

(Weston *et al.*, 2010; Bordes *et al.*, 2012)。

## 10.7 长期依赖的挑战

学习循环网络长期依赖的数学挑战在第 8.2.5 节中引入。根本问题是，经过许多阶段传播后的梯度倾向于消失（大部分情况）或爆炸（很少，但对优化过程影响很大）。即使我们假设循环网络是参数稳定的（可存储记忆，且梯度不爆炸），但长期依赖的困难来自比短期相互作用指数小的权重（涉及许多 Jacobian 相乘）。许多资料提供了更深层次的讨论 (Hochreiter, 1991a; Doya, 1993; Bengio *et al.*, 1994b; Pascanu *et al.*, 2013a)。在这一节中，我们会更详细地描述该问题。其余几节介绍克服这个问题的方法。

循环网络涉及相同函数的多次组合，每个时间步一次。这些组合可以导致极端非线性行为，如图 10.15 所示。

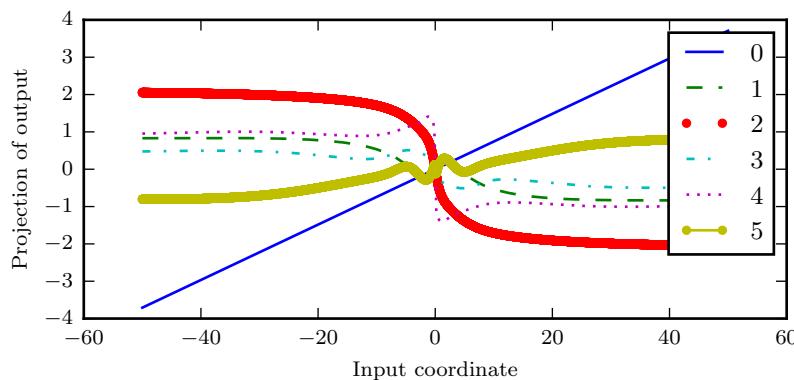


图 10.15：重复组合函数。当组合许多非线性函数（如这里所示的线性  $\tanh$  层）时，结果是高度非线性的，通常大多数值与微小的导数相关联，也有一些具有大导数的值，以及在增加和减小之间的多次交替。此处，我们绘制从 100 维隐藏状态降到单个维度的线性投影，绘制于  $y$  轴上。 $x$  轴是 100 维空间中沿着随机方向的初始状态的坐标。因此，我们可以将该图视为高维函数的线性截面。曲线显示每个时间步之后的函数，或者等价地，转换函数被组合一定次数之后。

特别地，循环神经网络所使用的函数组合有点像矩阵乘法。我们可以认为，循

环联系

$$\mathbf{h}^{(t)} = \mathbf{W}^\top \mathbf{h}^{(t-1)} \quad (10.36)$$

是一个非常简单的、缺少非线性激活函数和输入  $\mathbf{x}$  的循环神经网络。如第 8.2.5 节描述，这种递推关系本质上描述了幂法。它可以被简化为

$$\mathbf{h}^{(t)} = (\mathbf{W}^t)^\top \mathbf{h}^{(0)}, \quad (10.37)$$

而当  $\mathbf{W}$  符合下列形式的特征分解

$$\mathbf{W} = \mathbf{Q} \Lambda \mathbf{Q}^\top, \quad (10.38)$$

其中  $\mathbf{Q}$  正交，循环性可进一步简化为

$$\mathbf{h}^{(t)} = \mathbf{Q}^\top \Lambda^t \mathbf{Q} \mathbf{h}^{(0)}. \quad (10.39)$$

特征值提升到  $t$  次后，导致幅值不到一的特征值衰减到零，而幅值大于一的就会激增。任何不与最大特征向量对齐的  $\mathbf{h}^{(0)}$  的部分将最终被丢弃。

这个问题是针对循环网络的。在标量情况下，想象多次乘一个权重  $w$ 。该乘积  $w^t$  消失还是爆炸取决于  $w$  的幅值。然而，如果每个时刻使用不同权重  $w^{(t)}$  的非循环网络，情况就不同了。如果初始状态给定为 1，那么时刻  $t$  的状态可以由  $\prod_t w^{(t)}$  给出。假设  $w^{(t)}$  的值是随机生成的，各自独立，且有 0 均值  $v$  方差。乘积的方差就为  $\mathcal{O}(v^n)$ 。为了获得某些期望的方差  $v^*$ ，我们可以选择单个方差为  $v = \sqrt[n]{v^*}$  权重。因此，非常深的前馈网络通过精心设计的比例可以避免梯度消失和爆炸问题，如 Sussillo (2014) 所主张的。

RNN 梯度消失和爆炸问题是不同研究人员独立发现 (Hochreiter, 1991a; Bengio *et al.*, 1993, 1994b)。有人可能会希望通过简单地停留在梯度不消失或爆炸的参数空间来避免这个问题。不幸的是，为了储存记忆并对小扰动具有鲁棒性，RNN 必须进入参数空间中的梯度消失区域 (Bengio *et al.*, 1993, 1994b)。具体来说，每当模型能够表示长期依赖时，长期相互作用的梯度幅值就会变得指数小（相比短期相互作用的梯度幅值）。这并不意味着这是不可能学习的，由于长期依赖关系的信号很容易被短期相关性产生的最小波动隐藏，因而学习长期依赖可能需要很长的时间。实践中，Bengio *et al.* (1994b) 的实验表明，当我们增加了需要捕获的依赖关系的跨度，基于梯度的优化变得越来越困难，SGD 在长度仅为 10 或 20 的序列上成功训练传统 RNN 的概率迅速变为 0。

将循环网络作为动力系统更深入探讨的资料见 Doya (1993); Bengio *et al.* (1994b); Siegelmann and Sontag (1995) 及 Pascanu *et al.* (2013b) 的回顾。本章的其余部分将讨论目前已经提出的降低学习长期依赖（在某些情况下，允许一个 RNN 学习横跨数百步的依赖）难度的不同方法，但学习长期依赖的问题仍是深度学习中的一个主要挑战。

## 10.8 回声状态网络

从  $\mathbf{h}^{(t-1)}$  到  $\mathbf{h}^{(t)}$  的循环权重映射以及从  $\mathbf{x}^{(t)}$  到  $\mathbf{h}^{(t)}$  的输入权重映射是循环网络中最难学习的参数。研究者 (Jaeger, 2003; Maass *et al.*, 2002; Jaeger and Haas, 2004; Jaeger, 2007b) 提出避免这种困难的方法是设定循环隐藏单元，使其能很好地捕捉过去输入历史，并且只学习输出权重。回声状态网络 (echo state network) 或 ESN (Jaeger and Haas, 2004; Jaeger, 2007b)，以及流体状态机 (liquid state machine) (Maass *et al.*, 2002) 分别独立地提出了这种想法。后者是类似的，只不过它使用脉冲神经元 (二值输出) 而不是 ESN 中的连续隐藏单元。ESN 和流体状态机都被称为储层计算 (reservoir computing) (Lukoševičius and Jaeger, 2009)，因为隐藏单元形成了可能捕获输入历史不同方面的临时特征池。

储层计算循环网络类似于核机器，这是思考它们的一种方式：它们将任意长度的序列（到时刻  $t$  的输入历史）映射为一个长度固定的向量（循环状态  $\mathbf{h}^{(t)}$ ），之后可以施加一个线性预测算子（通常是一个线性回归）以解决感兴趣的问题。训练准则就可以很容易地设计为输出权重的凸函数。例如，如果输出是从隐藏单元到输出目标的线性回归，训练准则就是均方误差，由于是凸的就可以用简单的学习算法可靠地解决 (Jaeger, 2003)。

因此，重要的问题是：我们如何设置输入和循环权重才能让一组丰富的历史可以在循环神经网络的状态中表示？储层计算研究给出的答案是将循环网络视为动态系统，并设定让动态系统接近稳定边缘的输入和循环权重。

最初的想法是使状态到状态转换函数的 Jacobian 矩阵的特征值接近 1。如第 8.2.5 节解释，循环网络的一个重要特征就是 Jacobian 矩阵的特征值谱  $\mathbf{J}^{(t)} = \frac{\partial s^{(t)}}{\partial s^{(t-1)}}$ 。特别重要的是  $\mathbf{J}^{(t)}$  的谱半径 (spectral radius)，定义为特征值的最大绝对值。

为了解谱半径的影响，可以考虑反向传播中 Jacobian 矩阵  $\mathbf{J}$  不随  $t$  改变的简单

情况。例如当网络是纯线性时，会发生这种情况。假设  $\mathbf{J}$  特征值  $\lambda$  对应的特征向量为  $\mathbf{v}$ 。考虑当我们通过时间向后传播梯度向量时会发生什么。如果刚开始的梯度向量为  $\mathbf{g}$ ，然后经过反向传播的一个步骤后，我们将得到  $\mathbf{J}\mathbf{g}$ ， $n$  步之后我们会得到  $\mathbf{J}^n\mathbf{g}$ 。现在考虑如果我们向后传播扰动版本的  $\mathbf{g}$  会发生什么。如果我们刚开始是  $\mathbf{g} + \delta\mathbf{v}$ ，一步之后，我们会得到  $\mathbf{J}(\mathbf{g} + \delta\mathbf{v})$ 。 $n$  步之后，我们将得到  $\mathbf{J}^n(\mathbf{g} + \delta\mathbf{v})$ 。由此我们可以看出，由  $\mathbf{g}$  开始的反向传播和由  $\mathbf{g} + \delta\mathbf{v}$  开始的反向传播， $n$  步之后偏离  $\delta\mathbf{J}^n\mathbf{v}$ 。如果  $\mathbf{v}$  选择为  $\mathbf{J}$  特征值  $\lambda$  对应的一个单位特征向量，那么在每一步乘 Jacobian 矩阵只是简单地缩放。反向传播的两次执行分离的距离为  $\delta|\lambda|^n$ 。当  $\mathbf{v}$  对应于最大特征值  $|\lambda|$ ，初始扰动为  $\delta$  时这个扰动达到可能的最宽分离。

当  $|\lambda| > 1$ ，偏差  $\delta|\lambda|^n$  就会指数增长。当  $|\lambda| < 1$ ，偏差就会变得指数小。

当然，这个例子假定 Jacobian 矩阵在每个时间步是相同的，即对应于没有非线性循环网络。当非线性存在时，非线性的导数将在许多时间步后接近零，并有助于防止因过大的谱半径而导致的爆炸。事实上，关于回声状态网络的最近工作提倡使用远大于 1 的谱半径 (Yildiz *et al.*, 2012; Jaeger, 2012)。

我们已经说过多次，通过反复矩阵乘法的反向传播同样适用于没有非线性的正向传播的网络，其状态为  $\mathbf{h}^{(t+1)} = \mathbf{h}^{(t)\top} \mathbf{W}$ 。

如果线性映射  $\mathbf{W}^\top$  在  $L^2$  范数的测度下总是缩小  $\mathbf{h}$ ，那么我们说这个映射是 **收缩** (contractive) 的。当谱半径小于一，则从  $\mathbf{h}^{(t)}$  到  $\mathbf{h}^{(t+1)}$  的映射是收缩的，因此小变化在每个时间步后变得更小。当我们使用有限精度（如 32 位整数）来存储状态向量时，必然会使网络忘掉过去的信息。

Jacobian 矩阵告诉我们  $\mathbf{h}^{(t)}$  一个微小的变化如何向前一步传播，或等价的， $\mathbf{h}^{(t+1)}$  的梯度如何向后一步传播。需要注意的是， $\mathbf{W}$  和  $\mathbf{J}$  都不需要是对称的（尽管它们是实方阵），因此它们可能有复的特征值和特征向量，其中虚数分量对应于潜在的振荡行为（如果迭代地应用同一 Jacobian）。即使  $\mathbf{h}^{(t)}$  或  $\mathbf{h}^{(t)}$  中有趣的小变化在反向传播中是实值的，它们仍可以用这样的复数基表示。重要的是，当向量乘以矩阵时，这些复数基的系数幅值（复数的绝对值）会发生什么变化。幅值大于 1 的特征值对应于放大（如果反复应用则指数增长）或收缩（如果反复应用则指数减小）。

非线性映射情况时，Jacobian 会在每一步任意变化。因此，动态量变得更加复杂。然而，一个小的初始变化多步之后仍然会变成一个大的变化。纯线性和非线性情况的一个不同之处在于使用压缩非线性（如 tanh）可以使循环动态量有界。注意，即使前向传播动态量有界，反向传播的动态量仍然可能无界，例如，当 tanh 序列

都在它们状态中间的线性部分，并且由谱半径大于 1 的权重矩阵连接。然而，所有 tanh 单元同时位于它们的线性激活点是非常罕见的。

回声状态网络的策略是简单地固定权重使其具有一定的谱半径如 3，其中信息通过时间前向传播，但会由于饱和非线性单元（如 tanh）的稳定作用而不会爆炸。

最近，已经有研究表明，用于设置 ESN 权重的技术可以用来初始化完全可训练的循环网络的权重（通过时间反向传播来训练隐藏到隐藏的循环权重），帮助学习长期依赖 (Sutskever, 2012; Sutskever *et al.*, 2013)。在这种设定下，结合第 8.4 节中稀疏初始化的方案，设置 1.2 的初始谱半径表现不错。

## 10.9 渗漏单元和其他多时间尺度的策略

处理长期依赖的一种方法是设计工作在多个时间尺度的模型，使模型的某些部分在细粒度时间尺度上操作并能处理小细节，而其他部分在粗时间尺度上操作并能把遥远过去的信息更有效地传递过来。存在多种同时构建粗细时间尺度的策略。这些策略包括在时间轴增加跳跃连接，“渗漏单元”使用不同时间常数整合信号，并去除一些用于建模细粒度时间尺度的连接。

### 10.9.1 时间维度的跳跃连接

增加从遥远过去的变量到目前变量的直接连接是得到粗时间尺度的一种方法。使用这样跳跃连接的想法可以追溯到 Lin *et al.* (1996)，紧接是向前馈网络引入延迟的想法 (Lang and Hinton, 1988)。在普通的循环网络中，循环从时刻  $t$  的单元连接到时刻  $t + 1$  单元。构造较长的延迟循环网络是可能的 (Bengio, 1991)。

正如我们在第 8.2.5 节看到，梯度可能关于时间步数呈指数消失或爆炸。(Lin *et al.*, 1996) 引入了  $d$  延时的循环连接以减轻这个问题。现在导数指数减小的速度与  $\frac{\tau}{d}$  相关而不是  $\tau$ 。既然同时存在延迟和单步连接，梯度仍可能成  $t$  指数爆炸。这允许学习算法捕获更长的依赖性，但不是所有的长期依赖都能在这种方式下良好地表示。

### 10.9.2 渗漏单元和一系列不同时间尺度

获得导数乘积接近 1 的另一方式是设置线性自连接单元，并且这些连接的权重接近 1。

我们对某些  $v$  值应用更新  $\mu^{(t)} \leftarrow \alpha\mu^{(t-1)} + (1 - \alpha)v^{(t)}$  累积一个滑动平均值  $\mu^{(t)}$ ，其中  $\alpha$  是一个从  $\mu^{(t-1)}$  到  $\mu^{(t)}$  线性自连接的例子。当  $\alpha$  接近 1 时，滑动平均值能记住过去很长一段时间的信息，而当  $\alpha$  接近 0，关于过去的信息被迅速丢弃。线性自连接的隐藏单元可以模拟滑动平均的行为。这种隐藏单元称为 **渗漏单元** (leaky unit)。

$d$  时间步的跳跃连接可以确保单元总能被  $d$  个时间步前的那个值影响。使用权重接近 1 的线性自连接是确保该单元可以访问过去值的不同方式。线性自连接通过调节实值  $\alpha$  更平滑灵活地调整这种效果，而不是调整整数值的跳跃长度。

这个想法由 Mozer (1992) 和 El Hihi and Bengio (1996) 提出。在回声状态网络中，渗漏单元也被发现很有用 (Jaeger *et al.*, 2007)。

我们可以通过两种基本策略设置渗漏单元使用的时间常数。一种策略是手动将其固定为常数，例如在初始化时从某些分布采样它们的值。另一种策略是使时间常数成为自由变量，并学习出来。在不同时间尺度使用这样的渗漏单元似乎能帮助学习长期依赖 (Mozer, 1992; Pascanu *et al.*, 2013a)。

### 10.9.3 删除连接

处理长期依赖另一种方法是在多个时间尺度组织 RNN 状态的想法 (El Hihi and Bengio, 1996)，信息在较慢的时间尺度上更容易长距离流动。

这个想法与之前讨论的时间维度上的跳跃连接不同，因为它涉及主动删除长度为一的连接并用更长的连接替换它们。以这种方式修改的单元被迫在长时间尺度上运作。而通过时间跳跃连接是添加边。收到这种新连接的单元，可以学习在长时间尺度上运作，但也可以选择专注于自己其他的短期连接。

强制一组循环单元在不同时间尺度上运作有不同的方式。一种选择是使循环单元变成渗漏单元，但不同的单元组关联不同的固定时间尺度。这由 Mozer (1992) 提出，并被成功应用于 Pascanu *et al.* (2013a)。另一种选择是使显式且离散的更新发生在不同的时间，不同的单元组有不同的频率。这是 El Hihi and Bengio (1996) 和 Koutnik *et al.* (2014) 的方法。它在一些基准数据集上表现不错。

## 10.10 长短期记忆和其他门控 RNN

本文撰写之时，实际应用中最有效的序列模型称为**门控 RNN**（gated RNN）。包括基于**长短期记忆**（long short-term memory）和基于**门控循环单元**（gated recurrent unit）的网络。

像渗漏单元一样，门控 RNN 想法也是基于生成通过时间的路径，其中导数既不消失也不发生爆炸。渗漏单元通过手动选择常量的连接权重或参数化的连接权重来达到这一目的。门控 RNN 将其推广为在每个时间步都可能改变的连接权重。

渗漏单元允许网络在较长持续时间内积累信息（诸如用于特定特征或类的线索）。然而，一旦该信息被使用，让神经网络遗忘旧的状态可能是有用的。例如，如果一个序列是由子序列组成，我们希望渗漏单元能在各子序列内积累线索，我们需要将状态设置为 0 以忘记旧状态的机制。我们希望神经网络学会决定何时清除状态，而不是手动决定。这就是门控 RNN 要做的事。

### 10.10.1 LSTM

引入自循环的巧妙构思，以产生梯度长时间持续流动的路径是初始**长短期记忆**（long short-term memory, LSTM）模型的核心贡献 (Hochreiter and Schmidhuber, 1997)。其中一个关键扩展是使自循环的权重视上下文而定，而不是固定的 (Gers *et al.*, 2000)。门控此自循环（由另一个隐藏单元控制）的权重，累积的时间尺度可以动态地改变。在这种情况下，即使是具有固定参数的 LSTM，累积的时间尺度也可以因输入序列而改变，因为时间常数是模型本身的输出。LSTM 已经在许多应用中取得重大成功，如无约束手写识别 (Graves *et al.*, 2009)、语音识别 (Graves *et al.*, 2013; Graves and Jaitly, 2014)、手写生成 (Graves, 2013)、机器翻译 (Sutskever *et al.*, 2014)、为图像生成标题 (Kiros *et al.*, 2014b; Vinyals *et al.*, 2014b; Xu *et al.*, 2015) 和解析 (Vinyals *et al.*, 2014a)。

LSTM 块如图 10.16 所示。在浅循环网络的架构下，相应的前向传播公式如下。更深的架构也被成功应用 (Graves *et al.*, 2013; Pascanu *et al.*, 2014a)。LSTM 循环网络除了外部的 RNN 循环外，还具有内部的“LSTM 细胞”循环（自环），因此 LSTM 不是简单地向输入和循环单元的仿射变换之后施加一个逐元素的非线性。与普通的循环网络类似，每个单元有相同的输入和输出，但也有更多的参数和控制信息流动的门控单元系统。最重要的组成部分是状态单元  $s_i^{(t)}$ ，与前一节讨论的渗漏

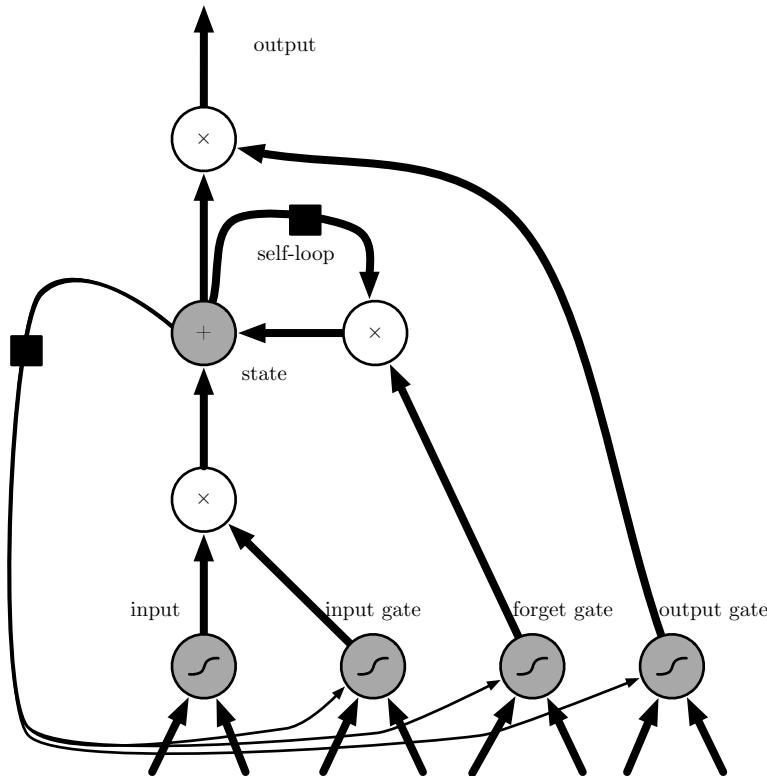


图 10.16: LSTM 循环网络“细胞”的框图。细胞彼此循环连接，代替一般循环网络中普通的隐藏单元。这里使用常规的人工神经元计算输入特征。如果 sigmoid 输入门允许，它的值可以累加到状态。状态单元具有线性自循环，其权重由遗忘门控制。细胞的输出可以被输出门关闭。所有门控单元都具有 sigmoid 非线性，而输入单元可具有任意的压缩非线性。状态单元也可以用作门控单元的额外输入。黑色方块表示单个时间步的延迟。

