(n)})$定义如下:

$$k(x,x^{(n)})=\frac{g(x-x^{(n)})}{\sum_m g(x-x^{(m)})}.$$

如果我们将$k(x,x^{(n)})$定义为RBF网络中的径向基函数$\phi(||x-x^{(n)}||)$，将$t^{(n)}$视为对应的RBF网络的权重w_n，则得到类似RBF网络形式。注意这里的核函数$k(x,x^{(n)})$是比$\phi(||x-x^{(n)}||)$更通用的形式。

从分类任务理解RBF

我们也可以从分类任务来理解RBF。一个典型的分类任务如图 3.7所示，其中数据属于三类，每个类C_k可用一个概率密度函数$p(x|C_k)$来表示。分类任务可基于最大后验概率准则，即计算测试样本属于每一类的后验概率，将后验概率最大的类作为对该样本的归类。依贝叶斯公式，该后验概率计算如下：

$$P(C_k|x)=\frac{P(x|C_k)P(C_K)}{p(x)}=\frac{P(x|C_k)P(C_K)}{\sum_{k'}p(x|C_{k'})P(C_{k'})}.$$

如果定义RBF如下：

$$\phi_k(x) = \frac{P(x|C_k)}{\sum_{k'} p(x|C_{k'})P(C_{k'})},$$

则分类任务可以写成类似RBF网络的形式：

$$P(C_k|x) = \phi_k(x)P(C_k).$$

因此基于贝叶斯公式的后验概率计算可理解为如下RBF网络：在第一层生成一个RBF函数$\phi_k(\cdot)$，在第二层基于每一类的先验概率作为连接权重将隐藏层的第k个隐藏结点和输出层的第k个结点连接起来。这一网络如图 3.8所示.

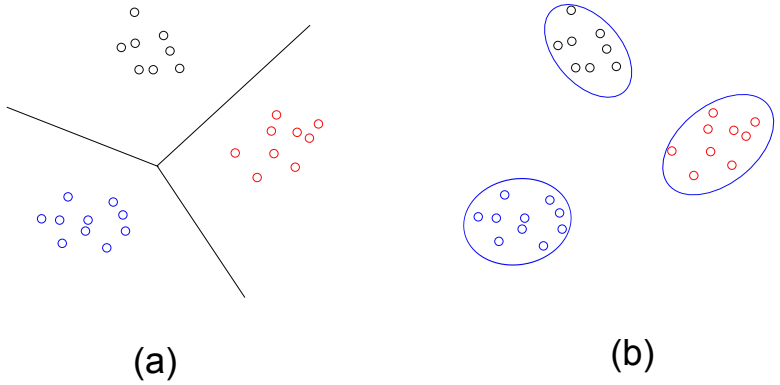


Fig. 3.7 基于贝叶斯方法的分类任务。(a)在特征空间的数据分布和分类面；（b）每一类对应的概率密度分布。

3.2.3.3 多层感知器与径向基函数网络的比较

上文描述了两个典型的神经网络结构：MLP和RBF网络。这两种神经网络采用不同的非线性扩展方法得到特征映射，当隐藏结点足够多时，均可描述任意连续函数。比较这两种模型可发现他们有很多不同之处。一个显著不同是MLP模型中的每个隐藏结点的“等激发线”是一个平面$c = w^T x$，而RBF 模型中的每个隐藏结点的“等激发线”是一个球面$c = ||x - v_j||$。这说明MLP中x的变动对隐藏层的影响是全局的，不论x在什么位置，对所有隐藏结点都会产生影响，而RBF中x的变动对隐藏层的影响是局部的，只对那些和x相近的v_j所对应的隐藏结点产生影响。换句话说，MLP的信息传递可认为是分散的（Distributed），x的信息通过所有隐藏结点分散向后传递，梯度信息也会通过所有隐藏结点分散地回传，所有参数都为描述某一信息进行修

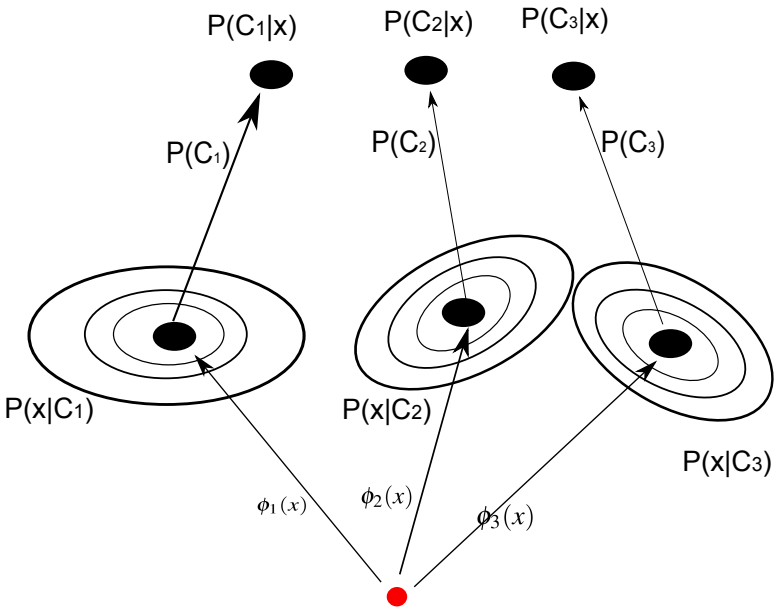


Fig. 3.8 基于贝叶斯公式的后验概率计算可认为是一个RBF网络，第一层求对每个类的RBF函数$\phi_k(x)$，第二层以每一类的先验概率作权重连接第k个隐藏结点和第k个输出结点。

正、协调。因此，MLP是一个参数高度共享的网络。RBF则不同，对任何输入x，其信息只通过少数和它邻近的v_j所对应的隐藏结点向后传递，对应地，梯度回传也仅与这些隐藏结点相关，因此RBF的参数共享性较弱。基于此，RBF需要更多隐藏结点，且泛化能力较弱，对没有被RBF有效覆盖的数据空间通常预测性能较差。

从训练角度看，RBF网络可用无监督学习来训练RBF函数，极大降低了训练难度。即便是监督学习，因为RBF的局部特性，参数共享较小，参数更新时的互相影响（Co-adaptation）较小，训练起来也比参数高度共享的MLP要容易。

3.2.4 神经网络模型与先验知识

标准MLP模型是一种全连接结构，具有强大的学习能力，但这一模型缺少先验知识，是一种纯数据驱动方法。这导致网络训练通常需要大量数据，且容易产生过训练或欠训练。在很多实际应用问题中，我们对问题本身是有

一定认识的，例如数据本身的数值特性和分布特性、和问题相关的显著特征、问题的复杂程度、可能的解决方法等。这些知识有些可以用在MLP的特征选择、目标函数设定、网络超参数选择（如层数，每层结点数）等方面，但很多时候并没有被有效利用。如果我们能将这些先验知识合理应用在神经网络模型设计和训练过程中，有可能极大降低模型复杂度，提高训练效率。我们介绍几种将先验知识用于神经网络建模的方法。

3.2.4.1 结构化模型与卷积神经网络

很多数据具有结构化特性，这些特性可用来设计更有针对性的网络。几中典型的结构化特性包括：

- 空间结构：例如在图像数据中，相近位置的像素往往具有相关性，不同位置的局部模式具有很大重复性。
- 时序结构：一些序列数据，如语音信号和文本串，往往包含很强的时序结构，时序相近数据相关性很强，而相同模式可能出现在一个序列的不同位置。
- 频域结构：在语音或图像数据中，相近频段具有较强的相关性，相同模式也有可能在不同频段上重复出现。

上述数据上的结构化特性可用来设计有针对性的结构化神经网络，其中卷积神经网络（Convolutional Neural Network, CNN）具有很强的代表性。CNN是利用上述结构化特性设计局部的、共享的网络子结构，使得每个子结构用于学习某种局部模式，且不同空间、时序、频域位置的子结构共享网络参数。图 3.9给出一个简单的CNN网络，其中包括一个卷积层和一个降采样层。卷积层利用一个局部网络将某一位置的输入映射到特征空间的某一结点，且不同位置的局部网络共享参数。这相当于利用一个由该局部网络组成的卷积核对输入平面进行卷积操作，生成一个特征平面（Feature Map）。降采样层利用一个简单的卷积核（如平均或取最大值）对特征平面进行降维。卷积核的作用类似于一个滤波器（Filter），可以用来学习输入信号中的重复模式。为提高表征性，一般CNN会通过多个卷积核生成多个特征平面，每个特征平面可学习输入数据的某一方面特性。由于每个卷积核对应的局部网络参数远少于全连接网络，CNN的模型的复杂度一般比全连接网络低很多。降采样层不仅可以对特征平面降维，还可以去除因输入数据的轻微变形引起的特征抖动，提高模型的泛化能力。

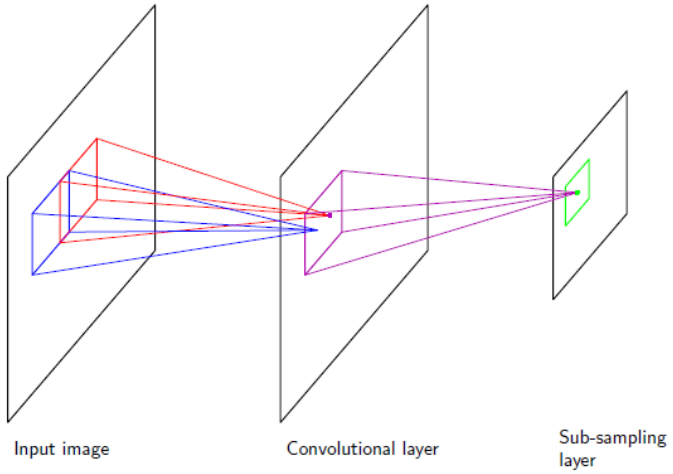


Fig. 3.9 CNN网络中的卷积和降采样。卷积层用多个卷积核提取特征，降采样层用来补偿因输入信号的轻微变形产生的特征抖动。注意卷积核在不同位置（红色和蓝色）进行卷积操作时参数不变。

图3.10给出一个用于手写数字识别任务的CNN结构。该结构包括两层卷积，每层卷积后接一个降采样层，最后通过一个全连接层实现对输入图片的分类。卷积神经网络的思路来源于1984年日本学者Fukushima提出的神经认知机（Neocognitron） [46]。神经认知机将一个视觉模式分解成许多子模式（特征），然后进入分层相连的特征平面进行处理。该方法试图将视觉系统模型化，使其能够在物体有位移或轻微变形时也能准确识别。LeCun等明确提出CNN结构，并提出有效的训练方法 [88]。其后，CNN被广泛用于各种模式识别和机器学习任务中。

总结起来，卷积神经网络具有下列性质：（1）使用卷积核可以学习重复性局部模式，因而可起到特征提取作用；（2）降采样增强对空间、时间、频域上轻微形变的鲁棒性；（3）参数量少，训练容易；（4）可以基于先验知识（如模式的大小）设计卷积核，因而有利于将知识结合到网络结构中，避免盲目学习。

3.2.4.2 混合密度网络

将先验知识与神经网络相结合的另一种方式是基于对数据分布的先验知识建立适当的概率模型，用神经网络来预测模型的参数。这种方式可以将概

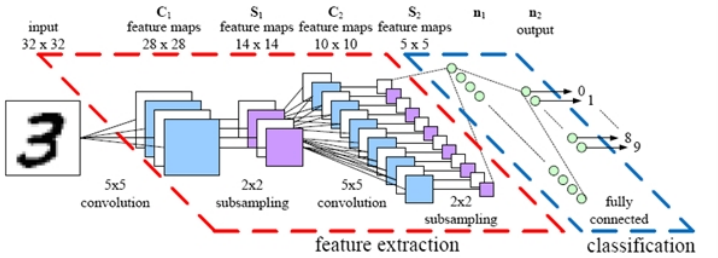


Fig. 3.10 用于手写数字识别的卷积神经网络。该结构包括两个卷积层，每个卷积层后接一个降采样层。这些卷积和降采样操作用于提取图像特征。最后通过一个全连接层对输入进行分类。图片本源于 [111]。

率模型的描述能力和神经网络强大的学习能力结合，由概率模型表达关于数据分布的先验知识，由神经网络增强参数估计能力。混合密度网络即是这样一种混合模型 [18]。以高斯混合密度网络为例，假设数据的条件概率符合如下高斯混合模型（GMM）：

$$p(t|x) = \sum_{k=1}^{K} \pi_k(x) N\left(t|\mu_k(x), \sigma_k^2(x)\right),$$

其中π_k、μ_k、σ_k^2分别代表第k个高斯成份的权重、均值和方差，他们都是输入x的参数，通过神经网络预测得到。K为高斯成份的个数。基于这一模型，即可对x和t之间的复杂概率关系建模。注意，传统基于平方误差的MLP事实上是上述混合密度网络模型取$K = 1$时的特例。

3.2.4.3 贝叶斯方法

贝叶斯方法是将先验知识引入神经网络的另一种有效途径。在线性模型一章中，我们知道贝叶斯方法通过对模型参数设定先验概率来约束模型的学习目标，从而改变模型的特性。因为这一先验概率多基于具体任务的数据、目标、预期解决方法等先验知识确定，这相当于将先验知识集成到神经网络建模中。一个典型的例子是稀疏先验知识的引入。生理学研究表明，人类神经系统是稀疏的、结构化的，只有特别必要时，神经元间的连接才会建立 [8, 44, 105]。这一先验知识可以通过在网络权重上引入一个拉普拉斯分布作为先验概率来实现：

$$p(w) = \frac{1}{2b} exp(-\frac{|w|_1}{b}), \tag{3.6}$$

其中b为控制分布集中度的参数，w为模型网络权重。以回归任务为例，对一个训练数据集$D = \{(x^{(n)}, t^{(n)}) : n = 1, 2, ..., N\}$, 模型对该数据集的概率计算如下：

$$p(D|w) = \prod_{n=1}^{N} P(t^{(n)}|y(x^{(n)}; w)),$$

则w的后验概率计算为：

$$p(w|D) = \frac{p(D|w)p(w)}{p(D)}.$$

模型训练的目的是优化模型参数w，使得$p(w|D)$最大。由于$p(D)$计算非常复杂，可采用如下近似目标函数：

$$\widetilde{L}(w) = \ln\ p(D|w) + \ln\ p(w).$$

注意上式中$\ln\ p(D|w)$即为传统神经网络训练的目标函数$L(w)$。代入式(3.6)，有：

$$\widetilde{L}(w) = L(w) - \frac{|w|_1}{b}.$$

由此可见，引入拉普拉斯先验的MLP相当于在训练准则上加入了一个l_1正则项，这一正则项除了鼓励模型选择较小的参数外，同时鼓励缺少显著关联的神经元之间的连接权重置零 [141]。因此，加入拉普拉斯先验鼓励生成稀疏模型，而这正是我们希望引入模型中的结构限制。推而广之，贝叶斯方法通过对模型权重引入不同先验概率，可以在模型训练中引入对应的先验知识，减少神经网络在建模和训练中的盲目性。

