CSLT

现代机器学习技术导论

2017 年 7 月 14 日

Springer

Contents

Chapter 1
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，这一假设在很多时候并不符合实际，它忽略了现实环境中大量非线性关系的存在。在上节讨论中，我们提到可以用一种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中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，随用性很难保证；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。基于数据的学习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。在参数学习时，我们预先定义一个变换的总体形式（如哪种函数样式），再对该形式中的参数进行学习来确定合理的变换函数；在非参数学习时，并没有一个参数集合可供学习，参数的多少与训练数据相关，典型的如支持向量机（SVM）模型 [26]。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变换函数来学习复杂的非线性变换，使得基于该变换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。事实上随着深度神经网络（DNN）的具大成功，这一观念已经被很多人接受。我们从最简单的神经网络讲起，并在下一章具体讨论DNN背后的深刻内涵。

1.1 神经网络概述

19世纪40年代，随着对人类大脑认知的不断深入，研究人员发现人类大脑与冯·诺依曼计算机体系相比使用完全不同的计算方式，计算机是符号化的，对不同事务定义不同的二进制表达，通过符号间的演绎规则（程序逻辑）进行计算；而人类在记忆和推理时是非符号的，是通过神经元间连接的强弱来存储知识并进行推理。换句话说，我们大脑里的神经细胞基本上是同质的，并没有哪个细胞用来确定表达哪个事物，也没有哪个细胞用来保存某一推理原则，各种记忆和推理功能以分布式方式保存在神经元的连接模式中。基于这一观察，Warren McCulloch和Walter Pitts提出了人工神经网络的概念[80], 基望通过模拟人类神经系统来实现类人的计算能力。

基于不同的研究兴趣，对人工神经网络的研究可分为两个方向，一个方向是研究如何描述真实人类大脑运作方式，如激励方式、抑制机理、传导模型等，这些研究对解开人类大脑的运行机理和智能的产生过程具有重要意义。另一方向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依照第二条思路，关注神经网络对数据的建模能力和推理功能。

1.1.1 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[48]：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：

- 知识通过学习从环境中获得。
- 知识被存储在神元之间的连接权重中。

不论如何定义，神经网络最重要的性质应包含如下三点：

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

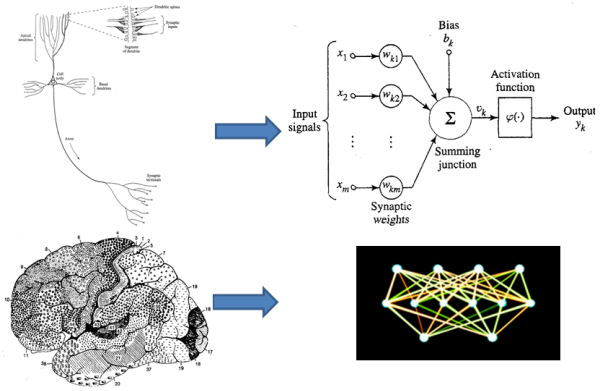


Fig. 1.1 基于仿生学的神经网络示意图

- 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，激发模式等方面都是相同的，致少在其基础方式上是如此；
- 连接性：神经网络中的神经元之间是互联的，通过组成网络的方式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，通过改变神经元连接之间的权重，使之适应环境经验，即可实现网络学习。

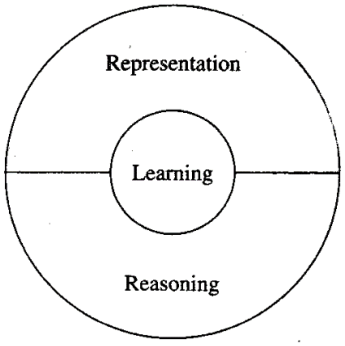


Fig. 1.2 神经网络的功能。通过学习，神经网络可实现对已经知识的记忆和对新知识的推理。

基于其强大的学习能力，人工神经网络可以近似人脑的多种功能，包括记忆功能和推理功能，如归纳（抽象）和演绎（预测）等。这些功能的实现

方式是不同的，比如预测功能一般用前馈网络实现，而记忆功能多用递归网络实现。我们将神经网络结构分成如下几种：基于映射的神经网络、基于结构的神经网络、基于过程的神经网络以及模拟人类大脑的神经图灵机。后续几节将对这些模型进行陆续介绍。

1.1.2 神经模型与人工智能

我们将基于通用单元的连接模式进行学习的模型统称为神经模型。如第一章中所说，神经模型的提出对机器学习乃至人工智能的发展起到了革命性作用。早期的人工智能多基于符号方法（symbolic AI）[2]，将人工智能问题设计成依某种计算规则的符号演算系统。符号方法是一个纯白箱，对问题进行精确描述，包括每一个概念和概念间的每一种关系和操作。这一方法是人类逻辑思维的形式化，具有清晰的问题描述，严格的形式化推理，除非人为设计，否则极少有不确定性。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：很显然是，对每一个概念都设计一个符号本身存在很大困难，因此该方法缺少通用性；另外，如果有新的经验出现，很难对现有符号系统进行学习改造。为解决这些困难，研究者提出了连接主义方法[3]，用一组简单一致的单元组成网络来描述问题的解决方案。在这种方法里，只关注对问题本身的解决而不必对该解决过程中每个概念进行精确定义，因此避免了概念符号化的困难；另一方面，推理方案被表示为网络结构及其参数本身，而这些参数的确定大量依赖于实际经验知识，因此当新的知识出现时，可以通过对网络参数的灵活调整实现对新知识的吸收。神经网络是最通用的连接主义方法。

神经模型的出现在从传统人工智能到现代人工智能的转变中，起着至关重要的推动作用。首先，神经网络的出现极大扩展了人工智能的处理范围。传统人工智能只能处理离散符号，而神经网络可处理各种离散和连续信号，甚至可以处理非常原始的感知信号。第二，神经网络改变了知识表示方法。传统符号方法将知识表达为概念和概念间的运算规则，而神经网络将知识‘存储’在网络结构中，这一存储既分散又紧凑：说其分散，是因为同一知识被多个神经元表示，说其紧凑，是因为不同知识间大量共享结构和参数。这一分散表示（distributed representation）在现代深度学习技术中具有重要意义。第

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

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

三，神经网络大量依赖经验数据进行学习，这极大改变了传统人工智能重人为知识轻经验学习的状态，最终发展出机器学习这门崭新的学科。

当然，今天的符号方法已经远非当年的规则推理和专家系统了，而是吸收了大量概率方法的自洽的可学习系统。一种自然的想法是将神经网络和符号方法结合起来，利用神经网络的学习能力和符号方法的推理能力和可扩展性，得到更高效的人工智能系统。事实上这一思路正是当前表示学习的主要内容，即基于深度神经网络从原始数据中学习有效表达，再基于符号（或概率）方法进行建模。另一个发展趋势是对符号任务的神经建模，基本思路是将概念表达成低维向量，再基于该向量进行神经计算。这一方法已经在某些传统符号任务上大量使用，如在自然语言处理领域，在很多传统符号任务中，包括语言模型、词性标注、句法分析、机器翻译、语言理解、多步推理等，神经模型都取得了极大成功。

还有一个有意思的方法是将符号方法和神经模型方法在应用领域上结合。神经模型方法通过共享参数得到一个紧凑模型来表示概念和规则，这一方法不可避免带来过度平滑的问题。这一平滑的好处是仅学习最普适的信息，去掉数据中无关紧要的变动性，因而可提高对未知数据的鲁棒性，这正是我们在上文提到的神经模型最重要的优势。然而，在很多任务中，个性化小概率事件是有价值的，有时甚至要比普适规律更重要。利如在机器翻译中，大量专有名词出现的概率很低，但这些词包含重要信息，对翻译出的句子质量至关重要。但基于神经模型，这些小概率词基本被当作噪声处理，代之的是一些在语法或类别上有些类似的但意义完全不同的词。符号方法基于精确的知识，对小概率事件更加重视（即使是基于概率模型也是如此），因此在处理这些低频词时具有明显优势。我们最近的实验表明，如果将神经模型和符号模型结合起来，让神经模型处理常用词，让符号模型处理低频词和外来语，可有效提高翻译系统的性能。

1.2 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，输出为基于该特征向量的预测目标，既可以是回归任务中的一个连续的预测值，也可以是分类任务中的后验概率。这一模型是上一章所介绍的线性预测模型的非线性扩展。

1.2.1 从线性模型开始

线性预测模型回顾

我们从上一章所介绍的线性回归模型开始来讨论神经网络。一个简单的线性模型回归模型可表示为:

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

其中$x_i : i = 1,2,...,d$为d维的输入变量，$w_i : i = 1,2,...d$为相应的参数，w_0为一个偏差值，x_0为一个'哑元' 输入，始终取值为1，y为输出变量。图1.3给出该模型的图形化表示。这一表示可以认为是一个非常简单的、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1.4表示。上一章讨论过，如果假设目标变量的观察值t符合以y为中心的高斯分布，则该模型的最大似然估计等价于以最小平方误差的准则的优化问题，即线性拟合。

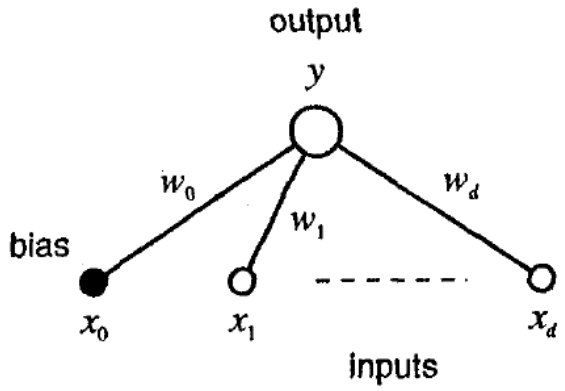


Fig. 1.3 线性模型

相应的，在二分类问题中，一般用Logistic回归，如下所示：

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

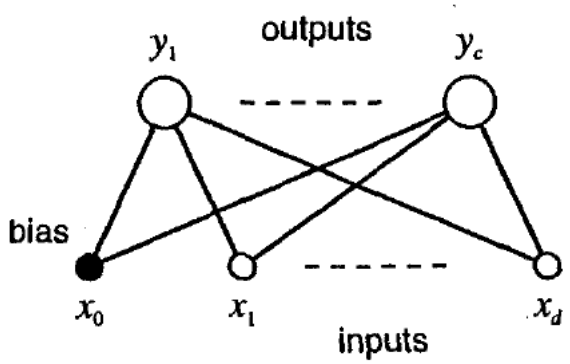


Fig. 1.4 线性回归模型

与线性回归相比，该模型和线性回归线性在形式上的区别仅在于对线性输出结果增加一个非线性Logistic变换，因此也称为对数线性模型。前一章讨论过，如果假设观察值（分类标记）是以y为参数的伯努利分布，则基于该模型的最大似然估计等价于一个以最大交叉熵为准则的优化问题。

另一种非线性变换是阶跃函数，形式化如下：

$$g(a) = \begin{cases} -1, \text{when } a < 0 \\ +1, \text{when } a \geq 0 \end{cases} \tag{1.1}$$

相应的预测模型为:

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

其中$y \in \{-1, +1\}$为对二分类问题的预测结果，-1和$+1$分别代表二分类问题中的两个类。这一模型即是神经网络发展早期著名的感知器模型。

模型优化方法

前面讨论过，对于上述线性模型或近似线性模型，可以采用两种方法进行优化：一种是闭式（closed form）解法，即基于模型的简单性，直接求出参数w的最优解；另一种是梯度下降法（Gradient Descend, GD），求目标函数（最小平方误差或最大交叉熵）对参数的梯度，再选择合适的步长将参数向

梯度方向做少量改动。这一过程往复进行，当步长选择合理时，可得到局部最优解。更复杂的优化方法将在本书后续章节介绍。

对感知器模型，由于阶跃函数是不连续的，因此无法直接利用梯度下降法求解。一种解决方法是不考虑阶跃函数，仅考虑线性预测$\mathbf{w}^T\mathbf{x}$部分，使其与目标变量t 符号相同，则得到如下目标函数：

$$E(\mathbf{w}) = -\sum_{n\in\mathscr{M}} \mathbf{w}^T\mathbf{x}_n t_n$$

其中x_n为第n个训练样本，$t_n \in \{-1,1\}$为相应目标变量，$\mathscr{M}$为识别错误（即y_n与t_n的符号相反）的样本集合。注意上述目标函数也是不连续的，因为$\mathscr{M}$会随着模型训练过程不断发生改变。仅管如此，我们在$\mathbf{w}$处于某一值时依然可以求出其梯度，基于梯度下降法进行一步迭代。通过简单计算可得到：

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \alpha\nabla E(\mathbf{w}) = \mathbf{w}^t + \alpha\mathbf{x}_n t_n$$

其中$\mathbf{w}^t$表示在第t次迭代后得到的参数值。很多早期研究[16]中都证明：利用上述迭代过程，在任何线性可分的数据集上都可确保在有限步中取得最优解，这一结论称为感知器收敛定理（Perceptron Conergence Theorem）。