单元有类似的线性自环。然而，此处自环的权重（或相关联的时间常数）由遗忘门（forget gate） $f_i^{(t)}$  控制（时刻  $t$  和细胞  $i$ ），由 sigmoid 单元将权重设置为 0 和 1 之间的值：

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}\right), \quad (10.40)$$

其中  $x^{(t)}$  是当前输入向量， $h^t$  是当前隐藏层向量， $h^t$  包含所有 LSTM 细胞的输出。 $b^f, U^f, W^f$  分别是偏置、输入权重和遗忘门的循环权重。因此 LSTM 细胞内部状态

以如下方式更新，其中有一个条件的自环权重  $f_i^{(t)}$ ：

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma \left( b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)} \right), \quad (10.41)$$

其中  $\mathbf{b}$ ,  $\mathbf{U}$ ,  $\mathbf{W}$  分别是 LSTM 细胞中的偏置、输入权重和遗忘门的循环权重。外部输入门 (external input gate) 单元  $g_i^{(t)}$  以类似遗忘门 (使用 sigmoid 获得一个 0 和 1 之间的值) 的方式更新，但有自身的参数：

$$g_i^{(t)} = \sigma \left( b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)} \right). \quad (10.42)$$

LSTM 细胞的输出  $h_i^{(t)}$  也可以由输出门 (output gate)  $q_i^{(t)}$  关闭 (使用 sigmoid 单元作为门控)：

$$h_i^{(t)} = \tanh(s_i^{(t)}) q_i^{(t)}, \quad (10.43)$$

$$q_i^{(t)} = \sigma \left( b_i^o + \sum_j U_{i,j}^o x_j^{(t)} + \sum_j W_{i,j}^o h_j^{(t-1)} \right), \quad (10.44)$$

其中  $\mathbf{b}^o$ ,  $\mathbf{U}^o$ ,  $\mathbf{W}^o$  分别是偏置、输入权重和遗忘门的循环权重。在这些变体中，可以选择使用细胞状态  $s_i^{(t)}$  作为额外的输入 (及其权重)，输入到第  $i$  个单元的三个门，如图 10.16 所示。这将需要三个额外的参数。

LSTM 网络比简单的循环架构更易于学习长期依赖，先是用于测试长期依赖学习能力的人工数据集 (Bengio *et al.*, 1994c; Hochreiter and Schmidhuber, 1997; Hochreiter *et al.*, 2001)，然后是在具有挑战性的序列处理任务上获得最先进的表现 (Graves, 2012, 2013; Sutskever *et al.*, 2014)。LSTM 的变体和替代也已经被研究和使用，这将在下文进行讨论。

## 10.10.2 其他门控 RNN

LSTM 架构中哪些部分是真正必须的？还可以设计哪些其他成功架构允许网络动态地控制时间尺度和不同单元的遗忘行为？

最近关于门控 RNN 的工作给出了这些问题的某些答案，其单元也被称为门控循环单元或 GRU (Cho *et al.*, 2014c; Chung *et al.*, 2014, 2015a; Jozefowicz *et al.*, 2015; Chrupala *et al.*, 2015)。与 LSTM 的主要区别是，单个门控单元同时控制遗忘因子

和更新状态单元的决定。更新公式如下：

$$h_i^{(t)} = u_i^{(t-1)} h_i^{(t-1)} + (1 - u_i^{(t-1)}) \sigma \left( b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} r_j^{(t-1)} h_j^{(t-1)} \right), \quad (10.45)$$

其中  $u$  代表“更新”门， $r$  表示“复位”门。它们的值就如通常所定义的：

$$u_i^{(t)} = \sigma \left( b_i^u + \sum_j U_{i,j}^u x_j^{(t)} + \sum_j W_{i,j}^u h_j^{(t)} \right), \quad (10.46)$$

和

$$r_i^{(t)} = \sigma \left( b_i^r + \sum_j U_{i,j}^r x_j^{(t)} + \sum_j W_{i,j}^r h_j^{(t)} \right). \quad (10.47)$$

复位和更新门能独立地“忽略”状态向量的一部分。更新门像条件渗漏累积器一样可以线性门控任意维度，从而选择将它复制（在 sigmoid 的一个极端）或完全由新的“目标状态”值（朝向渗漏累积器的收敛方向）替换并完全忽略它（在另一个极端）。复位门控制当前状态中哪些部分用于计算下一个目标状态，在过去状态和未来状态之间引入了附加的非线性效应。

围绕这一主题可以设计更多的变种。例如复位门（或遗忘门）的输出可以在多个隐藏单元间共享。或者，全局门的乘积（覆盖一整组的单元，例如整一层）和一个局部门（每单元）可用于结合全局控制和局部控制。然而，一些调查发现这些 LSTM 和 GRU 架构的变种，在广泛的任务中难以明显地同时击败这两个原始架构 (Greff *et al.*, 2015; Jozefowicz *et al.*, 2015)。Greff *et al.* (2015) 发现其中的关键因素是遗忘门，而 Jozefowicz *et al.* (2015) 发现向 LSTM 遗忘门加入 1 的偏置（由 Gers *et al.* (2000) 提倡）能让 LSTM 变得与已探索的最佳变种一样健壮。

## 10.11 优化长期依赖

我们已经在第 8.2.5 节和第 10.7 节中描述过在许多时间步上优化 RNN 时发生的梯度消失和爆炸的问题。

由 Martens and Sutskever (2011) 提出了一个有趣的想法是，二阶导数可能在一阶导数消失的同时消失。二阶优化算法可以大致被理解为将一阶导数除以二阶导数（在更高维数，由梯度乘以 Hessian 的逆）。如果二阶导数与一阶导数以类似的速率收缩，那么一阶和二阶导数的比率可保持相对恒定。不幸的是，二阶方法有许多缺

点，包括高的计算成本、需要一个大的小批量、并且倾向于被吸引到鞍点。Martens and Sutskever (2011) 发现采用二阶方法的不错结果。之后，Sutskever *et al.* (2013) 发现使用较简单的方法可以达到类似的结果，例如经过谨慎初始化的 Nesterov 动量法。更详细的内容参考 Sutskever (2012)。应用于 LSTM 时，这两种方法在很大程度上会被单纯的 SGD (甚至没有动量) 取代。这是机器学习中一个延续的主题，设计一个易于优化模型通常比设计出更加强大的优化算法更容易。

### 10.11.1 截断梯度

如第 8.2.4 节讨论，强非线性函数（如由许多时间步计算的循环网络）往往倾向于非常大或非常小幅度的梯度。如图 8.3 和图 10.17 所示，我们可以看到，目标函数（作为参数的函数）存在一个伴随“悬崖”的“地形”：宽且相当平坦区域被目标函数变化快的小区域隔开，形成了一种悬崖。

这导致的困难是，当参数梯度非常大时，梯度下降的参数更新可以将参数抛出很远，进入目标函数较大的区域，到达当前解所作的努力变成了无用功。梯度告诉我们，围绕当前参数的无穷小区域内最速下降的方向。这个无穷小区域之外，代价函数可能开始沿曲线背面而上。更新必须被选择为足够小，以避免过分穿越向上的曲面。我们通常使用衰减速度足够慢的学习率，使连续的步骤具有大致相同的学习率。适合于一个相对线性的地形部分的步长经常在下一步进入地形中更加弯曲的部分时变得不适合，会导致上坡运动。

一个简单的解决方案已被从业者使用多年：**截断梯度** (clipping the gradient)。此想法有不同实例 (Mikolov, 2012; Pascanu *et al.*, 2013a)。一种选择是在参数更新之前，逐元素地截断小批量产生的参数梯度 (Mikolov, 2012)。另一种是在参数更新之前截断梯度  $\mathbf{g}$  的范数  $\|\mathbf{g}\|$  (Pascanu *et al.*, 2013a)：

$$\text{if } \|\mathbf{g}\| > v \quad (10.48)$$

$$\mathbf{g} \leftarrow \frac{\mathbf{g}v}{\|\mathbf{g}\|}, \quad (10.49)$$

其中  $v$  是范数上界， $\mathbf{g}$  用来更新参数。因为所有参数（包括不同的参数组，如权重和偏置）的梯度被单个缩放因子联合重整合，所以后一方法具有的优点是保证了每个步骤仍然是在梯度方向上的，但实验表明两种形式类似。虽然参数更新与真实梯度具有相同的方向梯度，经过梯度范数截断，参数更新的向量范数现在变得有界。这种有界梯度能避免执行梯度爆炸时的有害一步。事实上，当梯度大小高于阈值时，即

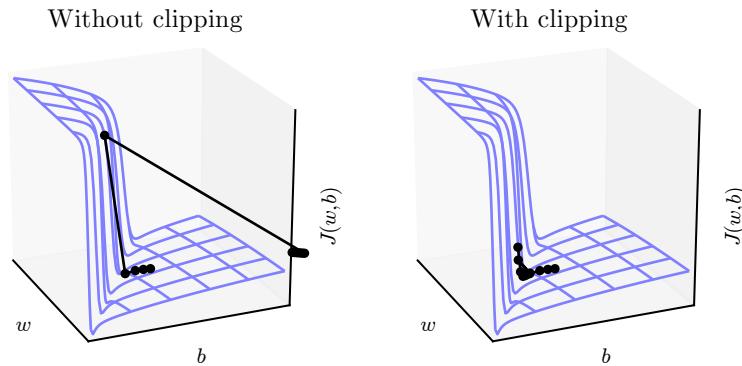


图 10.17: 梯度截断在有两个参数  $w$  和  $b$  的循环网络中的效果示例。梯度截断可以使梯度下降在极陡峭的悬崖附近更合理地执行。这些陡峭的悬崖通常发生在循环网络中，位于循环网络近似线性的附近。悬崖在时间步的数量上呈指数地陡峭，因为对于每个时间步，权重矩阵都自乘一次。(左)没有梯度截断的梯度下降越过这个小峡谷的底部，然后从悬崖面接收非常大的梯度。大梯度灾难性地将参数推到图的轴外。(右)使用梯度截断的梯度下降对悬崖的反应更温和。当它上升到悬崖面时，步长受到限制，使得它不会被推出靠近解的陡峭区域。经 Pascanu et al. (2013a) 许可改编此图。

使是采取简单的随机步骤往往工作得几乎一样好。如果爆炸非常严重，梯度数值上为 `Inf` 或 `Nan`（无穷大或不是一个数字），则可以采取大小为  $v$  的随机一步，通常会离开数值不稳定的状态。截断每小批量梯度范数不会改变单个小批量的梯度方向。然而，许多小批量使用范数截断梯度后的平均值不等同于截断真实梯度（使用所有的实例所形成的梯度）的范数。大导数范数的样本，和像这样的出现在同一小批量的样本，其对最终方向的贡献将消失。不像传统小批量梯度下降，其中真实梯度的方向是等于所有小批量梯度的平均。换句话说，传统的随机梯度下降使用梯度的无偏估计，而与使用范数截断的梯度下降引入了经验上是有用的启发式偏置。通过逐元素截断，更新的方向与真实梯度或小批量的梯度不再对齐，但是它仍然是一个下降方向。还有学者提出 (Graves, 2013)（相对于隐藏单元）截断反向传播梯度，但没有公布与这些变种之间的比较；我们推测，所有这些方法表现类似。

### 10.11.2 引导信息流的正则化

梯度截断有助于处理爆炸的梯度，但它无助于消失的梯度。为了解决消失的梯度问题并更好地捕获长期依赖，我们讨论了如下想法：在展开循环架构的计算图中，沿着与弧边相关联的梯度乘积接近 1 的部分创建路径。在第 10.10 节中已经讨论过，实现这一点的一种方法是使用 LSTM 以及其他自循环和门控机制。另一个想法是正则化或约束参数，以引导“信息流”。特别是即使损失函数只对序列尾部的输出作惩罚，我们也希望梯度向量  $\nabla_{\mathbf{h}^{(t)}} L$  在反向传播时能维持其幅度。形式上，我们要使

$$(\nabla_{\mathbf{h}^{(t)}} L) \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(t-1)}} \quad (10.50)$$

与

$$\nabla_{\mathbf{h}^{(t)}} L \quad (10.51)$$

一样大。在这个目标下，Pascanu *et al.* (2013a) 提出以下正则项：

$$\Omega = \sum_t \left( \frac{\left\| (\nabla_{\mathbf{h}^{(t)}} L) \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(t-1)}} \right\|}{\|\nabla_{\mathbf{h}^{(t)}} L\|} - 1 \right)^2. \quad (10.52)$$

计算这一梯度的正则项可能会出现困难，但 Pascanu *et al.* (2013a) 提出可以将后向传播向量  $\nabla_{\mathbf{h}^{(t)}} L$  考虑为恒值作为近似（为了计算正则化的目的，没有必要通过它们向后传播）。使用该正则项的实验表明，如果与标准的启发式截断（处理梯度爆炸）相结合，该正则项可以显著地增加 RNN 可以学习的依赖跨度。梯度截断特别重要，因为它保持了爆炸梯度边缘的 RNN 动态。如果没有梯度截断，梯度爆炸将阻碍学习的成功。

这种方法的一个主要弱点是，在处理数据冗余的任务时如语言模型，它并不像 LSTM 一样有效。

## 10.12 外显记忆

智能需要知识并且可以通过学习获取知识，这已促使大型深度架构的发展。然而，知识是不同的并且种类繁多。有些知识是隐含的、潜意识的并且难以用语言表达——比如怎么行走或狗与猫的样子有什么不同。其他知识可以是明确的、可陈述的以及可以相对简单地使用词语表达——每天常识性的知识，如“猫是一种动物”，

或者为实现自己当前目标所需知道的非常具体的事，如“与销售团队会议在 141 室于下午 3:00 开始”。

神经网络擅长存储隐性知识，但是他们很难记住事实。被存储在神经网络参数中之前，随机梯度下降需要多次提供相同的输入，即使如此，该输入也不会被特别精确地存储。Graves *et al.* (2014) 推测这是因为神经网络缺乏**工作存储** (working memory) 系统，即类似人类为实现一些目标而明确保存和操作相关信息片段的系统。这种外显记忆组件将使我们的系统不仅能够快速“故意”地存储和检索具体的事，也能利用他们循序推论。神经网络处理序列信息的需要，改变了每个步骤向网络注入输入的方式，长期以来推理能力被认为是重要的，而不是对输入做出自动的、直观的反应 (Hinton, 1990)。

为了解决这一难题，Weston *et al.* (2014) 引入了**记忆网络** (memory network)，其中包括一组可以通过寻址机制来访问的记忆单元。记忆网络原本需要监督信号指示他们如何使用自己的记忆单元。Graves *et al.* (2014) 引入的**神经网络图灵机** (neural Turing machine)，不需要明确的监督指示采取哪些行动而能学习从记忆单元读写任意内容，并通过使用基于内容的软注意机制（见 Bahdanau *et al.* (2015) 和第 12.4.5.1 节），允许端到端的训练。这种软寻址机制已成为其他允许基于梯度优化的模拟算法机制的相关架构的标准 (Sukhbaatar *et al.*, 2015; Joulin and Mikolov, 2015; Kumar *et al.*, 2015a; Vinyals *et al.*, 2015a; Grefenstette *et al.*, 2015)。

每个记忆单元可以被认为是 LSTM 和 GRU 中记忆单元的扩展。不同的是，网络输出一个内部状态来选择从哪个单元读取或写入，正如数字计算机读取或写入到特定地址的内存访问。

产生确切整数地址的函数很难优化。为了缓解这一问题，NTM 实际同时从多个记忆单元写入或读取。读取时，它们采取许多单元的加权平均值。写入时，他们对多个单元修改不同的数值。用于这些操作的系数被选择为集中在一小数目的单元，如通过 softmax 函数产生它们。使用这些具有非零导数的权重允许函数控制访问存储器，从而能使用梯度下降法优化。关于这些系数的梯度指示着其中每个参数是应该增加还是减少，但梯度通常只在接收大系数的存储器地址上变大。

这些记忆单元通常扩充为包含向量，而不是由 LSTM 或 GRU 存储单元所存储的单个标量。增加记忆单元大小的原因有两个。原因之一是，我们已经增加了访问记忆单元的成本。我们为产生用于许多单元的系数付出计算成本，但我们预期这些系数聚集在周围小数目的单元。通过读取向量值，而不是一个标量，我们可以抵

消部分成本。使用向量值的记忆单元的另一个原因是，它们允许基于内容的寻址 (content-based addressing)，其中从一个单元读或写的权重是该单元的函数。如果我们能够生产符合某些但并非所有元素的模式，向量值单元允许我们检索一个完整向量值的记忆。这类似于人们能够通过几个歌词回忆起一首歌曲的方式。我们可以认为基于内容的读取指令是说，“检索一首副歌歌词中带有‘我们都住在黄色潜水艇’的歌”。当我们要检索的对象很大时，基于内容的寻址更为有用——如果歌曲的每一个字母被存储在单独的记忆单元中，我们将无法通过这种方式找到他们。通过比较，基于位置的寻址 (location-based addressing) 不允许引用存储器的内容。我们可以认为基于位置的读取指令是说“检索 347 档的歌的歌词”。即使当存储单元很小时，基于位置的寻址通常也是完全合理的机制。

如果一个存储单元的内容在大多数时间步上会被复制（不被忘记），则它包含的信息可以在时间上向前传播，随时间向后传播的梯度也不会消失或爆炸。

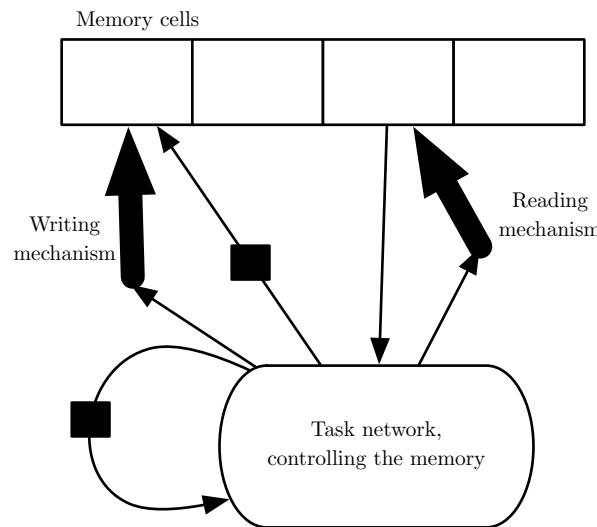


图 10.18：具有外显记忆网络的示意图，具备神经网络图灵机的一些关键设计元素。在此图中，我们将模型的“表示”部分（“任务网络”，这里是底部的循环网络）与存储事实的模型（记忆单元的集合）的“存储器”部分区分开。任务网络学习“控制”存储器，决定从哪读取以及在哪写入（通过读取和写入机制，由指向读取和写入地址的粗箭头指示）。

外显记忆的方法在图 10.18 说明，其中我们可以看到与存储器耦接的“任务神经网络”。虽然这一任务神经网络可以是前馈或循环的，但整个系统是一个循环网络。任务网络可以选择读取或写入的特定内存地址。外显记忆似乎允许模型学习普

通 RNN 或 LSTM RNN 不能学习的任务。这种优点的一个原因可能是因为信息和梯度可以在非常长的持续时间内传播（分别在时间上向前或向后）。

作为存储器单元的加权平均值反向传播的替代，我们可以将存储器寻址系数解释为概率，并随机从一个单元读取 (Zaremba and Sutskever, 2015)。优化离散决策的模型需要专门的优化算法，这将在第 20.9.1 节中描述。目前为止，训练这些做离散决策的随机架构，仍比训练进行软判决的确定性算法更难。

无论是软（允许反向传播）或随机硬性的，用于选择一个地址的机制与先前在机器翻译的背景下引入的注意力机制形式相同 (Bahdanau *et al.*, 2015)，这在第 12.4.5.1 节中也有讨论。甚至更早之前，注意力机制的想法就被引入了神经网络，在手写生成的情况下 (Graves, 2013)，有一个被约束为通过序列只向前移动的注意力机制。在机器翻译和记忆网络的情况下，每个步骤中关注的焦点可以移动到一个完全不同的地方 (相比之前的步骤)。

循环神经网络提供了将深度学习扩展到序列数据的一种方法。它们是我们的深度学习工具箱中最后一个主要的工具。现在我们的讨论将转移到如何选择和使用这些工具，以及如何在真实世界的任务中应用这些工具。

# 第十一章 实践方法论

要成功地使用深度学习技术，仅仅知道存在哪些算法和解释他们为何有效的原理是不够的。一个优秀的机器学习实践者还需要知道如何针对具体应用挑选一个合适的算法以及如何监控，并根据实验反馈改进机器学习系统。在机器学习系统的日常开发中，实践者需要决定是否收集更多的数据、增加或减少模型容量、添加或删除正则化项、改进模型的优化、改进模型的近似推断或调试模型的软件实现。尝试这些操作都需要大量时间，因此确定正确做法，而不盲目猜测尤为重要的。

本书的大部分内容都是关于不同的机器学习模型、训练算法和目标函数。这可能给人一种印象——成为机器学习专家的最重要因素是了解各种各样的机器学习技术，并熟悉各种不同的数学。在实践中，正确使用一个普通算法通常比草率地使用一个不清楚的算法效果更好。正确应用一个算法需要掌握一些相当简单的方法论。本章的许多建议都来自 Ng (2015)。