3.3 基于记忆的神经模型

上一节我们讨论了基于映射的神经网络模型，该模型学习一个从x到t的映射$y = f(x)$，使得y与目标变量t的误差最小。这一模型主要用于预测任务。在实际生活中，还有另一类问题，在这些问题中仅有数据x而没有明确的数据标记t，我们希望设计一个模型可以描述x的分布，或者得到代表x的抽象特征。对这类任务，基于映射的神经网络并不适合。研究者提出各种基于记忆

的神经网络来处理这一问题。当网络训练完成后，对于一个测试样本，可以基于网络中的记忆信息得到该样本的概率，采样出与之近似的训练样本，或得到该样本背后的隐藏结构。这一记忆模型有重要意义。一方面，对一个含有噪声的测试样本，通过提取近似样本，可以起到去噪效果；另一方面，如果训练样本足够多，这一网络可以学习样本中有价值的模式，进而提取有效特征。这一性质非常重要，是下一章要讨论的深度学习方法的基础。

本节将介绍几种典型的神经记忆模型，包括：Kohonen网络、Hopfield网络、玻尔兹曼机（Boltzmann Machine, BM）、自动编码器（Auto-Encoder, AE）等。

3.3.1 Kohonen网络

Kohonen网络又称自组织映射（Self Organization Map, SOM），由Kohonen在1982年提出 [84]。Kohonen网络的基本思想是将高维数据映射到一个低维空间，使得在高维空间中的分布结构在低维空间得以保持。一个Kohonen网络包含若干神经元结点，这些结点一般置于一个规整的平面上，如图 3.11所示。每个结点s_i对应一个D维向量v_i，其中D为数据空间的维度。在训练过程中，对一个输入向量x^n，可以计算该向量与所有神经元结点间的距离$\{d(v_i, x^{(n)})\}$，基于该距离可得到最相近的结点s_j，称为最佳匹配结点（Best Matching Unit, BMU）。找到BMU之后，对BMU对应的向量v_j进行更新，使之与x^n更加接近。一种简单的更新方法如下：

$$v_j^{t+1} = v_j^t + \eta(t)x^{(n)} \qquad s_j = BMU(x^{(n)}),$$

其中$\eta(t)$是t时刻的学习率，该学习率一般随t增加而衰减。上述更新对所有$\{x_n\}$循环迭代进行，即可使Kohonen网络的神经元结点收敛到原始数据的分布结构。这一过程如图 3.12所示，其中蓝色表示数据分布，网格中的结点位置由该结点对应的向量决定。对一个新的数据点（图 3.12中的白色点），寻找BMU（黄圈所示），将其往该数据点方向更新，如此循环往复，最后网络结点的对应向量即可充分代表训练数据。

如果我们仔细考察一下上述训练算法，发现它事实上是一个在线K-mean算法。这一方法可保证网络中的结点充分表达训练数据的分布，但并不能保证在高维平面上相近的点在低维平面上的激发结点（即BMU）是相近的。这是因为我们在更新神经元向量的时候并没有考虑各个神经元在低维空

间中的相邻性。这显然不能满足我们在低维空间描述数据分布的要求（可以参考在K-mean算法中，各个中心矢量是互相独立的，没有明确的近邻关系）。为解决这一问题，我们可以使低维空间中相邻的神经元被同一数据激发，如图 3.11所示，除了BMU（黄色结点）被激发外，周围结点（红色、紫色等）也同时被激发，只不过被激发的级别要低一些。注意，各个神经元结点之间的相邻关系是由事先定义的拓扑结构决定的，而非由其对应的向量计算。在实际训练过程中，对每一个训练数据$x^{(n)}$，神经元结点对应的向量更新公式如下：

$$v_k^{t+1} = v_k^t + \alpha(t)\theta(k,j)x^{(n)}; \quad s_j = BMU(x^{(n)}), \quad s_k \in N(s_j),$$

其中$N(s_j)$表示s_j的相邻结点集合，$\theta(k,j)$为相邻神经元结点间的相关性强度。通过这种相邻激发，拓扑结构所定义的近邻关系被引入到模型中，即可实现对高维空间中相邻关系的捕捉和呈现。注意图 3.12和图 3.13的训练的过程已经引入了这种相邻关系，因而在训练时不仅BMU的向量被更新，相邻的结点对应的向量也同时被更新，因而产生网络协同形变的效果。引入相邻关系是Kohonen网络区别于K-mean的主要特点。

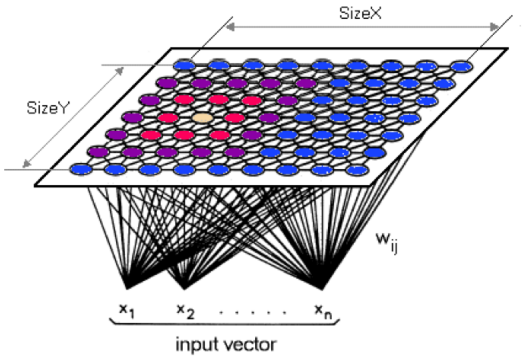


Fig. 3.11 Kohonen网络结构。神经元结点被均匀分布在一个平面上，每个神经元s_i对应一个向量v_i。每个输入$x^{(n)}$依$x^{(n)}$和v_i之间的距离激发一个最佳匹配结点（BMU）s_j（黄圈），同时激发s_j的相邻结点（红圈和紫圈）。图片来源于Adrian Horzyk的网页[5]。

Kohonen网络是一种局部映射。高维空间中的某个数据点只与和它最相似的网络结点有关，而与大多数结点无关。这种局部性和RBF有些类似，但在RBF网络中，所有RBF结点都会参与计算，虽然绝大多数结点并没有太大贡献，而Kohonen网络则可通过拓扑结构定义神经元的相邻关系。这种局部映射属性说明该映射天然是一种非线性映射，可描述复杂的分布

Fig. 3.12 Kohonen网络训练过程。给定一个训练数据，如图中白点所示，训练过程将对应的BMU及其相邻结点往该训练数据方向拉动。这一过程对所有数据循环迭代进行，直到网络结点中的向量可充分代表训练数据。图片来源于Wikipedia[7]。

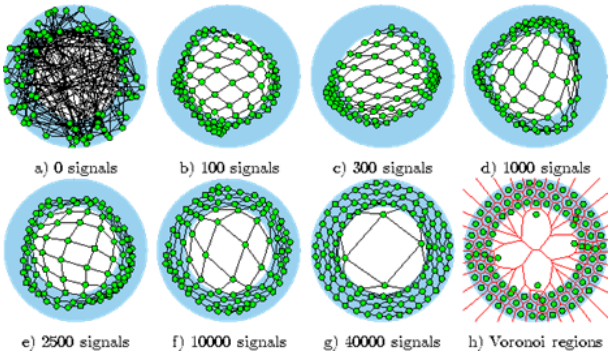


Fig. 3.13 Kohonen网络训练过程示例。随着迭代次数增加，网络结点越来越代表数据分布。图片来源于Bernd Fritzke的报告[9]。

情况。这一点和PCA显著不同，后者是全局线性映射，只对高斯分布数据有效。图 3.14给出在一个典型非高斯数据上Kohonen网络和PCA的对比，显然Kohonen网络在这种数据上的描述能力更强。

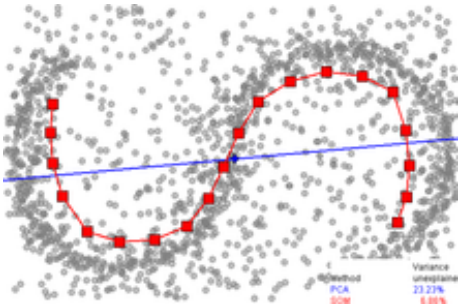


Fig. 3.14 Kohonen网络与PCA在非高斯数据上的对比。红色线条代表一维Kohonen网络，每个结点的位置由其对应的向量决定。PCA的映射空间在图中表示为一条蓝线。显然，Kohonen网络更能表示数据的实际分布情况。图片来源于Wikipedia[11]。

Kohonen网络是一种非常简单的记忆网络，该网络通过一些记忆结点（由网络结点对应的向量表示）对训练数据进行记忆和表达，在模式提取时将数据映射为最相似的结点。因为网络结点通常会记忆最有代表性的数据，Kohonen 网络可发现数据分布的基础模式，并可用于简单特征提取。

3.3.2 Hopfield网络

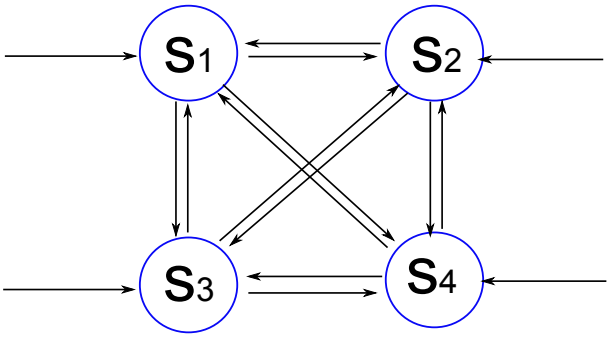


Fig. 3.15 Hopfield网络，每个结点是二值的，对应一个神经元，通过输入连接接收输入，通过输出连接向外输出。结点之间通过有向边连接在一起。

Kohonen网络的表达能力与矢量量化（Vector Quantitization, VQ）相似，不同结点有独立的代表向量，结点间缺少参数共享，没有形成结构化协同表示，因此记忆能力很低。

Hopfield等人在1982年受人类记忆模式的启发提出一种更有效的记忆网络 [71]，如图3.15所示。该网络包含若干二值的神经元结点（取+1或−1）$\{s_i\}$，每一对结点(s_i, s_j)间通过有向边w_{ij}相连。我们将所有结点组成一个向量，该向量的某种取值方式称为一个“模式”。Hopfield网络的目的是通过这一互联结构记住训练过程中遇到的模式，当训练完成后，可以通过“回忆”提取前期记住的模式。

3.3.2.1 模型学习

Hopfield网络的学习过程即是对训练数据所代表的模式进行记忆的过程。一种常见的学习方法采用Hebbian准则 [63]，该准则可简单表述为“同

时激发的单元互相连接" [95]。基于这一准则，给定N个训练样本$\{x^{(n)} : n = 1,2,...,N\}$，可计算结点$s_i$和$s_j$之间的连接权重为：

$$w_{ij} = \frac{1}{N}\sum_{n=1}^{N} x_i^{(n)} x_j^{(n)},$$

其中$x_i^{(n)}$为第n个训练数据的i维，对应网络中结点s_i的输入。由上式可见，如果x_i^n和x_j^n符号相同，则该样本对w_{ij} 的贡献为正值，意味着s_i和s_j之间的连接加强，恰好符合Hebbian准则。上述学习也可以采用增量模式，即：

$$w_{ij}^n = w_{ij}^{n-1} - \frac{1}{n}(w_{ij}^{n-1} - x_i^{(n)} x_j^{(n)}),$$

其中w_{ij}^n为学习第n个样本后神经元s_i和s_j之间的连接权重。

Hebbian学习可以理解为一个最大似然问题。给定训练集$D = \{x^{(n)} : n = 1,2,...,N\}$, 优化如下似然函数：

$$L(W) = p(D;W) = \prod_n p(x^{(n)};W),$$

其中$W = \{w_{ij}\}$为网络参数。设上述概率具有吉布斯形式（Gibbs Measure）如下：

$$p(x^{(n)};W) \propto e^{\sum_{ij} w_{ij} x_i^{(n)} x_j^{(n)} - \sum_i \theta_i x_i^{(n)}},$$

其中$\{\theta_i\}$为人为指定的模型参数（不需训练）。简单计算可知，基于上述假设的最大似然准则等价于Hebbian准则。因此，Hebbian准则的目的是使训练数据的概率最大化。对应上述概率形式，一个输入x的能量函数为：

$$E = -\sum_{i,j} w_{ij} x_i x_j + \sum_i \theta_i x_i, \tag{3.7}$$

因此，Hebbian准则也可以理解为使训练数据的能量最小。

图 3.16给出一个能量函数示意图，其中横轴表示Hopfield网络所处的模式（即输入x的不同取值），纵轴表示该模式对应的能量。经过学习以后，在网络记忆能力范围内，训练数据所对应的模式将处于能量局部最低状态，这些能量局部最低点称为"Attractor"。注意并不是所有能量局部最低点都对应一个训练模式。对于复杂网络，某些能量局部最低点并非实际需要记忆的模式，而是为了对目标函数进行优化而人为引入的"赝模式" [64]。另一方面，如果训练模式超出了网络记忆能力范围，有些模式不能很好记忆，有可能不对应能量最低点。

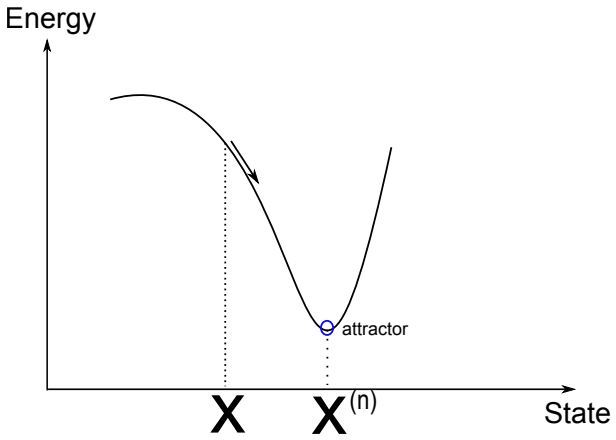


Fig. 3.16 Hopfield网络的能量函数示意图。在网络记忆能力范围内，训练样本被记忆成能量函数的局部最低点，称为“Attractor”。训练完成后，能量函数固定，这时输入一个新样本x，通过迭代更新，网络结点取值会收敛到与该输入邻近的Attractor，实现对记忆模式的提取。图中x 收敛到$x^{(n)}$对应的Attractor。

3.3.2.2 模式提取

在模式提取时，固定模型的权重，给定一个带噪声的样本作为图 3.15的初始模式，经过迭代运行，可得到一个与该输入样本最相似的记忆模式。这一过程如图 3.16所示：初始模式为样本x，通过迭代寻找邻近的能量局部最低点，最终得到被记忆的模式。为实现这一迭代搜索，首先将网络结点初始化为x，即$s_i = x_i$。依公式 3.7，可求由于某一神经元s_i取值变化时引起的能量变化：

$$\Delta E_i = E(s_i = +1) - E(s_i = -1) = -2\sum_j w_{ij}s_j + 2\theta_i.$$

上式意味着如果满足如下条件，则结点s_i取+1会使能量更低：

$$-\sum_j w_{ij}s_j + \theta_i < 0,$$

如果不满足上式条件，则应对s_i取-1。这一结论总结为如下迭代搜索准则：

$$s_i = \begin{cases} +1 & \text{if } \sum_j w_{ij}s_j \geq \theta_i \\ -1 & \text{otherwise} \end{cases}.$$