然而，现实生活中绝大部分问题是线性不可分的。典型的例子是异或运算问题$y = x_1 xor x_2$，其中$x_i \in \{0,1\}$ 为输入变量，$y \in \{0,1\}$代表类别标签，如图1.5 所示。可以看到，无论怎样设计，都无法找到一条线性分类器将黑色的点和白色的点完全分开。换句话说，线性模型无法模拟异或运算。

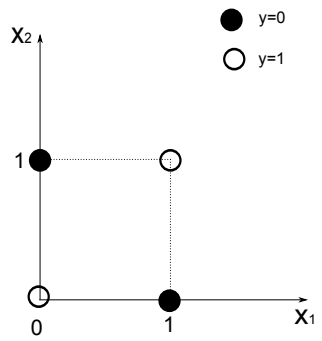


Fig. 1.5 异或运算表示图。x_1和x_2是两个输入变量，$y = x_1 xor X_2$。

非线性扩展

线性不可分问题的失败是导致早期神经模型走向低谷的原因之一。用今天的眼光看来，这一问题似乎并不像当初看起来那么严重。首先，很多问题虽然线性不可分，但总体上是满足线性关系的，引入随机变量以后很大一部分问题是可以解决的。对感知器的批评恐怕在很大程度上有符号主义的影子。另外，也不可能指望一个模型可以适用于所有数据分布情况，用线性模型解决线性问题是理所当然的事。不管如何，为回应在线性不可分问题上的挑战，研究者对传统线性模型进行了简单扩展，首先将输入变量进行非线性变换，然后在变换空间建立线性模型，即：

$$y(\mathbf{x}) = \mathbf{w}^T \phi(\mathbf{x})$$

其中$\phi(\mathbf{x})$是对$\mathbf{x}$的变换函数。依据变换函数的不同形式，可得到完全不同的模型。几种典型的非线性变换和其相应的模型形式如下：

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

注意，上一章在介绍线性模型时，我们同样提到过可对$\mathbf{x}$进行变换，且这一变换不会影响模型的线性属性。然而，上一章所提到的变换是固定的，不需要学习的，本章所提到的变换是通过学习得到的。因而从整个模型角度来讲，如果我们引入的变换是非线性的，则整个模型是非线性的。我们下面将着重讨论前两种变换方法，将核函数放在后续章节讨论。

1.2.2 多层感知器

1.2.2.1 模型结构

MLP基本结构

多层感知器（MLP）是对线性回归模型和线性分类模型的扩展。从结构上看，是将线性模型的一层网络扩展到多层，每一层的输入为前一层的输出

经过一个非线性变换得到的激发值，该输入经过一个线性变换传到下一层，由此得到一个信息逐渐向前传导的前向网络。图1.6给出一个包含一个隐藏层的两层MLP的结构（这里我们以变换次数来计算层数）。

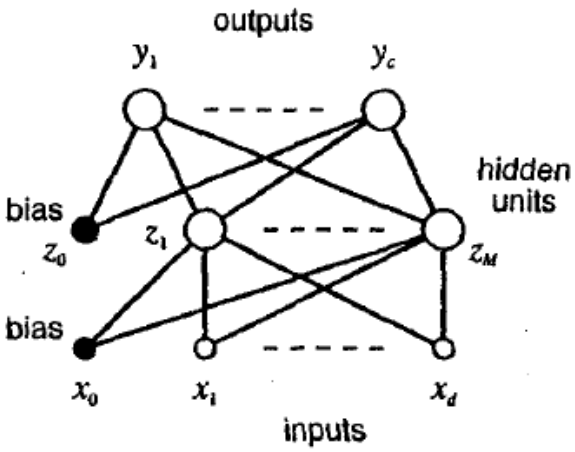


Fig. 1.6 多层感知器图

我们将该模型的计算过程形式化如下：

$$\begin{aligned} a_j &= \sum_{i=0}^{d} w_{ji}^{(1)} x_i \\ z_j &= g(a_j) \\ a_k &= \sum_{j=0}^{M} w_{kj}^{(2)} z_j \\ y_k &= \widetilde{g}(a_k) \end{aligned} \tag{1.2}$$

其中$g(\cdot)$和$\widetilde{g}$为第一层和第二层的激发函数，z_j为隐藏层的激发值。注意在回归任务中，一般取$\widetilde{g}(x) = x$，在分类任务中一般取$\widetilde{g}(x) = \sigma(x)$。

通过上述分层前向网络，相当于输入$\mathbf{x}$进行逐层变换，直到倒数第二层，得到变换$\boldsymbol{\phi}(\mathbf{x})$，再由最后一层线性模型完成回归或分类任务。简单计算可得到变换$\boldsymbol{\phi}(\mathbf{x})$如下：

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

输出层的激发值为:

$$y_k = \widetilde{g}\left(\sum_{j=0}^{M} w_{kj}^{(2)} g\left(\sum_{i=0}^{d} w_{ji}^{(1)} x_i\right)\right)$$

结构扩展

上述MLP的基础结构可做相应扩展。首先，在保证信息前向传递的前提下，可允许跨层连接，只需保证不存在循环边接即可，如图1.7所示。

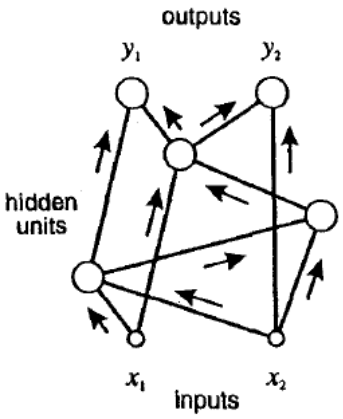


Fig. 1.7 存在跨层连接的多层感知器结构。

一般来说，如果激活函数选择适当（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 [57, 47, 16, 27]。

1.2.2.2 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过程类似，定义好一个目标函数后，即可通过各种优化方法对参数进行训练。不同于线性模型，由于层数的增多以及非线性激活函数等原因，多层感知器获取全局最优解比较困难，因此一般采用数值解法。最通用的是随机梯度下降法（Stochastic Gradient Descend）。

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在训练时每一步迭代都会随机选择一部分数据，基于该数据集进行参数调整。这一随机选择的数据集通常称为一个mini-batch。利用mini-batch，SGD可部分消除因参数初始化带来的偏差，且因为每个mini-batch都对参数进行更新，训练速度更快。需要提醒的是，不论GD还是SGD，都只能达到局部最优。这一训练过程如图 1.8所示，其中$\mathbf{w}_A$和$\mathbf{w}_B$是两个局部最优点，网络参数初始化在$\mathbf{w}_C$处，经过若干次SGD迭代过程，逐渐收敛到$\mathbf{w}_B$。

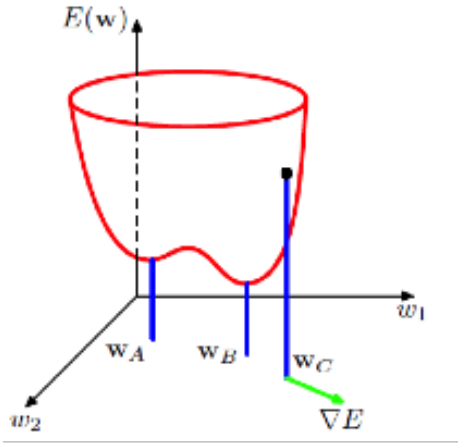


Fig. 1.8 梯度下降图

因为MLP具有层次结构，可利用偏导数的链式法则对参数进行顺序更新。即先对最后一层参数求梯度，因为最后一层只需考虑自身各个结点的误差，这一层的参数不依赖前面几层，因此梯度计算直接简单。基于最后一层的梯度，倒数第二层梯度可较容易求出。依次类推，直到求得第一层参数的梯度。这一梯度由后向前依次计算的方法可理解为预测误差的反向传递，因而称为后向传递算法(Back-propagation, BP)。图 1.9 给出了一个两层MLP的BP过程。图中，δ_k是在输出层的第k个结点得到的预测误差，基于该误差对$w_{k,j}$求梯度并对其进行更新，同时传导到中间隐藏层的第j个结点。所有输出结点的误差传导到第j个结点后，该误差的总和被继续向输入层的每个结点传递，并更新对应的连接权重w_{ji}。

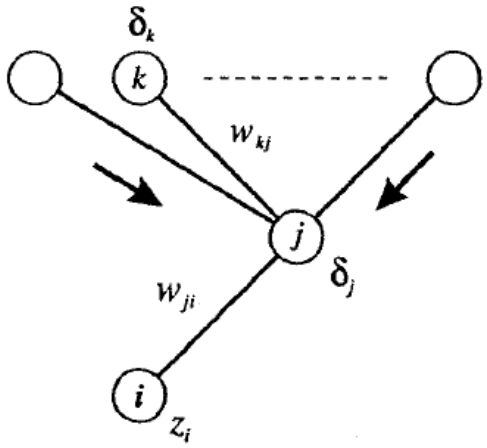


Fig. 1.9 BP算法示意图。

训练技巧

BP算法在原理上是清晰的，但实际应用中并不容易。因为层数增加以后，受到非线性函数的嵌套影响，误差向前传递变得越来越困难，或者逐渐消失，或者发生爆炸。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，模型质量可能会强烈依赖参数初始化。我们总结了一些以往在神经模型训练过程中的经验，希望能够帮助从业人员更容易地训练MLP模型。注意，这些经验在训练其它神经模型时也是适用的。

- **输入特征的归一化及相应变换(Feature normalization and transfer)**。过于分散的输入变量取值将增加训练难度。特别是由于大多数非线性函数会将激发值控制在0附近，过大或过小的输入会进入非线性函数饱和区，导致梯度传导效率下降。所以，将输入特征进行归一化是提高神经网络训练成功几率的重要方法。可选择的归一化方法包括最大-最小值归一（将一个mini-batch里的取值归一到0到1或-1到1之间）、均值-方差归一（对一个mini-batch里的取值减均值除标准差）、高斯化（将一个mini-batch里的取值归一化成高斯分布）。最近提出的Batch Norm方法不仅对输入层进行了归一化，对隐藏层也进行了类似在mini-batch内部的归一化。该方法可有效提高模型的性能。除了正规化外，一些变换方法也对神经网络的训练有重要意义，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音，从而降低神经网络的训练难度（否则噪音将引起参数更新过程的较大波动）。

- **选择合适的激发函数(Appropriate activation function)**。 对于不同的任务，应选取不同的激活函数。可选的激发函数有Sigmoid，Tanh，Relu等。实验表明，某些激发函数（如Sigmoid）容易陷入饱和区，导致训练困难; 有些激发函数（如Relu）具有无界性，容易导致训练发散。不同任务、不同的数据分布情况，必须要认真选择激发函数。
- **连接的初始化(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。如在很多自然语言处理任务中，使用标准正态分布初始化权重往往得不到最优解，通常建议的权重初始化方法为使用均匀分布在：$\left[-\frac{6}{\sqrt{a+b}},+\frac{6}{\sqrt{a+b}}\right]$，其中$a$为本层的结点数，$b$为下一层的节点数。
- **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：$w_t = w_{t-1} - \alpha(\beta\nabla_{t-1}(w) + (1-\beta)\nabla_t(w))$，其中$\beta$是动量参数，$\nabla_t(w)$为在$t$时刻目标函数的梯度。动量方法在一定程度上增强了训练的稳定性，同时也加快了学习速度。
- **二阶导数的信息(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，因此不论对哪个参数，其学习率都是一样的。事实上，牛顿-拉弗森方法（Newton-Raphson method）的学习效率更高。该方法可形式为：$w_t = w_{t-1} - H^{-1}(w)\nabla_{t-1}(w)$，其中$H(w)$为目标函数的Hessian矩阵。由上式可知，基于二阶信息可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二阶梯度。可以看到，对那些曲率较大的参数方向，意味着该方向的目标函数取值变化较大，需要对参数仔细调节，这时Hessian矩阵在该方向上的取值较大，参数的学习率自然调低，使学习不致于过度激进；而对那些曲率较小的参数方向，意味着该方向上目标函数取值变化不大，可放心学习，这时Hessian矩阵取值较小，参数的学习率自然增加，加快学习步伐。虽然二阶方法的优点明显，但计算Hessian矩阵在很多时候是不现实的，因此无法直接采用。一般采用一阶信息来近似二阶的方法，如Hessian Free [79]，AdaGrad [31],AdaDelta [127], Adam [30], Natural SGD [99, 3, 88, 90] 等。前面所述的动量方法事实上也是一种轻量级的二阶近似方法。
- **有组织学习或课程学习（Curriculum learning）** 研究表明[11]，认知学习中将相关问题聚类进行学习往往比混在一起学习更有效果。因此，在神经

网络训练中，可以将不同的学习内容进行预先分组，将比较容易的学习任务先行学习，再对较难的任务进行学习。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，等掌握的差不多了，再继续学习较深入的知识。对神经网络的训练也是如此，让它先学习一些较容易的目标，建立大概的分类面或拟合函数，再引入较困难的样本，学习分类或拟合任务的细节内容。