我们建议参考以下几个实践设计流程：

- 确定目标——使用什么样的误差度量，并为此误差度量指定目标值。这些目标和误差度量取决于该应用旨在解决的问题。
- 尽快建立一个端到端的工作流程，包括估计合适的性能度量。
- 搭建系统，并确定性能瓶颈。检查哪个部分的性能差于预期，以及是否是因为过拟合、欠拟合，或者数据或软件缺陷造成的。
- 根据具体观察反复地进行增量式的改动，如收集新数据、调整超参数或改进算法。

我们将使用街景地址号码转录系统 (Goodfellow *et al.*, 2014d) 作为一个运行示例。该应用的目标是将建筑物添加到谷歌地图。街景车拍摄建筑物，并记录与每张

建筑照片相关的 GPS 坐标。卷积网络识别每张照片上的地址号码，由谷歌地图数据库在正确的位置添加该地址。这个商业应用是一个很好的示例，它的开发流程遵循我们倡导的设计方法。

我们现在描述这个过程中的每一个步骤。

## 11.1 性能度量

确定目标，即使用什么误差度量，是必要的第一步，因为误差度量将指导接下来的所有工作。同时我们也应该了解大概能得到什么级别的目标性能。

值得注意的是对于大多数应用而言，不可能实现绝对零误差。即使你有无限的训练数据，并且恢复了真正的概率分布，贝叶斯误差仍定义了能达到的最小错误率。这是因为输入特征可能无法包含输出变量的完整信息，或是因为系统可能本质上是随机的。当然我们还会受限于有限的训练数据。

训练数据的数量会因为各种原因受到限制。当目标是打造现实世界中最好的产品或服务时，我们通常需要收集更多的数据，但必须确定进一步减少误差的价值，并与收集更多数据的成本做权衡。数据收集会耗费时间、金钱，或带来人体痛苦（例如，收集人体医疗测试数据）。科研中，目标通常是在某个确定基准下探讨哪个算法更好，一般会固定训练集，不允许收集更多的数据。

如何确定合理的性能期望？在学术界，通常我们可以根据先前公布的基准结果来估计预期错误率。在现实世界中，一个应用的错误率有必要是安全的、具有成本效益的或吸引消费者的。一旦你确定了想要达到的错误率，那么你的设计将由如何达到这个错误率来指导。

除了需要考虑性能度量之外，另一个需要考虑的是度量的选择。我们有几种不同的性能度量，可以用来度量一个含有机器学习组件的完整应用的有效性。这些性能度量通常不同于训练模型的代价函数。如第 5.1.2 节所述，我们通常会度量一个系统的准确率，或等价地，错误率。

然而，许多应用需要更高级的度量。

有时，一种错误可能会比另一种错误更严重。例如，垃圾邮件检测系统会有两种错误：将正常邮件错误地归为垃圾邮件，将垃圾邮件错误地归为正常邮件。阻止正常消息比允许可疑消息通过糟糕得多。我们希望度量某种形式的总代价，其中拦

截正常邮件比允许垃圾邮件通过的代价更高，而不是度量垃圾邮件分类的错误率。

有时，我们需要训练检测某些罕见事件的二元分类器。例如，我们可能会为一种罕见疾病设计医疗测试。假设每一百万人中只有一人患病。我们只需要让分类器一直报告没有患者，就能轻易地在检测任务上实现 99.9999% 的正确率。显然，正确率很难描述这种系统的性能。解决这个问题的方法是度量 精度 (precision) 和 召回率 (recall)。精度是模型报告的检测是正确的比率，而召回率则是真实事件被检测到的比率。检测器永远报告没有患者，会得到一个完美的精度，但召回率为零。而报告每个人都是患者的检测器会得到一个完美的召回率，但是精度会等于人群中患有该病的比例（在我们的例子是 0.0001%，每一百万人只有一人患病）。当使用精度和召回率时，我们通常会画 PR 曲线 (PR curve)， $y$  轴表示精度， $x$  轴表示召回率。如果检测到的事件发生了，那么分类器会返回一个较高的得分。例如，我们将前馈网络设计为检测一种疾病，估计一个医疗结果由特征  $\mathbf{x}$  表示的人患病的概率为  $\hat{y} = P(y = 1 | \mathbf{x})$ 。每当这个得分超过某个阈值时，我们报告检测结果。通过调整阈值，我们能权衡精度和召回率。在很多情况下，我们希望用一个数而不是曲线来概括分类器的性能。要做到这一点，我们可以将精度  $p$  和召回率  $r$  转换为 F 分数 (F-score)

$$F = \frac{2pr}{p + r}. \quad (11.1)$$

另一种方法是报告 PR 曲线下方的总面积。

在一些应用中，机器学习系统可能会拒绝做出判断。如果机器学习算法能够估计所作判断的置信度，这将会非常有用，特别是在错误判断会导致严重危害，而人工操作员能够偶尔接管的情况下。街景转录系统可以作为这种情况的一个示例。这个任务是识别照片上的地址号码，将照片拍摄地点对应到地图上的地址。如果地图是不精确的，那么地图的价值会严重下降。因此只在转录正确的情况下添加地址十分重要。如果机器学习系统认为它不太能像人一样正确地转录，那么最好办法当然是让人来转录照片。当然，只有当机器学习系统能够大量降低需要人工操作处理的图片时，它才是有用的。在这种情况下，一种自然的性能度量是 覆盖 (coverage)。覆盖是机器学习系统能够产生响应的样本所占的比率。我们权衡覆盖和精度。一个系统可以通过拒绝处理任意样本的方式来达到 100% 的精度，但是覆盖降到了 0%。对于街景任务，该项目的目标是达到人类级别的转录精度，同时保持 95% 的覆盖。在这项任务中，人类级别的性能是 98% 的精度。

还有许多其他的性能度量。例如，我们可以度量点击率、收集用户满意度调查

等等。许多专业的应用领域也有特定的标准。

最重要的是首先要确定改进哪个性能度量，然后专心提高性能度量。如果没有明确的目标，那么我们很难判断机器学习系统上的改动是否有所改进。

## 11.2 默认的基准模型

确定性能度量和目标后，任何实际应用的下一步是尽快建立一个合理的端到端的系统。本节给出了一些关于在不同情况下使用哪种算法作为第一个基准方法推荐。在本节中，我们提供了关于不同情况下使用哪种算法作为第一基准方法的推荐。值得注意的是，深度学习研究进展迅速，所以本书出版后很快可能会有更好的默认算法。

根据问题的复杂性，项目开始时可能无需使用深度学习。如果只需正确地选择几个线性权重就可能解决问题，那么项目可以开始于一个简单的统计模型，如逻辑回归。

如果问题属于“AI-完全”类的，如对象识别、语音识别、机器翻译等等，那么项目开始于一个合适的深度学习模型，效果会比较好。

首先，根据数据的结构选择一类合适的模型。如果项目是以固定大小的向量作为输入的监督学习，那么可以使用全连接的前馈网络。如果输入有已知的拓扑结构（例如，输入是图像），那么可以使用卷积网络。在这些情况下，刚开始可以使用某些分段线性单元（ReLU 或者其扩展，如 Leaky ReLU、PReLU 和 maxout）。如果输入或输出是一个序列，可以使用门控循环网络（LSTM 或 GRU）。

具有衰减学习率以及动量的 SGD 是优化算法一个合理的选择（流行的衰减方法有，衰减到固定最低学习率的线性衰减、指数衰减，或每次发生验证错误停滞时将学习率降低 2 – 10 倍，这些衰减方法在不同问题上好坏不一）。另一个非常合理的选择是 Adam 算法。批标准化对优化性能有着显著的影响，特别是对卷积网络和具有 sigmoid 非线性函数的网络而言。虽然在最初的基准中忽略批标准化是合理的，然而当优化似乎出现问题时，应该立刻使用批标准化。

除非训练集包含数千万以及更多的样本，否则项目应该在一开就包含一些温和的正则化。提前终止也被普遍采用。Dropout 也是一个很容易实现，且兼容很多模型和训练算法的出色正则化项。批标准化有时也能降低泛化误差，此时可以省略 Dropout 步骤，因为用于标准化变量的统计量估计本身就存在噪声。

如果我们的任务和另一个被广泛研究的任务相似，那么通过复制先前研究中已知性能良好的模型和算法，可能会得到很好的效果。甚至可以从该任务中复制一个训练好的模型。例如，通常会使用在 ImageNet 上训练好的卷积网络的特征来解决其他计算机视觉任务 (Girshick *et al.*, 2015)。

一个常见问题是项目开始时是否使用无监督学习，我们将在第三部分进一步探讨这个问题。这个问题和特定领域有关。在某些领域，比如自然语言处理，能够大大受益于无监督学习技术，如学习无监督词嵌入。在其他领域，如计算机视觉，除非是在半监督的设定下（标注样本数量很少）(Kingma *et al.*, 2014; Rasmus *et al.*, 2015)，目前无监督学习并没有带来益处。如果应用所在环境中，无监督学习被认为是很重要的，那么将其包含在第一个端到端的基准中。否则，只有在解决无监督问题时，才会第一次尝试时使用无监督学习。在发现初始基准过拟合的时候，我们可以尝试加入无监督学习。

## 11.3 决定是否收集更多数据

在建立第一个端到端的系统后，就可以度量算法性能并决定如何改进算法。许多机器学习新手都忍不住尝试很多不同的算法来进行改进。然而，收集更多的数据往往比改进学习算法要有用得多。

怎样判断是否要收集更多的数据？首先，确定训练集上的性能是否可接受。如果模型在训练集上的性能就很差，学习算法都不能在训练集上学习出良好的模型，那么就没必要收集更多的数据。反之，可以尝试增加更多的网络层或每层增加更多的隐藏单元，以增加模型的规模。此外，也可以尝试调整学习率等超参数的措施来改进学习算法。如果更大的模型和仔细调试的优化算法效果不佳，那么问题可能源自训练数据的质量。数据可能含太多噪声，或是可能不包含预测输出所需的正确输入。这意味着我们需要重新开始，收集更干净的数据或是收集特征更丰富的数据集。

如果训练集上的性能是可接受的，那么我们开始度量测试集上的性能。如果测试集上的性能也是可以接受的，那么就顺利完成了。如果测试集上的性能比训练集的要差得多，那么收集更多的数据是最有效的解决方案之一。这时主要的考虑是收集更多数据的代价和可行性，其他方法降低测试误差的代价和可行性，和增加数据数量能否显著提升测试集性能。在拥有百万甚至上亿用户的大型网络公司，收集大型数据集是可行的，并且这样做的成本可能比其他方法要少很多，所以答案几乎总是收

集更多的训练数据。例如，收集大型标注数据集是解决对象识别问题的主要因素之一。在其他情况下，如医疗应用，收集更多的数据可能代价很高或者不可行。一个可以替代的简单方法是降低模型大小或是改进正则化（调整超参数，如权重衰减系数，或是加入正则化策略，如Dropout）。如果调整正则化超参数后，训练集性能和测试集性能之间的差距还是不可接受，那么收集更多的数据是可取的。

在决定是否收集更多的数据时，也需要确定收集多少数据。如图 5.4 所示，绘制曲线显示训练集规模和泛化误差之间的关系是很有帮助的。根据走势延伸曲线，可以预测还需要多少训练数据来达到一定的性能。通常，加入总数目一小部分的样本不会对泛化误差产生显著的影响。因此，建议在对数尺度上考虑训练集的大小，例如在后续的实验中倍增样本数目。

如果收集更多的数据是不可行的，那么改进泛化误差的唯一方法是改进学习算法本身。这属于研究领域，并非对应用实践者的建议。

## 11.4 选择超参数

大部分深度学习算法都有许多超参数来控制不同方面的算法表现。有些超参数会影响算法运行的时间和存储成本。有些超参数会影响学习到的模型质量，以及在新输入上推断正确结果的能力。

有两种选择超参数的基本方法：手动选择和自动选择。手动选择超参数需要了解超参数做了些什么，以及机器学习模型如何才能取得良好的泛化。自动选择超参数算法大大减少了解这些想法的需要，但它们往往需要更高的计算成本。

### 11.4.1 手动调整超参数

手动设置超参数，我们必须了解超参数、训练误差、泛化误差和计算资源（内存和运行时间）之间的关系。这需要切实了解一个学习算法有效容量的基础概念，如第五章所描述的。

手动搜索超参数的目标通常是最小化受限于运行时间和内存预算的泛化误差。我们不去探讨如何确定各种超参数对运行时间和内存的影响，因为这高度依赖于平台。

手动搜索超参数的主要目标是调整模型的有效容量以匹配任务的复杂性。有

效容量受限于三个因素：模型的表示容量、学习算法成功最小化训练模型代价函数的能力以及代价函数和训练过程正则化模型的程度。具有更多网络层，每层有更多隐藏单元的模型具有较高的表示能力——能够表示更复杂的函数。然而，如果训练算法不能找到某个合适的函数来最小化训练代价，或是正则化项（如权重衰减）排除了这些合适的函数，那么即使模型的表达能力较高，也不能学习出合适的函数。

当泛化误差以某个超参数为变量，作为函数绘制出来时，通常会表现为 U 形曲线，如图 5.3 所示。在某个极端情况下，超参数对应着低容量，并且泛化误差由于训练误差较大而很高。这便是欠拟合的情况。另一种极端情况，超参数对应着高容量，并且泛化误差由于训练误差和测试误差之间的差距较大而很高。最优的模型容量位于曲线中间的某个位置，能够达到最低可能的泛化误差，由某个中等的泛化误差和某个中等的训练误差相加构成。

对于某些超参数，当超参数数值太大时，会发生过拟合。例如中间层隐藏单元的数量，增加数量能提高模型的容量，容易发生过拟合。对于某些超参数，当超参数数值太小时，也会发生过拟合。例如，最小的权重衰减系数允许为零，此时学习算法具有最大的有效容量，反而容易过拟合。

并非每个超参数都能对应着完整的 U 形曲线。很多超参数是离散的，如中间层单元数目或是 maxout 单元中线性元件的数目，这种情况只能沿曲线探索一些点。有些超参数是二值的。通常这些超参数用来指定是否使用学习算法中的一些可选部分，如预处理步骤减去均值并除以标准差来标准化输入特征。这些超参数只能探索曲线上的两点。其他一些超参数可能会有最小值或最大值，限制其探索曲线的某些部分。例如，权重衰减系数最小是零。这意味着，如果权重衰减系数为零时模型欠拟合，那么我们将无法通过修改权重衰减系数探索过拟合区域。换言之，有些超参数只能减少模型容量。

学习率可能是最重要的超参数。如果你只有时间调整一个超参数，那就调整学习率。相比其他超参数，它以一种更复杂的方式控制模型的有效容量——当学习率适合优化问题时，模型的有效容量最高，此时学习率是正确的，既不是特别大也不是特别小。学习率关于训练误差具有 U 形曲线，如图 11.1 所示。当学习率过大时，梯度下降可能会不经意地增加而非减少训练误差。在理想化的二次情况下，如果学习率是最佳值的两倍大时，会发生这种情况 (LeCun *et al.*, 1998b)。当学习率太小，训练不仅慢，还有可能永久停留在一个很高的训练误差。关于这种效应，我们知之甚少（不会发生于一个凸损失函数中）。

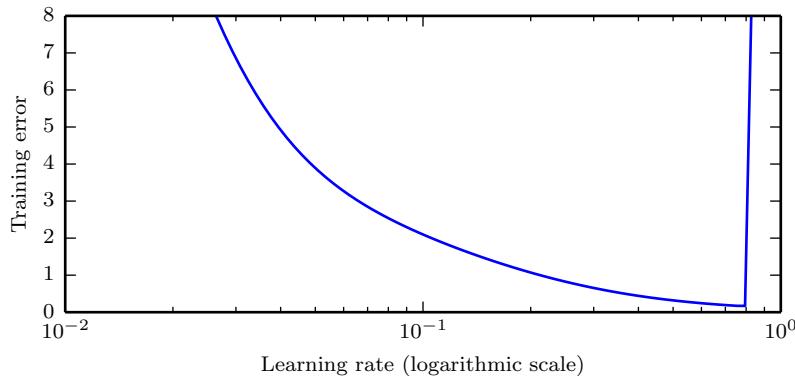


图 11.1: 训练误差和学习率之间的典型关系。注意当学习率大于最优值时误差会有显著的提升。此图针对固定的训练时间，越小的学习率有时候可以以一个正比于学习率减小量的因素来减慢训练过程。泛化误差也会得到类似的曲线，由于正则项作用在学习率过大或过小处比较复杂。由于一个糟糕的优化从某种程度上说可以避免过拟合，即使是训练误差相同的点也会拥有完全不同的泛化误差。

调整学习率外的其他参数时，需要同时监测训练误差和测试误差，以判断模型是否过拟合或欠拟合，然后适当调整其容量。

如果训练集错误率大于目标错误率，那么只能增加模型容量以改进模型。如果没有使用正则化，并且确信优化算法正确运行，那么有必要添加更多的网络层或隐藏单元。然而，令人遗憾的是，这增加了模型的计算代价。

如果测试集错误率大于目标错误率，那么可以采取两个方法。测试误差是训练误差和测试误差之间差距与训练误差的总和。寻找最佳的测试误差需要权衡这些数值。当训练误差较小（因此容量较大），测试误差主要取决于训练误差和测试误差之间的差距时，通常神经网络效果最好。此时目标是缩小这一差距，使训练误差的增长速率不快于差距减小的速率。要减少这个差距，我们可以改变正则化超参数，以减少有效的模型容量，如添加 Dropout 或权重衰减策略。通常，最佳性能来自正则化得很好的大规模模型，比如使用 Dropout 的神经网络。

大部分超参数可以通过推理其是否增加或减少模型容量来设置。部分示例如表11.1所示。

手动调整超参数时，不要忘记最终目标：提升测试集性能。加入正则化只是实现这个目标的一种方法。只要训练误差低，随时都可以通过收集更多的训练数据来

超参数	容量何时增加	原因	注意事项
隐藏单元数量	增加	增加隐藏单元数量会增加模型的表示能力。	几乎模型每个操作所需的时间和内存代价都会随隐藏单元数量的增加而增加。
学习率	调至最优	不正确的学习速率，不管是太高还是太低都会由于优化失败而导致低有效容量的模型。	
卷积核宽度	增加	增加卷积核宽度会增加模型的参数数量。	较宽的卷积核导致较窄的输出尺寸，除非使用隐式零填充减少此影响，否则会降低模型容量。较宽的卷积核需要更多的内存存储参数，并会增加运行时间，但较窄的输出会降低内存代价。
隐式零填充	增加	在卷积之前隐式添加零能保持较大尺寸的表示。	大多数操作的时间和内存代价会增加。
权重衰减系数	降低	降低权重衰减系数使得模型参数可以自由地变大。	
Dropout 比率	降低	较少地丢弃单元可以更多地让单元彼此“协力”来适应训练集。	

表 11.1: 各种超参数对模型容量的影响。

减少泛化误差。实践中能够确保学习有效的暴力方法就是不断提高模型容量和训练集的大小，直到解决问题。这种做法增加了训练和推断的计算代价，所以只有在拥有足够资源时才是可行的。原则上，这种做法可能会因为优化难度提高而失败，但对于许多问题而言，优化似乎并没有成为一个显著的障碍，当然，前提是选择了合适的模型。

### 11.4.2 自动超参数优化算法

理想的学习算法应该是只需要输入一个数据集，就可以输出学习的函数，而不需要手动调整超参数。一些流行的学习算法，如逻辑回归和支持向量机，流行的部

分原因是这类算法只有一到两个超参数需要调整，它们也能表现出不错的性能。有些情况下，所需调整的超参数数量较少时，神经网络可以表现出不错的性能；但超参数数量有几十甚至更多时，效果会提升得更加明显。当使用者有一个很好的初始值，例如由在相同类型的应用和架构上具有经验的人确定初始值，或者使用者在相似问题上具有几个月甚至几年的神经网络超参数调整经验，那么手动调整超参数能有很好的效果。然而，对于很多应用而言，这些起点都不可用。在这些情况下，自动算法可以找到合适的超参数。

如果我们仔细想想使用者搜索学习算法合适超参数的方式，我们会意识到这其实是一种优化：我们在试图寻找超参数来优化目标函数，例如验证误差，有时还会有一些约束（如训练时间，内存或识别时间的预算）。因此，原则上有可能开发出封装学习算法的超参数优化（hyperparameter optimization）算法，并选择其超参数，从而使用者不需要指定学习算法的超参数。令人遗憾的是，超参数优化算法往往有自己的超参数，如学习算法的每个超参数应该被探索的值的范围。然而，这些次级超参数通常很容易选择，这是说，相同的次级超参数能够很多不同的问题上具有良好的性能。

### 11.4.3 网格搜索

当有三个或更少的超参数时，常见的超参数搜索方法是网格搜索（grid search）。对于每个超参数，使用者选择一个较小的有限值集去探索。然后，这些超参数笛卡尔乘积得到一组组超参数，网格搜索使用每组超参数训练模型。挑选验证集误差最小的超参数作为最好的超参数。如图 11.2 所示超参数值的网络。

应该如何选择搜集集合的范围呢？在超参数是数值（有序）的情况下，每个列表的最小和最大的元素可以基于先前相似实验的经验保守地挑选出来，以确保最优解非常可能在所选范围内。通常，网格搜索大约会在对数尺度（logarithmic scale）下挑选合适的值，例如，一个学习率的取值集合是  $\{0.1, 0.01, 10^{-3}, 10^{-4}, 10^{-5}\}$ ，或者隐藏单元数目的取值集合  $\{50, 100, 200, 500, 1000, 2000\}$ 。

通常重复进行网格搜索时，效果会最好。例如，假设我们在集合  $\{-1, 0, 1\}$  上网格搜索超参数  $\alpha$ 。如果找到的最佳值是 1，那么说明我们低估了最优值  $\alpha$  所在的范围，应该改变搜索格点，例如在集合  $\{1, 2, 3\}$  中搜索。如果最佳值是 0，那么我们不妨通过细化搜索范围以改进估计，在集合  $\{-0.1, 0, 0.1\}$  上进行网格搜索。