基于上述单一神经结点的更新方式，可对整个网络进行更新。更新可采用两种方式：在同步更新中，对所有结点统一更新；在异步更新中，从某个结点开始更新，并利用更新后的结点值去更新其它结点。Hopfield证明这一非线性动态系统是稳定的，因此这一更新过程总会收敛到一个局部最低能量点（Attractor）。这一最低能量点通常是模型经过学习得到的一个记忆模式（也可能是一个赝模式）。

3.3.2.3 Hopfield网络的记忆功能

Hopfield网络的一个显著特点是在模式提取过程中，从一个初始模式出发，寻找与其最相近的记忆模式，因此也称为联想记忆（Associative Memory [107, 71]）。这一方法常用于基于内容的寻址（Content-based Addressing）。由于提取过程会收敛到一个记忆模式，Hopfield具有抗噪能力。例如，我们通过学习，可以让Hopfield网络记住十个数字的图片，在提取时输入某个数字的带噪图片，Hopfield网络会自动抽取到和这个数字最相近的记忆模型，这事实上提供了一种有效的去噪和正规化方法。

Hopfield网络的记忆能力是有限的，与网络结点数和结点间的连接数直接相关。Hertz等人证明，对一个有1000个结点的Hopfield网络，能有效记忆的模式大约为138个，即模式/结点比约为0.138 [64]。Liou 等人证明，这一比例可能会提高到0.14以上 [91]。显然，这一模式/结点比还是非常低的，说明Hopfield网络的记忆效率低下。

3.3.3 玻尔兹曼机

Hopfield网络有两个特点：一是其所有神经元结点都是可见的，二是结点的取值是确定的。这两点限制了该网络的表达能力。玻尔兹曼机（Boltzmann Machine）引入隐藏结点和结点取值的随机性来解决这一问题。这一模型由Hinton 和Sejnowski 在1985年提出 [1]。一个典型的玻尔兹曼机如图3.17所示，其中灰圈表示可见结点$\{v_i\}$，白圈表示隐藏结点$\{h_j\}$。不论是可见结点还是非可见结点都是二值随机变量，且每个结点的概率分布依赖与其相连的结点取值。后面我们会看到，这一模型事实上是一个无向图，亦称为马尔可夫随机场（Markov Random Field, MRF）。

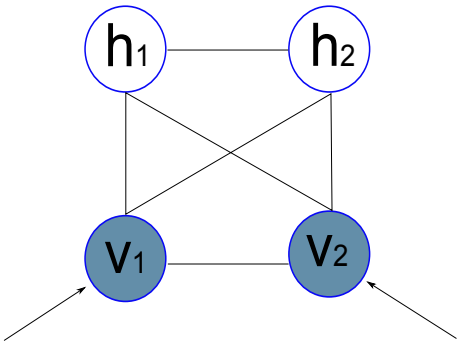


Fig. 3.17 玻尔兹曼机(Boltzmann Machine）结构图。灰色结点v_i为可见结点，白色结点h_i为隐藏结点。可见结点和隐藏结点都是随机变量。

3.3.3.1 运行与采样

在介绍Hopfield网络时，我们提到该网络在模式提取时，通过迭代更新每个神经元结点取值来寻找能量函数的局部最低点。玻尔兹曼机的运行方式类似，也是通过迭代方式达到能量最低。不同的是玻尔兹曼机是随机模型，因而这一能量最低不是状态取值上的能量最低，而是达到一种稳定分布状态。

和Hopfield网络类似，我们定义玻尔兹曼机的一个模式s为所有可见结点和隐藏结点的一种取值方式。定义模式s的能量如下：

$$E(s) = -\sum_{i,j} w_{ij} s_i s_j - \sum_i \theta_i s_i. \tag{3.8}$$

和Hopfield网络稍有不同的是，玻尔兹曼机习惯上取$s_i \in \{0,1\}$。假设该模式的概率分布为玻尔兹曼分布，即：

$$p(s) \propto e^{-\frac{E(s)}{T}}, \tag{3.9}$$

其中，T为一常数。我们考察某个结点s_i（可能是隐藏结点或可见结点），其能量变动为：

$$\Delta E_i = E(s_i = 0) - E(s_i = 1) = \sum_j w_{ij} s_j + \theta_i. \tag{3.10}$$

由玻尔兹曼分布公式，可知:

$$\frac{p_{i=0}}{p_{i=1}} = e^{-\frac{\Delta E_i}{T}}.$$

注意到$p_{i=0} = 1 - p_{i=1}$，经过简单计算可得:

$$p_{i=1} = \frac{1}{1+e^{-\frac{\Delta E_i}{T}}}. \tag{3.11}$$

上式表明在运行一个玻尔兹曼机时，应基于式 3.11对s_i进行抽样，其中ΔE_i依3.10计算。注意ΔE_i依赖与结点i相关的所有相邻结点s_j，因而是一个迭代采样过程。经过一段时间的运行后，玻尔兹曼机将达到稳定状态，该稳定状态与初始状态无关，只与模型参数相关。上述运行过程事实上是一个吉布斯采样过程，在该过程中，每次只对一个结点s_i 依条件概率$P(s_i|s_{-i})$进行采样，其中s_{-i}表示去除结点s_i外模式s的取值。在后续章节我们可以看到上述采样过程在较宽泛的假设下收敛到一个稳态分布。

和Hopfield网络相比，玻尔兹曼机是一个随机网络，不能像Hopfield网络那样收敛到一个与输入模式最接近的记忆模式，而是依概率生成各种模式。虽然在采样过程中有更大概率首先生成和输入模式接近的记忆模式，但最终会稳定到模型参数决定的概率分布上，与初始输入无关。

3.3.3.2 训练方法

玻尔兹曼机的参数包括每个神经元的偏置量$\{\theta_i\}$和神经元间的连接权重$\{w_{ij}\}$，确定了这些参数即确定了玻尔兹曼机所代表的概率分布，即式3.9和式 3.8。在实际操作时，我们希望一个玻尔兹曼机能尽可能代表训练样本的实际分布规律。注意到训练样本对应模型的可见结点，为便于区分，我们记训练数据为v，其实际分布为$P^+(v)$。同时，记玻尔兹曼机所代表的分布为$P^-(v)$，该分布可通过对隐藏结点的边缘化得到：

$$P^-(v) = \sum_h P(v,h).$$

模型训练的任务是调整玻尔兹曼机的参数$\{w_{ij}, \theta_i\}$，使该模型其所代表的分布$P^-(v)$和实际数据分布$P^+(v)$尽可能相似。采用Kullback－Leibler (KL) 散度来描述这两个分布间的距离，可得到学习目标函数如下：

$$L(\{w_{ij}, \theta_i\}) = \sum_v P^+(v) \ln \frac{P^+(v)}{P^-(v)}. \tag{3.12}$$

对上式进行最小化，即可完成对玻尔兹曼机的训练。实际实现时，可采用梯度下降法对上述KL散度进行优化，即：

$$w_{ij} = w_{ij} - \alpha \frac{\partial L}{\partial w_{ij}}, \tag{3.13}$$

$$\theta_i = \theta_i - \alpha \frac{\partial L}{\partial \theta_i}, \tag{3.14}$$

其中α为学习率。通过简单计算可得如下梯度公式 [1]：

$$\frac{\partial L}{\partial w_{ij}} = -(p_{ij}^{+} - p_{ij}^{-}), \tag{3.15}$$

$$\frac{\partial L}{\partial \theta_i} = -(p_i^{+} - p_i^{-}), \tag{3.16}$$

其中p_{ij}^{+}表示实际数据集中神经元s_i和s_j同时激发的概率，p_{ij}^{-}表示玻尔兹曼机的稳态分布中神经元s_i和s_j同时激发的概率。p_i^{+}和p_i^{-}分别表示依实际数据分布和玻尔兹曼机的稳态分布，神经元s_i的激发概率。

从形式上看，公式 3.15和公式 3.16代表的训练过程非常简单，对某一参数更新时只需考虑与该参数相关的神经元结点的激发概率，考察基于模型计算得到这一概率是否符合实际即可。从另一个角度看，这一训练事实上与Hebbian 准则是一致的：当在训练数据中看到两个神经元同时激发时，p_{ij}^{+}较大，这时如果p_{ij}^{-}较小，说明模型对这一协同激发性描述不够，则公式 3.15为负值，因此依式 3.13对w_{ij}进行更新时，会使w_{ij}增加。这正是Hebbian准则中的“同时激发的神经元连接增强”原则。

虽然形式上很简单，在实际实现时，玻尔兹曼机的训练却相当困难。这是因为$p^{-}(v)$的计算非常复杂，需要玻尔兹曼机运行到稳态分布才能得到较好的估计。然而，运行到稳态分布不仅要耗费大量时间，而且是否已经达到稳态很难判断。正是因为这些困难，通用的玻尔兹曼机在实际应用中并不普遍。尽管如此，这一模型所带来的理论价值非常明显：一是其训练过程与Hebbian准则的一致性，从某一方面证明该模型可以部分模拟人类神经系统的学习方式；二是这一模型连接了贝叶斯方法和神经模型方法两大学派，一方面证明了神经学习中可利用概率方法，另一方面表明概率模型可以允许更通用的结构，如同质的结点和统一的条件概率；三是这一模型连接了物理学中某些简单的动态系统（如铁磁化过程）和机器学习中的概率模型 [71]，揭示了概率方法与物理学的某些深刻联系。

3.3.4 受限玻尔兹曼机

通用玻尔兹曼机很难训练，但如果对其结构进行若干限制，则可得到有效的训练方法。一种限制结构是将可见结点$\{v_i\}$和隐藏结点$\{h_j\}$分为两组，只有不同组的两个结点可以互相连接。这一结构称为限制性玻尔兹曼机（Restricted Boltzmann Machine, RBM），最初由Paul Smolensky 在1986年提出，当时称为Harmonium [132]。一个RBM的结构如图 3.18所示，其中灰色结点表示可见结点，白色结点表示隐藏结点，这两组结点间有无向边连接。

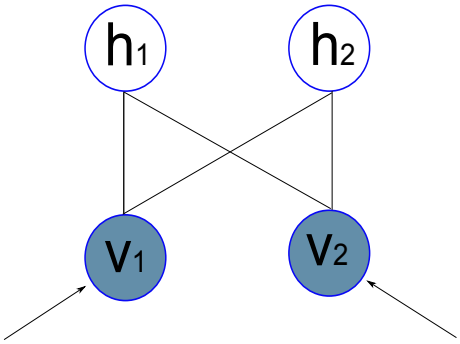


Fig. 3.18 限制性玻尔兹曼机（Restricted Boltzmann Machine, RBM）结构图。灰色结点为可见结点，白色结点为隐藏结点。不同组间的结点间可以无向边连接，同组结点间无连接。

RBM有广泛应用。首先，RBM可以认为是一种模式学习方法，其隐变量可以表达数据的显著模式。Hinton等利用这一能力在文本分析中学习主题模型 [67]。其次，如果隐变量比较少，可以认为是一种数据降维方法 [66]。第三，因为RBM可以学习数据中的主要特征，去掉干扰，因此可用作特征提取模型 [29]。第四，RBM多用于非监督学习，但如果可见变量中包含某些目标变量，则RBM也可用于监督学习，如分类任务 [87]。

3.3.4.1 RBM的运行

RBM运行方法和通用玻尔兹曼机类似，但由于引入受限结构，运行起来更加简单。考虑RBM的能量函数如下：

$$E(v,h) = -\sum_i a_i v_i - \sum_i b_i h_i - \sum_{ij} w_{ij} v_i h_j,$$

基于这一能量函数可定义如下联合概率分布：

$$P(v,h) = \frac{1}{Z}e^{-E(v,h)}, \tag{3.17}$$

其中$Z = \sum_{v,h} e^{-E(v,h)}$是归一化因子（也称为Partition Function）。这一因子由模型参数决定，与模型状态无关。RBM的受限结构极大简化了条件概率计算：给定隐藏结点，每个可见结点的概率分布是条件独立的（Conditional Independent）；反之，给定可见结点，隐藏结点也具有同样属性。这一条件独立属性形式化如下：

$$P(v|h) = \prod_i P(v_i|h), \tag{3.18}$$

$$P(h|v) = \prod_j P(h_j|v). \tag{3.19}$$

通过简单计算可得：

$$P(v_i = 1|h) = \sigma(a_i + \sum_j w_{ij}h_j), \tag{3.20}$$

$$P(h_j = 1|v) = \sigma(b_j + \sum_i w_{ij}v_i). \tag{3.21}$$

可见，RBM中的条件独立假设极大简化了模型运行，使得吉布斯采样得以分块进行（Block Gibbs Sampling）。具体运行过程如下：给定一个初始状态，基于当前观察变量v，依公式 3.19和公式 3.21采样出h；基于采样得到的h，依公式 3.18和公式 3.20采样出v。如此循环采样，即可收敛到稳态分布。

3.3.4.2 RBM的训练

在训练过程中，我们希望模型在训练数据上的概率更高，或能量更低。由式 3.17 可知：

$$P(v) = \frac{1}{Z}\sum_h e^{-E(v,h)}.$$

对一个训练集$D = \{v^{(n)} : n = 1,2,...,N\}$，其目标函数可写成如下最大似然形式：

$$L(\{w_{ij},a_i,b_i\}) = \frac{1}{N}\sum_n \ln P(v^{(n)}) = \frac{1}{N}\sum_n \{\ln[\sum_h e^{-E(v^{(n)},h)}] - \ln[Z]\}.$$

可采用梯度下降法对上式中的参数进行优化。以w_{ij}为例，注意到$-E(v,h)$中只有一项和w_{ij}相关，因而有：

$$E(w_{ij}) = -v_i h_j w_{ij} + const.$$

代入目标函数：

$$L(w_{ij}) = \frac{1}{N}\sum_n \{\ln[\sum_h e^{v_i^{(n)} h_j w_{ij} + const}] - \ln[Z]\}.$$