- **迁移学习（Transfer learning）** 在神经网络的学习过程中，很多时候因为数据量的限制，无法学习得到较高质量的网络。一种解决方法是利用已经学习到的网络，用其中所包含的知识来学习新的任务。这一方法称为转移学习。研究表明[13, 8]，如果我们有一个已经学习成功的模型，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模型的前向层直接拿来用作新任务模型的特征提取层，再基于新任务进行微调。也可以用原来模型对新任务进行指导，如使得新任务生成的目标不得与原模型的预测目标相差太远，或将原模型生成的目标作为新模型的预测目标。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，这一方法更具价值。
- **正规化(Regularization)** 神经网络模型缺少先验知识，完全依赖数据驱动来优化参数，这意味着这种模型很容易陷入过拟合。因此，训练过程中加入约束项可有效提高模型的可扩展性。一种正规化方法是在目标函数中引入对参数或神经元的正规化因子，如l-1 [75]，l-2 [68, 103]等。另一种常用的方法是Eearly stop，即选择一个验证集，当在验证集上的性能开始下降时即停止训练 [91, 92, 21]。除些之外，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范围进行限制，防止过大或过小。带噪训练方法也可认为是一种正规化方法，通过在训练数据中随机加入一些噪声，可以让神经网络学习到更有价值的数据模式和规律 [124]。最近提出的Drop out [**?** 106]方法可认为是对网络参数的加噪训练，即在训练过程中随机将一些连接置零。研究表明，这一方法可有效防止参数间的协同训练问题，使得每个参数发挥更有效的代表性，同时，该方法也被认为是模拟多个网络共同学习的有效方法。各种模型剪裁方法 [28, 95, 125, 76, 75]也可认为是一种引入稀疏性的正规化方法。正规化方法是当前研究的热点之一。

1.2.3 径向基函数网络

MLP用基于函数嵌套或复合（function composition）方法设计非线性变换$\phi(\mathbf{x})$，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）网络基于另一个思路设计这一特征变换。该方法在特征空间设计一系列标识点（Anchor points）$\{\mathbf{v}_j\}$，基于这些标识点，可以将每一个采样点$\mathbf{x}$用该点到$\mathbf{v}_j$的之间的距离表示出来。由此，每个标识点代表了变换空间的一个维度，基于该标识点的距离函数称为一个径向基，即RBF。RBF网络 [19, 102, 77]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。

1.2.3.1 RBF 网络与训练方法

一个包含M个RBF的网络如图1.10所示，其中第二层结点为特征映射层，每个ϕ_j对应一个RBF，定义如下：

$$\phi_j(\mathbf{x}) = \phi_j(\|\mathbf{x} - \mathbf{v}_j\|)$$

其中ϕ_j是任意一个变换函数，该函数以$\mathbf{x}$和$\mathbf{v}_j$之间的距离为变量。注意ϕ_0是个和输入无关的偏移量。输出层计算与MLP 类似：

$$y_k(\mathbf{x}) = \sum_{j=0}^{M} w_{kj}\phi_j(\mathbf{x}) \tag{1.3}$$

RBF中的ϕ_j形式可以是多样的。最常用的可能是如下的高斯形式:

$$\phi_j(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mu_j\|^2}{2\sigma_j^2}\right) \tag{1.4}$$

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。注意上式的高斯形式是距离$||\mathbf{x} - \mu_j||$的单变量高斯分布，而非以$\mathbf{x}$为变量的多变量高斯分布。高斯RBF网络如图1.11所示。Hartman等人证明[46] 如式 1.4所示的高斯RBF网络可以近似一切连续函数。Park和Sandberg等人的推广了该结论，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 [87]。

RBF网络的核心思想是用特征空间里的一些具有代表性的点（标识点）作为新的坐标来实现特征变换，因此这些标识点的选择显得至关重要。选择

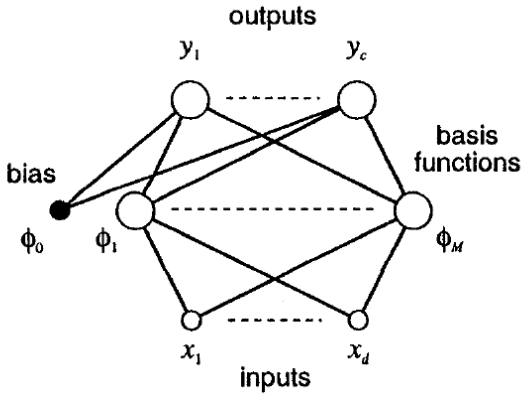


Fig. 1.10 包含M个RBF的径向基函数网络。

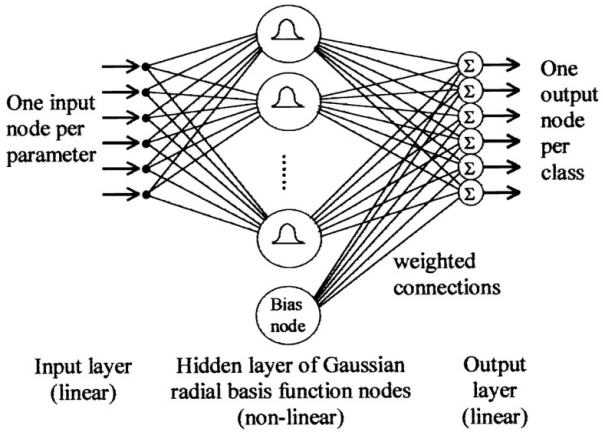


Fig. 1.11 基于高斯核函数的的RBF网络。

这些标识点时，可采用无监督学习的方法，如K均值（K-means）聚类，高斯混合模型等得到中心点和相应参数（如高斯RBF里的宽度参数）。这样得到的标识点具有较强的代表性，一般比较适合RBF网络。确定好RBF之后，相当于线性模型里的变换函数$\phi(\cdot)$已经确定，因此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。这样学习得到的RBF对当前任务的代表性更性，但可能失去一些局部表征属性，引起过拟合。

1.2.3.2 RBF网络的意义

我们可以从几个方面理解RBF网络的意义，包括插值分析、回归和分类任务等。本节对这些思路做简要介绍，以理解RBF的表征性。

从插值分析到RBF

在插值分析中，已知一组数据$\{(\mathbf{x}_n,t_n)_{n=1}^N\}$，对一个新的输入$\mathbf{x}'$，利用已有的数据得到合适的输出$t'$。设核函数为$\phi(\cdot)$，对任何一个输入$\mathbf{x}$，其相应的输出$t$由如下映射函数得到：

$$f(\mathbf{x})=\sum_{n=1}^{N} w_n\phi(\|\mathbf{x}-\mathbf{x}_n\|) \tag{1.5}$$

对于训练集中的每个数据$(\mathbf{x}_n,t_n)$，都可列出如上式形式，则一共可得N个等式。由于上式中的参数$\{w_n\}$一共N个，因而可以解出这些参数，使得该映射函数对训练集上所有的点都恰好满足。对于不在训练集上的点，映射函数1.5提供了一种基于训练集中所有数据的插值方法。

式1.5所示的插值方法与式1.3所示的RBF网络非常相似，$\phi(\cdot)$相当于RBF网络中的RBF函数。不同的是插值公式1.5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RBF的中心可以灵活取值，且RBF的数目不定，则得到RBF网络，即：

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

注意当上式中的M与样本数不等时，上述插值RBF网络对训练集中的点未必能精确预测，这时一般以最小平方误差为准则进行拟合，对应一个回归任务的RBF。

从核回归理解RBF

给定训练数据$\{x_n,t_n\}_{n=1}^N$，假设数据(x,t)服从如图1.12的联合分布，即，

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

其中$f(x-x_n,t-t_n)$为以某一个训练数据样本(x_n,t_n)为中心的概率分布。如果$f(\cdot,\cdot)$取高斯分布，则上式类似一个先验概率为$\frac{1}{N}$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：

$$y(x) = E[t|x] = \int t\,p(t|x)dt \tag{1.6}$$

$$= \frac{\sum_n \int t f(x-x_n,t-t_n)dt}{\sum_m \int f(x-x_m,t-t_m)dt} \tag{1.7}$$

令$g(x) = \int f(x,t)dt$，通过变量代换，可得:

$$y(x) = \frac{\sum_n g(x-x_n)t_n}{\sum_m g(x-x_m)} = \sum_n k(x,x_n)t_n$$

其中我们已经定义了一个核函数$k(x,x_n)$:

$$k(x,x_n) = \frac{g(x-x_n)}{\sum_m g(x-x_m)}$$

如果我们将$k(x,x_n)$视为RBF网络中的径向基函数$\phi(||x-x_n||)$，将t_n视为对应的RBF的权重，则我们得到类似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。

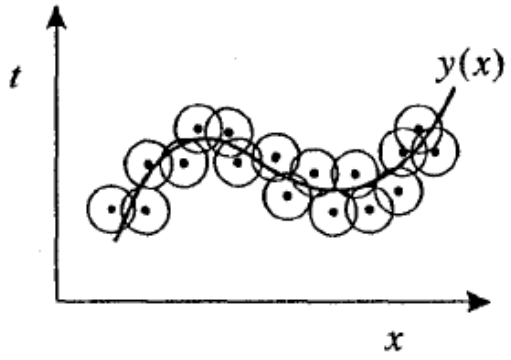


Fig. 1.12 核回归方法中对数据分布的假设。

从分类任务理解RBF

我们也可以从分类任务来理解RBF，会得到更有趣的结果。一个典型的分类任务如图1.13所示，其中数据包括三个类，每个类C_k可用一个概率密度$P(x|C_k)$来表示。分类任务的目标是给定一个新的样本，求出该样本属于各个类的后验概率，如下式所述：

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

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 1.14.

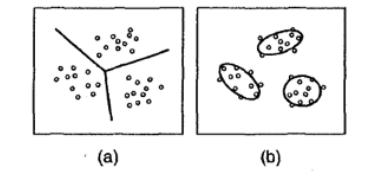


Fig. 1.13 基于贝叶斯方法的分类任务。(a)在特征空间的数据分布和分类面；（b）每一类对应的概率密度分布。

1.2.3.3 多层感知器与径向基函数网络的比较

上文描述了两个基于特征提取的神经网络：多层感知器MLP和RBF网络，两个神经网络均可以描述任意连续函数，但具有不同的特点。首先，对MLP模型中的每个隐藏结点，其‘等激发线’是一个平面，这意味着输入的变化会在隐藏层较大范围内产生影响；而RBF模型中的每个隐藏结点的‘等激发线’是一个球面，因而输入的变化只会在局部空间产生影响。这一特性如

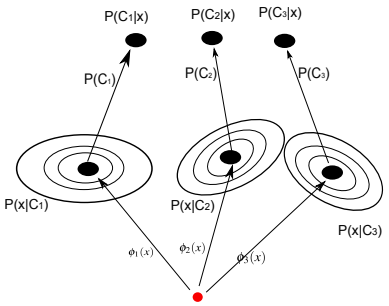


Fig. 1.14 基于贝叶斯公式的后验概率计算可认为是一个RBF网络，第一层求对每个类的RBF函数$\phi_k(x)$，第二层以每一类的先验概率作权重连接第k个隐藏结点和第k个输出结点。

图 1.15所示。由此可见，MLP的信息传递可认为是分散的，RBF的信息传递是局部的。由此产生的一个结果可能是MLP在不同输入空间的参数共享性更强，可扩展性更强，而RBF的共享性比较弱，需要有代表不同输入空间的数据来对模型进行训练。因此，MLP需要的隐藏结点数比RBF 网络的隐藏结点数要少的多。

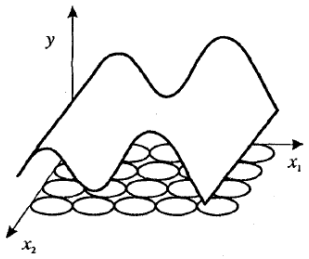


Fig. 1.15 MLP与RBF的等激发线。图中x_1和x_2是两维输入，y是隐藏结点的输出。

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是局部的，参数共享较小，对每一部分数据训练时对其它数据的影响较小，因此训练起来比参数高度共享的MLP要容易。

1.2.4 被约束的神经网络模型

标准MLP模型是一种全连接结构，具有强大的学习能力，但在实际应用中如果不考虑数据分布属性而盲目学习，则很容易进入过训练。为解决这一问题，我们在建模时应该尽量考虑数据本身的结构性质，基于这些先验知识设计带有结构的网络，往往后得到更好的结果。

实际数据（如图像数据、文本数据、语音数据等）往往具有鲜明的结构化特性，例如：很多数据往往处于一个子空间中，数据往往具有聚类性，数据各维度间往往具有相关性，数据的取值往往是稀疏的（仅可取有限的几个值），等等。在实际研究中，我们常遇到和可利用的结构化性质包括：

- 空间结构：例如在图象数据中，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：在语音信号或文本数据中，我们看到的是一个时间序列，相近时间的信号或文本的相关性很强，而相同模式可能出现在一个序列的不同时刻。
- 频域结构：在语音或图象数据中，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。

基于这些先验知识，研究人员对全连接的MLP模型进行约束，即引入一定的先验结构，通过限制连接和共享参数的方法，不仅减小了网络参数规模，而且减小了计算量，更重要的是引入先验知识使得模型结构和训练方法更有针对性。典型的结构化神经网络模型如由Lecun等[71]提出的卷积神经网络模型、由Bishop等[15] 提出的混合密度网络以及由Pearl等[89, 36]提出的贝叶斯网络等。