网格搜索带来的一个明显问题是，计算代价会随着超参数数量呈指数级增长。如

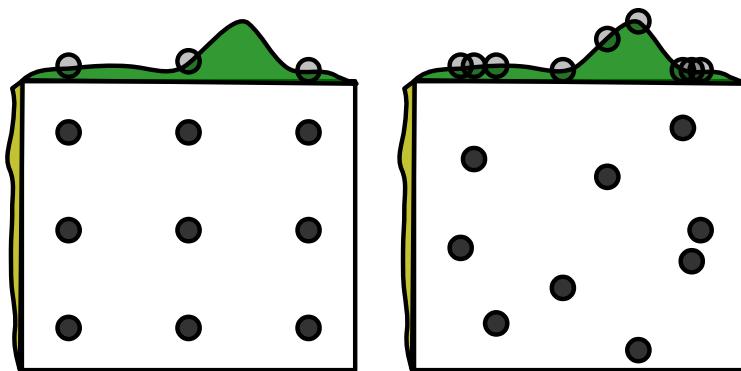


图 11.2: 网格搜索和随机搜索的比较。为了方便地说明, 我们只展示两个超参数的例子, 但是我们关注的问题中超参数个数通常会更多。(左) 为了实现网格搜索, 我们为每个超参数提供了一个值的集合。搜索算法对每一种在这些集合的交叉积中的超参数组合进行训练。(右) 为了实现随机搜索, 我们给联合超参数赋予了一个概率分布。通常超参数之间是相互独立的。常见的这种分布的选择是均匀分布或者是对数均匀(从对数均匀分布中抽样, 就是对从均匀分布中抽取的样本进行指数运算)的。然后这些搜索算法从联合的超参数空间中采样, 然后运行每一个样本。网格搜索和随机搜索都运行了验证集上的误差并返回了最优的解。这个图说明了通常只有一个超参数对结果有着重要的影响。在这个例子中, 只有水平轴上的超参数对结果有重要的作用。网格搜索将大量的计算浪费在了指数量级的对结果无影响的超参数中, 相比之下随机搜索几乎每次测试都测试了对结果有影响的每个超参数的独一无二的值。此图经 Bergstra and Bengio (2011) 允许转载。

如果有  $m$  个超参数, 每个最多取  $n$  个值, 那么训练和估计所需的试验数将是  $O(n^m)$ 。我们可以并行地进行实验, 并且并行要求十分宽松(进行不同搜索的机器之间几乎没有必要进行通信)。令人遗憾的是, 由于网格搜索指数量级增长计算代价, 即使是并行, 我们也无法提供令人满意的搜索规模。

#### 11.4.4 随机搜索

幸运的是, 有一个替代网格搜索的方法, 并且编程简单, 使用更方便, 能更快地收敛到超参数的良好取值: 随机搜索 (Bergstra and Bengio, 2012)。

随机搜索过程如下。首先, 我们为每个超参数定义一个边缘分布, 例如, Bernoulli 分布或范畴分布 (分别对应着二元超参数或离散超参数), 或者对数尺度上的均匀分

布（对应着正实值超参数）。例如，

$$\text{log\_learning\_rate} \sim u(-1, -5), \quad (11.2)$$

$$\text{learning\_rate} = 10^{\text{log\_learning\_rate}}, \quad (11.3)$$

其中， $u(a, b)$  表示区间  $(a, b)$  上均匀采样的样本。类似地，`log_number_of_hidden_units` 可以从  $u(\log(50), \log(2000))$  上采样。

与网格搜索不同，我们不需要离散化超参数的值。这允许我们在一个更大的集合上进行搜索，而不产生额外的计算代价。实际上，如图 11.2 所示，当有几个超参数对性能度量没有显著影响时，随机搜索相比于网格搜索指数级地高效。Bergstra and Bengio (2012) 进行了详细的研究并发现相比于网格搜索，随机搜索能够更快地减小验证集误差（就每个模型运行的试验数而言）。

与网格搜索一样，我们通常会重复运行不同版本的随机搜索，以基于前一次运行的结果改进下一次搜索。

随机搜索能比网格搜索更快地找到良好超参数的原因是，没有浪费的实验，不像网格搜索有时会对一个超参数的两个不同值（给定其他超参数值不变）给出相同结果。在网格搜索中，其他超参数将在这两次实验中拥有相同的值，而在随机搜索中，它们通常会具有不同的值。因此，如果这两个值的变化所对应的验证集误差没有明显区别的话，网格搜索没有必要重复两个等价的实验，而随机搜索仍然会对其他超参数进行两次独立地探索。

#### 11.4.5 基于模型的超参数优化

超参数搜索问题可以转化为一个优化问题。决策变量是超参数。优化的代价是超参数训练出来的模型在验证集上的误差。在简化的设定下，可以计算验证集上可导误差函数关于超参数的梯度，然后我们遵循这个梯度更新 (Bengio *et al.*, 1999; Bengio, 2000; Maclaurin *et al.*, 2015)。令人遗憾的是，在大多数实际设定中，这个梯度是不可用的。这可能是因为其高额的计算代价和存储成本，也可能是因为验证集误差在超参数上本质上不可导，例如超参数是离散值的情况。

为了弥补梯度的缺失，我们可以对验证集误差建模，然后通过优化该模型来提出新的超参数猜想。大部分基于模型的超参数搜索算法，都是使用贝叶斯回归模型来估计每个超参数的验证集误差期望和该期望的不确定性。因此，优化涉及到探索（探索高度不确定的超参数，可能带来显著的效果提升，也可能效果很差）和

使用（使用已经确信效果不错的超参数——通常是先前见过的非常熟悉的超参数）之间的权衡。关于超参数优化的最前沿方法还包括 Spearmint (Snoek *et al.*, 2012), TPE (Bergstra *et al.*, 2011) 和 SMAC (Hutter *et al.*, 2011)。

目前，我们无法明确确定，贝叶斯超参数优化是否是一个能够实现更好深度学习结果或是能够事半功倍的成熟工具。贝叶斯超参数优化有时表现得像人类专家，能够在有些问题上取得很好的效果，但有时又会在某些问题上发生灾难性的失误。看看它是否适用于一个特定的问题是值得尝试的，但目前该方法还不够成熟或可靠。就像所说的那样，超参数优化是一个重要的研究领域，通常主要受深度学习所需驱动，但是它不仅能贡献于整个机器学习领域，还能贡献于一般的工程学。

大部分超参数优化算法比随机搜索更复杂，并且具有一个共同的缺点，在它们能够从实验中提取任何信息之前，它们需要运行完整的训练实验。相比于人类实践者手动搜索，考虑实验早期可以收集的信息量，这种方法是相当低效的，因为手动搜索通常可以很早判断出某组超参数是否是完全病态的。Swersky *et al.* (2014) 提出了一个可以维护多个实验的早期版本算法。在不同的时间点，超参数优化算法可以选择开启一个新实验，“冻结”正在运行但希望不大的实验，或是“解冻”并恢复早期被冻结的，但现在根据更多信息后又有希望的实验。

## 11.5 调试策略

当一个机器学习系统效果不好时，通常很难判断效果不好的原因是算法本身，还是算法实现错误。由于各种原因，机器学习系统很难调试。

在大多数情况下，我们不能提前知道算法的行为。事实上，使用机器学习的整个出发点是，它会发现一些我们自己无法发现的有用行为。如果我们在一个新的分类任务上训练一个神经网络，它达到 5% 的测试误差，我们没法直接知道这是期望的结果，还是次优的结果。

另一个难点是，大部分机器学习模型有多个自适应的部分。如果一个部分失效了，其他部分仍然可以自适应，并获得大致可接受的性能。例如，假设我们正在训练多层神经网络，其中参数为权重  $\mathbf{W}$  和偏置  $\mathbf{b}$ 。进一步假设，我们单独手动实现了每个参数的梯度下降规则。而我们在偏置更新时犯了一个错误：

$$\mathbf{b} \leftarrow \mathbf{b} - \alpha, \quad (11.4)$$

其中  $\alpha$  是学习率。这个错误更新没有使用梯度。它会导致偏置在整个学习中不断变为负值，对于一个学习算法来说这显然是错误的。然而只是检查模型输出的话，该错误可能并不是显而易见的。根据输入的分布，权重可能可以自适应地补偿负的偏置。

大部分神经网络的调试策略都是解决这两个难题的一个或两个。我们可以设计一种足够简单的情况，能够提前得到正确结果，判断模型预测是否与之相符；我们也可以设计一个测试，独立检查神经网络实现的各个部分。

一些重要的调试检测如下所列。

**可视化计算中模型的行为：**当训练模型检测图像中的对象时，查看一些模型检测到部分重叠的图像。在训练语音生成模型时，试听一些生成的语音样本。这似乎是显而易见的，但在实际中很容易只注意量化性能度量，如准确率或对数似然。直接观察机器学习模型运行其任务，有助于确定其达到的量化性能数据是否看上去合理。错误评估模型性能可能是最具破坏性的错误之一，因为它们会使你在系统出问题时误以为系统运行良好。

**可视化最严重的错误：**大多数模型能够输出运行任务时的某种置信度量。例如，基于 softmax 函数输出层的分类器给每个类分配一个概率。因此，分配给最有可能的类的概率给出了模型在其分类决定上的置信估计值。通常，相比于正确预测的概率最大似然训练会略有高估。但是由于实际上模型的较小概率不太可能对应着正确的标签，因此它们在一定意义上还是有些用的。通过查看训练集中很难正确建模的样本，通常可以发现该数据预处理或者标记方式的问题。例如，街景转录系统原本有个问题是，地址号码检测系统会将图像裁剪得过于紧密，而省略掉了一些数字。然后转录网络会给这些图像的正确答案分配非常低的概率。将图像排序，确定置信度最高的错误，显示系统的裁剪有问题。修改检测系统裁剪更宽的图像，从而使整个系统获得更好的性能，但是转录网络需要能够处理地址号码中位置和范围更大变化的情况。

**根据训练和测试误差检测软件：**我们往往很难确定底层软件是否是正确实现。训练和测试误差能够提供一些线索。如果训练误差较低，但是测试误差较高，那么很有可能训练过程是在正常运行，但模型由于算法原因过拟合了。另一种可能是，测试误差没有被正确地度量，可能是由于训练后保存模型再重载去度量测试集时出现问题，或者是因为测试数据和训练数据预处理的方式不同。如果训练和测试误差都很高，那么很难确定是软件错误，还是由于算法原因模型欠拟合。这种情况需要进一步的测试，如下面所述。

**拟合极小的数据集：**当训练集上有很大的误差时，我们需要确定问题是真正的欠拟合，还是软件错误。通常，即使是小模型也可以保证很好地拟合一个足够小的数据集。例如，只有一个样本的分类数据可以通过正确设置输出层的偏置来拟合。通常，如果不能训练一个分类器来正确标注一个单独的样本，或不能训练一个自编码器来成功地精准再现一个单独的样本，或不能训练一个生成模型来一致地生成一个单独的样本，那么很有可能是由于软件错误阻止训练集上的成功优化。此测试可以扩展到只有少量样本的小数据集上。

**比较反向传播导数和数值导数：**如果读者正在使用一个需要实现梯度计算的软件框架，或者在添加一个新操作到求导库中，必须定义它的 `bprop` 方法，那么常见的错误原因是没能正确地实现梯度表达。验证这些求导正确性的一种方法是比较实现的自动求导和通过**有限差分** (finite difference) 计算的导数。因为

$$f'(x) = \lim_{\epsilon \rightarrow 0} \frac{f(x + \epsilon) - f(x)}{\epsilon}, \quad (11.5)$$

我们可以使用小的、有限的  $\epsilon$  近似导数：

$$f'(x) \approx \frac{f(x + \epsilon) - f(x)}{\epsilon}. \quad (11.6)$$

我们可以使用**中心差分** (centered difference) 提高近似的准确率：

$$f'(x) \approx \frac{f(x + \frac{1}{2}\epsilon) - f(x - \frac{1}{2}\epsilon)}{\epsilon}. \quad (11.7)$$

扰动大小  $\epsilon$  必须足够大，以确保该扰动不会由于数值计算的有限精度问题产生舍入误差。

通常，我们会测试向量值函数  $g : \mathbb{R}^m \rightarrow \mathbb{R}^n$  的梯度或 Jacobian 矩阵。令人遗憾的是，有限差分只允许我们每次计算一个导数。我们可以使用有限差分  $mn$  次评估  $g$  的所有偏导数，也可以将该测试应用于一个新函数（在函数  $g$  的输入输出都加上随机投影）。例如，我们可以将导数实现的测试用于函数  $f(x) = \mathbf{u}^T g(\mathbf{v}x)$ ，其中  $\mathbf{u}$  和  $\mathbf{v}$  是随机向量。正确计算  $f'(x)$  要求能够正确地通过  $g$  反向传播，但是使用有限差分能够高效地计算，因为  $f$  只有一个输入和一个输出。通常，一个好的方法是在多个  $\mathbf{u}$  值和  $\mathbf{v}$  值上重复这个测试，可以减少测试忽略了垂直于随机投影的错误的几率。

如果我们可以在复数上进行数值计算，那么使用复数作为函数的输入会有非常高效的数值方法估算梯度 (Squire and Trapp, 1998)。该方法基于如下观察

$$f(x + i\epsilon) = f(x) + i\epsilon f'(x) + O(\epsilon^2), \quad (11.8)$$

$$\text{real}(f(x + i\epsilon)) = f(x) + O(\epsilon^2), \quad \text{image}\left(\frac{f(x + i\epsilon)}{\epsilon}\right) = f'(x) + O(\epsilon^2), \quad (11.9)$$

其中  $i = \sqrt{-1}$ 。和上面的实值情况不同，这里不存在消除影响，因为我们在不同点上计算差分。因此我们可以使用很小的  $\epsilon$ ，比如  $\epsilon = 10^{-150}$ ，其中误差  $O(\epsilon^2)$  对所有实用目标都是微不足道的。

**监控激活函数值和梯度的直方图：**可视化神经网络在大量训练迭代后（也许是一个轮）收集到的激活函数值和梯度的统计量往往是有用的。隐藏单元的预激活值可以告诉我们该单元是否饱和，或者它们饱和的频率如何。例如，对于整流器，它们多久关一次？是否有单元一直关闭？对于双曲正切单元而言，预激活绝对值的平均值可以告诉我们该单元的饱和程度。在深度网络中，传播梯度的快速增长或快速消失，可能会阻碍优化过程。最后，比较参数梯度和参数的量级也是有帮助的。正如 (Bottou, 2015) 所建议的，我们希望参数在一个小批量更新中变化的幅度是参数量值 1% 这样的级别，而不是 50% 或者 0.001%（这会导致参数移动得太慢）。也有可能是某些参数以良好的步长移动，而另一些停滞。如果数据是稀疏的（比如自然语言），有些参数可能很少更新，检测它们变化时应该记住这一点。

最后，许多深度学习算法为每一步产生的结果提供了某种保证。例如，在第三部分，我们将看到一些使用代数解决优化问题的近似推断算法。通常，这些可以通过测试它们的每个保证来调试。某些优化算法提供的保证包括，目标函数值在算法的迭代步中不会增加，某些变量的导数在算法的每一步中都是零，所有变量的梯度在收敛时会变为零。通常，由于舍入误差，这些条件不会在数字计算机上完全成立，因此调试测试应该包含一些容差参数。

## 11.6 示例：多位数字识别

为了端到端的说明如何在实践中应用我们的设计方法论，我们从设计深度学习组件出发，简单地介绍下街景转录系统。显然，整个系统的许多其他组件，如街景车、数据库设施等等，也是极其重要的。

从机器学习任务的视角出发，首先这个过程要采集数据。街景车收集原始数据，然后操作员手动提供标签。转录任务开始前有大量的数据处理工作，包括在转录前使用其他机器学习技术探测房屋号码。

转录项目开始于性能度量的选择和对这些度量的期望值。一个重要的总原则是度量的选择要符合项目的业务目标。因为地图只有是高准确率时才有用，所以为这个项目设置高准确率的要求非常重要。具体地，目标是达到人类水平，98% 的准确

率。这种程度的准确率并不是总能达到。为了达到这个级别的准确率，街景转录系统牺牲了覆盖。因此在保持准确率 98% 的情况下，覆盖成了这个项目优化的主要性能度量。随着卷积网络的改进，我们能够降低网络拒绝转录输入的置信度阈值，最终超出了覆盖 95% 的目标。

在选择量化目标后，我们推荐方法的下一步是要快速建立一个合理的基准系统。对于视觉任务而言，基准系统是带有整流线性单元的卷积网络。转录项目开始于一个这样的模型。当时，使用卷积网络输出预测序列并不常见。开始时，我们使用一个尽可能简单的基准模型，该模型输出层的第一个实现包含  $n$  个不同的 softmax 单元来预测  $n$  个字符的序列。我们使用与训练分类任务相同的方式来训练这些 softmax 单元，独立地训练每个 softmax 单元。

我们建议反复细化这些基准，并测试每个变化是否都有改进。街景转录系统的第一个变化受激励于覆盖指标的理论理解和数据结构。具体地，当输出序列的概率低于某个值  $t$  即  $p(\mathbf{y} | \mathbf{x}) < t$  时，网络拒绝为输入  $\mathbf{x}$  分类。最初， $p(\mathbf{y} | \mathbf{x})$  的定义是临时的，简单地将所有 softmax 函数输出乘在一起。这促使我们发展能够真正计算出合理对数似然的特定输出层和代价函数。这种方法使得样本拒绝机制更有效。

此时，覆盖仍低于 90%，但该方法没有明显的理论问题了。因此，我们的方法论建议综合训练集和测试集性能，以确定问题是否是欠拟合或过拟合。在这种情况下，训练和测试集误差几乎是一样的。事实上，这个项目进行得如此顺利的主要原因是有数以千万计的标注样本数据集可用。因为训练和测试集的误差是如此相似，这表明要么是这个问题欠拟合，要么是训练数据的问题。我们推荐的调试策略之一是可视化模型最糟糕的错误。在这种情况下，这意味着可视化不正确而模型给了最高置信度的训练集转录结果。结果显示，主要是输入图像裁剪得太紧，有些和地址相关的数字被裁剪操作去除了。例如，地址“1849”的图片可能裁切得太紧，只剩下“849”是可见的。如果我们花费几周时间改进确定裁剪区域的地址号码检测系统的准确率，或许也可以解决这个问题。与之不同，项目团队采取了更实际的办法，简单地系统性扩大裁剪区域的宽度，使其大于地址号码检测系统预测的区域宽度。这种单一改变将转录系统的覆盖提高了 10 个百分点。

最后，性能提升的最后几个百分点来自调整超参数。这主要包括在保持一些计算代价限制的同时加大模型的规模。因为训练误差和测试误差保持几乎相等，所以明确表明性能不足是由欠拟合造成的，数据集本身也存在一些问题。

总体来说，转录项目是非常成功的，可以比人工速度更快、代价更低地转录数

以亿计的地址。

我们希望本章中介绍的设计原则能带来其他更多类似的成功。

# 第十二章 应用

在本章中，我们将介绍如何使用深度学习来解决计算机视觉、语音识别、自然语言处理以及其他商业领域中的应用。首先我们将讨论在许多最重要的 AI 应用中所需的大规模神经网络的实现。接着，我们将回顾深度学习已经成功应用的几个特定领域。尽管深度学习的一个目标是设计能够处理各种任务的算法，然而截止目前深度学习的应用仍然需要一定程度的特化。例如，计算机视觉中的任务对每一个样本都需要处理大量的输入特征（像素）。自然语言处理任务的每一个输入特征都需要对大量的可能值（词汇表中的词）建模。

## 12.1 大规模深度学习

深度学习的基本思想基于联结主义：尽管机器学习模型中单个生物性的神经元或者说是单个特征不是智能的，但是大量的神经元或者特征作用在一起往往能够表现出智能。我们必须着重强调神经元数量必须很大这个事实。相比 20 世纪 80 年代，如今神经网络的精度以及处理任务的复杂度都有一定提升，其中一个关键的因素就是网络规模的巨大提升。正如我们在第 1.2.3 节中看到的一样，在过去的三十年内，网络规模是以指数级的速度递增的。然而如今的人工神经网络的规模也仅仅和昆虫的神经系统差不多。

由于规模的大小对于神经网络来说至关重要，因此深度学习需要高性能的硬件设施和软件实现。

### 12.1.1 快速的 CPU 实现

传统的神经网络是用单台机器的 CPU 来训练的。如今，这种做法通常被视为是不可取的。现在，我们通常使用 GPU 或者许多台机器的 CPU 连接在一起进行计算。在使用这种昂贵配置之前，为论证 CPU 无法承担神经网络所需的巨大计算量，研究者们付出了巨大的努力。

描述如何实现高效的数值 CPU 代码已经超出了本书的讨论范围，但是我们在这里还是要强调通过设计一些特定的 CPU 上的操作可以大大提升效率。例如，在 2011 年，最好的 CPU 在训练神经网络时使用定点运算能够比浮点运算跑得更快。通过调整定点运算的实现方式，Vanhoucke *et al.* (2011) 获得了 3 倍于一个强浮点运算系统的速度。因为各个新型 CPU 都有各自不同的特性，所以有时候采用浮点运算实现会更快。一条重要的准则就是，通过特殊设计的数值运算，我们可以获得巨大的回报。除了选择定点运算或者浮点运算以外，其他的策略还包括了如通过优化数据结构避免高速缓存缺失、使用向量指令等。如果模型规模不会限制模型表现（不会影响模型精度）时，机器学习的研究者们一般忽略这些实现的细节。

### 12.1.2 GPU 实现

许多现代神经网络的实现基于 **图形处理器** (Graphics Processing Unit, GPU)。图形处理器 (GPU) 最初是为图形应用而开发的专用硬件组件。视频游戏系统的消费市场刺激了图形处理硬件的发展。它为视频游戏所设计的特性也可以使神经网络的计算受益。