计算目标函数对w_{ij}梯度为：

$$\begin{aligned}
\frac{\partial L(w_{ij})}{\partial w_{ij}} &= \frac{1}{N}\sum_n \frac{[v_i^{(n)} h_j e^{v_i^{(n)} h_j w_{ij} + const}]_{h_j=1}}{\sum_h e^{v_i h_j w_{ij} + const}} - \frac{\partial \ln[Z]}{\partial w_{ij}} \\
&= \frac{1}{N}\sum_n [(v_i^{(n)} h_j) P(h_j = 1 | v^{(n)})] - \frac{\partial \ln[Z]}{\partial w_{ij}} \\
&= < v_i h_j >_{data} - \frac{\partial \ln[Z]}{\partial w_{ij}},
\end{aligned}$$

其中$< v_i h_j >_{data}$表示基于实际训练数据得到的$v_i h_j$的期望。注意到$Z = \sum_v \sum_h e^{-E(v,h)}$，因此可用同样的方法提取出和$w_{ij}$相关的项进行求导，可得

$$\frac{\partial \ln[Z]}{\partial w_{ij}} = v_i h_j P(v_i, h_j) =< v_i h_j >_{model},$$

其中$< v_i h_j >_{model}$表示基于当前模型的稳态分布得到的$v_i h_j$的期望。综合起来，有：

$$\frac{\partial \ln P(v)}{\partial w_{ij}} =< v_i h_j >_{data} - < v_i h_j >_{model} . \tag{3.22}$$

基于相似的过程，可对偏置a和b进行更新：

$$\frac{\partial \ln P(v)}{\partial a_i} =< v_i >_{data} - < v_i >_{model},$$

$$\frac{\partial \ln P(v)}{\partial b_j} =< h_j >_{data} - < h_j >_{model} .$$

如果我们回顾下通用玻尔兹曼机的训练公式 3.15，就会发现上述RBM的训练公式 3.22 和式 3.15 十分相似，都是“局部训练”，即对某一参数w_{ij}的更新仅考虑与该连接相关的两个结点。这一特点显然是和玻尔兹曼机的能量函数形式相关的。

3.3.4.3 对比散度训练

上述训练方法看起来虽然简单，但在实际实现时效率依然很低，因为计算$< v_i h_j >_{model}$需要模型运行到稳态分布。Hinton在2002年提出了一种对比散度（Contrastive Divergence，CD）算法 [65]，该方法只需几次吉布斯采样即可完成一次参数更新，而不需系统运行到稳态分布。具体过程如下所示：

while *Not Converge* **do**
  从训练数据中随机一个样本v;
  以v为输入变量，基于公式 3.19采样出隐变量h;
  基于h，利用公式 3.18得到v的重构样本$\hat{v}$;
  基于$\hat{v}$，重复利用公式 3.19采样出隐变量$\hat{h}$;
  对模型更新如下：

$$\Delta w_{ij} = \eta(< vh > - < \hat{v}\hat{h} >),$$

$$\Delta a_i = \eta(< v > - < \hat{v} >),$$

$$\Delta b_j = \eta(< h > - < \hat{h} >),$$

  其中η是学习率，期望$< \cdot >$由上述采样过程得到的训练样本和重构样本计算得到。
end

CD算法的优化目标并不是求最大似然函数，而是模拟对比散度（即两个KL散度的差）的梯度，但也仅是近似的。Sutskever和Tieleman证明，CD的更新方式并不对应任何一个目标函数的梯度 [135]。仅管如此，CD算法在实际应用中依然有良好表现，极大提高了模型训练效率。对CD方法的一个改进是对模型的采样持续进行（模型参数在采样过程中会被持续更新），而不是对每个训练数据重新开始采样。这一方法称为Persistent CD （PCD） [142]。CD算法为RBM提供了高效的训练工具，使RBM 得以广泛应用，特别是有力推动了深度学习技术的发展 [68]。

3.3.4.4 RBM模型变种

研究者基于RBM提出很多改进模型。Larochelle在2008年提出用于监督学习的RBM [87]。在这种RBM中，可见变量中除了包含输入特征外，还包含输入的类别标记。运行时固定输入特征，对类别标记计算后验概率，即可实现分类任务。这一模型可基于最大似然或对比散度训练，同时也可以基于区分性目标训练，即最大化$P(c|v)$，其中c是类别变量，v是数据特征，二者都是可见变量。不仅如此，监督学习和非监督学习还可以同时做为训练目标，实现混合训练或多任务训练。

Lee等人在2009年提出卷积RBM（Convolutional RBM, CRBM） [90]。类似卷积神经网络，CRBM将隐藏结点分为若干组，每一组称为一个特征平面（Feature Map），每个特征平面的结点共享连接参数。Nair等人在2009年提出一种混合RBM模型 [102]，该模型类似混合高斯模型，引入第三组向量（类似高斯混合模型的权重）来控制不同组RBM在能量函数中的贡献。

3.3.5 自编码器

自编码器（Auto Encoder，AE）是另一种基于记忆的神经模型。一个标准AE包括两个部分：一个编码器（Encoder）和一个解码器（Decoder），其中编码器$f_\phi(x)$将原始数据x编码到一个特征空间，生成特征h，解码器$g_\theta(h)$基于特征h对原始数据进行重构$\hat{x} = g(h)$。一个典型的AE结构如图 3.19所示。AE 的学习目标是使得重构的数据和原始输入数据尽可能接近，即恢复误差最小。对大多数连续数据，这一目标可通过将网络训练目标函数设为最小平方误差来实现，即：

$$L(\theta, \phi) = ||\hat{x} - x||^2 = ||g_\theta(f_\phi(x)) - x||^2.$$

定义了上述训练目标后，基于BP算法即可对网络参数ϕ, θ进行优化。

AE的概念早在1987年就出现了[153, 7]，只是直到深度学习发展起来后才受到更多重视。一个可能的原因是浅层网络对特征的学习能力较弱，而深层网络的训练一直存在困难，直到Hinton等提出用RBM进行预训练之后才得以解决 [66]。关于深度学习的知识，我们将在下一章具体讨论。

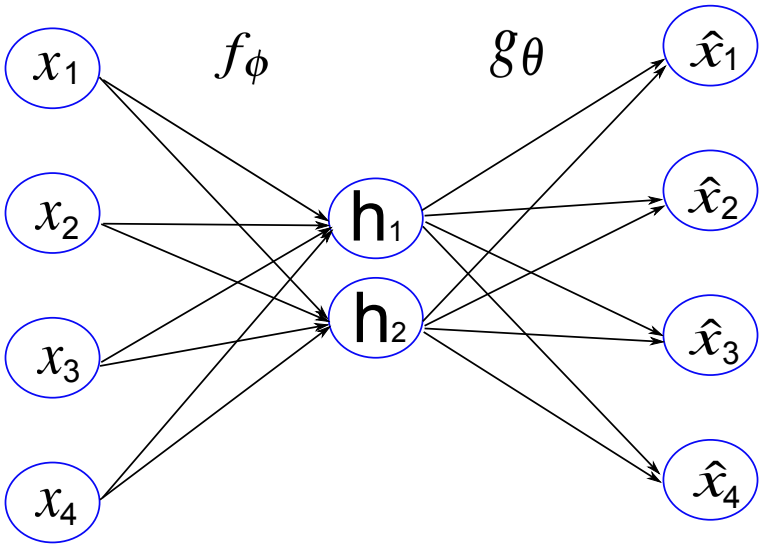


Fig. 3.19 自编码器（AE）的网络结构。编码器$f_\phi(x)$将x编码到特征空间，生成特征（编码）h，解码器$g_\theta(h)$基于该特征生成x的重构$\hat{x}$。

3.3.5.1 AE与其它模型的关系

AE和PCA有天然联系 [21, 25]。在线性模型一章我们提到过，PCA的训练目标是使得线性变换后得到特征在对输入进行重构时误差最小，因此PCA可以认为是一种特殊的AE，其中编码器和解码器是线性的，且二者共享参数。显然，AE的结构比PCA更灵活，允许非线性编码和解码，允许解码器和编码器有独立的参数，允许更灵活的目标函数。因此，AE具有比PCA更强大的学习能力 [66]。

另一方面，AE与RBM也有紧密关系。由RBM的对比散度训练过程可知，RBM的训练目标也是对原始输入数据的重构误差最小，这与AE的训练目标非常相似。Bengio [11]讨论了这种相关性，证明RBM中的CD训练等价于AE中的梯度。AE和RBM的区别在于RBM的编码和解码都是随机的，而AE的编解码过程是确定的。这一区别启发研究者将随机变量引入到AE中，由此产生了一种随机的变分自编码器（Variational AE, VAE） [83]。VAE假设数据由解码器$p_\theta(x|h)$随机产生，其中h符合某一先验概率$p(h)$，而编码器用来模拟该生成过程的后验概率$q_\phi(h|x)$。VAE提供了一种将贝叶斯方法和神经网络相结合的新思路，既可利用神经网络的强大学习能力，也可以对学习过程进行概率约束。

3.3.5.2 其它约束

AE的学习目标是对数据进行重构，因此需要一定的约束条件才可避免学习到平凡解（等值映射）。在传统AE结构中，特征层的维度小于数据维度，这事实上提供了一种低维约束，强制网络学习显著特征（类似PCA中的主成分）。低维约束有一定局限性，研究者提出了更多约束方法来提高AE的学习能力。一种约束是使得生成的特征具有稀疏性，即稀疏编码（Sparse Coding [104]）。传统的稀疏编码不存在一个参数化的编码器，而是基于一个优化过程，在优化目标中加入一个鼓励稀疏特征的正则项:

$$h^* = \arg\min_h \{L(g(h),x) + \lambda\Omega(h)\},$$

其中$L(g(h),x)$是重构误差，λ是控制稀疏性的参数，$\Omega(h)$是鼓励稀疏性的正则项。常见的正则项包括l_0范数和l_1范数。上述优化过程一般计算量较大，一种可能的方法是用神经网络来学习这一编码过程，如PSD方法 [82]。稀疏自编码器（Sparse AE）则是将鼓励稀疏编码的正则项引入到AE中，形式化如下：

$$L_{sparse}(\theta,\phi) = ||g_\theta(f_\phi(x)) - x||^2 + \lambda\Omega(h).$$

加入该正则项后，特征空间不再受低维限制，其维度甚至可以大于输入向量维度，因而可学习更复杂的分布结构；另外，稀疏编码将非显著维度置零，使得特征向量具有更强的解释性。

另一种引入稀疏特征的方法是对编码器的Jacobian矩阵进行约束，目的是使特征对输入的变化更加鲁棒，不受局部变化的影响。写成目标函数形式为：

$$L_{CAE}(\theta,\phi) = ||g_\theta(f_\phi(x)) - x||^2 + \lambda||\frac{\partial h}{\partial x}||^2.$$

这一模型称为Contractive AE （CAE） [122, 121]。显然，如果特征层的激活函数在$h=0$的梯度为0，CAE和Sparse AE一样倾向于生成稀疏特征。

除了稀疏性，另一种约束方法是向输入数据中加入噪音，包括白噪音、实际场景噪音或某些维度的数据破坏或缺失，利用AE从这些加噪数据中恢复原始数据。这种加噪训练相当于在原始目标函数上加入一个正则项，使得目标函数对训练数据变化的敏感性降低 [53]。这一模型称为去噪自编码机（Denoising Auto Encoder, DAE) [145, 146].

最近研究表明，DAE具有学习数据分布的能力 [144, 2, 15]。这些研究得到的一个结论是对一个编码/解码系统，如果噪音和重构余量（Reconstruction Residual）都符合高斯分布，则DAE可以学习数据的概率密度函数。具体来说，设噪音符合如下规律：

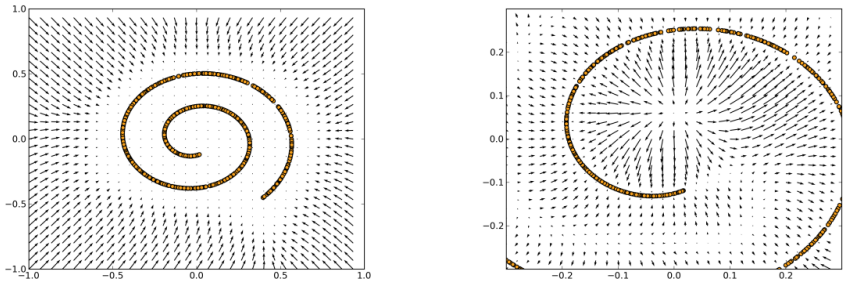
$$C(\tilde{x}|x) = N(\tilde{x}; \mu = x, \Sigma = \sigma^2 \mathbf{I}),$$

其中$\tilde{x}$为加入噪声后的数据。设DAE训练准则为：

$$||g(f(\tilde{x})) - x||^2,$$

则$\frac{g(f(x))-x}{\sigma^2}$是$\frac{\partial \ln(Q(x))}{\partial x}$的一致估计，其中$Q(x)$表示数据的实际分布。

从这一结果可以得到若干重要结论。首先，如果对那些可以精确重构的点x，有$\frac{g(f(x))-x}{\sigma^2} = 0$，则在点$x$处的数据分布概率$Q(x)$最大化。第二，对那些无法精确重构的点，重构误差$\frac{g(f(x))-x}{\sigma^2}$事实上与$\ln(Q(x))$在该点的梯度方向是一致的。注意$\ln(Q(x))$可以认为是数据$x$的能量场，这说明DAE学习了数据能量场中的梯度，梯度越大的地方，重构误差越大。图 3.20给出DAE学习到的能量场 [2]。基于上述发现，Bengio等对DAE提出了一种概率解释，认为DAE可以视作一个生成模型，利用DAE进行反复迭代（即将DAE的输出结果重新加入噪声作为下一次输入），可以生成对数据分布的采样 [15]。



(a) $r(x) - x$ vector field, acting as sink, zoomed out　　(b) $r(x) - x$ vector field, close-up

Fig. 3.20 DAE可以学习数据能量场 [2]。左图是远景图，右图是放大图。图中曲面表示实际数据分布位置，每个位置x处的箭头表示$r(x) - x$，其中$r(x) = g(f(x))$。可以看到在实际数据分布位置，能量场的梯度为零，达到局部极小值。注意右图中间位置，能量梯度也为零，但这些位置并非数据分布位置，因此是能量局部极大值点而非局部极小值点。

3.4 基于过程的模型

上文我们提到的映射模型和记忆模型可以认为是一种“静态模型”，即仅描述数据的分布特性。在实际应用中，我们还常遇到另一种问题，在这些问题中，样本的出现具有很强的序列性，且序列中的样本间存在很强的时序相关性，例如语音信号中不同时刻的采样点，自然语言理解任务中的文本序列，股票交易信号中不同时刻的交易记录，脑电波信号中不同时段的样本等。这类和时序相关的问题称为序列问题。解决序列问题的模型通常称为动态模型或过程模型。