1.2.4.1 卷积神经网络模型(Convolutional neural network)

一种结构化约束方法是利用时域、频域或空间上的模式重复特性，使不同位置的特征提取共享参数，这样得到的网络结构称为卷积神经网络（Convolutional neural network, CNN）。图 1.16给出一个简单CNN网络，其中包括一个卷积层和一个降采样层。卷积层利用一个卷积核与原始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。由于每个卷积核的参

数通常远少于全连接网络的参数，因此模型复杂度一般低很多。降采样层不仅可以对得到的卷积特征进行降维，还可以去除因输入信号的轻微变形引起的特征抖动，提高系统的鲁棒性。

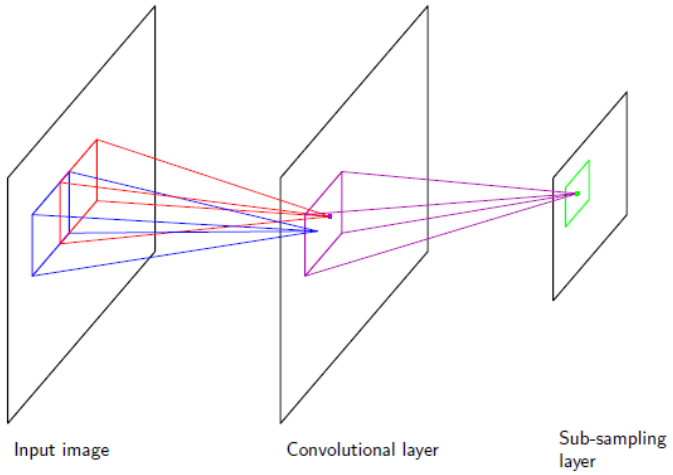


Fig. 1.16 CNN网络中的卷积和降采样。卷积层用多个滤波器提取特征，降采样层用来补偿因轻微变形产生的相邻特征抖动。

图1.17给出一个应用在图像识别任务中的CNN结构。该结构包括两层卷积，每层卷积后接一个降采样层。卷积神经网络的思路来源于1984年日本学者Fukushima[35]提出的神经认知机（Neocognitron）。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。该模型试图将视觉系统模型化，使其能够在即使物体有位移或轻微变形时也能完成识别。

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局部重复模式，因而可起到特征提取作用；（2）降采样层对空间、时间、频域上的轻微形变有抵消作用；（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，避免重复学习。

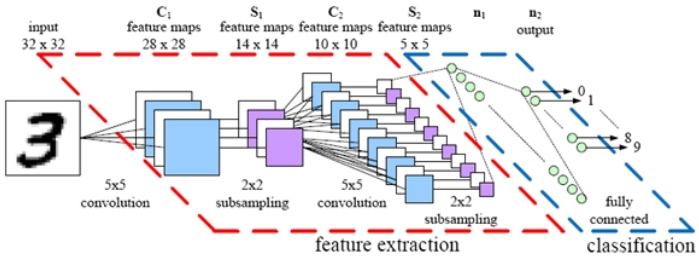


Fig. 1.17 用于图象识别的卷积神经网络结构。该结构包括两层卷积层，每个卷积层后接一个降采样层。

1.2.4.2 混合密度网络(Mixture-Density network)

将先验知识与神经网络相结合的另一种方式是对数据分布建立概率模型，用神经网络预测模型的参数。这种方式可以将概率模型的鲁棒性（因为有人为分布假设）和神经网络强大的学习能力结合，由概率模型来防止过拟合，由神经网络来增强参数估计能力。混合密度网络即是这样一种混合模型。例如，我们可以假设数据分布是一个高斯混合模型（GMM），

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

而模型参数，包括每个高斯的均值和方差$\{\mu_k(x)\}$及不同高斯的权重$\{\pi_k(x)\}$由神经网络学习得到。注意，在这一模型中，GMM的参数是输入x的函数，这与传统GMM不同。这一不同带来深刻影响，意味着对(x,t)的概率描述更为精确。

1.2.4.3 贝叶斯网络(Bayesian network)

上一章讲过贝叶斯模型，通过对模型参数设定先验概率来对其进行约束。为简便起见，一般选择高斯分布作为先验概率，即：

$$p(\mathbf{w}|\alpha^{-1}I)$$

其中$\mathbf{w}$为神经网络的参数向量。以回归任务为例，对一个训练数据集$D = \{\mathbf{x}_n, t_n\}_{n=1}^N$, 其条件概率为：

$$p(D|\mathbf{w},\beta^{-1}) = \Pi_{n=1}^{N} N(t|y(\mathbf{x}_n,\mathbf{w}))$$

则对$\mathbf{w}$的后验概率：

$$p(\mathbf{w}|D) \propto p(D|\mathbf{w},\beta^{-1})p(\mathbf{w}|\alpha^{-1}I)$$

由于$p(D|\mathbf{w})$比较复杂，上式很难得到解析解，一般需要采用近似解来估计$p(\mathbf{w}|D)$。一种估计方法是拉普拉斯近似，即用一个以$p(\mathbf{w}|D)$的最大值为中心的高斯分布来近似$p(\mathbf{w}|D)$。

1.3 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，当系统给定一组样本对$\{(x,t)\}$时，该模型学习一个对x的映射$y = f(x)$，使得y与目标变量t的距离最小。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，在这些问题中我们仅有数据x而没有明确的数据标记t，我们希望有模型可以描述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显然是不合适的。研究者提出各种基于记忆的神经网络来处理这一问题。在这种神经网络中，网络结构用来记住某些训练样本。当网络训练完成后，对于一个测试样本，可以从网络中提取出与之最近似的训练样本。这一记忆结构具有重要意义。一方面，对一个含有噪声的测试样本，通过近似样本提取，事实上起到了去噪效果；另一方面，如果训练样本足够多，这一网络则学到了训练样本中最有价值的模式，这意味着在对测试样本提取过程中，将只关注显著模式，而忽略非显著模式，因而是一种有效的特征提取方法。这一性质非常重要，是下章要讨论的深度学习方法的基础。

本节将介绍几种不同的神经模型，包括：Kohonen网络、霍普菲尔德网络（Hopfield Net），双向联想记忆网络（Bidirectional associative memory），玻尔兹曼机（Boltzmann machine），自动编码机（Auto-encoder）等。

1.3.1 Kohonen网络

Kohonen网络又称自组织图（Self Organization Map, SOM），由Kohonen在1982年提出 [67]。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，这些结点一般会放置在一个规整的平面上，如图 1.18所示。每个结点s_i对应一个d维向量v_i，其中d为高维空间（数据空间）的维度。在训练过程中，对一个d维输入向量x_n，可以计算该向量与所有结点间的距离$\{d(v_i,x_n)\}$，基于该距离寻找到最相近的结点s_j进行激发，这一结点s_j称为最佳匹配结点（Best Matching Unit, BMU）。找到BMU之后，对BMU对应的向量进行更新，使之与v_j 更加接近。例如，可采用如下形式：

$$v_j^t = v_j^{t-1} + \alpha(t)x_n \quad s_j = BMU(x_n)$$

其中$\alpha(t)$是t时刻的学习率（该学习率一般随时间t增加而衰减）。上述更新对所有$\{x_n\}$循环进行，即可使Kohonen网络的结点代表高维数据。这一过程如图 1.19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 1.19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代表训练数据，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任何相似性。这显然不能满足我们在低维空间保持数据分布结构的要求（可以参考在k-mean算法中，各个中心矢量是各自独立的，并没有任何近邻关系）。为解决这一问题，我们可以使低维空间中相邻的点被同一数据激发，如图1.18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，网络结点向量的更新公式如下：

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

其中$N(s_j)$表示s_j的相邻结点集合，$\theta(k,j)$为相邻结点相关的强度。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的呈现。注意图 1.19和图 1.20的训练的过程中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。

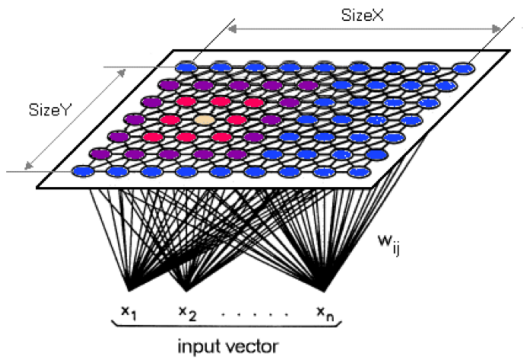


Fig. 1.18 Kohonen网络图

Fig. 1.19 Kohonen网络训练过程。给定一个训练数据，如图中白色点所示，训练过程将对应的BMU及其相近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。

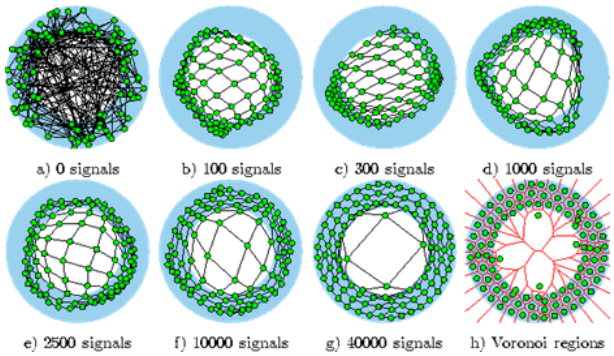


Fig. 1.20 Kohonen网络训练过程示例。随着迭代次数增加，网络结点越来越代表数据分布。

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。这种局部映射属性说明该映射天然是一种非线性映射，可描述各种复

杂的分布情况。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 1.21给出在一个典型非高斯数据上Kohonen网络和PCA的对比，显然Kohonen网络在这种数据分布上更能描述数据分布特性。

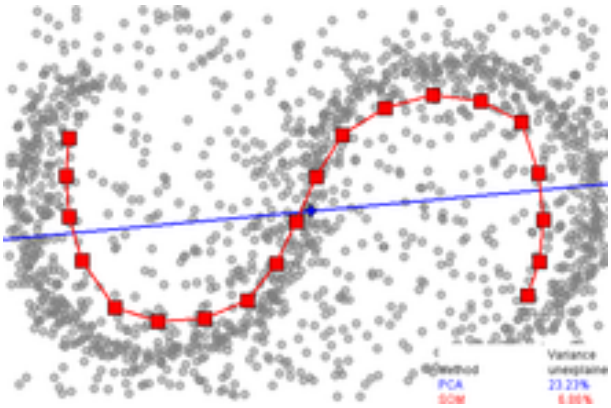


Fig. 1.21 Kohonen网络与PCA在非高斯数据上的对比。红色线条代表一维Kohonen网络，每个结点的位置由其对应的向量决定。PCA在图中表示为一条蓝线。

Kohonen网络是非常简单的一种记忆网络，该网络通过一些记忆结点（由网络结点的对应向量表示）对训练数据进行表达，在模式提取时将数据映射为最相似的结点。考虑到网络结点通常会记忆最有代表性的数据，这一方法可以认为是一种朴素的典型数据记忆。

1.3.2 霍普菲尔德网络

Kohonen网络，抛去其对数据表示的同构性（低维空间保持高维空间的相近属性），其表达能力与矢量量化（Vector Quantitization, VQ）类似。一个主要原因在于不同结点都有自己的向量，结点之间缺少参数共享，没有形成结构化协同表达，因此记忆能力很低。

Hopfield等人在1982年受到人类记忆模式的启发提出一种记忆网络[56]，如图1.22所示。该网络包含若干二值的神经元结点（取+1或−1）$\{s_j\}$，每一对结点(s_i, s_j)间通过无向边w_{ij}相连。所有结点组成一个向量，该向量的某一取值称为一种“模式”。Hopfield网络的目的是通过这一互联结构，记住训练过程中教给它的某些模式，当记忆完成后，在“回忆”阶段可以将前期记住的模式提取出来。

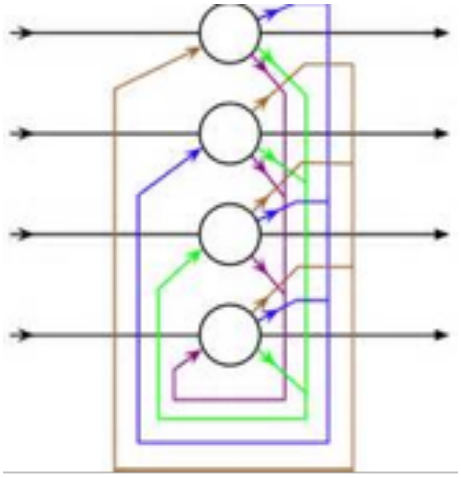


Fig. 1.22 霍普菲尔德网络

1.3.2.1 模型学习

Hopfield网络的学习过程即是对训练模式进行记忆的过程。将μ^n表示第n个模式，μ_i^n表示该模式的第i个元素（结点）取值。一种常见的学习方法采用Hebbian准则 [49]，该准则可简单表述为‘同时激发的单元互相连接’ [78]。基于这一准则，给定N个训练模式$\{\mu^n\}_{n=1}^N$，可计算结点s_i和s_j之间的连接权重为：

$$w_{ij} = \frac{1}{N}\sum_{n=1}^{N}\mu_i^n\mu_j^n$$

注意，上式中如果μ_i^n和μ_j^n符号相同，则基于该模式对$w_{i,j}$的贡献为正值，意味着同时激发的神经元间的连接加强，恰好符合Hebbian准则。上述学习也可以采用增量模式，即：

$$w_{ij}^n = w_{ij}^{n-1} - \frac{1}{n}(w_{ij}^{n-1} - \mu_i^n\mu_j^n)$$

其中$w_{i,j}^n$为学习第n个模式后神经元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$D = \{\mu^n\}_{n=1}^N$, 优化如下似然函数：

$$L(W) = p(D;W) = \Pi_n p(\mu^n;W)$$

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

$$p(\mu^n; W) \propto e^{\Sigma_{ij} w_{i,j} \mu_i^n \mu_j^n + \Sigma_i \theta_i \mu_i^n}$$