视频游戏的渲染要求许多操作能够快速并行地执行。环境和角色模型通过一系列顶点的 3D 坐标确定。为了将大量的 3D 坐标转化为 2D 显示器上的坐标，显卡必须并行地对许多顶点执行矩阵乘法与除法。之后，显卡必须并行地在每个像素上执行诸多计算，来确定每个像素点的颜色。在这两种情况下，计算都是非常简单的，并且不涉及 CPU 通常遇到的复杂的分支运算。例如，同一个刚体内的每个顶点都会乘上相同的矩阵；也就是说，不需要通过 `if` 语句来判断确定每个顶点需要乘哪个矩阵。各个计算过程之间也是完全相互独立的，因此能够实现并行操作。计算过程还涉及处理大量内存缓冲以及描述每一个需要被渲染的对象的纹理（颜色模式）的位图信息。总的来说，这使显卡设计为拥有高度并行特性以及很高的内存带宽，同时也付出了一些代价，如相比传统的 CPU 更慢的时钟速度以及更弱的处理分支运算

的能力。

与上述的实时图形算法相比，神经网络算法所需要的性能特性是相同的。神经网络算法通常涉及大量参数、激活值、梯度值的缓冲区，其中每个值在每一次训练迭代中都要被完全更新。这些缓冲过大，会超出传统的桌面计算机的高速缓存 (cache)，所以内存带宽通常会成为主要瓶颈。相比 CPU，GPU 一个显著的优势是其极高的内存带宽。神经网络的训练算法通常并不涉及大量的分支运算与复杂的控制指令，所以更适合在 GPU 硬件上训练。由于神经网络能够被分为多个单独的“神经元”，并且独立于同一层内其他神经元进行处理，所以神经网络可以从 GPU 的并行特性中受益匪浅。

GPU 硬件最初专为图形任务而设计。随着时间的推移，GPU 也变得更灵活，允许定制的子程序处理转化顶点坐标或者计算像素颜色的任务。原则上，GPU 不要求这些像素值实际基于渲染任务。只要将计算的输出值作为像素值写入缓冲区，GPU 就可以用于科学计算。Steinkrau *et al.* (2005) 在 GPU 上实现了一个两层全连接的神经网络，并获得了相对基于 CPU 的基准方法三倍的加速。不久以后，Chellapilla *et al.* (2006) 也论证了相同的技术可以用来加速监督卷积网络的训练。

在通用 GPU 发布以后，使用显卡训练神经网络的热度开始爆炸性地增长。这种通用 GPU 可以执行任意的代码，而并非仅仅渲染子程序。NVIDIA 的 CUDA 编程语言使得我们可以用一种像 C 一样的语言实现任意代码。由于相对简便的编程模型，强大的并行能力以及巨大的内存带宽，通用 GPU 为我们提供了训练神经网络的理想平台。在它发布以后不久，这个平台就迅速被深度学习的研究者们所采纳 (Raina *et al.*, 2009b; Ciresan *et al.*, 2010)。

如何在通用 GPU 上写高效的代码依然是一个难题。在 GPU 上获得良好表现所需的技术与 CPU 上的技术非常不同。比如说，基于 CPU 的良好代码通常被设计为尽可能从高速缓存中读取更多的信息。然而在 GPU 中，大多数可写内存位置并不会被高速缓存，所以计算某个值两次往往比计算一次然后从内存中读取更快。GPU 代码是天生多线程的，不同线程之间必须仔细协调好。例如，如果能够把数据 **级联** (coalesced) 起来，那么涉及内存的操作一般会更快。当几个线程同时需要读/写一个值时，像这样的级联会作为一次内存操作出现。不同的 GPU 可能采用不同的级联读/写数据的方式。通常来说，如果在  $n$  个线程中，线程  $i$  访问的是第  $i + j$  处的内存，其中  $j$  是 2 的某个幂的倍数，那么内存操作就易于级联。具体的设定在不同的 GPU 型号中有所区别。GPU 另一个常见的设定是使一个组中的所有线程都同时执行同一指令。这意味着 GPU 难以执行分支操作。线程被分为一个个称

作 warp 的小组。在一个 warp 中的每一个线程在每一个循环中执行同一指令，所以当同一个 warp 中的不同线程需要执行不同的指令时，需要使用串行而非并行的方式。

由于实现高效 GPU 代码的困难性，研究人员应该组织好他们的工作流程，避免对每一个新的模型或算法都编写新的 GPU 代码。通常来讲，人们会选择建立一个包含高效操作（如卷积和矩阵乘法）的软件库解决这个问题，然后再从库中调用所需要的操作确定模型。例如，机器学习库 Pylearn2 (Goodfellow *et al.*, 2013e) 将其所有的机器学习算法都通过调用 Theano (Bergstra *et al.*, 2010c; Bastien *et al.*, 2012a) 和 cuda-convnet (Krizhevsky, 2010) 所提供的高性能操作来指定。这种分解方法还可以简化对多种硬件的支持。例如，同一个 Theano 程序可以在 CPU 或者 GPU 上运行，而不需要改变调用 Theano 的方式。其他库如 Tensorflow (Abadi *et al.*, 2015) 和 Torch (Collobert *et al.*, 2011b) 也提供了类似的功能。

### 12.1.3 大规模的分布式实现

在许多情况下，单个机器的计算资源是有限的。因此，我们希望把训练或者推断的任务分摊到多个机器上进行。

分布式的推断是容易实现的，因为每一个输入的样本都可以在单独的机器上运行。这也被称为 **数据并行** (data parallelism)。

同样地，**模型并行** (model parallelism) 也是可行的，其中多个机器共同运行一个数据点，每一个机器负责模型的一个部分。对于推断和训练，这都是可行的。

在训练过程中，数据并行某种程度上来说更加困难。对于随机梯度下降的单步来说，我们可以增加小批量的大小，但是从优化性能的角度来说，我们得到的回报通常并不会线性增长。使用多个机器并行地计算多个梯度下降步骤是一个更好的选择。不幸的是，梯度下降的标准定义完全是一个串行的过程：第  $t$  步的梯度是第  $t - 1$  步所得参数的函数。

这个问题可以使用**异步随机梯度下降** (Asynchronous Stochastic Gradient Descent) (Bengio *et al.*, 2001b; Recht *et al.*, 2011) 解决。在这个方法中，几个处理器的核共用存有参数的内存。每一个核在无锁情况下读取这些参数并计算对应的梯度，然后在无锁状态下更新这些参数。由于一些核把其他的核所更新的参数覆盖了，因此这种方法减少了每一步梯度下降所获得的平均提升。但因为更新步数的速率增

加，总体上还是加快了学习过程。Dean *et al.* (2012) 率先提出了多机器无锁的梯度下降方法，其中参数是由 **参数服务器** (parameter server) 管理而非存储在共用的内存中。分布式的异步梯度下降方法保留了训练深度神经网络的基本策略，并被工业界很多机器学习组所使用 (Chilimbi *et al.*, 2014; Wu *et al.*, 2015)。学术界的深度学习研究者们通常无法负担那么大规模的分布式学习系统，但是一些研究仍关注于如何在校园环境中使用相对廉价的硬件系统构造分布式网络 (Coates *et al.*, 2013)。

### 12.1.4 模型压缩

在许多商业应用的机器学习模型中，一个时间和内存开销较小的推断算法比一个时间和内存开销较小的训练算法要更为重要。对于那些不需要个性化设计的应用来说，我们只需要一次性的训练模型，然后它就可以被成千上万的用户使用。在许多情况下，相比开发者，终端用户的可用资源往往更有限。例如，开发者们可以使用巨大的计算机集群训练一个语音识别的网络，然后将其部署到移动手机上。

减少推断所需开销的一个关键策略是 **模型压缩** (model compression) (Buciluă *et al.*, 2006)。模型压缩的基本思想是用一个更小的模型取代原始耗时的模型，从而使得用来存储与评估所需的内存与运行时间更少。

当原始模型的规模很大，且我们需要防止过拟合时，模型压缩就可以起到作用。在许多情况下，拥有最小泛化误差的模型往往是多个独立训练而成的模型的集成。评估所有  $n$  个集成成员的成本很高。有时候，当单个模型很大（例如，如果它使用 Dropout 正则化）时，其泛化能力也会很好。

这些巨大的模型能够学习到某个函数  $f(\mathbf{x})$ ，但选用的参数数量超过了任务所需的参数数量。只是因为训练样本数是有限的，所以模型的规模才变得必要。只要我们拟合了这个函数  $f(\mathbf{x})$ ，我们就可以通过将  $f$  作用于随机采样点  $\mathbf{x}$  来生成有无穷多训练样本的训练集。然后，我们使用这些样本训练一个新的更小的模型，使其能够在这些点上拟合  $f(\mathbf{x})$ 。为了更加充分地利用了这个新的小模型的容量，最好从类似于真实测试数据（之后将提供给模型）的分布中采样  $\mathbf{x}$ 。这个过程可以通过损坏训练样本或者从原始训练数据训练的生成模型中采样完成。

此外，我们还可以仅在原始训练数据上训练一个更小的模型，但只是为了复制模型的其他特征，比如在不正确的类上的后验分布 (Hinton *et al.*, 2014, 2015)。

### 12.1.5 动态结构

一般来说，加速数据处理系统的一种策略是构造一个系统，这个系统用**动态结构** (dynamic structure) 描述图中处理输入的所需计算过程。在给定一个输入的情况下，数据处理系统可以动态地决定运行神经网络系统的哪一部分。单个神经网络内部同样也存在动态结构，给定输入信息，决定特征（隐藏单元）哪一部分用于计算。这种神经网络中的动态结构有时被称为**条件计算** (conditional computation) (Bengio, 2013; Bengio *et al.*, 2013b)。由于模型结构许多部分可能只跟输入的一小部分有关，只计算那些需要的特征可以起到加速的目的。

动态结构计算是一种基础的计算机科学方法，广泛应用于软件工程项目。应用于神经网络的最简单的动态结构基于决定神经网络（或者其他机器学习模型）中的哪些子集需要应用于特定的输入。

在分类器中加速推断的可行策略是使用**级联** (cascade) 的分类器。当目标是检测罕见对象（或事件）是否存在时，可以应用级联策略。要确定对象是否存在，我们必须使用具有高容量、运行成本高的复杂分类器。然而，因为对象是罕见的，我们通常可以使用更少的计算拒绝不包含对象的输入。在这些情况下，我们可以训练一序列分类器。序列中的第一个分类器具有低容量，训练为具有高召回率。换句话说，他们被训练为确保对象存在时，我们不会错误地拒绝输入。最后一个分类器被训练为具有高精度。在测试时，我们按照顺序运行分类器进行推断，一旦级联中的任何一个拒绝它，就选择抛弃。总的来说，这允许我们使用高容量模型以较高的置信度验证对象的存在，而不是强制我们为每个样本付出完全推断的成本。有两种不同的方式可以使得级联实现高容量。一种方法是使级联中靠后的成员单独具有高容量。在这种情况下，由于系统中的一些个体成员具有高容量，因此系统作为一个整体显然也具有高容量。还可以使用另一种级联，其中每个单独的模型具有低容量，但是由于许多小型模型的组合，整个系统具有高容量。Viola and Jones (2001) 使用级联的增强决策树实现了适合在手持数字相机中使用的快速并且鲁棒的面部检测器。本质上，它们的分类器使用滑动窗口方法来定位面部。分类器会检查许多的窗口，如果这些窗口内不包含面部则被拒绝。级联的另一个版本使用早期模型来实现一种硬注意力机制：级联的先遣成员定位对象，并且级联的后续成员在给定对象位置的情况下执行进一步处理。例如，Google 使用两步级联从街景视图图像中转换地址编号：首先使用一个机器学习模型查找地址编号，然后使用另一个机器学习模型将其转录 (Goodfellow *et al.*, 2014d)。

决策树本身是动态结构的一个例子，因为树中的每个节点决定应该使用哪个子树来评估输入。一个结合深度学习和动态结构的简单方法是训练一个决策树，其中每个节点使用神经网络做出决策 (Guo and Gelfand, 1992)，虽然这种方法没有实现加速推断计算的目标。

类似的，我们可以使用称为 **选通器** (gater) 的神经网络来选择在给定当前输入的情况下将使用几个 **专家网络** (expert network) 中的哪一个来计算输出。这个想法的第一个版本被称为 **专家混合体** (mixture of experts) (Nowlan, 1990; Jacobs *et al.*, 1991)，其中选通器为每个专家输出一个概率或权重 (通过非线性的softmax 函数获得)，并且最终输出由各个专家输出的加权组合获得。在这种情况下，使用选通器不会降低计算成本，但如果每个样本的选通器选择单个专家，我们就会获得一个特殊的**硬专家混合体** (hard mixture of experts) (Collobert *et al.*, 2001, 2002)，这可以加速推断和训练。当选通器决策的数量很小时，这个策略效果会很好，因为它不是组合的。但是当我们想要选择不同的单元或参数子集时，不可能使用“软开关”，因为它需要枚举 (和计算输出) 所有的选通器配置。为了解决这个问题，许多工作探索了几种方法来训练组合的选通器。Bengio *et al.* (2013b) 提出使用选通器概率梯度的若干估计器，而 Bacon *et al.* (2015); Bengio *et al.* (2015a) 使用强化学习技术 (**策略梯度** (policy gradient)) 来学习一种条件的 Dropout 形式 (作用于隐藏单元块)，减少了实际的计算成本，而不会对近似的质量产生负面影响。

另一种动态结构是开关，其中隐藏单元可以根据具体情况从不同单元接收输入。这种动态路由方法可以理解为 **注意力机制** (attention mechanism) (Olshausen *et al.*, 1993)。目前为止，硬性开关的使用在大规模应用中还没有被证明是有效的。较为先进的方法一般采用对许多可能的输入使用加权平均，因此不能完全得到动态结构所带来的计算益处。先进的注意力机制将在第 12.4.5.1 节中描述。

使用动态结构化系统的主要障碍是由于系统针对不同输入的不同代码分支导致的并行度降低。这意味着网络中只有很少的操作可以被描述为对样本小批量的矩阵乘法或批量卷积。我们可以写更多的专用子程序，用不同的核对样本做卷积，或者通过不同的权重列来乘以设计矩阵的每一行。不幸的是，这些专用的子程序难以高效地实现。由于缺乏高速缓存的一致性，CPU 实现会十分缓慢。此外，由于缺乏级联的内存操作以及 warp 成员使用不同分支时需要串行化操作，GPU 的实现也会很慢。在一些情况下，我们可以通过将样本分成组，并且都采用相同的分支并且同时处理这些样本组的方式来缓解这些问题。在离线环境中，这是最小化处理固定量样本所需时间的一项可接受的策略。然而在实时系统中，样本必须连续处理，对工作

负载进行分区可能会导致负载均衡问题。例如，如果我们分配一台机器处理级联中的第一步，另一台机器处理级联中的最后一步，那么第一台机器将倾向于过载，最后一个机器倾向于欠载。如果每个机器被分配以实现神经决策树的不同节点，也会出现类似的问题。

### 12.1.6 深度网络的专用硬件实现

自从早期的神经网络研究以来，硬件设计者已经致力于可以加速神经网络算法的训练和/或推断的专用硬件实现。读者可以查看早期和更近的专用硬件深度网络的评论 (Lindsey and Lindblad, 1994; Beiu *et al.*, 2003; Misra and Saha, 2010)。

不同形式的专用硬件 (Graf and Jackel, 1989; Mead and Ismail, 2012; Kim *et al.*, 2009; Pham *et al.*, 2012; Chen *et al.*, 2014b,a) 的研究已经持续了好几十年，比如**专用集成电路** (application-specific integrated circuit, ASIC) 的数字 (基于数字的二进制表示)，模拟 (Graf and Jackel, 1989; Mead and Ismail, 2012) (基于以电压或电流表示连续值的物理实现) 和混合实现 (组合数字和模拟组件)。近年来更灵活的**现场可编程门阵列** (field programmable gated array, FPGA) 实现 (其中电路的具体细节可以在制造完成后写入芯片) 也得到了长足发展。

虽然 CPU 和 GPU 上的软件实现通常使用 32 或 64 位的精度来表示浮点数，但是长期以来使用较低的精度在更短的时间内完成推断也是可行的 (Holt and Baker, 1991; Holi and Hwang, 1993; Presley and Haggard, 1994; Simard and Graf, 1994; Wawrynek *et al.*, 1996; Savich *et al.*, 2007)。这已成为近年来更迫切的问题，因为深度学习在工业产品中越来越受欢迎，并且由于更快的硬件产生的巨大影响已经通过 GPU 的使用得到了证明。激励当前对深度网络专用硬件研究的另一个因素是单个 CPU 或 GPU 核心的进展速度已经减慢，并且最近计算速度的改进来自于核心的并行化 (无论 CPU 还是 GPU)。这与 20 世纪 90 年代的情况 (上一个神经网络时代) 的不同之处在于，神经网络的硬件实现 (从开始到芯片可用可能需要两年) 跟不上快速进展和价格低廉的通用 CPU 的脚步。因此，在针对诸如手机等低功率设备开发新的硬件设计，并且想要用于深度学习的一般公众应用 (例如，具有语音、计算机视觉或自然语言功能的设施) 等时，研究专用硬件能够进一步推动其发展。

最近对基于反向传播神经网络的低精度实现的工作 (Vanhoucke *et al.*, 2011; Courbariaux *et al.*, 2015; Gupta *et al.*, 2015) 表明，8 和 16 位之间的精度足以满足使用或训练基于反向传播的深度神经网络的要求。显而易见的是，在训练期间需要

比在推断时更高的精度，并且数字某些形式的动态定点表示能够减少每个数需要的存储空间。传统的定点数被限制在了一个固定范围之内（其对应于浮点表示中的给定指数）。而动态定点表示在一组数字（例如一个层中的所有权重）之间共享该范围。使用定点代替浮点表示并且每个数使用较少的比特能够减少执行乘法所需的硬件表面积、功率需求和计算时间。而乘法已经是使用或训练反向传播的现代深度网络中要求最高的操作。

## 12.2 计算机视觉

一直以来，计算机视觉就是深度学习应用中几个最活跃的研究方向之一。因为视觉是一个对人类以及许多动物毫不费力，但对计算机却充满挑战的任务 (Ballard *et al.*, 1983)。深度学习中许多流行的标准基准任务包括对象识别以及光学字符识别。

计算机视觉是一个非常广阔的发展领域，其中包括多种多样的处理图片的方式以及应用方向。计算机视觉的应用广泛：从复现人类视觉能力（比如识别人脸）到创造全新的视觉能力。举个后者的例子，近期一个新的计算机视觉应用是从视频中可视物体的振动中识别相应的声波 (Davis *et al.*, 2014)。大多数计算机视觉领域的深度学习研究未曾关注过这样一个奇异的应用，它扩展了图像的范围，而不是仅仅关注于人工智能中较小的核心目标——复制人类的能力。无论是报告图像中存在哪个物体，还是给图像中每个对象周围添加注释性的边框，或从图像中转录符号序列，或给图像中的每个像素标记它所属对象的标识，大多数计算机视觉中的深度学习往往用于对象识别或者某种形式的检测。由于生成模型已经是深度学习研究的指导原则，因此还有大量图像合成工作使用了深度模型。尽管图像合成（“无中生有”）通常不包括在计算机视觉内，但是能够进行图像合成的模型通常用于图像恢复，即修复图像中的缺陷或从图像中移除对象这样的计算机视觉任务。

### 12.2.1 预处理

由于原始输入往往以深度学习架构难以表示的形式出现，许多应用领域需要复杂精细的预处理。计算机视觉通常只需要相对少的这种预处理。图像应该被标准化，从而使得它们的像素都在相同并且合理的范围内，比如  $[0, 1]$  或者  $[-1, 1]$ 。将  $[0, 1]$  中的图像与  $[0, 255]$  中的图像混合通常会导致失败。将图像格式化为具有相同的比例严格上说是唯一一种必要的预处理。许多计算机视觉架构需要标准尺寸的图像，

因此必须裁剪或缩放图像以适应该尺寸。然而，严格地说即使是这种重新调整比例的操作并不总是必要的。一些卷积模型接受可变大小的输入并动态地调整它们的池化区域大小以保持输出大小恒定 (Waibel *et al.*, 1989)。其他卷积模型具有可变大小的输出，其尺寸随输入自动缩放，例如对图像中的每个像素进行去噪或标注的模型 (Hadsell *et al.*, 2007)。

数据集增强可以被看作是一种只对训练集做预处理的方式。数据集增强是减少大多数计算机视觉模型泛化误差的一种极好方法。在测试时可用的一个相关想法是将同一输入的许多不同版本传给模型（例如，在稍微不同的位置处裁剪的相同图像），并且在模型的不同实例上决定模型的输出。后一个想法可以被理解为集成方法，并且有助于减少泛化误差。

其他种类的预处理需要同时应用于训练集和测试集，其目的是将每个样本置于更规范的形式，以便减少模型需要考虑的变化量。减少数据中的变化量既能够减少泛化误差，也能够减小拟合训练集所需模型的大小。更简单的任务可以通过更小的模型来解决，而更简单的解决方案泛化能力一般更好。这种类型的预处理通常被设计为去除输入数据中的某种可变性，这对于人工设计者来说是容易描述的，并且人工设计者能够保证不受到任务影响。当使用大型数据集和大型模型训练时，这种预处理通常是不必要的，并且最好只是让模型学习哪些变化性应该保留。例如，用于分类 ImageNet 的 AlexNet 系统仅具有一个预处理步骤：对每个像素减去训练样本的平均值 (Krizhevsky *et al.*, 2012b)。