解决序列问题的基本思路是使模型本身带有时序性，使之可以描述序列信号中的动态发展。传统方法包括各种动态概率模型，如离散状态空间的隐马尔科夫模型（HMM） [9, 117]、连续状态空间的线性卡尔曼滤波器（Kalman Filter） [80, 124, 59]，或更通用的动态贝叶斯模型（Dynamic Bayesian Network, DBN） [34, 35, 48, 45, 101, 37]方法等。这些方法都对数据的动态性和随机性做出某种概率假设（一般为线性和高斯的），基于该假设对模型进行训练和推理。简单的动态概率模型很容易训练，但适用性不强，不能描述真实数据的实际动态特性；复杂的动态模型不论训练和推理都较困难。虽然研究者提出了一些近似方法以提高训练和推理效率（如变分或采样方法），但这些近似方法有可能带来较大偏差。

基于过程（序列）的神经模型利用神经网络来模拟这种动态性。在这种神经网络中，网络输出不仅依赖当前输入，还可以依赖前序所有输入和输出，因而可学习数据中的序列相关性。这种网络通常称为递归神经网络（Recurrent Neural Network, RNN）。值得注意的是，序列问题通常是和时间序列相关的，因此RNN通常用在时序信号建模上，但RNN可以处理更广义上的序列，如逻辑序列。比如我们解一道数学题，完成一个化学实验，这些任务一般需要几个步骤，这些步骤之间固然有时序性，但更重要的是逻辑上的先后性。近年来RNN在这些逻辑序列建模上取得了一系列成果 [58, 17]。

RNN是个庞大的家族，最简单的是全连接网络，允许每个结点对所有其它结点进行递归连接。如上文所述的Hopfield网络 [71]，即可认为是这种全连接RNN。这一结构虽然通用，但训练起来很困难。可以考虑两种解决方法，一是对这些递归连接保持随机初值，不必参与训练，如Echo State Network （ESN） [76]。另一种方法是引入某些结构化限制，以降低网络复杂度。RBM可以认为是将结点分为两组，两组之间存在递归连接的RNN。更常见的方式是将结点分层，只允许同层结点之间存在跨时间的递归连

接，如Elman网络 [40]，或只允许输出层和下一时刻的隐藏层存在递归连接，如Jordan网络 [77, 78]。这两种网络是最常用的RNN结构。我们从Elman网络开始讲起。

3.4.1 Elman RNN

Elman RNN的结构如图 3.21所示。和MLP相比，可以看到隐藏层的输出被回传到隐藏层，作为下一时刻的输入，因此形成一个递归网络。数学表示为：

$$h_t = \sigma_h(W_{(h)}x_t + U_{(h)}h_{t-1} + b_{(h)}),$$
$$y_t = \sigma_y(W_{(y)}h_t + b_{(y)}),$$

其中x_t和y_t为网络在t时刻的输入和输出，h_t为t时刻的隐藏结点，$\{W_{(h)}, U_{(h)}, b_{(h)}, W_{(y)}, b_{(y)}\}$为模型参数。

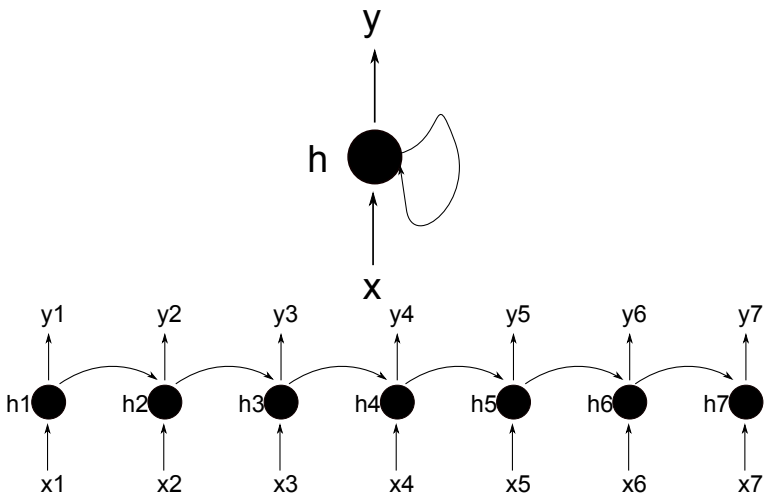


Fig. 3.21 Elman RNN网络结构。上图是带有递归连接的网络结构，下图是将递归连接按时间展开后的等价网络。

Elman RNN训练可采用传统BP算法。如图 3.21所示，如果将RNN的递归结构依时间轴展开，可以发现它等价于一个无限长的深层前向网络。基于这一等价网络，我们可以将每一时刻的预测误差沿时间轴反方向回传，

对RNN中的参数进行修正。这一沿时间BP的算法一般称为BP Through Time（BPTT） [152, 150]。理论上说，任何一个时刻的预测误差都会回传到所有历史状态，但在实际中我们一般都会限制一个回传长度。这是因为随着时间增长，信号间的相关性变弱，较远处的状态不会对当前预测产生显著影响；另一方面，BPTT回传步骤越多，梯度发生爆炸或消失的可能性越大 [12]，训练越困难，即使不做长度限制，RNN也很难将信息回传到过去的状态。这种Truncated BPTT如图 3.22所示。

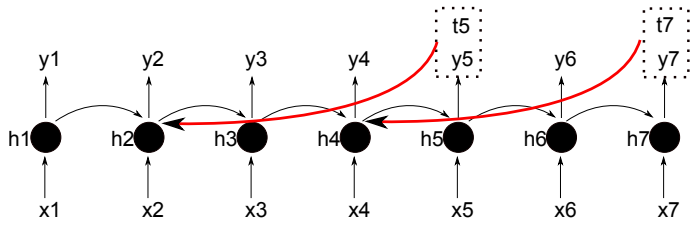


Fig. 3.22 Truncated BPTT。在时刻n，网络输出为y_n，对应的目标为t_n，由此产生的误差向前回传（红色曲线）。回传时我们限制传递长度为3。

Elman网络很容易扩展到其它更复杂的结构。图 3.23给出了几种RNN的扩展结构。在这些结构中，双向RNN不仅考虑过去历史，还考虑未来，通常会带来更好的建模能力。近年来，深层RNN得到广泛应用。与单层RNN相比，深层RNN可以在抽象特征上学习时序相关性，事实上提供了一种将特征学习和时序学习结合在一起的有效方式，因而受到广泛重视。如百度公司的DeepSpeech2语音识别系统，其声学模型就是一个包含了3 个卷积层和7个递归层的深度RNN网络 [4]。

3.4.2 门网络

前面讨论的RNN模型存在一个显著缺陷：它虽然在理论上具有学习长时特性的能力，但因为训练上的困难 [12]，往往只能学到较短的时序关系。Gers等在论文中提到，标准RNN大约只能学到5-10个时长的序列模式 [47]。研究者提出了各种解决方案，包括时间延迟网络 [86]，引入时间常数对隐藏层结点的输出进行平滑 [100]，利用多种时间常数以学习不同尺度的时序结构 [100]，引入卡尔曼滤波器对隐藏层结点输出进行平滑 [116]。这些方法都

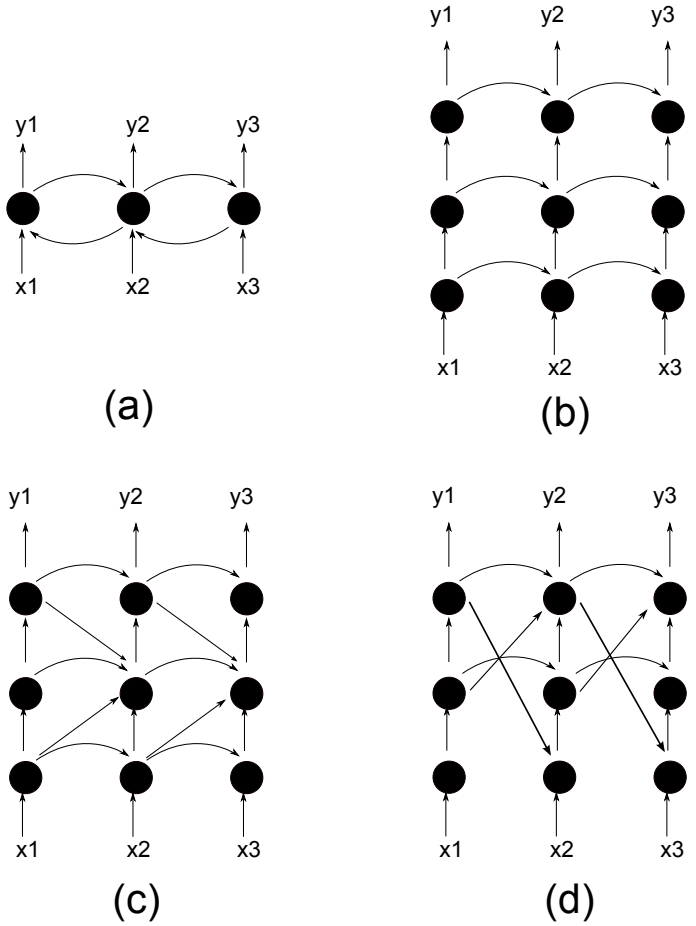


Fig. 3.23 几种RNN扩展结构。(a)双向RNN；(b)深层RNN；(c)深层RNN包含层间递归连接；(d)深层RNN包含跨层递归连接。

起到一定效果，但直到1997年Hochreiter等人提出门网络（Gate Network），才真正较好地解决了RNN的长时学习问题。

Hochreiter等人提出的门网络称为Long Short-Term Memory （LSTM），或称为长短时记忆单元网络 [70]，其结构如图 3.24所示。和标准RNN不同的是，LSTM网络将隐藏结点替换成一个个复杂的记忆单元（LSTM）。这些单元本身具有记忆功能，因此不再需要显式的递归连接。

一个LSTM单元结构如图 3.25所示。这一结构包括三个依赖输入变量x的门结构，分别是输入门（Input Gate）、遗忘门（Forget Gate）和输出门（Output Gate）。这些门结构用来控制信息的记忆、更新和输出。具体来

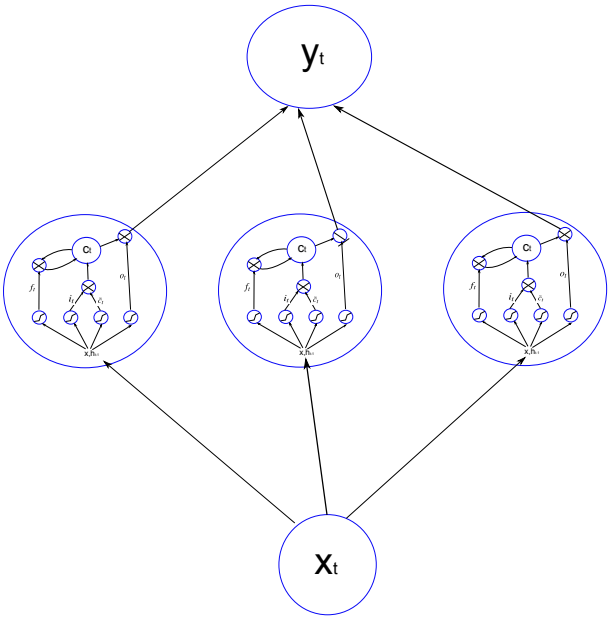


Fig. 3.24 LSTM网络。输入结点顺序读入时间序列，输出结点顺序进行序测。隐藏层的每个结点为一个LSTM单元，该单元本身具有记忆功能，因此不需要一个时序上的递归连接。

说，输入门决定当前信息是否重要到需要被记忆；遗忘门控制对当前记忆单元内容的更新；输出门控制在当前输入情况下是否应该对记忆内容进行输出。设t时刻的输入、遗忘、输出门分别为i_t,f_t,o_t，令c_t代表该时刻的记忆单元内容，h_t代表网络输出，则该LSTM单元的动态性可写成如下公式：

$$
\begin{aligned}
i_t &= \sigma(W_{(i)}x_t + U_{(i)}h_{t-1}) \\
f_t &= \sigma(W_{(f)}x_t + U_{(f)}h_{t-1}) \\
o_t &= \sigma(W_{(o)}x_t + U_{(o)}h_{t-1}) \\
\tilde{c}_t &= \tanh(W_{(c)}x_t + U_{(c)}h_{t-1}) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\
h_t &= o_t \circ \tanh(c_t).
\end{aligned}
$$

从直观上，LSTM通过引入门结构，可以控制记忆信息随不同环境（输入）发生相应变化的动态性，从而可区分重要信息和非重要信息，进行有选

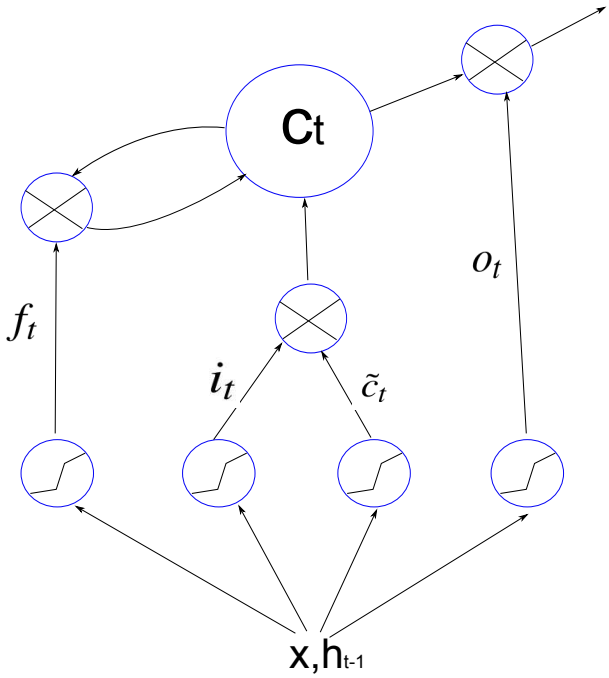
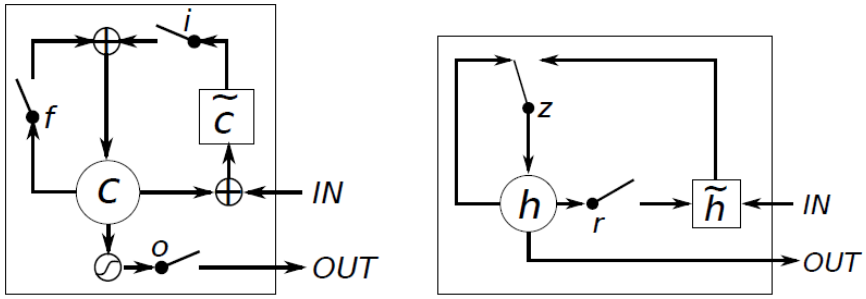


Fig. 3.25 LSTM单元。输入通过非线性变换，分别形成控制信息流动的输入门、遗忘门和输出门，这些门结构的输出分别作用到输入信号、记忆单元、输出结果上，实现对复杂动态性的学习。