其中$\{\theta_i\}$为人为指定的模型参数（不需训练）。基于上述假设的最大似然优化即可导出Hebbian学习准则。对应上述概率形式，一个模式μ的能量函数为：

$$E = -\sum_{i,j} w_{ij} \mu_i \mu_j + \sum_i \theta_i \mu_i \tag{1.8}$$

因此，Hebbian学习准则的目的是使训练数据的概率最大化，或相对应的，能量最小化。图 1.23给出一个能量函数示意图，其中横轴表示不同函数的不同状态，纵轴表示状态对应的能量。经过学习以后，也就是记忆完成后，训练模式都处于能量局部最低状态。注意并不是所有能量局部最低点都对应一个训练模式，由于神经网络连接的复杂性，某些能量局部最低点并非期望记忆的模式，这些模式可称为‘赝模式’ [50]。

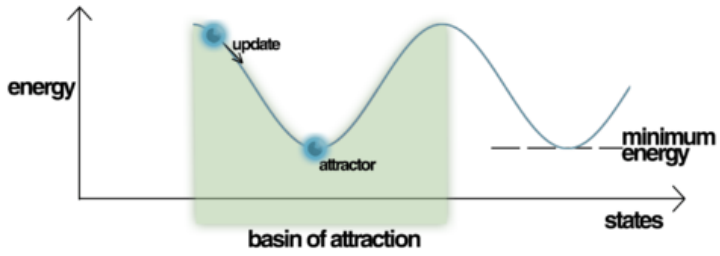


Fig. 1.23 Hopfield网络的能量函数图

1.3.2.2 模式提取

在模式提取时，固定模型的权重，给定一个带噪音的初始模式（如部分记忆的模式）作为图 1.22的输入，得到一个与该初值最近的模式。这一过程如图 1.23所示：初始的输入状态在μ^I，通过迭代法寻找邻近的能量局部最低点，最终得到被记忆的模式μ^0。为实现这一迭代搜寻，可以对公式 1.8求由于某一神经元s_i的取值变化引起的能量变化：

$$\Delta E_i = E(\mu_i = +1) - E(\mu_i = -1) = -2\sum_j w_{ij}\mu_j + 2\theta_i$$

上式意味着如果满足如下条件，则s_i取+1会使能量更低：

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

如果不满足上式条件，则应对s_i取-1。这一结论总结如下：

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

上面我们推导了对某一神经结点的更新方式。对整个网络所有结点的更新可采用两种方式：在同步更新中，对所有结点统一更新结点值；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。上述更新过程迭代进行，直到不再有结点值发生变动。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小能量点。这一最小能量点通常是模型需要记的模式，但也可能是一个赝模式。

1.3.2.3 Hopfield网络的记忆功能

基于Hopfield网络的模式提取方式的一个显著特点是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 [86, 56]）。这一记忆方式可用于基于内容的寻址（Content-based Addressing）。基于该机制，给定部分模式或带噪音的模式，即可通过Hopfield网络得到和这些模式最相近的目标模式。例如，我们通过学习，让Hopfield网络记住了十个数字图片，在提取时输入某个变形带噪音的数字，Hopfield网络会自动抽取到和这个数字最相近的数字，这事实上提供了一种有效的正规化方式。当然，Hopfield网络的应用远不止此，而这种记忆功能是所有基于记忆的神经网络的共同特征。

Hopfield网络的记忆能力是有限的，并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 [50]。Liou 等人证明，这一比例可能会提高到0.14以上 [74]。

1.3.3 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]，如图1.24所示，其中白圈表示可见结点$\{v_i\}$，灰圈表示不可见结点$\{h_j\}$。不论是可见结点还是非可见结点都是随机变量，且每个结点的概率分布依赖于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有向图模型，亦称为马尔可夫随机场（Markov Random Field, MRF）。

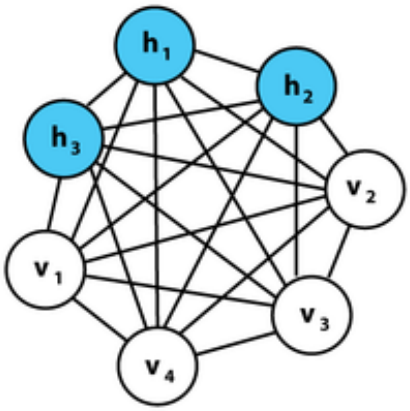


Fig. 1.24 玻尔兹曼机(Boltzmann Machine）.

1.3.3.1 运行与采样

在前面介绍Hopfield网络时，我们提到对模式进行提取时，通过迭代更新每个神经元结点来达到局部最低能量点。玻尔兹曼机的运行方式类似，也是通过迭代达到通量最低。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，而是达到一种稳定的状态分布。

和Hopfield网络类似，我们定义一个模式μ的能量如下：

$$E(\mu) = -\sum_{i,j} w_{ij}\mu_i\mu_j - \sum_i \theta_i\mu_i \tag{1.9}$$

和Hopfiled网络稍有不同的是，玻尔兹曼机习惯上取$\mu_i \in \{0,1\}$。假设基于该模型的状态（即模式）概率分布为玻尔兹曼分布，即：

$$p(\mu) \propto e^{-\frac{E(\mu)}{T}} \tag{1.10}$$

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：

$$\Delta E_i = E(\mu_i = 0) - E(\mu_i = 1) = \sum_j w_{ij}\mu_j + \theta_i \tag{1.11}$$

由基于玻尔兹曼分布公式，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:

$$p_{i=1} = \frac{1}{1 + e^{-\frac{\Delta E_i}{T}}} \tag{1.12}$$

上式表明在运行一个玻尔兹曼机时，应基于式 1.12对s_i随机取值，而式中ΔE_i应基于式 1.11计算。注意在计算ΔE_i时，依赖与结点i相关的所有相邻结点s_j和s_i上的偏置量θ_i。这一计算是一个迭代过程。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。

上述运行过程亦可看作一个吉布斯采样过程，在该过程中，每次采样只对一个结点s_i 依条件概率$P(\mu_i|\mu_{-i})$进行随机取值，其中μ_{-i}表示去除结点s_i外模式μ的取值。在后续章节我们可以看到上述采样过程将收敛到一个稳态分布。因此，玻尔兹曼机是一个生成模型，可用来生成适合该模型的稳态分布样本，这是与Hopfield 网络的一个典型区别。

1.3.3.2 训练方法

玻尔兹曼机的参数包括每个神经模型的偏置量$\{\theta_i\}$和神经模型间的连接权重$\{w_{ij}\}$，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 1.10和式 1.9。在实际应用中，我们希望一个玻尔兹曼机能代表可见的训练样本的分布规律，记为$P^+(v)$。注意到一个玻尔兹曼机中包括隐藏结点和观察结点，此模型中的数据分布规律记为$P^-(v)$，可以肯定的是其中包含隐藏结

点的作用成分，但它仍为由观察结点联合概率表示的样本分布，这是由于我们进行了隐藏变量边缘化：

$$P^{-}(v) = \sum_{h} P(\mu)$$

我们的任务是调整玻尔兹曼机的各个参数，使其所代表的分布$P^{-}(v)$和实际数据分布$P^{+}(v)$尽可能相似。用Kullback－Leibler (KL) 散度来描述这两个分布间的距离，得到学习目标函数如下：

$$L(W,\Theta) = \sum_{v} P^{+}(v) ln \frac{P^{+}(v)}{P^{-}(v)} \tag{1.13}$$

其中(W,Θ)表示玻尔兹曼机的参数集合。对上式进行最小化，即可得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：

$$w_{ij} = w_{ij} - \alpha \frac{\partial L}{\partial w_{ij}} \tag{1.14}$$

$$\theta_i = \theta_i - \alpha \frac{\partial L}{\partial \theta_i} \tag{1.15}$$

其中α为学习率。

通过简单计算可得如下梯度公式 [1]：

$$\frac{\partial L}{\partial w_{ij}} = -(p_{ij}^{+} - p_{ij}^{-}) \tag{1.16}$$

$$\frac{\partial L}{\partial \theta_i} = -(p_i^{+} - p_i^{-}) \tag{1.17}$$

其中$p_{i,j}^{+}$表示实际数据集中神经元s_i和s_j同时激发的概率，$p_{i,j}^{-}$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。

公式 1.16和公式 1.17说明玻尔兹曼机的训练非常简单，对某一参数的训练只需考虑与该参数相关的神经元结点的概率是否与实际训练数据的概率情况一致即可。从另一个角度看，这一训练事实上与Hebbian 准则是一致的：当两个神经元同时激发时，p_{ij}^{+}较大，如果p_{ij}^{-}较小，说明模型对这一同步激发性描述不够，则公式 1.16为负值，则意味着依式 1.14对w_{ij}进行更新时，会使w_{ij}增加。这正是Hebbian准则中描述的‘同时激发的神经元连接增强’中心思想。

理论上，玻尔兹曼机的训练非常简单（公式 1.14， 1.15），但在实际操作中，$p^{-}(v)$ 的计算非常麻烦，需要玻尔兹曼机运行到稳态分布才可计算。达到稳态分布不仅要耗费大量时间，而且是否达到稳态本身就很难判断。当模型的神经元较多时，这种朴素的训练方法几乎是不可能的。因此，玻尔兹曼机并没有在实际中应用很多。仅管如此，这一模型所带来的理论价值是巨大的：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对人类神经系统的学习方式进行了某些方面的模拟；二是这种模型连接了机器学习中贝叶斯方法和神经模型方法两大学派，一方面证明了在神经学习可利用概率方法，另一方面表明概率模型可以允许更通用的结构；三是这一模型连接了物理学中的某些简单的动态系统（如铁磁化过程）与机器学习中的概率模型 [56]，表明概率方法与物理学的某些深刻联系。

1.3.4 受限玻尔兹曼机

通用的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。一种限制结构是将观察变量$\{v_i\}$和隐藏变量$\{h_j\}$分为两组，只有不同组的两个神经结点可以连接。这一结构称为限制性玻尔兹曼机（Restricted Boltzmann Machine, RBM），最初由Paul Smolensky in 1986年提出，当时称为Harmonium。一个RBM的结构如图 1.25所示，其中红色结点表示可见结点，蓝色结点表示隐藏结点，这两组结点间有无向边连接。

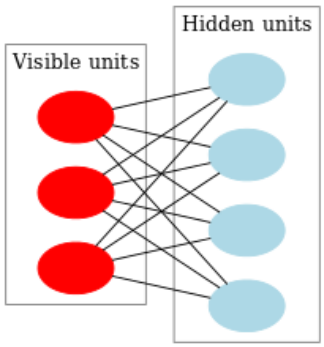


Fig. 1.25 限制性玻尔兹曼机(Restricted Boltzmann Machine, RBM）.