### 12.2.1.1 对比度归一化

在许多任务中，对比度是能够安全移除的最为明显的变化源之一。简单地说，对比度指的是图像中亮像素和暗像素之间差异的大小。量化图像对比度有许多方式。在深度学习中，对比度通常指的是图像或图像区域中像素的标准差。假设我们有一个张量表示的图像  $\mathbf{X} \in \mathbb{R}^{r \times c \times 3}$ ，其中  $X_{i,j,1}$  表示第  $i$  行第  $j$  列红色的强度， $X_{i,j,2}$  对应的是绿色的强度， $X_{i,j,3}$  对应的是蓝色的强度。然后整个图像的对比度可以表示如下：

$$\sqrt{\frac{1}{3rc} \sum_{i=1}^r \sum_{j=1}^c \sum_{k=1}^3 (X_{i,j,k} - \bar{\mathbf{X}})^2} \quad (12.1)$$

其中  $\bar{\mathbf{X}}$  是整个图片的平均强度，满足

$$\bar{\mathbf{X}} = \frac{1}{3rc} \sum_{i=1}^r \sum_{j=1}^c \sum_{k=1}^3 X_{i,j,k}. \quad (12.2)$$

**全局对比度归一化** (Global contrast normalization, GCN) 旨在通过从每个图像中减去其平均值，然后重新缩放其使得其像素上的标准差等于某个常数  $s$  来防止图像具有变化的对比度。这种方法非常复杂，因为没有缩放因子可以改变零对比度图像（所有像素都具有相等强度的图像）的对比度。具有非常低但非零对比度的图像通常几乎没有信息内容。在这种情况下除以真实标准差通常仅能放大传感器噪声或压缩伪像。这种现象启发我们引入一个正的正则化参数  $\lambda$  来平衡估计的标准差。或者，我们至少可以约束分母使其大于等于  $\epsilon$ 。给定一个输入图像  $\mathbf{X}$ ，全局对比度归一化产生输出图像  $\mathbf{X}'$ ，定义为

$$X'_{i,j,k} = s \frac{X_{i,j,k} - \bar{X}}{\max\{\epsilon, \sqrt{\lambda + \frac{1}{3rc} \sum_{i=1}^r \sum_{j=1}^c \sum_{k=1}^3 (X_{i,j,k} - \bar{X})^2}\}}. \quad (12.3)$$

从大图像中剪切感兴趣的对象所组成的数据集不可能包含任何强度几乎恒定的图像。在这些情况下，通过设置  $\lambda = 0$  来忽略小分母问题是安全的，并且在非常罕见的情况下为了避免除以 0，通过将  $\epsilon$  设置为一个非常小的值比如说  $10^{-8}$ 。这也是 Goodfellow et al. (2013c) 在 CIFAR-10 数据集上所使用的方法。随机剪裁的小图像更可能具有几乎恒定的强度，使得激进的正则化更有用。在处理从 CIFAR-10 数据中随机选择的小区域时，Coates et al. (2011) 使用  $\epsilon = 0, \lambda = 10$ 。

尺度参数  $s$  通常可以设置为 1 (如 Coates et al. (2011) 所采用的)，或选择使所有样本上每个像素的标准差接近 1 (如 Goodfellow et al. (2013c) 所采用的)。

式 (12.3) 中的标准差仅仅是对图片  $L^2$  范数的重新缩放 (假设图像的平均值已经被移除)。我们更偏向于根据标准差而不是  $L^2$  范数来定义 GCN，因为标准差包括除以像素数量这一步，从而基于标准差的 GCN 能够使用与图像大小无关的固定的  $s$ 。然而，观察到  $L^2$  范数与标准差成比例，这符合我们的直觉。我们可以把 GCN 理解成到球壳的一种映射。图 12.1 对此有所说明。这可能是一个有用的属性，因为神经网络往往更好地响应空间方向，而不是精确的位置。响应相同方向上的多个距离需要具有共线权重向量但具有不同偏置的隐藏单元。这样的情况对于学习算法来说可能是困难的。此外，许多浅层的图模型把多个分离的模式表示在一条线上会出现问题。GCN 采用一个样本一个方向<sup>1</sup>而不是不同的方向和距离来避免这些问题。

<sup>1</sup>译者：所有样本相似的距离

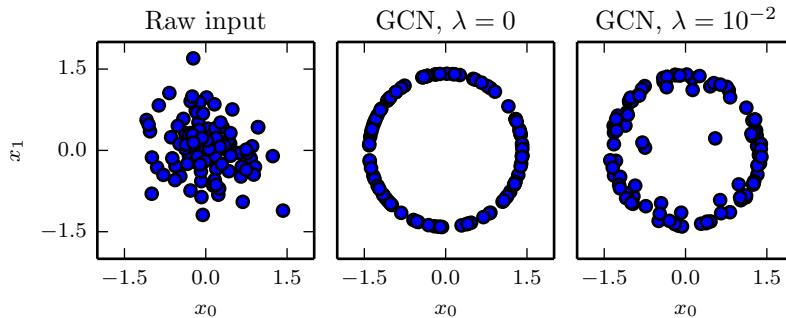


图 12.1: GCN 将样本投影到一个球上。(左) 原始的输入数据可能拥有任意的范数。(中) $\lambda = 0$  时候的 GCN 可以完美地将所有的非零样本投影到球上。这里我们令  $s = 1$ ,  $\epsilon = 10^{-8}$ 。由于我们使用的 GCN 是基于归一化标准差而不是  $L^2$  范数, 所得到的球并不是单位球。(右) $\lambda > 0$  的正则化 GCN 将样本投影到球上, 但是并没有完全地丢弃其范数中变化。 $s$  和  $\epsilon$  的取值与之前一样。

与直觉相反的是, 存在被称为 **sphering** 的预处理操作, 并且它不同于 GCN。sphering 并不会使数据位于球形壳上, 而是将主成分重新缩放以具有相等方差, 使得 PCA 使用的多变量正态分布具有球形等高线。sphering 通常被称为白化 (whitening)。

全局对比度归一化常常不能突出我们想要突出的图像特征, 例如边缘和角。如果我们有一个场景, 包含了一个大的黑暗区域和一个大的明亮的区域 (例如一个城市广场有一半的区域处于建筑物的阴影之中), 则全局对比度归一化将确保暗区域的亮度与亮区域的亮度之间存在大的差异。然而, 它不能确保暗区内的边缘突出。

这催生了 **局部对比度归一化** (local contrast normalization, LCN)。局部对比度归一化确保对比度在每个小窗口上被归一化, 而不是作为整体在图像上被归一化。关于局部对比度归一化和全局对比度归一化的比较可以参考图 12.2。

局部对比度归一化的各种定义都是可行的。在所有情况下, 我们可以通过减去邻近像素的平均值并除以邻近像素的标准差来修改每个像素。在一些情况下, 要计算以当前要修改的像素为中心的矩形窗口中所有像素的平均值和标准差 (Pinto *et al.*, 2008)。在其他情况下, 使用的则是以要修改的像素为中心的高斯权重的加权平均和加权标准差。在彩色图像的情况下, 一些策略单独处理不同的颜色通道, 而其他策略组合来自不同通道的信息以使每个像素归一化 (Sermanet *et al.*, 2012)。

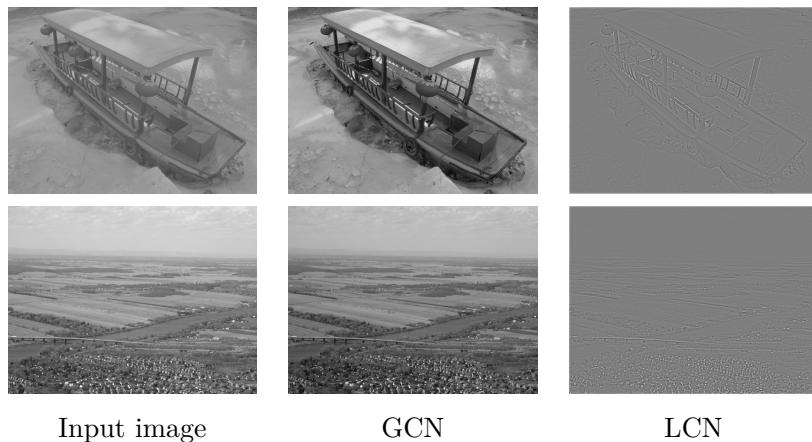


图 12.2: 全局对比度归一化和局部对比度归一化的比较。直观上说，全局对比度归一化的效果很巧妙。它使得所有的图片的尺度都差不多，这减轻了学习算法处理多个尺度的负担。局部对比度归一化更多地改变了图像，丢弃了所有相同强度的区域。这使得模型能够只关注于边缘。较好的纹理区域，如第二行的屋子，可能会由于归一化核的过高带宽而丢失一些细节。

局部对比度归一化通常可以通过使用可分离卷积（参考第 9.8 节）来计算特征映射的局部平均值和局部标准差，然后在不同的特征映射上使用逐元素的减法和除法。

局部对比度归一化是可微分的操作，并且还可以作为一种非线性作用应用于网络隐藏层，以及应用于输入的预处理操作。

与全局对比度归一化一样，我们通常需要正则化局部对比度归一化来避免出现除以零的情况。事实上，因为局部对比度归一化通常作用于较小的窗口，所以正则化更加重要。较小的窗口更可能包含彼此几乎相同的值，因此更可能具有零标准差。

## 12.2.2 数据集增强

如第 7.4 节中讲到的一样，我们很容易通过增加训练集的额外副本本来增加训练集的大小，进而改进分类器的泛化能力。这些额外副本可以通过对原始图像进行一些变化来生成，但是并不改变其类别。对象识别这个分类任务特别适合于这种形式的数据集增强，因为类别信息对于许多变换是不变的，而我们可以简单地对输入应用诸多几何变换。如前所述，分类器可以受益于随机转换或者旋转，某些情况下输入的翻转可以增强数据集。在专门的计算机视觉应用中，存在很多更高级的用以数据集增强的变换。这些方案包括图像中颜色的随机扰动 (Krizhevsky *et al.*, 2012b),

以及对输入的非线性几何变形 (LeCun *et al.*, 1998c)。

## 12.3 语音识别

语音识别任务在于将一段包括了自然语言发音的声学信号投影到对应说话人的词序列上。令  $\mathbf{X} = (\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(T)})$  表示语音的输入向量 (传统做法以 20ms 为一帧分割信号)。许多语音识别的系统通过特殊的手工设计方法预处理输入信号, 从而提取特征, 但是某些深度学习系统 (Jaity and Hinton, 2011) 直接从原始输入中学习特征。令  $\mathbf{y} = (y_1, y_2, \dots, y_N)$  表示目标的输出序列 (通常是一个词或者字符的序列)。自动语音识别 (Automatic Speech Recognition, ASR) 任务指的是构造一个函数  $f_{\text{ASR}}^*$ , 使得它能够在给定声学序列  $\mathbf{X}$  的情况下计算最有可能的语言序列  $\mathbf{y}$ :

$$f_{\text{ASR}}^*(\mathbf{X}) = \arg \max_{\mathbf{y}} P^*(\mathbf{y} \mid \mathbf{X} = \mathbf{X}), \quad (12.4)$$

其中  $P^*$  是给定输入值  $\mathbf{X}$  时对应目标  $\mathbf{y}$  的真实条件分布。

从 20 世纪 80 年代直到约 2009-2012 年, 最先进的语音识别系统是 **隐马尔可夫模型** (Hidden Markov Model, HMM) 和 **高斯混合模型** (Gaussian Mixture Model, GMM) 的结合。GMM 对声学特征和 **音素** (phoneme) 之间的关系建模 (Bahl *et al.*, 1987), HMM 对音素序列建模。GMM-HMM 模型将语音信号视作由如下过程生成: 首先, 一个 HMM 生成了一个音素的序列以及离散的子音素状态 (比如每一个音素的开始, 中间, 结尾), 然后 GMM 把每一个离散的状态转化为一个简短的声音信号。尽管直到最近 GMM-HMM 一直在 ASR 中占据主导地位, 语音识别仍然是神经网络所成功应用的第一个领域。从 20 世纪 80 年代末期到 90 年代初期, 大量语音识别系统使用了神经网络 (Bourlard and Wellekens, 1989; Waibel *et al.*, 1989; Robinson and Fallside, 1991; Bengio *et al.*, 1991, 1992; Konig *et al.*, 1996)。当时, 基于神经网络的 ASR 的表现和 GMM-HMM 系统的表现差不多。比如说, Robinson and Fallside (1991) 在 TIMIT 数据集 (Garofolo *et al.*, 1993) (有 39 个区分的音素) 上达到了 26% 的音素错误率, 这个结果优于或者说是可以与基于 HMM 的结果相比。从那时起, TIMIT 成为了音素识别的一个基准数据集, 在语音识别中的作用就和 MNIST 在对象识别中的作用差不多。然而, 由于语音识别软件系统中复杂的工程因素以及在基于 GMM-HMM 的系统中已经付出的巨大努力, 工业界并没有迫切转向神经网络的需求。结果, 直到 21 世纪 00 年代末期, 学术界和工业界的研究者们更多的是用神经网络为 GMM-HMM 系统学习一些额外的特征。

之后，随着更大更深的模型以及更大的数据集的出现，通过使用神经网络代替 GMM 来实现将声学特征转化为音素（或者子音素状态）的过程可以大大地提高识别的精度。从 2009 年开始，语音识别的研究者们将一种无监督学习的深度学习方法应用于语音识别。这种深度学习方法基于训练一个被称作是受限玻尔兹曼机的无向概率模型，从而对输入数据建模。受限玻尔兹曼机将会在第三部分中描述。为了完成语音识别任务，无监督的预训练被用来构造一个深度前馈网络，这个神经网络每一层都是通过训练受限玻尔兹曼机来初始化的。这些网络的输入是从一个固定规格的输入窗（以当前帧为中心）的谱声学表示抽取，预测了当前帧所对应的 HMM 状态的条件概率。训练一个这样的神经网络能够显著提高在 TIMIT 数据集上的识别率 (Mohamed *et al.*, 2009, 2012a)，并将音素级别的错误率从大约 26% 降到了 20.7%。关于这个模型成功原因的详细分析可以参考 Mohamed *et al.* (2012b)。对于基本的电话识别工作流程的一个扩展工作是添加说话人自适应相关特征 (Mohamed *et al.*, 2011) 的方法，这可以进一步地降低错误率。紧接着的工作则将结构从音素识别 (TIMIT 所主要关注的) 转向了大规模词汇语音识别 (Dahl *et al.*, 2012)，这不仅包含了识别音素，还包括了识别大规模词汇的序列。语音识别上的深度网络从最初的使用受限玻尔兹曼机进行预训练发展到了使用诸如整流线性单元和 Dropout 这样的技术 (Zeiler *et al.*, 2013; Dahl *et al.*, 2013)。从那时开始，工业界的几个语音研究组开始寻求与学术圈的研究者之间的合作。Hinton *et al.* (2012a) 描述了这些合作所带来的突破性进展，这些技术现在被广泛应用在产品中，比如移动手机端。

随后，当研究组使用了越来越大的带标签的数据集，加入了各种初始化，训练方法以及调试深度神经网络的结构之后，他们发现这种无监督的预训练方式是没有必要的，或者说不能带来任何显著的改进。

用语音识别中词错误率来衡量，在语音识别性能上的这些突破是史无前例的（大约 30% 的提高）。在这之前的长达十年左右的时间内，尽管数据集的规模是随时间增长的（见 Deng and Yu (2014) 的图 2.4），但基于 GMM-HMM 的系统的传统技术已经停滞不前了。这也导致了语音识别领域快速地转向深度学习的研究。在大约的两年时间内，工业界的大多数的语音识别产品都包含了深度神经网络，这种成功也激发了 ASR 领域对深度学习算法和结构的一波新的研究浪潮，并且影响至今。

其中的一个创新点是卷积网络的应用 (Sainath *et al.*, 2013)。卷积网络在时域与频域上复用了权重，改进了之前的仅在时域上使用重复权值的时延神经网络。这种新的二维的卷积模型并不是将输入的频谱当作一个长的向量，而是当成是一个图像，其中一个轴对应着时间，另一个轴对应的是谱分量的频率。

完全抛弃 HMM 并转向研究端到端的深度学习语音识别系统是至今仍然活跃的另一个重要推动。这个领域第一个主要的突破是 Graves *et al.* (2013)，其中训练了一个深度的长短期记忆循环神经网络（见第 10.10 节），使用了帧—音素排列的 MAP 推断，就像 LeCun *et al.* (1998c) 以及 CTC 框架 (Graves *et al.*, 2006; Graves, 2012) 中一样。一个深度循环神经网络 (Graves *et al.*, 2013) 每个时间步的各层都有状态变量，两种展开图的方式导致两种不同深度：一种是普通的根据层的堆叠衡量的深度，另一种根据时间展开衡量的深度。这个工作把 TIMIT 数据集上音素的错误率记录降到了的新低 17.7%。关于应用于其他领域的深度循环神经网络的变种可以参考 Pascanu *et al.* (2014a); Chung *et al.* (2014)。

另一个端到端的深度学习语音识别方向的最新方法是让系统学习如何利用语音 (phonetic) 层级的信息“排列”声学 (acoustic) 层级的信息 (Chorowski *et al.*, 2014; Lu *et al.*, 2015)。

## 12.4 自然语言处理

自然语言处理 (Natural Language Processing) 让计算机能够使用人类语言，例如英语或法语。为了让简单的程序能够高效明确地解析，计算机程序通常读取和发出特殊化的语言。而自然的语言通常是模糊的，并且可能不遵循形式的描述。自然语言处理中的应用如机器翻译，学习者需要读取一种人类语言的句子，并用另一种人类语言发出等同的句子。许多 NLP 应用程序基于语言模型，语言模型定义了关于自然语言中的字、字符或字节序列的概率分布。

与本章讨论的其他应用一样，非常通用的神经网络技术可以成功地应用于自然语言处理。然而，为了实现卓越的性能并扩展到大型应用程序，一些领域特定的策略也很重要。为了构建自然语言的有效模型，通常必须使用专门处理序列数据的技术。在很多情况下，我们将自然语言视为一系列词，而不是单个字符或字节序列。因为可能的词总数非常大，基于词的语言模型必须在极高维度和稀疏的离散空间上操作。为使这种空间上的模型在计算和统计意义上都高效，研究者已经开发了几种策略。

### 12.4.1 $n$ -gram

语言模型 (language model) 定义了自然语言中标记序列的概率分布。根据模型的设计，标记可以是词、字符、甚至是字节。标记总是离散的实体。最早成功的语言

模型基于固定长度序列的标记模型，称为  $n$ -gram。一个  $n$ -gram 是一个包含  $n$  个标记的序列。

基于  $n$ -gram 的模型定义一个条件概率——给定前  $n - 1$  个标记后的第  $n$  个标记的条件概率。该模型使用这些条件分布的乘积定义较长序列的概率分布：

$$P(x_1, \dots, x_\tau) = P(x_1, \dots, x_{n-1}) \prod_{t=n}^{\tau} P(x_t | x_{t-n+1}, \dots, x_{t-1}). \quad (12.5)$$

这个分解可以由概率的链式法则证明。初始序列  $P(x_1, \dots, x_{n-1})$  的概率分布可以通过带有较小  $n$  值的不同模型建模。

训练  $n$ -gram 模型是简单的，因为最大似然估计可以通过简单地统计每个可能的  $n$ -gram 在训练集中出现的次数来获得。几十年来，基于  $n$ -gram 的模型都是统计语言模型的核心模块 (Jelinek and Mercer, 1980; Katz, 1987; Chen and Goodman, 1999)。

对于小的  $n$  值，模型有特定的名称： $n = 1$  称为一元语法 (unigram)， $n = 2$  称为二元语法 (bigram) 及  $n = 3$  称为三元语法 (trigram)。这些名称源于相应数字的拉丁前缀和希腊后缀 “-gram”，分别表示所写之物。

通常我们同时训练  $n$ -gram 模型和  $n - 1$  gram 模型。这使得下式可以简单地通过查找两个存储的概率来计算。

$$P(x_t | x_{t-n+1}, \dots, x_{t-1}) = \frac{P_n(x_{t-n+1}, \dots, x_t)}{P_{n-1}(x_{t-n+1}, \dots, x_{t-1})} \quad (12.6)$$

为了在  $P_n$  中精确地再现推断，我们训练  $P_{n-1}$  时必须省略每个序列最后一个字符。

举个例子，我们演示三元模型如何计算句子 “THE DOG RAN AWAY.” 的概率。句子的第一个词不能通过上述条件概率的公式计算，因为句子的开头没有上下文。取而代之，在句子的开头我们必须使用词的边缘概率。因此我们计算  $P_3(\text{THE DOG RAN})$ 。最后，可以使用条件分布  $P(\text{AWAY} | \text{DOG RAN})$  (典型情况) 来预测最后一个词。将这与式(12.6)放在一起，我们得到：

$$P(\text{THE DOG RAN AWAY}) = P_3(\text{THE DOG RAN})P_3(\text{DOG RAN AWAY})/P_2(\text{DOG RAN}). \quad (12.7)$$

$n$ -gram 模型最大似然的基本限制是，在许多情况下从训练集计数估计得到的  $P_n$  很可能为零（即使元组  $(x_{t-n+1}, \dots, x_t)$  可能出现在测试集中）。这可能会导致两种不同的灾难性后果。当  $P_{n-1}$  为零时，该比率是未定义的，因此模型甚至不能

产生有意义的输出。当  $P_{n-1}$  非零而  $P_n$  为零时，测试样本的对数似然为  $-\infty$ 。为避免这种灾难性的后果，大多数  $n$ -gram 模型采用某种形式的平滑 (smoothing)。平滑技术将概率质量从观察到的元组转移到类似的未观察到的元组。见 Chen and Goodman (1999) 的综述和实验对比。其中一种基本技术基于向所有可能的下一个符号值添加非零概率质量。这个方法可以被证明是，计数参数具有均匀或 Dirichlet 先验的贝叶斯推断。另一个非常流行的想法是包含高阶和低阶  $n$ -gram 模型的混合模型，其中高阶模型提供更多的容量，而低阶模型尽可能地避免零计数。如果上下文  $x_{t-n+k}, \dots, x_{t-1}$  的频率太小而不能使用高阶模型，回退方法 (back-off methods) 就查找低阶  $n$ -gram。更正式地说，它们通过上下文  $x_{t-n+k}, \dots, x_{t-1}$  估计  $x_t$  上的分布，并增加  $k$  直到找到足够可靠的估计。