择地记忆和遗忘。这种重点记忆方法有利于学习重要的长时模式。另一方面，在记忆结构中的信息保持了对过去历史信息的缩影，这等价于在当前状态和历史状态之间引入了一条直连边（Shortcut Path），这些直连边使得当前误差信号可以传递到较远的历史状态，进而学习长时信息。从更抽象的层次看，LSTM中引入了较复杂的计算结构，特别是在门结构的输出与信息变量（输入，输出和记忆单元）之间引入乘法关系。这事实上突破了传统神经网络中的矩阵乘加操作，因而引入更大自由度来学习较复杂的记忆规则。后面我们会看到，当引入更丰富的操作时，可能会得到更有效的学习模型，这一结构被称为“神经图灵机”（Neural Turing Machine, NTM） [56]。

LSTM在计算上较复杂。最近，Cho等人提出了门递归单元（Gated Recurrent Unit, GRU） [26]代替LSTM结构。GRU用一个更新门和一个重置门来控制信息的记忆和更新，其计算公式如下：

$$
\begin{aligned}
z_t &= \sigma(W_{(z)}x_t + U_{(z)}h_{t-1}) \\
r_t &= \sigma(W_{(r)}x_t + U_{(r)}h_{t-1}) \\
h_t &= (1-z_t)\circ h_{t-1} + z_t \circ \tanh(W_{(h)}x_t + U_{(h)}(r_t \circ h_{t-1}) + b_{(h)}).
\end{aligned}
$$



(a) Long Short-Term Memory (b) Gated Recurrent Unit

Fig. 3.26 LSTM和GRU对比。图片来自 [26]。

图 3.26给出LSTM和GRU的结构对比。从图中可以看到，LSTM和GRU主要有两点区别：一是LSTM在输出时由一个输出门控制，而GRU直接输出当前记忆内容；二是LSTM用一个输入门控制对记忆单元的更新内容，用一个遗忘门来控制历史信息的保存，这两个门是独立的，而GRU用一个更新门来控制二者的比例。第二点区别影响可能更深刻一些，它使得LSTM的记忆单元的值是无界的，而GRU中的记忆单元是有界的。我们最近研究发现，归因于这一区别，GRU倾向于用更极化的表达来描述信息 [140]。Chung等人对GRU和LSTM做了一些对比研究，发现在这两种门结构在音乐和语音信号建模任务中表现出类似的性能 [28]。Zaremba等人对各种RNN结构进行了一些实证研究。通过搜索和验证大量可能的RNN结构，他们发现虽然可以找到一些网络结构在某些任务上超过LSTM和GRU，但这些模型并不能在所有任务上一致性地超过LSTM和GRU，说明LSTM和GRU还是非常有效的模型 [156]。Karpathy等人用可视化工具对RNN的学习方式做了探讨。基于一个字母级的语言模型建模任务，他们发现RNN的记忆单元中确实可以学到有价值的语义信息 [81]。

RNN/LSTM近年来在序列学习方面取得一系列重要进展，在语言模型 [98, 134]、语音识别 [55, 126]、语音合成 [41]、乐谱合成 [20]、语种检测 [51]、

韵律检测 [43]、机器翻译 [137]、社交信号分析 [23]等任务上都取得了不错效果。

3.4.3 序列对序列网络

RNN提供了强大的序列编码能力。给定一个序列，RNN，特别是基于各种门结构的RNN，可以通过递归方式顺序积累每一步输入的信息，并将这些信息保存在记忆单元中，从而将不定长序列编码成定长的表征向量。这一向量可以表征输入序列的主要信息。基于该向量，可以对时序数据进行分类、聚类等监督和非监督学习 [73]。另一方面，RNN也提供了强大的解码能力：给定一个初始状态，RNN可以自动运行，递归生成随机序列。这一生成模型已经被成功用在手写体字母生成 [54]和文本生成中 [136]。图 3.27给出RNN用作编码和解码任务的结构。

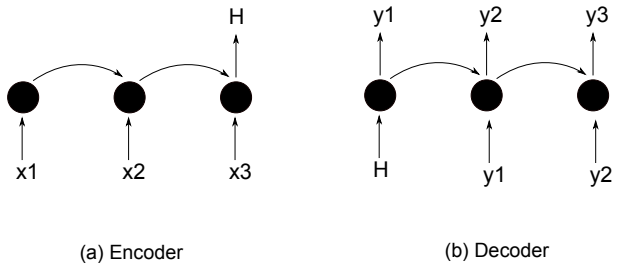


Fig. 3.27 基于RNN的（a）编码器（b）解码器。

上述编码和解码过程可以结合起来，用一个RNN对输入序列进行编码，用另一个RNN基于编码结果进行解码，这样就可以学习一个序列到另一个序列的映射关系。这一模型称为序列对序列网络（Sequence to Sequence Network, S2S），由Sutskever等人在2014年提出 [138]。S2S网络可以认为是自编码器（Auto Encoder，AE）的扩展。不同于传统AE的是，AE中的输入和输出是同一个原始特征向量，而S2S模型输入和输出是两个序列，这两个序列一般是不同的。一个典型的S2S网络如图 3.28所示。

序列对序列模型在解码时依赖由编码器生成的表征向量。将不定长的输入序列转换成定长的表征向量具有重要意义，基于这一转换，现有基于定长特征向量的机器学习方法都可用于序列学习。然而，这种方法也存在很大局限性，特别是RNN的长时学习能力还不够强，将一个序列压缩成一个定长向

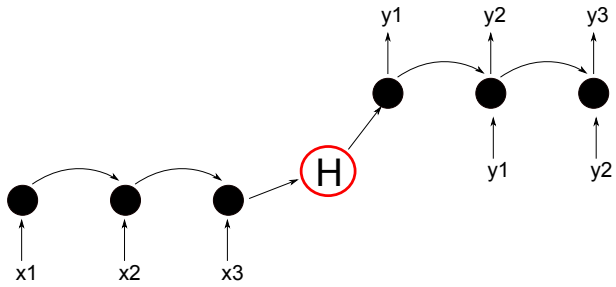


Fig. 3.28 序列对序列网络结构。输入序列通过一个RNN编码器压缩成一个全局向量H，另一个RNN作为解码器，以H为初始输入，递归生成输出序列。

量可能会损失大量信息。一种可能的方法是引入双向RNN，从序列的两个方向进行编码，并将两组编码结合起来作为表征向量。这一方法并不能完全解决RNN的信息丢失问题，特别是对较长的输入序列，双向RNN无法描述局部细节信息。

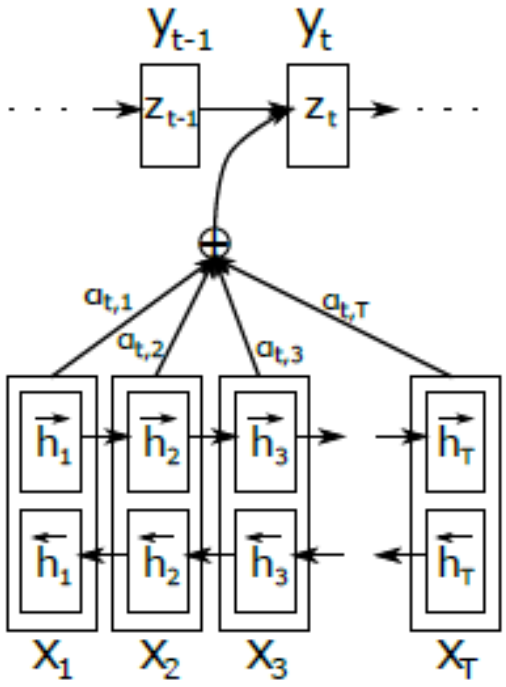


Fig. 3.29 基于关注的序列到序列网络。图片来自 [6]。

为解决这一问题，研究者提出基于关注的序列对序列模型（Attention S2S） [6, 130, 27]。在这一模型中，每一个生成步骤关注输入序列中的某一部分信息，从而可以对局部细节进行学习。如图 3.29所示，首先将输入序列编码成一个记忆单元序列$\{h_i\}$（一般使用双向RNN以提高编码精度），在生成t时刻的输出y_t时，依前一时刻的生成状态z_{t-1}对$\{h_i\}$进行选择，选择方法是计算每个h_i对当前生成步骤的贡献权重α_{ti}。具体计算公式如下：

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{k=1}^{T_x} \exp(e_{tk})},$$
$$e_{ti} = g(z_{t-1}, h_i),$$

其中g为任意一个距离测度模型（如Cosine距离或神经网络）。基于Attention权重α_{ti}，可计算当前生成所关注的局部输入信息c_t如下：

$$c_t = \sum_i \alpha_{ti} h_i.$$

由此可生成对y_t的预测和对z_t的更新：

$$y_t = f_y(z_{t-1}, c_t),$$
$$z_t = f_z(z_{t-1}, c_t),$$

其中f_y和f_z为解码RNN中相应的预测函数。

序列对序列模型，特别是引入Attention机制后，极大增强了对时序数据的建模能力。特别是可对不同领域的序列进行学习，极大扩展了神经模型的应用范围。例如在机器翻译中 [6]，Attention S2S 模型可以将各种不同语言映射到同一个语义空间，基于该语义空间实现多语言翻译。另一个例子是图像理解 [27]，将图像和对应的文本映射到同一个语义空间，可生成对图片的文本描述 [147]。类似的方法在智能问答[130]、视频理解 [143] 等领域都取得了很大成功。

3.4.4 基于*Attention*模型的诗词生成

本节以诗词生成为例介绍Attention S2S模型的强大建模能力 [149, 158]。诗词是中华民族的文化瑰宝，无数优秀诗篇至今为人传诵。诗词生成一向被认为是人类的独有能力，包含创新性、审美性、个性化等看起来机器无法模

拟的事情。然而，诗词生成本身又是一种长期学习过程，所谓“熟读唐诗三百首，不会作诗也会吟”。这意味着机器有可能从已有的诗词作品中学到某些规律，从而实现自动创作。

传统诗词生成方法一般是拼凑法：给定一个主题，可以在大量诗词库中搜索相关诗句，将这些诗句打乱后，挑选可连接在一起的片段通过一定规则组合起来成为一首新诗 [161]。这种机械拼凑的方法显然过于机器化了，既没有对句子意义的理解，也没有对规则的学习，生成的诗除了合规外观赏价值较低。

神经网络模型，特别是序列对序列模型提供了更有效的方法。和拼凑法不同的是，神经模型方法将历史上的诗词通过RNN映射到语义空间，并在该语义空间中进行句子生成。这意味着该模型在生成之前需要对句子意义进行理解，虽然这一理解仅是隐变量空间的某种表达，但却提供了深层的语义和情感信息，基于此，生成的诗句不仅在形式上更加连贯（源于RNN模型的时序连续性），而且具有更强的语义和情感约束。因为神经模型方法的出现，高质量的诗词生成已经成为可能。

早期基于神经网络的诗词生成基于句子向量 [160]，这一结构比较复杂，不利于扩展到较复杂的诗词结构（如宋词）。Wang等人在2016年提出利用Attention S2S模型解决这一问题 [149]。这一模型将整首诗作为一个完整的序列（包括断句符号）生成，其中输入是人为给定的关键词。序列对序列模型使得整个写作过程围绕同一个主题，Attention机制使生成过程在不同关键词间切换。

图 3.30 给出了该模型的主要结构，其中编码RNN将用户提供的关键词“春花秋月何时了”编码成一组隐藏向量（图下方的矩形序列），该向量包含用户的生成意图。在生成过程中，一个单向RNN网络递归运行，逐字生成整首诗。在生成每一个字的时候，对用户的意图向量进行查看，找到与当前生成状态最相关的用户意图进行下一字的生成。在生成过程中，强制加入断句、押韵、平仄等需要遵守的规则，这样就保证了生成的字串既能最大程度地符合语法和语义规则，又能使句子围绕用户的意图展开，成为一首合格的诗。下面我们给出一些生成的例子。

美人	海棠花
花香粉脸胭脂染，	红霞淡艳媚妆水，
帘影鸳鸯绿嫩妆。	万朵千峰映碧垂。
翠袖红蕖春色冷，	一夜东风吹雨过，
柳梢褪叶暗烟芳。	满城春色在天辉。

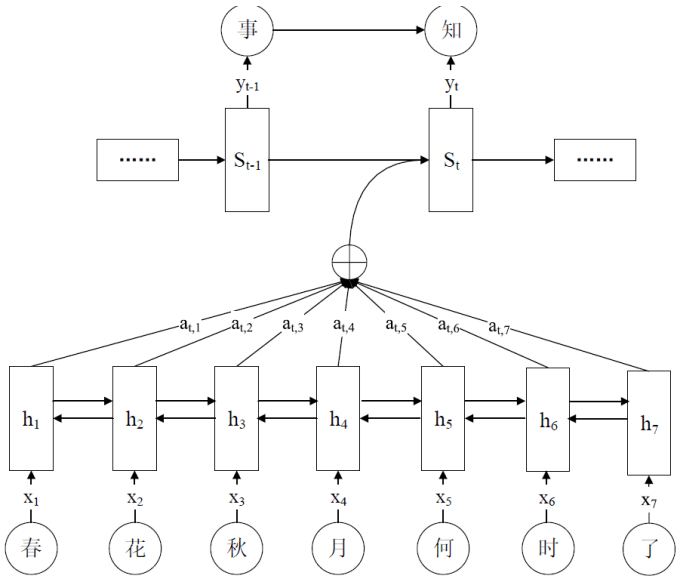


Fig. 3.30 用于古诗生成的Attention Sequence to Sequence模型。图片来自 [149]。

菩萨蛮
哀筝一弄湘江曲，
风流水上人家绿。
小艇子规啼，
不堪春去时。
花前杨柳下，
红叶满庭洒。
月落尽成秋，
愁思欲寄留。

3.5 神经图灵机

回顾神经模型的发展，可以发现模型越来越异构化。传统的前向网络仅包含矩阵乘加操作，发展到LSTM网络后，每个单元引入了门限操作，且增加了记忆单元。特别重要的是，门限值和记忆单元的更新方法都是通过数据

学习出来的。这种“端对端”的黑箱学习方式需要更多数据，学习起来也更困难，但却可以描述更复杂的实际系统。

一个很自然的想法是，如果对神经网络中的操作进一步扩充，定义一系列元操作，通过这些元操作进行组合，有可能学习更复杂的动态性。基于这一思路，Graves等人提出了神经图灵机的概念 [56]。所谓神经图灵机，是基于神经模型对传统图灵机模型的一种模拟。一个图灵机包括如下几个部分：一条无限长的纸带，一个读写头，一个控制器，一个状态寄存器。Graves利用神经网络来模拟这些单元，将一些内存开辟出来代表纸带，定义一系列寻址和读写操作来模拟读写头，用神经网络模型和内部记忆单元分别作为控制器和寄存器。经过这些定义的神经网络即可视为一个虚拟的图灵机，称为神经图灵机（Neural Turing Machine, NTM）。Graves等提出的神经图灵机模型如图 3.31所示。