RBM有广泛应用。首先，RBM可以认为是一种结构学习方法，其隐变量分布用以表达数据的内部结构。这一性质可用来在文本分析中学习主题模型 [53]。其次，如果隐变量比较少，可以认为是一种数据降维方法 [52]。第三，因为RBM可以学习数据中的主要特征，去掉无关噪音，因此可用作特征提取 [25]。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也可用于非监督学习，如分类任务 [70]。

1.3.4.1 RBM的运行

区分隐藏结点和可见结点，RBM的能量函数可定义如下：

$$E(v,h) = -a^T v - b^T h - v^T W h$$

由这一能量函数定义的分布函数为：

$$P(v,h) = \frac{1}{Z} e^{-E(v,h)} \tag{1.18}$$

其中$Z = \sum_{v,h} e^{-E(v,h)}$是规一化因子，或Partition function，由模型参数决定。注意，上式中我们已经用$E(v,h)$中的参数把玻尔兹曼分布中的常数项吸收。RBM的受限结构带来一个重要简化，即给定隐藏结点，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，即：

$$P(v|h) = \prod_i P(v_i|h) \tag{1.19}$$

$$P(h|v) = \prod_j P(h_j|v) \tag{1.20}$$

代入玻尔兹曼分布形式，可得：

$$P(v_i = 1|h) = \sigma(a_i + \sum_j w_{ij} h_j) \tag{1.21}$$

$$P(h_j = 1|v) = \sigma(b_j + \sum_i w_{ij} v_i) \tag{1.22}$$

RBM中的条件独立性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）：给定一个初始状态，基于当前观察量变量v，依公式 1.20和公式 1.22采样出h；基于采样得到的h，依公式 1.19和公式 1.21采样出v。

1.3.4.2 RBM的训练

在训练过程中，我们希望模型对训练数据相对其它数据的概率更高（能量更低）。由式 1.18 可知：

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

对一个训练集$D = v^n$，得到目标函数为：

$$L(W,a,b) = \frac{1}{N}\sum_{n} logP(v^n) = \frac{1}{N}\sum_{n}\{log[\sum_{h} e^{-E(v^n,h)}] - log[Z]\}$$

用梯度下降法可对上式中的模型参数优化。以参数w_{ij}为例，注意到$-E(v,h)$中只有一项和w_{ij}相关，可表示为：

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

代入目标函数：

$$L(w_{ij}) = \frac{1}{N}\sum_{n}\{log[\sum_{h} e^{-v_i^n h_j w_{ij} + const(h)}] - log[Z]\}$$

注意上式中如果对某个训练样本，如果$v_i^n h_j = 0$，则$-v_i^n h_j w_{ij}$对梯度无贡献，因而有:

$$\sum_{h} e^{-v_i^n h_j w_{ij}} = e^{-v_i^n w_{ij}} P(h_j = 1|v^n) + const$$

由此得目标函数对w_{ij}梯度为：

$$\frac{\partial logP(v)}{\partial w_{ij}} = \frac{1}{N}\sum_{n}(v_i^n)P(h_j|v^n) - \frac{\partial log[Z]}{\partial w_{ij}} \tag{1.23}$$

$$= < v_i h_j >_{data} - \frac{\partial log[Z]}{\partial w_{ij}} \tag{1.24}$$

其中$< v_i h_j >_{data}$表示基于实际训练数据$v_i h_j$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

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

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：

$$\frac{\partial logP(v)}{\partial w_{ij}} =< v_i h_j >_{data} - < v_i h_j >_{model} \tag{1.25}$$

基于相似的过程，可对偏置a和b进行更新：

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

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

如果我们回顾下通用玻尔兹曼机的训练公式 1.16，就会发现上述RBM的训练公式 1.25 和式 1.16 十分相似，都是‘局部训练‘，即某一参数w_{ij}的更新仅包括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。

1.3.4.3 对比散度训练

上述训练方法看起来虽然简单，但在实际应用中依然需要大量计算，因为$< v_i h_j >_{model}$的计算需要基于模型的稳态分布，这通常要对RBM运行很久才能得到较好的估计。Hinton在2002年提出了一种基于对比散度（Contrastive Divergence，CD）的算法 [51]，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：

- 从训练数据中随机一个样本v
- 以v为输入变量，基于公式 1.20采样出隐变量h
- 基于h，利用公式 1.19得到v的重构样本$\hat{v}$
- 基于$\hat{v}^n$，再利用公式 1.20采样出隐变量$\hat{h}$
- 对模型更新如下：

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

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

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

其中ε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。

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函数的梯度 [108] 仅管如此，CD算法在实际应用中表现的非常好，极大提高了模型训练效率。CD方法的一个改进是对模型的采样持续进行（不管模型参数已经发生了变质），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） [113]。CD算法为RBM提供了有效的训练工具，从此以后RBM 获得广泛应用，特别是作为初始动力推动了深度学习技术的发展 [54]。

1.3.4.4 RBM模型变种

近年来，对RBM的研究发展讯速。Larochelle在2008年提出监督学习的RBM [70]。在这种RBM中，可见变量除了包含数据特征外，还包含类别标记。这一模型可用最大似然方法或对比散度进行训练，同时也可以用区分性目标训练，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年卷积RBM（Convolutional RBM, CRBM） [73]。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 [83]，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。

1.3.5 自编码机

自编码机（Auto Encoder，AE）是另一种学习数据内部结构的神经网络模型，该模型包括两个部分：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$r = g(h)$。一个典型的结构如图1.26所示。AE 的学习任务是使得重构的数据和原始输入数据尽可能相近。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：

$$L(\theta,\phi) = ||r-x||^2 = ||g_\theta(f_\phi(x)) - x||^2$$

定义了上述训练目标后，应用传统神经网络训练方法即可对网络参数ϕ,θ进行优化。

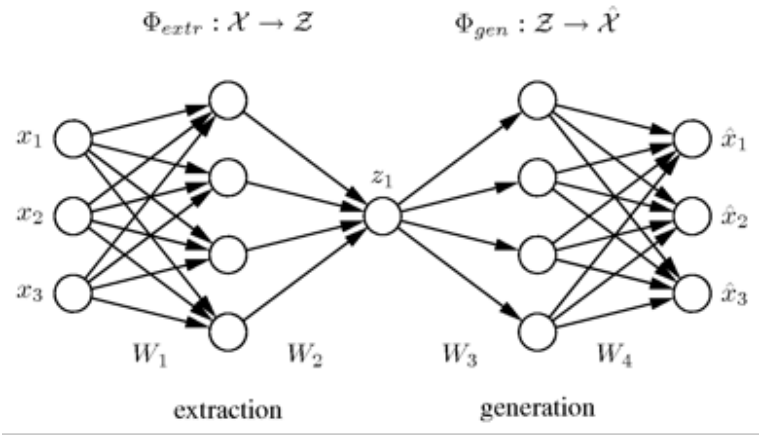


Fig. 1.26 自编码机（AE）网络结构。

AE的概念可能早在1987年就出现了[123, 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 [52]。

1.3.5.1 AE与其它模型的关系

AE和PCA有天然联系 [18, 22]。上一章我们提到过，PCA的训练目标对线性变换得到特征进行重构时的重构误差最小，这意味着PCA可以认为是编码器和解码器是线性的，且共享参数的AE。显然，AE的结构远比PCA灵活，通过增加编码器和解码器的非线性，允许各自独立的参数，允许更灵活的目标函数（基于平方误差的目标函数对应高斯分布假设），AE具有强大的学习能力 [52]。

另一方面，AE与RBM也有紧密关系。由前面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 [9]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） [66]。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，既可利用网络的强大学习能力，也可以对学习过程提供概率约束。

1.3.5.2 其它约束

AE的学习目标是对数据的重构，因此需要一定的约束条件才可避免学习平凡解（如等值映射）。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（如PCA中的主成分）。低维约束有一定局限性，研究者提出了一系列约束方法来提高特征学习能力。一种约束是使得生成的特征具有稀疏性，即稀疏编码（Sparse Coding [84]）。传统的稀疏编码不存在一个参数化的编码器，而是通过优化方法得到特征。一种常见的编码方法是在优化目标函数中加入鼓励稀疏性特征的正则项:

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

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规范和l-1规范。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 [65]。稀疏自编码（sparse AE）将稀疏规范引入AE中，更有效地解决了这一问题。引入稀疏规范后的目标函数如下：

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

加入稀疏约束后，特征空间不再受低维限制，其维度甚至可以高于输入向量维度，因而可学习更复杂的分布结构;另外，稀疏特征将无关信息置零，因而特征向量具有更强的解释性。

另一种稀疏约束方法是对编码器的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：

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

这一模型称为contractive AE [98, 97]。显然，如果特征层的激发函数在$h=0$时的梯度为0，则contractive AE会和sparse AE一样产生稀疏特征。

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少，通过AE学习没有加入噪音

的原始数据。这种加噪训练相当于在原始训练目标函数上加入一个正规化项，增加目标函数对输入变化敏感性降低 [40]。这一模型称为去噪自编码机（Denoising Auto Encoder, DAE) [116, 117].

最近研究表明，DAE具有学习数据分布的重要性质 [115, 2, 12]。这些研究得到的一个中心结论是对一个编码/解码系统，如果噪音和重构余量（Reconstruction Residual）都符合高斯分布，则DAE可以学习数据的分布。具体来说，设噪音符合如下规律：

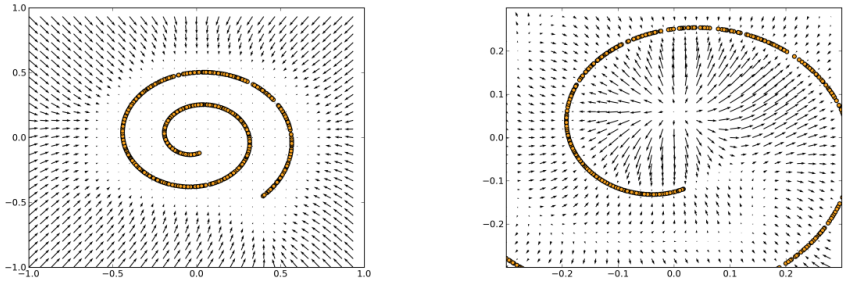
$$C(\tilde{x}|x) = N(\tilde{x}; \mu = x, \Sigma = \sigma^2 I)$$

DAE训练准则为：

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

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

从这一结果我们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，$\frac{g(f(x))-x}{\sigma^2} = 0$，则在该点$x$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$log(Q(x))$在该点的梯度方向是一致的。注意$log(Q(x))$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1.27给出DAE学习到的能量场 []。基于这一点，Beingio对DAE提出了一种概率解释，依此解释DAE可以视作一个生成模型，利用DAE进行反复迭代（即将DAE的输出重新加入噪声作为DAE的下一次输入）生成生成数据x [12]。



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

Fig. 1.27 DAE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$r(x) = g(f(x))$。注意右图中间位置，能量梯度为零，但这些位置并非数据分布位置，而是能量局部高点。

1.4 基于过程的模型

上文我们提到的映射模型和记忆模型都是一种‘静态模型’，即仅考虑数据的分布特性，不同样本一般假设是独立同分布的。在实际生活中，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，后来的样本在分布特性上和以前出现过的样本具有很强相关性。如语音识别任务中的音频信号，自然语言理解任务中的上下文信息，股票交易信号以及脑科学中的脑电波信号等，都天然具有很强的时序相关性。我们将这类问题称为序列问题。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，包括离散状态空间的隐马尔科夫模型（HMM）、连续状态空间的线性动态模型（Linear Dynamic System, LDS），或更通用的动态贝叶斯模型（Dynamic Bayesian Network, DBN）方法等。这些方法都对数据的动态性做出某种概率假设，基于该假设训练模型和进行推理。简单的动态概率模型训练简单，但适用性不强；复杂的动态模型不论训练和推理都较困难，虽然可采用一些近似方法（如变分或采样方法），但这些近似方法有可能在本身就不很精确的模型假设上带来更大偏差。

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，还依赖前面所有输入和输出，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，即不仅是时间序列，也可能是一种逻辑序列。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完成，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。RNN对这些逻辑序列的建模通常表现出强大的能力 [45, 14]。

RNN是个庞大的家族，最简单的是结点间全连接的网络，允许每个结点对所有其它结点进行递归连接。如上文所述的霍普菲尔德网络 [56]，即可认为是这种全连接的RNN：当给定一个输入时，该网络通过递归运行，当前状态作为计算下一状态的依据。这一结构虽然通用，但训练起来很困难。解决训练困难包括两个思路，一是对这些递归连接保持随机初值，不必重新训练，如Echo State Network （ESN） [60]。另一个思路是引入一定的结构化限制。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，只允许同层之间存在递归连接，如Elman网络 [32]，或只允许输出层向隐藏层的递归连接，如Jordan网络 [61, 62]。这两种网络是最常用的RNN结构。我们从Elman网络开始讲起。

1.4.1 Elman递归神经网络

Elman RNN的结构如图 1.28所示。和MLP相比，可以看到隐藏层的输出被回传回隐藏层，作为下一时刻的输入，因此形成一个循环网络。数学表示为：

$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h)$$

$$y_t = \sigma_y(W_y h_t + b_y)$$

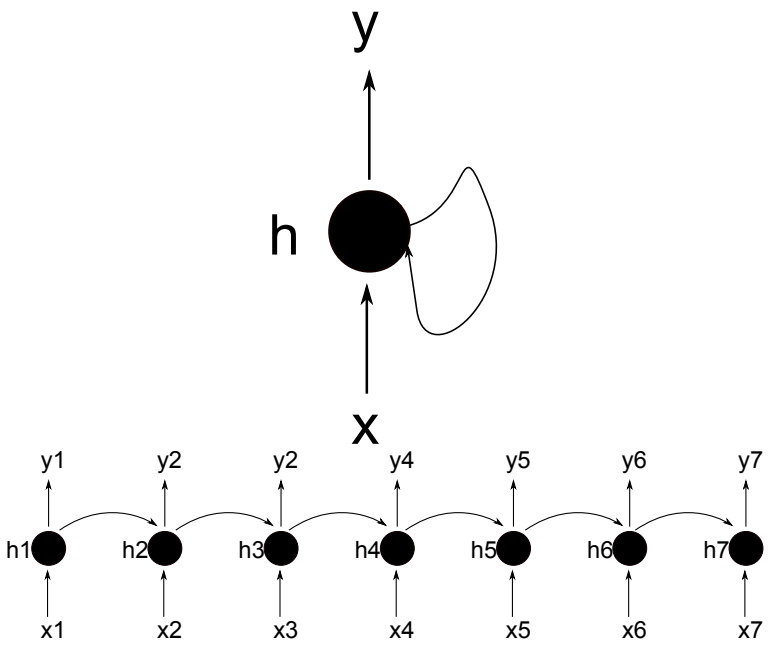


Fig. 1.28 经典RNN网络图构。上图是网络连接结构，下图是将递归连接按时间展开后的等价结构。

Elman RNN的训练可采用传统BP算法，只不过需要做少许改动。如图1.28所示，如果将RNN的递归结构依时间轴无限展开，可以发现它等价于一个无限长的深层网络。基于这一等价网络，我们可以将每一时刻的预测误差信号沿时间轴方向回传，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） [122, 120]。理论上来说，任何一个时刻的预测误差都会回传到所有已经经历过的状态，然而，在实际中，我们一般不会让BPTT 考虑过早的状态。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 [10]，训练越困难，即使不做限制，RNN也很难学到较长时的相关性。这种Truncated BPTT如图 1.29所示。