经典的  $n$ -gram 模型特别容易引起维数灾难。因为存在  $|\mathcal{V}|^n$  可能的  $n$ -gram，而且  $|\mathcal{V}|$  通常很大。即使有大量训练数据和适当的  $n$ ，大多数  $n$ -gram 也不会出现在训练集中。经典  $n$ -gram 模型的一种观点是执行最近邻查询。换句话说，它可以被视为局部非参数预测器，类似于  $k$ -最近邻。这些极端局部预测器面临的统计问题已经在第 5.11.2 节中描述过。语言模型的问题甚至比普通模型更严重，因为任何两个不同的词在 one-hot 向量空间中的距离彼此相同。因此，难以大量利用来自任意“邻居”的信息——只有重复相同上下文的训练样本对局部泛化有用。为了克服这些问题，语言模型必须能够在同一个词和其他语义相似的词之间共享知识。

为了提高  $n$ -gram 模型的统计效率，**基于类的语言模型** (class-based language model) (Brown *et al.*, 1992; Ney and Kneser, 1993; Niesler *et al.*, 1998) 引入词类别的概念，然后属于同一类别的词共享词之间的统计强度。这个想法使用了聚类算法，基于它们与其他词同时出现的频率，将该组词分成集群或类。随后，模型可以在条件竖杠的右侧使用词类 ID 而不是单个词 ID。混合 (或回退) 词模型和类模型的复合模型也是可能的。尽管词类提供了在序列之间泛化的方式，但其中一些词被相同类的另一个替换，导致该表示丢失了很多信息。

### 12.4.2 神经语言模型

**神经语言模型** (Neural Language Model, NLM) 是一类用来克服维数灾难的语言模型，它使用词的分布式表示对自然语言序列建模 (Bengio *et al.*, 2001b)。不同于基于类的  $n$ -gram 模型，神经语言模型在能够识别两个相似的词，并且不丧失将每个词编码为彼此不同的能力。神经语言模型共享一个词 (及其上下文) 和其他类似词

(和上下文之间)的统计强度。模型为每个词学习的分布式表示，允许模型处理具有类似共同特征的词来实现这种共享。例如，如果词 `dog` 和词 `cat` 映射到具有许多属性的表示，则包含词 `cat` 的句子可以告知模型对包含词 `dog` 的句子做出预测，反之亦然。因为这样的属性很多，所以存在许多泛化的方式，可以将信息从每个训练语句传递到指数数量的语义相关语句。维数灾难需要模型泛化到指数多的句子（指数相对句子长度而言）。该模型通过将每个训练句子与指数数量的类似句子相关联克服这个问题。

我们有时将这些词表示称为 **词嵌入** (word embedding)。在这个解释下，我们将原始符号视为维度等于词表大小的空间中的点。词表示将这些点嵌入到较低维的特征空间中。在原始空间中，每个词由一个one-hot向量表示，因此每对词彼此之间的欧氏距离都是  $\sqrt{2}$ 。在嵌入空间中，经常出现在类似上下文（或共享由模型学习的一些“特征”的任何词对）中的词彼此接近。这通常导致具有相似含义的词变得邻近。图 12.3 放大了学到的词嵌入空间的特定区域，我们可以看到语义上相似的词如何映射到彼此接近的表示。

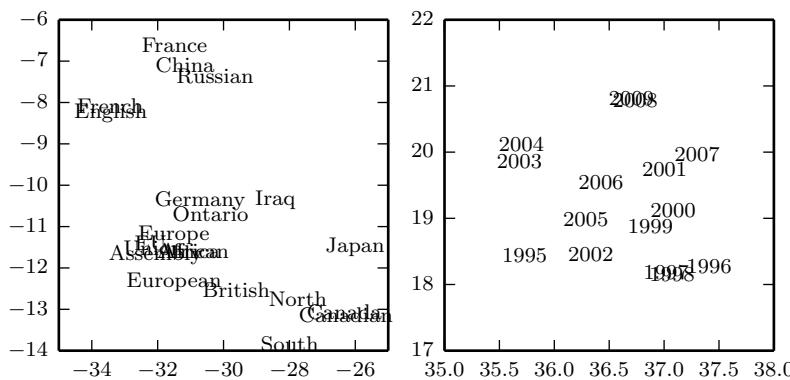


图 12.3: 从神经机器翻译模型获得的词嵌入的二维可视化 (Bahdanau *et al.*, 2015)。此图在语义相关词的特定区域放大，它们具有彼此接近的嵌入向量。国家在左图，数字在右图。注意，这些嵌入是为了可视化才表示为 2 维。在实际应用中，嵌入通常具有更高的维度并且可以同时捕获词之间多种相似性。

其他领域的神经网络也可以定义嵌入。例如，卷积网络的隐藏层提供“图像嵌入”。因为自然语言最初不在实值向量空间上，所以 NLP 从业者通常对嵌入的这个想法更感兴趣。隐藏层在表示数据的方式上提供了更质变的戏剧性变化。

使用分布式表示来改进自然语言处理模型的基本思想不必局限于神经网络。它还可以用于图模型，其中分布式表示是多个潜变量的形式 (Mnih and Hinton, 2007)。

### 12.4.3 高维输出

在许多自然语言应用中，我们通常希望我们的模型产生词（而不是字符）作为输出的基本单位。对于大词汇表，由于词汇量很大，在词的选择上表示输出分布的计算成本可能非常高。在许多应用中， $\mathbb{V}$  包含数十万词。表示这种分布的朴素方法是应用一个仿射变换，将隐藏表示转换到输出空间，然后应用 softmax 函数。假设我们的词汇表  $\mathbb{V}$  大小为  $|\mathbb{V}|$ 。因为其输出维数为  $|\mathbb{V}|$ ，描述该仿射变换线性分量的权重矩阵非常大。这造成了表示该矩阵的高存储成本，以及与之相乘的高计算成本。因为 softmax 要在所有  $|\mathbb{V}|$  输出之间归一化，所以在训练时以及测试时执行全矩阵乘法是必要的——我们不能仅计算与正确输出的权重向量的点积。因此，输出层的高计算成本在训练期间（计算似然性及其梯度）和测试期间（计算所有或所选词的概率）都有出现。对于专门的损失函数，可以有效地计算梯度 (Vincent *et al.*, 2015)，但是应用于传统 softmax 输出层的标准交叉熵损失时会出现许多困难。

假设  $\mathbf{h}$  是用于预测输出概率  $\hat{\mathbf{y}}$  的顶部隐藏层。如果我们使用学到的权重  $\mathbf{W}$  和学到的偏置  $\mathbf{b}$  参数化从  $\mathbf{h}$  到  $\hat{\mathbf{y}}$  的变换，则仿射 softmax 输出层执行以下计算：

$$a_i = b_i + \sum_j W_{ij} h_j \quad \forall i \in \{1, \dots, |\mathbb{V}|\}, \quad (12.8)$$

$$\hat{y}_i = \frac{e^{a_i}}{\sum_{i'=1}^{|\mathbb{V}|} e^{a_{i'}}}. \quad (12.9)$$

如果  $\mathbf{h}$  包含  $n_h$  个元素，则上述操作复杂度是  $O(|\mathbb{V}|n_h)$ 。在  $n_h$  为数千和  $|\mathbb{V}|$  数十万的情况下，这个操作占据了神经语言模型的大多数计算。

#### 12.4.3.1 使用短列表

第一个神经语言模型 (Bengio *et al.*, 2001b, 2003) 通过将词汇量限制为 10,000 或 20,000 来减轻大词汇表上 softmax 的高成本。Schwenk and Gauvain (2002) 和 Schwenk (2007) 在这种方法的基础上建立新的方式，将词汇表  $\mathbb{V}$  分为最常见词汇（由神经网络处理）的短列表 (shortlist)  $\mathbb{L}$  和较稀有词汇的尾列表  $\mathbb{T} = \mathbb{V} \setminus \mathbb{L}$ （由  $n$ -gram 模型处理）。为了组合这两个预测，神经网络还必须预测在上下文  $C$  之后出现

的词位于尾列表的概率。我们可以添加额外的 sigmoid 输出单元估计  $P(i \in \mathbb{T} | C)$  实现这个预测。额外输出则可以用来估计  $\mathbb{V}$  中所有词的概率分布，如下：

$$\begin{aligned} P(y = i | C) &= 1_{i \in \mathbb{L}} P(y = i | C, i \in \mathbb{L}) (1 - P(i \in \mathbb{T} | C)) \\ &\quad + 1_{i \in \mathbb{T}} P(y = i | C, i \in \mathbb{T}) P(i \in \mathbb{T} | C), \end{aligned} \quad (12.10)$$

其中  $P(y = i | C, i \in \mathbb{L})$  由神经语言模型提供  $P(y = i | C, i \in \mathbb{T})$  由  $n$ -gram 模型提供。稍作修改，这种方法也可以在神经语言模型的 softmax 层中使用额外的输出值，而不是单独的 sigmoid 单元。

短列表方法的一个明显缺点是，神经语言模型的潜在泛化优势仅限于最常用的词，这大概是最没用的。这个缺点引发了处理高维输出替代方法的探索，如下所述。

#### 12.4.3.2 分层 Softmax

减少大词汇表  $\mathbb{V}$  上高维输出层计算负担的经典方法 (Goodman, 2001) 是分层地分解概率。 $|\mathbb{V}|$  因子可以降低到  $\log |\mathbb{V}|$  一样低，而无需执行与  $|\mathbb{V}|$  成比例数量（并且也与隐藏单元数量  $n_h$  成比例）的计算。Bengio (2002) 和 Morin and Bengio (2005) 将这种因子分解方法引入神经语言模型中。

我们可以认为这种层次结构是先建立词的类别，然后是词类别的类别，然后是词类别的类别的类别等等。这些嵌套类别构成一棵树，其叶子为词。在平衡树中，树的深度为  $\log |\mathbb{V}|$ 。选择一个词的概率是由路径（从树根到包含该词叶子的路径）上的每个节点通向该词分支概率的乘积给出。图 12.4 是一个简单的例子。Mnih and Hinton (2009) 也描述了使用多个路径来识别单个词的方法，以便更好地建模具有多个含义的词。计算词的概率则涉及在导向该词所有路径上的求和。

为了预测树的每个节点所需的条件概率，我们通常在树的每个节点处使用逻辑回归模型，并且为所有这些模型提供与输入相同的上下文  $C$ 。因为正确的输出编码在训练集中，我们可以使用监督学习训练逻辑回归模型。我们通常使用标准交叉熵损失，对应于最大化正确判断序列的对数似然。

因为可以高效地计算输出对数似然（低至  $\log |\mathbb{V}|$  而不是  $|\mathbb{V}|$ ），所以也可以高效地计算梯度。这不仅包括关于输出参数的梯度，而且还包括关于隐藏层激活的梯度。

优化树结构最小化期望的计算数量是可能的，但通常不切实际。给定词的相对频率，信息理论的工具可以指定如何选择最佳的二进制编码。为此，我们可以构造树，使得与词相关联的位数量近似等于该词频率的对数。然而在实践中，节省计算通

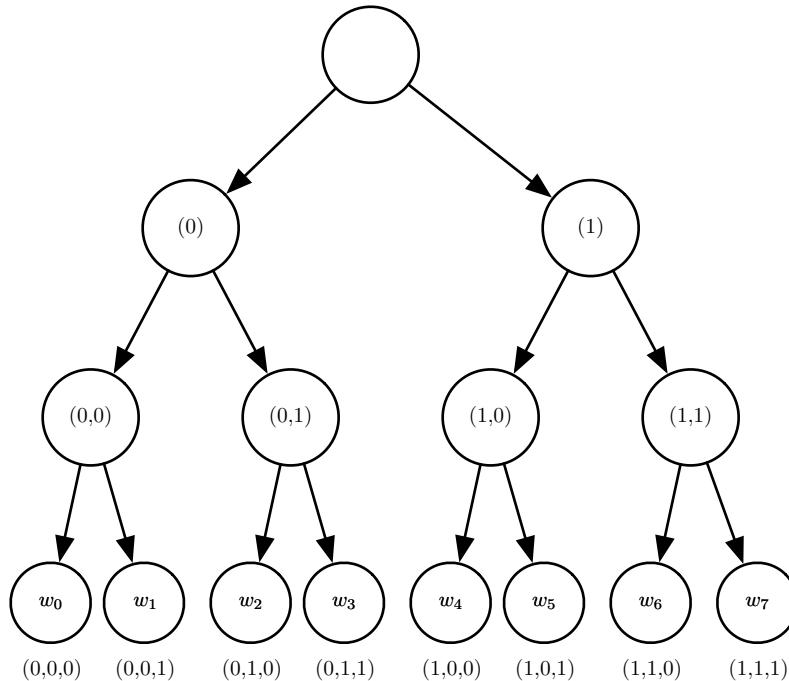


图 12.4: 词类别简单层次结构的示意图, 其中 8 个词  $w_0, \dots, w_7$  组织成三级层次结构。树的叶子表示实际特定的词。内部节点表示词的组别。任何节点都可以通过二值决策序列 ( $0 = \text{左}, 1 = \text{右}$ ) 索引, 从根到达节点。超类  $(0)$  包含类  $(0, 0)$  和  $(0, 1)$ , 其中分别包含词  $\{w_0, w_1\}$  和  $\{w_2, w_3\}$  的集合, 类似地超类  $(1)$  包含类  $(1, 0)$  和  $(1, 1)$ , 分别包含词  $\{w_4, w_5\}$  和  $\{w_6, w_7\}$ 。如果树充分平衡, 则最大深度 (二值决策的数量) 与词数  $|V|$  的对数同阶: 从  $|V|$  个词中选一个词只需执行  $\mathcal{O}(\log |V|)$  次操作 (从根开始的路径上的每个节点一次操作)。在该示例中, 我们乘三次概率就能计算词  $y$  的概率, 这三次概率与从根到节点  $y$  的路径上每个节点向左或向右的二值决策相关联。令  $b_i(y)$  为遍历树移向  $y$  时的第  $i$  个二值决策。对输出  $y$  进行采样的概率可以通过条件概率的链式法则分解为条件概率的乘积, 其中每个节点由这些位的前缀索引。例如, 节点  $(1, 0)$  对应于前缀  $(b_0(w_4) = 1, b_1(w_4) = 0)$ , 并且  $w_4$  的概率可以如下分解:

$$P(y = w_4) = P(b_0 = 1, b_1 = 0, b_2 = 0) \quad (12.11)$$

$$= P(b_0 = 1)P(b_1 = 0 | b_0 = 1)P(b_2 = 0 | b_0 = 1, b_1 = 0). \quad (12.12)$$

常事倍功半，因为输出概率的计算仅是神经语言模型中总计算的一部分。例如，假设有  $l$  个全连接的宽度为  $n_h$  的隐藏层。令  $n_b$  是识别一个词所需比特数的加权平均值，其加权由这些词的频率给出。在这个例子中，计算隐藏激活所需的操作数增长为  $O(ln_h^2)$ ，而输出计算增长为  $O(n_h n_b)$ 。只要  $n_b \leq ln_h$ ，我们可以通过收缩  $n_h$  比收缩  $n_b$  减少更多的计算量。事实上， $n_b$  通常很小。因为词汇表的大小很少超过一千万而  $\log_2(10^6) \approx 20$ ，所以可以将  $n_b$  减小到大约 20，但  $n_h$  通常大得多，大约为  $10^3$  或更大。我们可以定义深度为 2 和分支因子为  $\sqrt{|\mathbb{T}|}$  的树，而不用仔细优化分支因子为 2 的树。这样的树对应于简单定义一组互斥的词类。基于深度为 2 的树的简单方法可以获得层级策略大部分的计算益处。

一个仍然有点开放的问题是如何最好地定义这些词类，或者如何定义一般的词层次结构。早期工作使用现有的层次结构 (Morin and Bengio, 2005)，但也可以理想地与神经语言模型联合学习层次结构。学习层次结构很困难。对数似然的精确优化似乎难以解决，因为词层次的选择是离散的，不适于基于梯度的优化。然而，我们可以使用离散优化来近似地最优化词类的分割。

分层 softmax 的一个重要优点是，它在训练期间和测试期间（如果在测试时我们想计算特定词的概率）都带来了计算上的好处。

当然即使使用分层 softmax，计算所有  $|\mathbb{V}|$  个词概率的成本还是很高的。另一个重要的操作是在给定上下文中选择最可能的词。不幸的是，树结构不能为这个问题提供高效精确的解决方案。

缺点是在实践中，分层 softmax 倾向于更差的测试结果（相对基于采样的方法），我们将在下文描述。这可能是因为词类选择得不好。

#### 12.4.3.3 重要采样

加速神经语言模型训练的一种方式是，避免明确地计算所有未出现在下一位置的词对梯度的贡献。每个不正确的词在此模型下具有低概率。枚举所有这些词的计算成本可能会很高。相反，我们可以仅采样词的子集。使用式 (12.8) 中引入的符号，

梯度可以写成如下形式：

$$\frac{\partial \log P(y | C)}{\partial \theta} = \frac{\partial \log \text{softmax}_y(\mathbf{a})}{\partial \theta} \quad (12.13)$$

$$= \frac{\partial}{\partial \theta} \log \frac{e^{a_y}}{\sum_i e^{a_i}} \quad (12.14)$$

$$= \frac{\partial}{\partial \theta} (a_y - \log \sum_i e^{a_i}) \quad (12.15)$$

$$= \frac{\partial a_y}{\partial \theta} - \sum_i P(y = i | C) \frac{\partial a_i}{\partial \theta}, \quad (12.16)$$

其中  $\mathbf{a}$  是 presoftmax 激活（或得分）向量，每个词对应一个元素。第一项是正相 (positive phase) 项，推动  $a_y$  向上；而第二项是负相 (negative phase) 项，对于所有  $i$  以权重  $P(i | C)$  推动  $a_i$  向下。由于负相项是期望值，我们可以通过蒙特卡罗采样估计。然而，这将需要从模型本身采样。从模型中采样需要对词汇表中所有的  $i$  计算  $P(i | C)$ ，这正是我们试图避免的。

我们可以从另一个分布中采样，而不是从模型中采样，这个分布称为 **提议分布** (proposal distribution) (记为  $q$ )，并通过适当的权重校正从错误分布采样引入的偏差 (Bengio and Sénecal, 2003; Bengio and Sénecal, 2008)。这是一种称为 **重要采样** (Importance Sampling) 的更通用技术的应用，我们将在第 12.4.3.3 节中更详细地描述。不幸的是，即使精确重要采样也不一定有效，因为我们需要计算权重  $p_i/q_i$ ，其中的  $p_i = P(i | C)$  只能在计算所有得分  $a_i$  后才能计算。这个应用采取的解决方案称为有偏重要采样，其中重要性权重被归一化加和为 1。当对负词  $n_i$  进行采样时，相关联的梯度被加权为：

$$w_i = \frac{p_{n_i}/q_{n_i}}{\sum_{j=1}^N p_{n_j}/q_{n_j}}. \quad (12.17)$$

这些权重用于对来自  $q$  的  $m$  个负样本给出适当的重要性，以形成负相估计对梯度的贡献：

$$\sum_{i=1}^{|V|} P(i | C) \frac{\partial a_i}{\partial \theta} \approx \frac{1}{m} \sum_{i=1}^m w_i \frac{\partial a_{n_i}}{\partial \theta}. \quad (12.18)$$

一元语法或二元语法分布与提议分布  $q$  工作得一样好。从数据估计这种分布的参数是很容易。在估计参数之后，也可以非常高效地从这样的分布采样。

**重要采样** (Importance Sampling) 不仅可以加速具有较大 softmax 输出的模型。更一般地，它可以加速具有大稀疏输出层的训练，其中输出是稀疏向量而不是

$n$  选 1。其中一个例子是词袋 (bag of words)。词袋具有稀疏向量  $\mathbf{v}$ , 其中  $v_i$  表示词汇表中的词  $i$  存不存在文档中。或者,  $v_i$  可以指示词  $i$  出现的次数。由于各种原因, 训练产生这种稀疏向量的机器学习模型的成本可能很高。在学习的早期, 模型可能不会真的使输出真正稀疏。此外, 将输出的每个元素与目标的每个元素进行比较, 可能是描述训练的损失函数最自然的方式。这意味着稀疏输出并不一定能带来计算上的好处, 因为模型可以选择使大多数输出非零, 并且所有这些非零值需要与相应的训练目标进行比较 (即使训练目标是零)。Dauphin *et al.* (2011) 证明可以使用重要采样加速这种模型。高效算法最小化“正词”(在目标中非零的那些词)和相等数量的“负词”的重构损失。负词是被随机选取的, 如使用启发式采样更可能被误解的词。该启发式过采样引入的偏差则可以使用重要性权重校正。

在所有这些情况下, 输出层梯度估计的计算复杂度被减少为与负样本数量成比例, 而不是与输出向量的大小成比例。

#### 12.4.3.4 噪声对比估计和排名损失

为减少训练大词汇表的神经语言模型的计算成本, 研究者也提出了其他基于采样的方法。早期的例子是 Collobert and Weston (2008a) 提出的排名损失, 将神经语言模型每个词的输出视为一个得分, 并试图使正确词的得分  $a_y$  比其他词  $a_i$  排名更高。提出的排名损失则是

$$L = \sum_i \max(0, 1 - a_y + a_i). \quad (12.19)$$

如果观察到词的得分  $a_y$  远超过负词的得分  $a_i$  (相差大于 1), 则第  $i$  项梯度为零。这个准则的一个问题是它不提供估计的条件概率, 条件概率在很多应用中是有用的, 包括语音识别和文本生成 (包括诸如翻译的条件文本生成任务)。