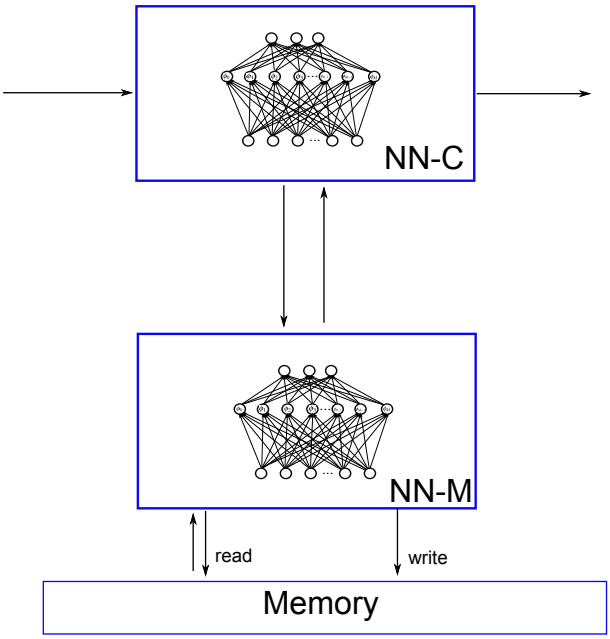


Fig. 3.31 神经图灵机模型。内存单元相当于读写纸带，用神经网络NN-C作为中心控制器（CPU），其中记忆单元类似寄存器。中心控制器通过发出读写指令给一个神经网络NN-M，该神经网络通过定义一系列寻址操作来模拟图灵机中的读写头。读写指令被NN-M转化成对内存的访问和读写。

NTM并不是图灵机的简单复制。首先，NTM可以定义更灵活的寻址方式，如基于内容的寻址和基于地址的寻址，对应的读写头操作可以非常复杂。同时，读写方式也可以灵活定义。如在Graves等人的实现中，可利用Attention机制定位与当前寄存器相关的内存单元，再基于Attention的强度对该内存做读写操作。除了这一基于内容的寻址机制，也可以通过当前状态直接计算出下一个读写位置（如基于移位操作）。最近，Graves等人扩展了NTM方面的工作，提出可微分神经计算机（Differentiable Neural Computer, DNC）模型，引入更有效的寻址方法来提高复杂任务上的计算能力 [57]。图3.32给出了DNC的基础结构。

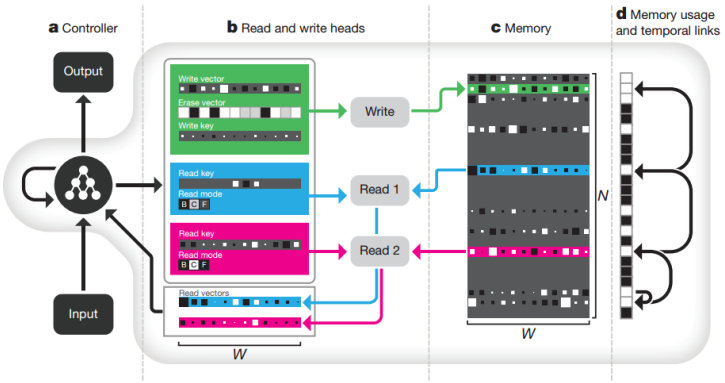


Fig. 3.32 可微分神经计算机（DNC）结构 [57]。注意图（d）中引入的Temporal link寻址机制，可实现更灵活的寻址，进而处理更复杂的任务。

NTM是一个非常强大的计算框架，给机器学习研究者提供了广阔的想象空间，有可能在未来带来AI领域的深刻变革。这种强大的建模能力可以归因于两个方面：内存机制的引入和控制逻辑的可学习性。首先，引入内存机制相当于为神经网络提供了一个记事本，极大扩展了神经网络信息记忆能力和推理能力。特别是内存的读写机制是由数据驱动学习得到的，这意味着寻址和数据的协同优化。Weston在2014年提出了类似的概念，认为外部内存是神经网络的重要补充 [151]。另一方面，NTM引入的可学习控制逻辑极大丰富了机器学习的内容。传统机器学习中，虽然模型是可学习的，但完成一个任务仍然需要人为定义好逻辑控制过程（即算法），程序依此逻辑机械执行。NTM打破了这种固有观念，实现了真正可学习的神经机器。在NTM中，所有模块都是可训练的，包括控制逻辑本身（即神经网络参数）。这意味着未来人们可能不必再为机器编写程序，只需为它提供足够的数据，设定好学习

目标，机器即可自动组织好处理逻辑，有效完成设定的目标。Neural Program Interpreter （NPI）在这方面做出了有益探索 [120, 131]。通过给定若干例子，NPI可以学会加减、排序、交换等简单操作。

3.6 本章小结

本章简要介绍了神经模型的基本概念和一些简单神经模型的结构和训练方法。我们将神经模型分为四种：用来学习输入输出关系的映射模型，用来学习内部模式的记忆模型，用来学习时序过程的动态模型，和用来学习复杂操作的神经图灵机。在这些模型中，映射模型结构和概念最简单，训练起来相对容易。记忆模型和动态模型具有直接相关性，有些模型（如Hopfield网络）既是记忆模型，也是动态模型。这一概念在LSTM模型和NTM模型中变得更加清晰：之所以某些模型同时具有记忆性和动态性，是因为其记忆单元同时用来记忆知识和描述系统状态。NTM模型清晰定义了记忆（外部内存）、状态（寄存器）和控制逻辑（神经网络），事实上包含了其余三种模型结构。

神经模型是机器学习的重要分枝。和贝叶斯模型相比，神经模型通过同质的计算单元和灵活的连接结构来学习各种模式和过程。“同质的单元”和“灵活的结构”这两点使得神经模型具有强大的学习能力，这种能力和大量数据结合起来，可以部分模拟人类的学习过程。然而，大数据和复杂模型带来计算量的急剧增加。幸好，我们今天有了更高性能的计算工具（如GPU [118]和TPU [79]）和更高效的并行训练方法 [36]，使大数据训练成为可能。

神经模型的成功很大程度上要归功于深度神经网络（DNN）的兴起。表面上看，DNN只是加深了神经网络的层次，但稍做研究后，我们会发现这一方法带来的变革要深刻得多。我们将在下一章对深度学习方法进行介绍。

3.7 相关资源

- 本章主要参考资料包括Simon Kaykin的《Neural networks and learning machines》 [62]，Christopher Bishop的《Neural networks for pattern recognition》 [19]和Ian Goodfellow、Yoshua Bengio的《Deep Learning》 [52]。

- 文中关于神经网络发展的部分参考了Schmidhuber最近的综述论文 [127][12]。Schmidhuber的个人主页还包含关于RNN众多有价值的资料[13]。
- 关于神经网络训练，可以参考的资料包括《Neural networks: tricks of the trade》 [106]，特别是其中LeCun的“Efficient Backprop”一文很值得一读 [89]。Anthony和Bartlett的《Neural network learning: Theoretical foundations》 [5]也是很不错的参考书。递归神经网络的训练可参考Jaeger的《Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the “echo state network” approach》 [75]。
- 文中关于Kohonen网络和Hopfield网络部分参考了Wikipedia相关页面[14][15]。

[12] http://people.idsia.ch/ juergen/deep-learning-overview.html

[13] http://people.idsia.ch/ juergen/rnn.html

[14] https://en.wikipedia.org/wiki/Self-organizing_map

[15] https://en.wikipedia.org/wiki/Hopfield_network

Chapter 4
深度学习

Chapter 5
核方法

Chapter 6
图模型

Chapter 7
非监督学习

Chapter 8
非参数模型

Chapter 9
遗传学习

Chapter 10
强化学习

Chapter 11
优化方法

References

[1] Ackley DH, Hinton GE, Sejnowski TJ (1985) A learning algorithm for boltzmann machines. Cognitive science 9(1):147–169

[2] Alain G, Bengio Y (2014) What regularized auto-encoders learn from the data-generating distribution. Journal of Machine Learning Research 15(1):3563–3593

[3] Amari SI (1998) Natural gradient works efficiently in learning. Neural computation 10(2):251–276

[4] Amodei D, Anubhai R, Battenberg E, Case C, Casper J, Catanzaro B, Chen J, Chrzanowski M, Coates A, Diamos G, et al (2015) Deep speech 2: End-to-end speech recognition in english and mandarin. arXiv preprint arXiv:151202595

[5] Anthony M, Bartlett PL (2009) Neural network learning: Theoretical foundations. cambridge university press

[6] Bahdanau D, Cho K, Bengio Y (2014) Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:14090473

[7] Ballard DH (1987) Modular learning in neural networks. In: AAAI, pp 279–284

[8] Barlow H (1972) Single units and sensation: a neuron doctrine for perceptual psychology? Perception 1:371–394

[9] Baum LE, Petrie T (1966) Statistical inference for probabilistic functions of finite state markov chains. The annals of mathematical statistics 37(6):1554–1563

[10] Bengio Y (2009) Learning deep architectures for ai. Foundations and trends® in Machine Learning 2(1):1–127

[11] Bengio Y, Delalleau O (2009) Justifying and generalizing contrastive divergence. Neural computation 21(6):1601–1621

[12] Bengio Y, Simard P, Frasconi P (1994) Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks 5(2):157–166

[13] Bengio Y, Ducharme R, Vincent P, Jauvin C (2003) A neural probabilistic language model. Journal of machine learning research 3(Feb):1137–1155

[14] Bengio Y, Louradour J, Collobert R, Weston J (2009) Curriculum learning. In: Proceedings of the 26th annual international conference on machine learning, ACM, pp 41–48

[15] Bengio Y, Yao L, Alain G, Vincent P (2013) Generalized denoising auto-encoders as generative models. In: Advances in Neural Information Processing Systems, pp 899–907

[16] Bengio Y, et al (2012) Deep learning of representations for unsupervised and transfer learning. ICML Unsupervised and Transfer Learning 27:17–36

[17] Bilen H, Vedaldi A (2016) Integrated perception with recurrent multi-task neural networks. In: Advances in neural information processing systems, pp 235–243

[18] Bishop CM (1994) Mixture density networks

[19] Bishop CM (1995) Neural networks for pattern recognition. Oxford university press

[20] Boulanger-Lewandowski N, Bengio Y, Vincent P (2012) Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. arXiv preprint arXiv:12066392

[21] Bourlard H, Kamp Y (1988) Auto-association by multilayer perceptrons and singular value decomposition. Biological cybernetics 59(4):291–294

[22] Broomhead DS, Lowe D (1988) Radial basis functions, multi-variable functional interpolation and adaptive networks. Tech. rep., DTIC Document

[23] Brueckner R, Schulter B (2014) Social signal classification using deep BLSTM recurrent neural networks. In: Proceedings 39th IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2014, Florence, Italy, pp 4856–4860

[24] Caruana R, Lawrence S, Giles L (2000) Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In: NIPS, pp 402–408

[25] Chicco D, Sadowski P, Baldi P (2014) Deep autoencoder neural networks for gene ontology annotation predictions. In: Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics, ACM, pp 533–540

[26] Cho K, Van Merriënboer B, Bahdanau D, Bengio Y (2014) On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:14091259

[27] Cho K, Courville A, Bengio Y (2015) Describing multimedia content using attention-based encoder-decoder networks. IEEE Transactions on Multimedia 17(11):1875–1886

[28] Chung J, Gulcehre C, Cho K, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:14123555

[29] Coates A, Lee H, Ng AY (2010) An analysis of single-layer networks in unsupervised feature learning. Ann Arbor 1001(48109):2

[30] Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P (2011) Natural language processing (almost) from scratch. J Mach Learn Res 12:2493–2537

[31] Cortes C, Vapnik V (1995) Support-vector networks. Machine Learning 20(3):273–297

[32] Csáji BC (2001) Approximation with artificial neural networks. Faculty of Sciences, Etvs Lornd University, Hungary 24:48

[33] Cun YL, Denker JS, Solla SA (1990) Optimal brain damage. In: Proc. NIPS'90

[34] Dagum P, Galper A, Horvitz E (1992) Dynamic network models for forecasting. In: Proceedings of the eighth international conference on uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc., pp 41–48

[35] Dagum P, Galper A, Horvitz E, Seiver A (1995) Uncertain reasoning and forecasting. International Journal of Forecasting 11(1):73–87

[36] Dean J, Corrado G, Monga R, Chen K, Devin M, Mao M, Senior A, Tucker P, Yang K, Le QV, et al (2012) Large scale distributed deep networks. In: Advances in neural information processing systems, pp 1223–1231

[37] Deng L (2006) Dynamic speech models: theory, algorithms, and applications. Synthesis Lectures on Speech and Audio Processing 2(1):1–118

[38] Diederik P Kingma JLB (2015) Adam: A method for stochastic optimization. In: Proc. of ICLR

[39] Duchi J, Hazan E, Singer Y (2011) Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research 12(Jul):2121–2159

[40] Elman JL (1990) Finding structure in time. Cognitive science 14(2):179–211

[41] Fan Y, Qian Y, Xie F, Soong FK (2014) TTS synthesis with bidirectional LSTM based recurrent neural networks. In: Proc. Interspeech

[42] Feng Y, Zhang S, Zhang A, Wang D, Abel A (2017) Memory-augmented neural machine translation. EMNLP 2017

[43] Fernandez R, Rendel A, Ramabhadran B, Hoory R (2014) Prosody contour prediction with Long Short-Term Memory, bi-directional, deep recurrent neural networks. In: Proc. Interspeech

[44] Földiák P, Young MP (1995) Sparse coding in the primate cortex. The handbook of brain theory and neural networks 1:1064–1068

[45] Friedman N, Murphy K, Russell S (1998) Learning the structure of dynamic probabilistic networks. In: Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc., pp 139–147

[46] Fukushima K (1980) Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological cybernetics 36(4):193–202

[47] Gers FA, Schmidhuber J, Cummins F (2000) Learning to forget: Continual prediction with lstm. Neural computation 12(10):2451–2471

[48] Ghahramani Z (1998) Learning dynamic bayesian networks. In: Adaptive processing of sequences and data structures, Springer, pp 168–197

[49] Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: Aistats, vol 9, pp 249–256

[50] Glorot X, Bordes A, Bengio Y (2011) Deep sparse rectifier neural networks. In: Proceedings of the 14th international conference on Artificial Intelligence and Statistics (AISTATS), pp 315–323