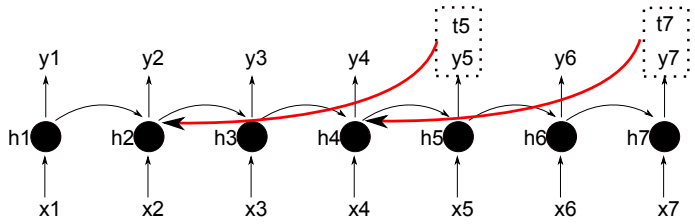


Fig. 1.29 Truncated BPTT。在时刻n，网络输出为y_n，对应的标记为t_n，由此产生的误差往前传播。传播时我们限制传播长度为3。

Elman网络很容易扩展到其它更复杂的模型。图 1.30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，还考虑未来特征，因此通常会带来更好的建模能力。近年来，深层RNN得到广泛应用。与一层RNN相比，深层RNN有更强的特征学习能力。事实上，在抽象特征上学习时序依赖性，往往比在原始特征上学习更有效。这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，受到广泛重视。如百度公司的DeepSpeech2语音识别系统，其声学模型即是一个包含了3 个卷积层和7个递归层的深度RNN网络 [4]。

1.4.2 门网络

前面所述的RNN模型存在一个显著缺陷：它虽然具有一定记忆能力，但因为训练上的困难 [10]，只能学习较短的时序关系。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 [37]。研究者提出了各种

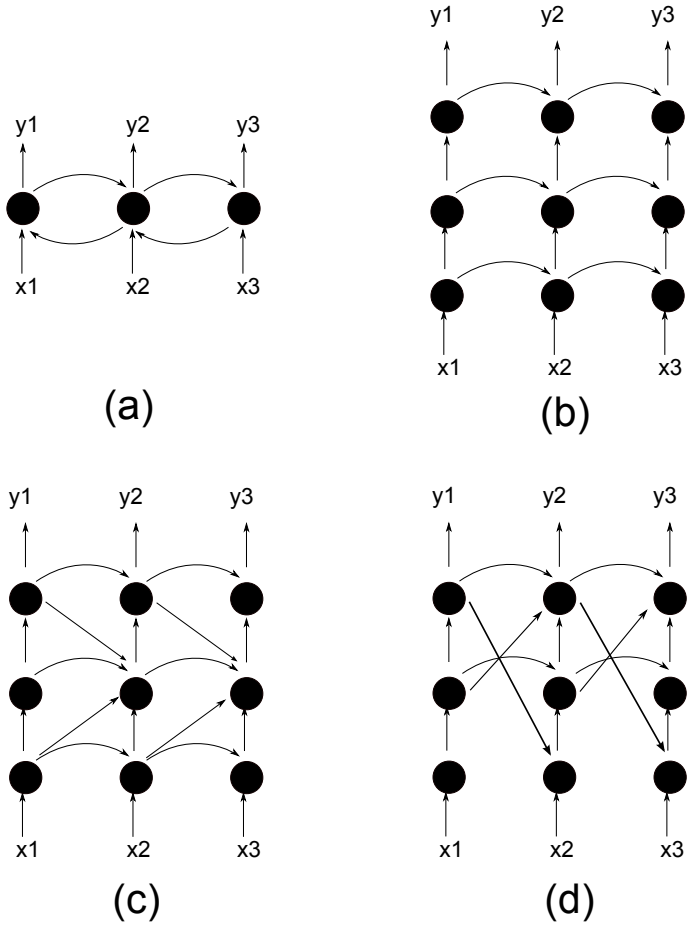


Fig. 1.30 几种RNN扩展结构。(a)双向RNN；(b)深层RNN；(c)深层RNN包含层间递归连接；(d)深层RNN包含跨层递归连接。

解决方案，包括时间延迟网络 [69]，引入时间常数对隐藏层结点的输出进行平滑 [82]，利用不等的时间常数以学习不同尺度的结构 [82]，引入卡尔曼滤波器对隐藏层结点输出进行平滑 [93]。这些方法都起到一定效果，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）网络，长短时记忆单元网络 [55]，如图 1.31 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，因此不再需要时序上的显式的递归连接。这一结构再一次说明递归网络和记忆网络本质上具有非常强的相似性。

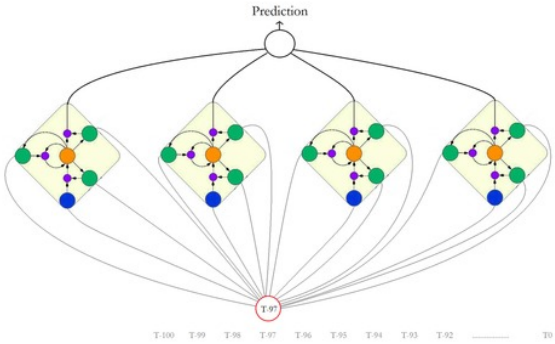


Fig. 1.31 LSTM网络。输入结点顺序输入时间序列，输出结点顺序进行序测。隐藏层的每个结点替换成一个LSTM单元，该单元本身具有记忆功能，因此不需要一个时序上的递归连接。

一个LSTM单元结构如图 1.32所示。这一结构包括三个依赖输入x的门结构，分别是输入门，遗忘门和输出门。这些门结构用来控制信息的记忆、更新和输出。具体来说，输入门控制当前输入是否足够重要到被被记忆；遗忘门控制对当前记忆单元内容的更新，即依当前输入，记忆单元应该让出多少给新的信息；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i_t,f_t,o_t分别代表这三个门，c_t代表该时刻的记忆单元，h_t代表网络输出，则该LSTM单元的动态性可写成如下公式：

$$i_t = \sigma(W_{(i)}x_t + U_{(i)}h_{t-1}) \tag{1.26}$$

$$f_t = \sigma(W_{(f)}x_t + U_{(f)}h_{t-1}) \tag{1.27}$$

$$o_t = \sigma(W_{(o)}x_t + U_{(o)}h_{t-1}) \tag{1.28}$$

$$\tilde{c}_t = \tanh(W_{(c)}x_t + U_{(c)}h_{t-1}) \tag{1.29}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \tag{1.30}$$

$$h_t = o_t \circ \tanh(c_t) \tag{1.31}$$

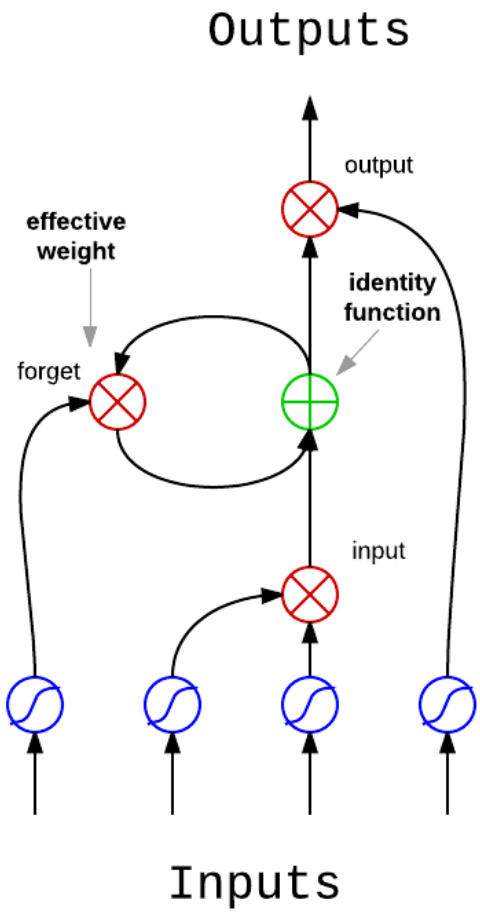


Fig. 1.32 LSTM单元。

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）而改变的动态性，从而可区分重要信息和非重要信息，进行有选择的记忆和遗忘。这种重点记忆方法有利于记忆重要的长时模式。另一方面，在记忆结构中的信息事实上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。从更抽象层次看，LSTM中引入了较复杂的计算结构，特别是门结构输出与信息变量（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

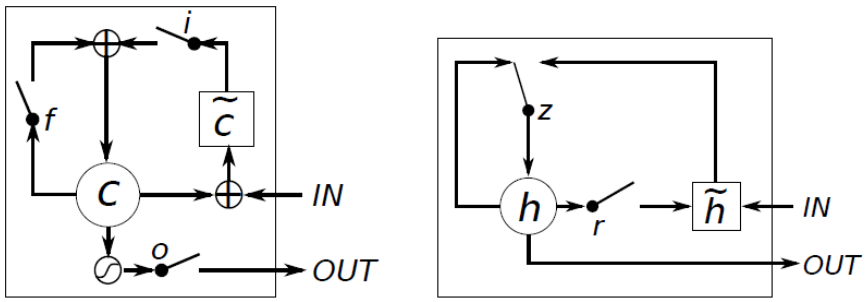
阵乘加操作，因而引入更大自由度来学习较复杂的记忆原则。后面我们会看到，当引入更丰富的操作时，可能会得到更有效的学习模型，这一结构被称为‘神经图灵机（Neural Turing Machine, MTM）。

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） [23]代替LSTM结构。GRU用两个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：

$$z_t = \sigma(W_{(z)}x_t + U_{(z)}h_{t-1}) \tag{1.32}$$

$$r_t = \sigma(W_{(r)}x_t + U_{(r)}h_{t-1}) \tag{1.33}$$

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \tanh(W^{(h)}x_t + U^{(h)}(r_t \circ h_{t-1}) + b^{(h)}) \tag{1.34}$$



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

Fig. 1.33 LSTM和GRU对比。

图 1.33给出LSTM和GRU的结构对比。从图中看，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，因于这一区别，GRU倾向于用更极化的表达来描述信息 [112]。Chung等人对GRU和LSTM做了一些对比研究，发现在这两种门结构具有类似效果。Zaremba等人最近对各种RNN结构进行了一些实证研究 [126]，Karpathy等人则用可视化工具对RNN的学习方式做了探讨 [64]。

RNN/LSTM近年来在序列学习方面取得一系列重要进展，如语言模型 [81, 107]，语音识别 [42, 100]，语音合成 [33]，乐谱合成 [17]，语种检测 [38]，韵律检测 [34]，机器翻译 [110]，社交信号分析 [20]等。

1.4.3 序列对序列网络

RNN提供了强大的序列编码能力。给定一串序列，RNN，特别是基于各种门结构的RNN，可以通过递归方式顺序学习每一步输入特征的信息，并将这些信息保存在记忆单元中，从而将不定长序列压缩成定长的向量。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 [58]。另一方面，RNN也提供了强大的序列生成能力：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生成 [41]和文本生成中 [109]。图 1.34给出RNN用作编码和解码任务的结构。

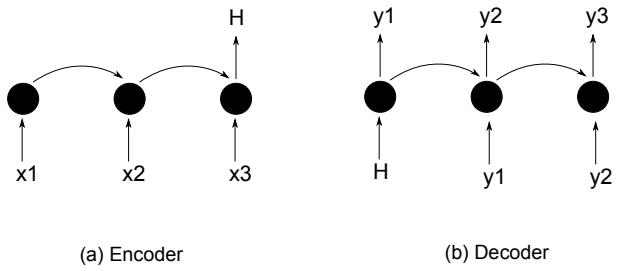


Fig. 1.34 基于RNN的（a）编码器（b）解码器。

一个有意思的想法是能否将这两个模型结合起来，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 [111]。S2S网络可以认为是自编码机（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 1.35所示。

S2S模型在解码时依赖由编码器生成的一个固定维度的全局特征向量。将不定长的输入序列转换成定长特征向量具有重要意义，如可利用现有大量基于定长特征的机器学习模型方法。然而，这种方法也带来很大局限性，特