[51] Gonzalez-Dominguez J, Lopez-Moreno I, Sak H, Gonzalez-Rodriguez J, Moreno PJ (2014) Automatic language identification using Long Short-Term Memory recurrent neural networks. In: Proc. Interspeech

[52] Goodfellow I, Bengio Y, Courville A (2016) Deep Learning. MIT Press, http://www.deeplearningbook.org

[53] Grandvalet Y, Canu S (1995) Comments on" noise injection into inputs in back propagation learning". IEEE Transactions on Systems, Man, and Cybernetics 25(4):678–681

[54] Graves A (2013) Generating sequences with recurrent neural networks. arXiv preprint arXiv:13080850

[55] Graves A, Mohamed Ar, Hinton G (2013) Speech recognition with deep recurrent neural networks. In: 2013 IEEE international conference on acoustics, speech and signal processing, IEEE, pp 6645–6649

[56] Graves A, Wayne G, Danihelka I (2014) Neural turing machines. arXiv preprint arXiv:14105401

[57] Graves A, Wayne G, Reynolds M, Harley T, Danihelka I, Grabska-Barwińska A, Colmenarejo SG, Grefenstette E, Ramalho T, Agapiou J, et al (2016) Hybrid computing using a neural network with dynamic external memory. Nature 538(7626):471–476

[58] Gregor K, Danihelka I, Graves A, Rezende DJ, Wierstra D (2015) Draw: A recurrent neural network for image generation. arXiv preprint arXiv:150204623

[59] Hamilton JD (1994) Time series analysis, vol 2. Princeton university press Princeton

[60] Hartman E, Keeler JD (1991) Predicting the future: Advantages of semilocal units. Neural Computation 3(4):566–578

[61] Hassoun MH (1995) Fundamentals of artificial neural networks. MIT press

[62] Haykin SS, Haykin SS, Haykin SS, Haykin SS (2009) Neural networks and learning machines, vol 3. Pearson Upper Saddle River, NJ, USA:

[63] Hebb DO (1949) The organization of behavior: A neuropsychological theory. Psychology Press

[64] Hertz J, Krogh A, Palmer RG (1991) Introduction to the theory of neural computation, vol 1. Basic Books

[65] Hinton GE (2002) Training products of experts by minimizing contrastive divergence. Neural computation 14(8):1771–1800

[66] Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. science 313(5786):504–507

[67] Hinton GE, Salakhutdinov RR (2009) Replicated softmax: an undirected topic model. In: Advances in neural information processing systems, pp 1607–1614

[68] Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. Neural computation 18(7):1527–1554

[69] Hinton GE, Vinyals O, Dean J (2014) Distilling the knowledge in a neural network. In: NIPS 2014 Deep Learning Workshop

[70] Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural computation 9(8):1735–1780

[71] Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. Proceedings of the national academy of sciences 79(8):2554–2558

[72] Hornik K (1991) Approximation capabilities of multilayer feedforward networks. Neural networks 4(2):251–257

[73] Hüsken M, Stagge P (2003) Recurrent neural networks for time series classification. Neurocomputing 50:223–235

[74] Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:150203167

[75] Jaeger H (2002) Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach, vol 5. GMD-Forschungszentrum Informationstechnik

[76] Jaeger H, Haas H (2004) Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. science 304(5667):78–80

[77] Jordan MI (1986) Serial order: A parallel distributed processing approach. Tech. rep., DTIC Document

[78] Jordan MI (1997) Serial order: A parallel distributed processing approach. Advances in psychology 121:471–495

[79] Jouppi NP, Young C, Patil N, Patterson D (2017) In datacenter performance analysis of a tensor processing unit. In: ISCA 17

[80] Kalman RE, et al (1960) A new approach to linear filtering and prediction problems. Journal of basic Engineering 82(1):35–45

[81] Karpathy A, Johnson J, Fei-Fei L (2015) Visualizing and understanding recurrent networks. arXiv preprint arXiv:150602078

[82] Kavukcuoglu K, Ranzato M, LeCun Y (2010) Fast inference in sparse coding algorithms with applications to object recognition. arXiv preprint arXiv:10103467

[83] Kingma DP, Welling M (2013) Auto-encoding variational bayes. arXiv preprint arXiv:13126114

[84] Kohonen T (1982) Self-organized formation of topologically correct feature maps. Biological cybernetics 43(1):59–69

[85] Krogh A, Hertz JA (1991) A simple weight decay can improve generalization. In: NIPS, vol 4, pp 950–957

[86] Lang KJ, Waibel AH, Hinton GE (1990) A time-delay neural network architecture for isolated word recognition. Neural networks 3(1):23–43

[87] Larochelle H, Bengio Y (2008) Classification using discriminative restricted boltzmann machines. In: Proceedings of the 25th international conference on Machine learning, ACM, pp 536–543

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

[89] LeCun YA, Bottou L, Orr GB, Müller KR (2012) Efficient backprop. In: Neural networks: Tricks of the trade, Springer, pp 9–48

[90] Lee H, Grosse R, Ranganath R, Ng AY (2009) Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: Proceedings of the 26th annual international conference on machine learning, ACM, pp 609–616

[91] Liou CY, Lin SL (2006) Finite memory loading in hairy neurons. Natural Computing 5(1):15–42

[92] Liu B, Wang M, Foroosh H, Tappen M, Pensky M (2015) Sparse convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 806–814

[93] Liu C, Zhang Z, Wang D (2014) Pruning deep neural networks by optimal brain damage. In: Fifteenth Annual Conference of the International Speech Communication Association

[94] Lowe D, Broomhead D (1988) Multivariable functional interpolation and adaptive networks. Complex systems 2(3):321–355

[95] Lowel S, Singer W (1992) Selection of intrinsic horizontal connections in the visual cortex by correlated neuronal activity. Science 255(5041):209

[96] Martens J (2010) Deep learning via hessian-free optimization. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10), pp 735–742

[97] McCulloch WS, Pitts W (1943) A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics 5(4):115–133

[98] Mikolov T, Karafiát M, Burget L, Cernockỳ J, Khudanpur S (2010) Recurrent neural network based language model. In: Interspeech, vol 2, p 3

[99] Minsky M, Papert S (1969) Perceptrons. MIT press

[100] Mozer MC (1993) Induction of multiscale temporal structure. Advances in neural information processing systems pp 275–275

[101] Murphy KP (2002) Dynamic bayesian networks. Probabilistic Graphical Models, M Jordan 7

[102] Nair V, Hinton GE (2009) Implicit mixtures of restricted boltzmann machines. In: Advances in neural information processing systems, pp 1145–1152

[103] Novikoff AB (1963) On convergence proofs for perceptrons. Tech. rep., STANFORD RESEARCH INST MENLO PARK CALIF

[104] Olshausen BA, Field DJ (1996) Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature 381(6583):607

[105] Olshausen BA, Field DJ (1997) Sparse coding with an overcomplete basis set: A strategy employed by v1? Vision Research 37(23):3311–3325

[106] Orr GB, Müller KR (2003) Neural networks: tricks of the trade. Springer

[107] Palm G (1980) On associative memory. Biological cybernetics 36(1):19–31

[108] Park J, Sandberg IW (1994) Nonlinear approximations using elliptic basis function networks. Circuits, Systems and Signal Processing 13(1):99–113

[109] Pascanu R, Bengio Y (2013) Revisiting natural gradient for deep networks. arXiv preprint arXiv:13013584

[110] Pascanu R, Mikolov T, Bengio Y (2013) On the difficulty of training recurrent neural networks. ICML (3) 28:1310–1318

[111] Peemen M, Mesman B, Corporaal H (2011) Speed sign detection and recognition by convolutional neural networks. In: Proceedings of the 8th International Automotive Congress, pp 162–170

[112] Polyak BT (1964) Some methods of speeding up the convergence of iteration methods. USSR Computational Mathematics and Mathematical Physics 4(5):1–17

[113] Povey D, Zhang X, Khudanpur S (2014) Parallel training of deep neural networks with natural gradient and parameter averaging. CoRR, vol abs/14107455

[114] Prechelt L (1998) Automatic early stopping using cross validation: quantifying the criteria. Neural Networks 11(4):761–767

[115] Prechelt L (1998) Early stopping-but when? In: Neural Networks: Tricks of the trade, Springer, pp 55–69

[116] Puskorius GV, Feldkamp LA (1994) Neurocontrol of nonlinear dynamical systems with kalman filter trained recurrent networks. IEEE Transactions on neural networks 5(2):279–297

[117] Rabiner L, Juang B (1986) An introduction to hidden markov models. ieee assp magazine 3(1):4–16

[118] Raina R, Madhavan A, Ng AY (2009) Large-scale deep unsupervised learning using graphics processors. In: Proceedings of the 26th annual international conference on machine learning, ACM, pp 873–880

[119] Reed R (1993) Pruning algorithms-a survey. IEEE transactions on Neural Networks 4(5):740–747

[120] Reed S, De Freitas N (2015) Neural programmer-interpreters. arXiv preprint arXiv:151106279

[121] Rifai S, Mesnil G, Vincent P, Muller X, Bengio Y, Dauphin Y, Glorot X (2011) Higher order contractive auto-encoder. Machine Learning and Knowledge Discovery in Databases pp 645–660

[122] Rifai S, Vincent P, Muller X, Glorot X, Bengio Y (2011) Contractive auto-encoders: Explicit invariance during feature extraction. In: Proceedings of the 28th international conference on machine learning (ICML-11), pp 833–840

[123] Roux NL, Fitzgibbon AW (2010) A fast natural Newton method. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10), pp 623–630

[124] Roweis S, Ghahramani Z (1999) A unifying review of linear gaussian models. Neural computation 11(2):305–345

[125] Rumelhart DE, Hinton GE, Williams RJ (1988) Learning representations by back-propagating errors. Cognitive modeling 5(3):1

[126] Sak H, Senior AW, Beaufays F (2014) Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In: INTERSPEECH, pp 338–342

[127] Schmidhuber J (2015) Deep learning in neural networks: An overview. Neural Networks 61:85–117, DOI 10.1016/j.neunet.2014.09.003, published online 2014; based on TR arXiv:1404.7828 [cs.NE]

[128] Schwenker F, Kestler HA, Palm G (2001) Three learning phases for radial-basis-function networks. Neural networks 14(4):439–458

[129] Setiono R (1994) A penalty function approach for prunning feedforward neural networks. Neural computation 9(1):185–204

[130] Shang L, Lu Z, Li H (2015) Neural responding machine for short-text conversation. arXiv preprint arXiv:150302364

[131] Shu C, Zhang H (2017) Neural programming by example. arXiv preprint arXiv:170304990

[132] Smolensky P (1986) Information processing in dynamical systems: Foundations of harmony theory. Tech. rep., COLORADO UNIV AT BOULDER DEPT OF COMPUTER SCIENCE

[133] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 15(1):1929–1958

[134] Sundermeyer M, Schlüter R, Ney H (2012) Lstm neural networks for language modeling. In: Interspeech, pp 194–197

[135] Sutskever I, Tieleman T (2010) On the convergence properties of contrastive divergence. In: AISTATS, vol 9, pp 789–795

[136] Sutskever I, Martens J, Hinton GE (2011) Generating text with recurrent neural networks. In: Proceedings of the 28th International Conference on Machine Learning (ICML-11), pp 1017–1024

[137] Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. Tech. Rep. arXiv:1409.3215 [cs.CL], Google, nIPS'2014

[138] Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. In: Advances in neural information processing systems, pp 3104–3112

[139] Tang Z, Wang D, Pan Y, Zhang Z (2015) Knowledge transfer pre-training. arXiv preprint arXiv:150602256

[140] Tang Z, Shi Y, Wang D, Feng Y, Zhang S (2016) Memory visualization for gated recurrent neural networks in speech recognition. arXiv preprint arXiv:160908789

[141] Tibshirani R (1996) Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society Series B (Methodological) pp 267–288

[142] Tieleman T (2008) Training restricted boltzmann machines using approximations to the likelihood gradient. In: Proceedings of the 25th international conference on Machine learning, ACM, pp 1064–1071

[143] Venugopalan S, Xu H, Donahue J, Rohrbach M, Mooney R, Saenko K (2014) Translating videos to natural language using deep recurrent neural networks. arXiv preprint arXiv:14124729

[144] Vincent P (2011) A connection between score matching and denoising autoencoders. Neural computation 23(7):1661–1674

[145] Vincent P, Larochelle H, Bengio Y, Manzagol PA (2008) Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on Machine learning, ACM, pp 1096–1103

[146] Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol PA (2010) Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of Machine Learning Research 11(Dec):3371–3408

[147] Vinyals O, Toshev A, Bengio S, Erhan D (2015) Show and tell: A neural image caption generator. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp 3156–3164

[148] Wang D, Liu C, Tang Z, Zhang Z, Zhao M (2015) Recurrent neural network training with dark knowledge transfer. arXiv preprint arXiv:150504630

[149] Wang Q, Luo T, Wang D, Xing C (2016) Chinese song iambics generation with neural attention-based model. arXiv preprint arXiv:160406274

[150] Werbos PJ (1988) Generalization of backpropagation with application to a recurrent gas market model. Neural networks 1(4):339–356

[151] Weston J, Chopra S, Bordes A (2014) Memory networks. arXiv preprint arXiv:14103916

[152] Williams RJ, Zipser D (1995) Gradient-based learning algorithms for recurrent networks and their computational complexity. Backpropagation: Theory, architectures, and applications 1:433–486

[153] YANN L (1987) Modèles connexionnistes de l鈥檃pprentissage. PhD thesis, These de Doctorat, Universite Paris 6

[154] Yin S, Liu C, Zhang Z, Lin Y, Wang D, Tejedor J, Zheng TF, Li Y (2015) Noisy training for deep neural networks in speech recognition. In: EURASIP Journal on Audio, Speech, and Music Processing

[155] Yu D, Seide F, Li G, Deng L (2012) Exploiting sparseness in deep neural networks for large vocabulary speech recognition. In: Proc. ICASSP2012

[156] Zaremba W (2015) An empirical exploration of recurrent network architectures

[157] Zeiler MD (2012) AdaDelta: An adaptive learning rate method. In: arXiv preprint

[158] Zhang J, Feng Y, Wang D, Abel A, Wang Y, Zhang S, Zhang A (2017) Flexible and creative chinese poetry generation using neural memory. In: ACL 2017

[159] Zhang W, Li R, Zeng T, Sun Q, Kumar S, Ye J, Ji S (2015) Deep model based transfer and multi-task learning for biological image analysis. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, pp 1475–1484

[160] Zhang X, Lapata M (2014) Chinese poetry generation with recurrent neural networks. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp 670–680

[161] Zhou CL, You W, Ding X (2010) Genetic algorithm and its implementation of automatic generation of Chinese Songci. Journal of Software 21(3):427–437