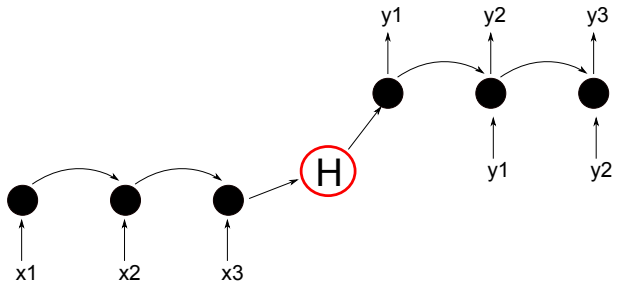


Fig. 1.35 序列对序列网络结构。

别是当RNN的长时特征学习能力还有待提高的前提下，压缩成的全局向量表征性可能有限。一种可能的方法是引入双向RNN，从两个方向进行同时编码，并将每个方向累积的记忆单元作为特征表达。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，通常无法描述局部细节信息。

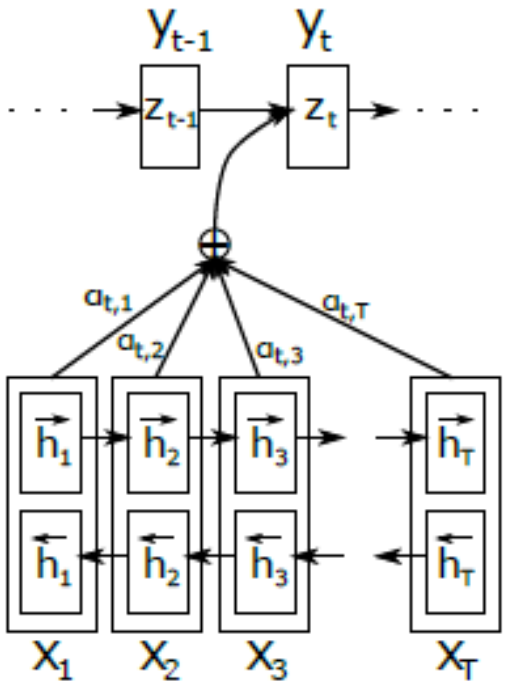


Fig. 1.36 基于关注的序列到序列网络

为解决这一问题，研究者人员提出基于关注的S2S模型（Attention S2S）[6, 104, 24]。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 1.36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对当前生成的权重α_{t_i}。具体计算公式如下：

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{k=1}^{T_x} \exp(e_{tk})}$$

$$e_{ti} = g(z_{t-1}, h_j)$$

其中g为任意一个预测模型（如cosine距离），但一般可基于一个神经网络。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。

Sequence to Sequence 模型，特别是引入Attention机制后，极大增强了对序列的建模能力。特别有意思的是，S2S模型可以学习完全不同领域的随机序列的对应关系。例如，在机器翻译中[6]，S2S模型可以将源语言和目标语言映射到同一个语义空间，基于该语义空间中的向量表达实现多语言翻译。区别于传统概率模型方法，神经模型机器翻译（Neural Machine Translation, NMT）将输入句子映射到语义空间的过程相当于对语言的理解过程，因此可实现‘意译’。另一个例子是图象理解 [24]，将图象和对应的文本映射到同一个语义空间，事实上实现了对图象的语言化理解，基于此还可生成对图片的描述 [118]。类似的方法在智能问答[104]，视频理解 [114] 等领域都取得了巨大成功。

1.4.4 基于*Attention*模型的诗词生成

本节介绍以诗词生成作为一个例子来介绍基于Attention的S2S模型的强大能力 [119, 128]。诗词是中华民族的文化瑰宝，是无数优秀诗人辛勤创作的结果。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，不会吟诗也会吟’。这意味着机器有可能从历史上已有的诗词中学习到规律，从而完成自动创作。

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，挑选可连接在一起形成新句的片段，通过一定规则组合起来，成为一首新诗 [130]。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很难有观赏价值。

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惯（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，使高质量的诗词生成成为可能。

早期基于神经网络的诗词生成基于句子的向量 [129]，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年提出一种基于Attention的S2S模型解决这一问题。在这一模型中，输入是一些关键词，输出是整首诗词。S2S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 1.37 给出了该S2S模型结构，其中编码RNN将用户给予的信息‘春花秋月何时了’编码成一组隐藏向量（图中下部的矩形序列），该向量作为用户意图的编码。在生成过程中，一个单向RNN网络不断循环运行，一个字一个字地生成整首诗。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、平仄等诗词要遵守的规则，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，又保证了生成的句子紧紧围绕用户的意图展开，成为一首合格的诗词。下面我们给出几首生成的例子。

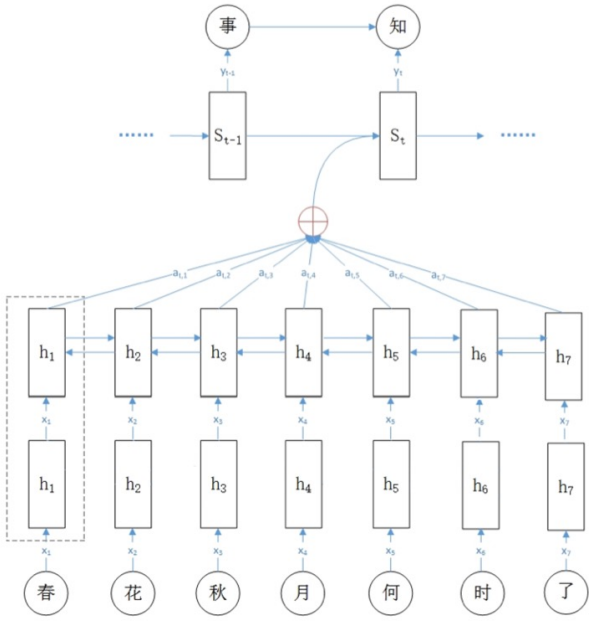


Fig. 1.37 用于古诗生成的Attention-based S2S模型。

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。

1.5 神经图灵机

回顾神经模型的发展，可以发现一个很有意思的现象，即模型的异构化。传统的前向网络仅包含矩阵乘加操作，且不包含记忆单元，发展

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。

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’的黑箱学习方式比基于概率贝叶斯学习需要更多数据，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，定义一系列的元操作，通过这些元操作组合成更复杂的操作，有可能会学得更复杂的动态性。基于这点考虑，Graves等人提出了神经图灵机的概念 [43]。所谓神经图灵机，是基于神经模型对传统图灵机模型进行模拟。一个图灵机应该如下几个部分：一个无限长的纸带，一个读写头，一个控制规则，一个状态寄存器。Graves利用神经网络来模拟这些单元，将一些内存单元开僻出来代表纸带，定义一系列寻址和读写操作模拟读写头，用神经网络模型和内部记忆单元分别作为控制规则和寄存器。经过这些定义的神经网络即可视为一个虚拟的图灵机，即神经图灵机。Grave提出的神经图灵机模型如图 1.38所示。

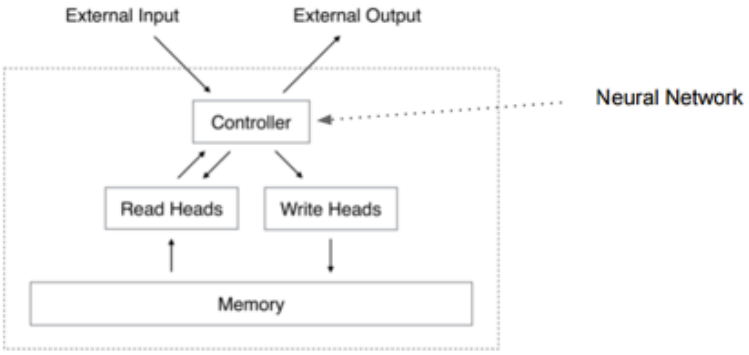


Fig. 1.38 神经图灵机

NMT并不是图灵机的简单复制。首先，NMT可以定义任何寻址方式，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，读写方式也可以灵活定义。如在Graves等人的实现中，可利用当前寄存器内容应该关注的内存单元，再依该关注权重读写相应内存，这与RNN的Attention机制非常相似。在写内存时，同样基于当前寄存器内容计算对每个内存单元的写入权重，再对相应单元依该权重进行删除或写入。除了基容的寻址机制，也可以通过当前状态直接计算出下一个读写位置（如基于移位操作等）。最近，Graves等人扩展了NMT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 [44]。图 1.39给出了DNC的基础结构。

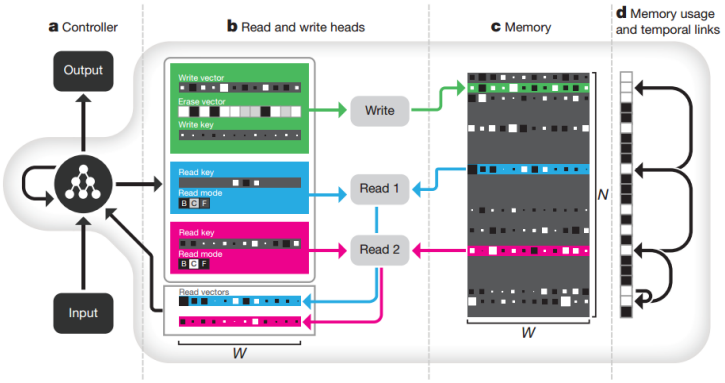


Fig. 1.39 可微分神经计算机结构 [44]。

NMT是一个非常自由强大的计算框架，给机器学习研究者提供了非常广阔的相象空间，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。特别是对内存的读写方式是由数据学习出来的，这意味着寻址和数据的优化是协同的。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 [121]。另一方面，NMT引入的可学习控制逻辑极大丰富了机器学习的内容。传统图灵机需要人为定义好的逻辑控制过程（即计算机程序），机器依此逻辑进行运行。机器学习中，虽然模型是可学习的，模型参数是可调节的，但学习方法本身是确定的，是人为定好的。NMT打存了这种固有观念，实现了真正端对端可学习的神经机器。在NMT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具体编程，只需为NMT提供足够的数据，设定好学习目标，机器可能会自动学习到如何组织动作流程以更有效地完成这一目标。

Neural Program Interpreter （NPI）即在学习控制逻辑方面做出有益探索 [96]。通过给定若干例子，NPI可以学会如加减、排序、交换等简单操作。同时，NPI还提供了另一种领域自适应的方法：保持数据模型一致，但对学习到的操作程序进行自适应。Shu等人最近发现，NPI可以学习字串的各种操作过程 [105]。

1.6 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：用来学习输出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，映射模型可能是结构和概念最简单的，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）既可以算作记忆模型，也可以算作动态模型。这一概念的意义在LSTM，特别在NMT中变得清晰：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。NMT以图灵机模型为概念蓝本，将记忆（外部内存）、状态（寄存器）和控制逻辑（网络参数）清晰分开，事实上将各种神经模型统一到了一个整体框架中。

神经模型是机器学习中的重要分枝。和贝叶斯模型相比，神经模型通过同质的单元和灵活的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。这种学习能力和大量数据结合起来，可以强大到模拟人类的学习过程。今天人工智能领域的飞速发展，和神经模型是分不开的，但更重要的是这种模型和大数据的结合。同时，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 [94]和TPU [63]，和更有效的并行训练方法 [29]，这些方法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。简单地看，DNN只是更深层次的神经网络，但当我们深入考虑这一方法后，会惊讶地发现这一方法给我们思想带来的变革要深刻的多。我们将在下一章讨论深度学习是如何对神经模型进行了加冕，然而，我们还是要再次强调，这一加冕的背景是大数据的积累和计算资源的强大；在这一背景下，神经模型的成功是必然的。

1.7 相关资源

- 本章主要参考资料包括Simon Kaykin的《Neural networks and learning machines》[48]，Christopher Bishop的《Neural networks for pattern recognition》[16]和Ian Goodfellow、Yoshua Bengio的《Deep Learning》 [39]。

- 文中关于神经网络发展的部分参考了Schmidhuber最近的综述论文 [101][4]。Schmidhuber的个人主页还包含关于RNN众多有价值的资料[5]。
- 关于神经网络训练，可以有考的资料包括《Neural networks: tricks of the trade》 [85]，特别是其中LeCun的“Efficient Backprop”一文很值得一读 [72]。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》 [59]。
- 文中关于Kohonen网络和Hopfield网络部分参考了Wikipedia相关页面[6][7]。

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

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

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

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

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

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

[10] 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

[11] 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

[12] 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

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

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

[15] Bishop CM (1994) Mixture density networks

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

[17] 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

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

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

[20] 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

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

[22] 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

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

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

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

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

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

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

[29] 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

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

[31] 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

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

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

[34] 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

[35] 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

[36] Geiger D, Verma T, Pearl J (1990) Identifying independence in bayesian networks. Networks 20(5):507–534

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

[38] 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

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

[40] 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

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

[42] 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

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

[44] 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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[70] 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

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

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

[73] 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

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

[75] 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

[76] 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

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

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

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

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

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

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

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

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

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

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

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

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

[89] Pearl J (1985) How to do with probabilities what people say you can't. University of California, Computer Science Department

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

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

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

[93] 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

[94] 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

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

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

[97] 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

[98] 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

[99] 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

[100] 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

[101] 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]

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

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

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

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

[106] 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

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

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

[109] 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

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

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

[112] 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

[113] 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

[114] 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

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

[116] 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

[117] 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

[118] 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

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

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

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

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

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

[124] 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

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

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

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

[128] 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

[129] 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

[130] 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