最近用于神经语言模型的训练目标是噪声对比估计, 将在第 18.6 节中介绍。这种方法已成功应用于神经语言模型 (Mnih and Teh, 2012; Mnih and Kavukcuoglu, 2013)。

#### 12.4.4 结合 $n$ -gram 和神经语言模型

$n$ -gram 模型相对神经网络的主要优点是  $n$ -gram 模型具有更高的模型容量 (通过存储非常多的元组的频率), 并且处理样本只需非常少的计算量 (通过查找只匹配

当前上下文的几个元组)。如果我们使用哈希表或树来访问计数,那么用于  $n$ -gram 的计算量几乎与容量无关。相比之下,将神经网络的参数数目加倍通常也大致加倍计算时间。当然,避免每次计算时使用所有参数的模型是一个例外。嵌入层每次只索引单个嵌入,所以我们可以增加词汇量,而不会增加每个样本的计算时间。一些其他模型,例如平铺卷积网络,可以在减少参数共享程度的同时添加参数以保持相同的计算量。然而,基于矩阵乘法的典型神经网络层需要与参数数量成比例的计算量。

因此,增加容量的一种简单方法是将两种方法结合,由神经语言模型和  $n$ -gram 语言模型组成集成 (Bengio *et al.*, 2001b, 2003)。

对于任何集成,如果集成成员产生独立的错误,这种技术可以减少测试误差。集成学习领域提供了许多方法来组合集成成员的预测,包括统一加权和在验证集上选择权重。Mikolov *et al.* (2011a) 扩展了集成,不是仅包括两个模型,而是包括大量模型。我们也可以将神经网络与最大熵模型配对并联合训练 (Mikolov *et al.*, 2011b)。该方法可以被视为训练具有一组额外输入的神经网络,额外输入直接连接到输出并且不连接到模型的任何其他部分。额外输入是输入上下文中特定  $n$ -gram 是否存在的指示器,因此这些变量是非常高维且非常稀疏的。

模型容量的增加是巨大的(架构的新部分包含高达  $|sV|^n$  个参数),但是处理输入所需的额外计算量是很小的(因为额外输入非常稀疏)。

#### 12.4.5 神经机器翻译

机器翻译以一种自然语言读取句子并产生等同含义的另一种语言的句子。机器翻译系统通常涉及许多组件。在高层次,一个组件通常会提出许多候选翻译。由于语言之间的差异,这些翻译中的许多翻译是不符合语法的。例如,许多语言在名词后放置形容词,因此直接翻译成英语时,它们会产生诸如“apple red”的短语。提议机制提出建议翻译的许多变体,理想情况下应包括“red apple”。翻译系统的第二个组成部分(语言模型)评估提议的翻译,并可以评估“red apple”比“apple red”更好。

最早的机器翻译神经网络探索中已经纳入了编码器和解码器的想法 (Allen 1987; Chrisman 1991; Forcada and Neco 1997),而翻译中神经网络的第一个大规模有竞争力的用途是通过神经语言模型升级翻译系统的语言模型 (Schwenk *et al.*, 2006; Schwenk, 2010)。之前,大多数机器翻译系统在该组件使用  $n$ -gram 模型。机器翻译中基于  $n$ -gram 的模型不仅包括传统的回退  $n$ -gram 模型 (Jelinek and Mercer, 1980; Katz, 1987; Chen and Goodman, 1999),而且包括最大熵语言模型 (maximum

entropy language models) (Berger *et al.*, 1996), 其中给定上下文中常见的词, affine-softmax 层预测下一个词。

传统语言模型仅仅报告自然语言句子的概率。因为机器翻译涉及给定输入句子产生输出句子, 所以将自然语言模型扩展为条件的是有意义的。如第 6.2.1.1 节所述可以直接地扩展一个模型, 该模型定义某些变量的边缘分布, 以便在给定上下文  $C$  ( $C$  可以是单个变量或变量列表) 的情况下定义该变量的条件分布。Devlin *et al.* (2014) 在一些统计机器翻译的基准中击败了最先进的技术, 他给定源语言中的短语  $s_1, s_2, \dots, s_k$  后使用 MLP 对目标语言的短语  $t_1, t_2, \dots, t_k$  进行评分。这个 MLP 估计  $P(t_1, t_2, \dots, t_k | s_1, s_2, \dots, s_k)$ 。这个 MLP 的估计替代了条件  $n$ -gram 模型提供的估计。

基于 MLP 方法的缺点是需要将序列预处理为固定长度。为了使翻译更加灵活, 我们希望模型允许可变的输入长度和输出长度。RNN 具备这种能力。第 10.2.4 节描述了给定某些输入后, 关于序列条件分布 RNN 的几种构造方法, 并且第 10.4 节描述了当输入是序列时如何实现这种条件分布。在所有情况下, 一个模型首先读取输入序列并产生概括输入序列的数据结构。我们称这个概括为“上下文”  $C$ 。上下文  $C$  可以是向量列表, 或者向量或张量。读取输入以产生  $C$  的模型可以是 RNN (Cho *et al.*, 2014b; Sutskever *et al.*, 2014; Jean *et al.*, 2014) 或卷积网络 (Kalchbrenner and Blunsom, 2013)。另一个模型 (通常是 RNN), 则读取上下文  $C$  并且生成目标语言的句子。在图 12.5 中展示了这种用于机器翻译的编码器-解码器框架的总体思想。

为生成以源句为条件的整句, 模型必须具有表示整个源句的方式。早期模型只能表示单个词或短语。从表示学习的观点来看, 具有相同含义的句子具有类似表示是有用的, 无论它们是以源语言还是以目标语言书写。研究者首先使用卷积和 RNN 的组合探索该策略 (Kalchbrenner and Blunsom, 2013)。后来的工作介绍了使用 RNN 对所提议的翻译进行打分 (Cho *et al.*, 2014b) 或生成翻译句子 (Sutskever *et al.*, 2014)。Jean *et al.* (2014) 将这些模型扩展到更大的词汇表。

#### 12.4.5.1 使用注意力机制并对齐数据片段

使用固定大小的表示概括非常长的句子 (例如 60 个词) 的所有语义细节是非常困难的。这需要使用足够大的 RNN, 并且用足够长时间训练得很好才能实现, 如 Cho *et al.* (2014b) 和 Sutskever *et al.* (2014) 所表明的。然而, 更高效的方法是先读取整个句子或段落 (以获得正在表达的上下文和焦点), 然后一次翻译一个词,

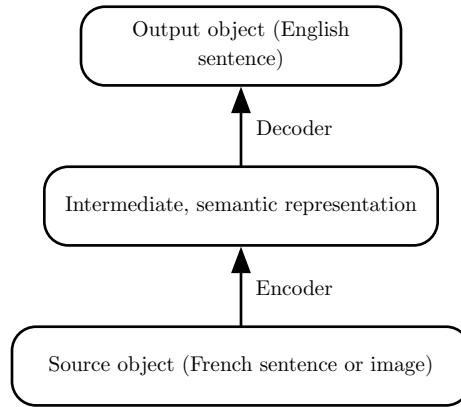


图 12.5: 编码器-解码器架构在直观表示（例如词序列或图像）和语义表示之间来回映射。使用来自一种模态数据的编码器输出（例如从法语句子到捕获句子含义的隐藏表示的编码器映射）作为用于另一模态的解码器输入（如解码器将捕获句子含义的隐藏表示映射到英语），我们可以训练将一种模态转换到另一种模态的系统。这个想法已经成功应用于很多领域，不仅仅是机器翻译，还包括为图像生成标题。

每次聚焦于输入句子的不同部分来收集产生下一个输出词所需的语义细节。这正是 Bahdanau *et al.* (2015) 第一次引入的想法。图 12.6 中展示了注意力机制，其中每个时间步关注输入序列的特定部分。

我们可以认为基于注意力机制的系统有三个组件：

- 读取器读取原始数据（例如源语句中的源词）并将其转换为分布式表示，其中一个特征向量与每个词的位置相关联。
- 存储器存储读取器输出的特征向量列表。这可以被理解为包含事实序列的存储器，而之后不必以相同的顺序从中检索，也不必访问全部。
- 最后一个程序利用存储器的内容顺序地执行任务，每个时间步聚焦于某个存储器元素的内容（或几个，具有不同权重）。

第三组件可以生成翻译语句。

当用一种语言书写的句子中的词与另一种语言的翻译语句中的相应词对齐时，可以使对应的词嵌入相关联。早期的工作表明，我们可以学习将一种语言中的词嵌入与另一种语言中的词嵌入相关联的翻译矩阵 (Kočiský *et al.*, 2014)，与传统的基于短语表中频率计数的方法相比，可以产生较低的对齐错误率。更早的工作

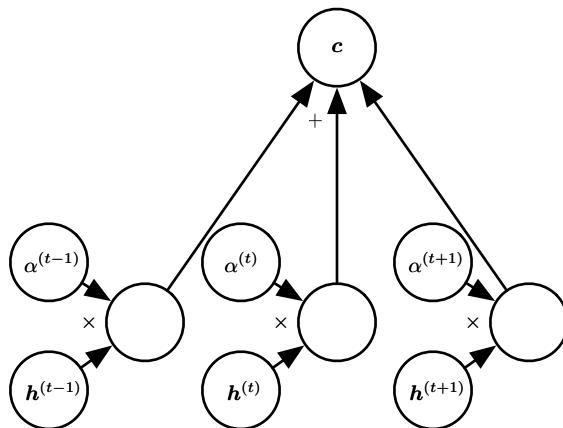


图 12.6: 由 Bahdanau *et al.* (2015) 引入的现代注意力机制，本质上是加权平均。注意力机制对具有权重  $\alpha^{(t)}$  的特征向量  $h^{(t)}$  进行加权平均形成上下文向量  $c$ 。在一些应用中，特征向量  $h$  是神经网络的隐藏单元，但它们也可以是模型的原始输入。权重  $\alpha^{(t)}$  由模型本身产生。它们通常是区间  $[0, 1]$  中的值，并且旨在仅仅集中在单个  $h^{(t)}$  周围，使得加权平均精确地读取接近一个特定时间步的特征向量。权重  $\alpha^{(t)}$  通常由模型另一部分发出的相关性得分应用 softmax 函数后产生。注意力机制在计算上需要比直接索引期望的  $h^{(t)}$  付出更高的代价，但直接索引不能使用梯度下降训练。基于加权平均的注意力机制是平滑、可微的近似，可以使用现有优化算法训练。

(Klementiev *et al.*, 2012) 也对跨语言词向量进行了研究。这种方法存在很多的扩展。例如，允许在更大数据集上训练的更高效的跨语言对齐 (Gouws *et al.*, 2014)。

### 12.4.6 历史展望

在对反向传播的第一次探索中，Rumelhart *et al.* (1986a) 等人提出了分布式表示符号的思想，其中符号对应于族成员的身份，而神经网络捕获族成员之间的关系，训练样本形成三元组如 (Colin, Mother, Victoria)。神经网络的第一层学习每个族成员的表示。例如，Colin 的特征可能代表 Colin 所在的族树，他所在树的分支，他来自哪一代等等。我们可以将神经网络认为是将这些属性关联在一起的计算学习规则，可以获得期望预测。模型则可以进行预测，例如推断谁是 Colin 的母亲。

Deerwester *et al.* (1990) 将符号嵌入的想法扩展到对词的嵌入。这些嵌入使用 SVD 学习。之后，嵌入将通过神经网络学习。

自然语言处理的历史是由流行表示（对模型输入不同方式的表示）的变化为标志的。在早期对符号和词建模的工作之后，神经网络在 NLP 上一些最早的应用 (Miikkulainen and Dyer, 1991; Schmidhuber, 1996) 将输入表示为字符序列。

Bengio *et al.* (2001b) 将焦点重新引到对词建模并引入神经语言模型，能产生可解释的词嵌入。这些神经模型已经从在一小组符号上的定义表示（20世纪80年代）扩展到现代应用中的数百万字（包括专有名词和拼写错误）。这种计算扩展的努力导致了第 12.4.3 节中描述的技术发明。

最初，使用词作为语言模型的基本单元可以改进语言建模的性能 (Bengio *et al.*, 2001b)。而今，新技术不断推动基于字符 (Sutskever *et al.*, 2011) 和基于词的模型向前发展，最近的工作 (Gillick *et al.*, 2015) 甚至建模 Unicode 字符的单个字节。

神经语言模型背后的思想已经扩展到多个自然语言处理应用，如解析 (Henderson, 2003, 2004; Collobert, 2011)、词性标注、语义角色标注、分块等，有时使用共享词嵌入的单一多任务学习架构 (Collobert and Weston, 2008a; Collobert *et al.*, 2011a)。

随着 t-SNE 降维算法的发展 (van der Maaten and Hinton, 2008) 以及 Joseph Turian 在 2009 年引入的专用于可视化词嵌入的应用，用于分析语言模型嵌入的二维可视化成为一种流行的工具。

## 12.5 其他应用

在本节中，我们介绍深度学习一些其他类型的应用，它们与上面讨论的标准对象识别、语音识别和自然语言处理任务不同。本书的第三部分将扩大这个范围，甚至进一步扩展到仍是目前主要研究领域的任务。

### 12.5.1 推荐系统

信息技术部门中机器学习的主要应用之一是向潜在用户或客户推荐项目。这可以分为两种主要的应用：在线广告和项目建议（通常这些建议的目的仍然是为了销售产品）。两者都依赖于预测用户和项目之间的关联，一旦向该用户展示了广告或推荐了该产品，推荐系统要么预测一些行为的概率（用户购买产品或该行为的一些代替）或预期增益（其可取决于产品的价值）。目前，互联网的资金主要来自于各种形式的在线广告。经济的主要部分依靠网上购物。包括 Amazon 和 eBay 在内的公司都使用了机器学习（包括深度学习）推荐他们的产品。有时，项目不是实际出售的产品。如选择在社交网络新闻信息流上显示的帖子、推荐观看的电影、推荐笑话、推荐专家建议、匹配视频游戏的玩家或匹配约会的人。

通常，这种关联问题可以作为监督学习问题来处理：给出一些关于项目和关于用户的信息，预测感兴趣的行为（用户点击广告、输入评级、点击“喜欢”按钮、购买产品，在产品上花钱、花时间访问产品页面等）。通常这最终会归结到回归问题（预测一些条件期望值）或概率分类问题（预测一些离散事件的条件概率）。

早期推荐系统的工作依赖于这些预测输入的最小信息：用户 ID 和项目 ID。在这种情况下，唯一的泛化方式依赖于不同用户或不同项目的目标变量值之间的模式相似性。假设用户 1 和用户 2 都喜欢项目 A, B 和 C. 由此，我们可以推断出用户 1 和用户 2 具有类似的口味。如果用户 1 喜欢项目 D，那么这可以强烈提示用户 2 也喜欢 D。基于此原理的算法称为 **协同过滤** (collaborative filtering)。非参数方法（例如基于估计偏好模式之间相似性的最近邻方法）和参数方法都可能用来解决这个问题。参数方法通常依赖于为每个用户和每个项目学习分布式表示（也称为嵌入）。目标变量的双线性预测（例如评级）是一种简单的参数方法，这种方法非常成功，通常被认为是最先进的系统的组成部分。通过用户嵌入和项目嵌入之间的点积（可能需要使用仅依赖于用户 ID 或项目 ID 的常数来校正）获得预测。令  $\hat{\mathbf{R}}$  是包含我们预测的矩阵， $\mathbf{A}$  矩阵行中是用户嵌入， $\mathbf{B}$  矩阵列中具有项目嵌入。令  $\mathbf{b}$  和  $\mathbf{c}$  是分别包

含针对每个用户（表示用户平常坏脾气或积极的程度）以及每个项目（表示其大体受欢迎程度）的偏置向量。因此，双线性预测如下获得：

$$\hat{R}_{u,i} = b_u + c_i + \sum_j A_{u,j} B_{j,i}. \quad (12.20)$$

通常，人们希望最小化预测评级  $\hat{R}_{u,i}$  和实际评级  $R_{u,i}$  之间的平方误差。当用户嵌入和项目嵌入首次缩小到低维度（两个或三个）时，它们就可以方便地可视化，或者可以将用户或项目彼此进行比较（就像词嵌入）。获得这些嵌入的一种方式是对实际目标（例如评级）的矩阵  $\mathbf{R}$  进行奇异值分解。这对应于将  $\mathbf{R} = \mathbf{UDV}'$ （或归一化的变体）分解为两个因子的乘积，低秩矩阵  $\mathbf{A} = \mathbf{UD}$  和  $\mathbf{B} = \mathbf{V}'$ 。SVD 的一个问题是它以任意方式处理缺失条目，如同它们对应于目标值 0。相反，我们希望避免为缺失条目做出的预测付出任何代价。幸运的是，观察到的评级的平方误差总和也可以使用基于梯度的优化最小化。SVD 和式 (12.20) 中的双线性预测在 Netflix 奖竞赛中（目的是仅基于大量匿名用户的之前评级预测电影的评级）表现得非常好 (Bennett and Lanning, 2007)。许多机器学习专家参加了 2006 年和 2009 年之间的这场比赛。它提高了使用先进机器学习的推荐系统的研究水平，并改进了推荐系统。即使简单的双线性预测或 SVD 本身并没有赢得比赛，但它是大多数竞争对手提出的整体模型中一个组成部分，包括胜者 (Töscher *et al.*, 2009; Koren, 2009)。

除了这些具有分布式表示的双线性模型之外，第一次用于协同过滤的神经网络之一是基于 RBM 的无向概率模型 (Salakhutdinov *et al.*, 2007)。RBM 是 Netflix 比赛获胜方法的一个重要组成部分 (Töscher *et al.*, 2009; Koren, 2009)。神经网络社群中也已经探索了对评级矩阵进行因子分解的更高级变体 (Salakhutdinov and Mnih, 2008)。

然而，协同过滤系统有一个基本限制：当引入新项目或新用户时，缺乏评级历史意味着无法评估其与其他项目或用户的相似性，或者说无法评估新的用户和现有项目的联系。这被称为冷启动推荐问题。解决冷启动推荐问题的一般方式是引入单个用户和项目的额外信息。例如，该额外信息可以是用户简要信息或每个项目的特征。使用这种信息的系统被称为 **基于内容的推荐系统** (content-based recommender system)。从丰富的用户特征或项目特征集到嵌入的映射可以通过深度学习架构学习 (Huang *et al.*, 2013; Elkahky *et al.*, 2015)。

专用的深度学习架构，如卷积网络已经应用于从丰富内容中提取特征，如提取用于音乐推荐的音乐音轨 (van den Oörd *et al.*, 2013)。在该工作中，卷积网络将声学特征作为输入并计算相关歌曲的嵌入。该歌曲嵌入和用户嵌入之间的点积则可以

预测用户是否将收听该歌曲。

### 12.5.1.1 探索与利用

当向用户推荐时，会产生超出普通监督学习范围的问题，并进入强化学习的领域。理论上，许多推荐问题最准确的描述是contextual bandit(Langford and Zhang, 2008; Lu *et al.*, 2010)。问题是，当我们使用推荐系统收集数据时，我们得到的是一个有偏且不完整的用户偏好观：我们只能看到用户对推荐给他们项目的反应，而不是其他项目。此外，在某些情况下，我们可能无法获得未向其进行推荐的用户的任何信息（例如，在广告竞价中，可能是广告的建议价格低于最低价格阈值，或者没有赢得竞价，因此广告不会显示）。更重要的是，我们不知道推荐任何其他项目会产生什么结果。这就像训练一个分类器，为每个训练样本  $x$  挑选一个类别  $\hat{y}$ （通常是基于模型最高概率的类别），然后只能获得该类别正确与否的反馈。显然，每个样本传达的信息少于监督的情况（其中真实标签  $y$  是可直接访问的），因此需要更多的样本。更糟糕的是，如果我们不够小心，即使收集越来越多的数据，我们得到的系统可能会继续选择错误的决定，因为正确的决定最初只有很低的概率：直到学习者选择正确的决定之前，该系统都无法学习正确的决定。这类似于强化学习的情况，其中仅观察到所选动作的奖励。一般来说，强化学习会涉及许多动作和许多奖励的序列。bandit 情景是强化学习的特殊情况，其中学习者仅采取单一动作并接收单个奖励。bandit 问题在学习者知道哪个奖励与哪个动作相关联的时候，是更容易的。在一般的强化学习场景中，高奖励或低奖励可能是由最近的动作或很久以前的动作引起的。术语 **contextual bandit** (contextual bandit) 指的是在一些输入变量可以通知决定的上下文中采取动作的情况。例如，我们至少知道用户身份，并且我们要选择一个项目。从上下文到动作的映射也称为 **策略** (policy)。学习者和数据分布（现在取决于学习者的动作）之间的反馈循环是强化学习和bandit研究的中心问题。

强化学习需要权衡探索 (exploration) 与 利用 (exploitation)。利用指的是从目前学到的最好策略采取动作，也就是我们所知的将获得高奖励的动作。探索 (exploration) 是指采取行动以获得更多的训练数据。如果我们知道给定上下文  $x$ ，动作  $a$  给予我们 1 的奖励，但我们不知道这是否是最好的奖励。我们可能想利用我们目前的策略，并继续采取行动  $a$  相对肯定地获得 1 的奖励。然而，我们也可能想通过尝试动作  $a'$  来探索。我们不知道尝试动作  $a'$  会发生什么。我们希望得到 2 的奖励，但有获得 0 奖励的风险。无论如何，我们至少获得了一些知识。

探索 (exploration) 可以以许多方式实现，从覆盖可能动作的整个空间的随机动作到基于模型的方法（基于预期回报和模型对该回报不确定性的量来计算动作的选择）。

许多因素决定了我们喜欢探索或利用的程度。最突出的因素之一是我们感兴趣的时间尺度。如果代理只有短暂的时间积累奖励，那么我们喜欢更多的利用。如果代理有很长时间积累奖励，那么我们开始更多的探索，以便使用更多的知识更有效地规划未来的动作。

监督学习在探索或利用之间没有权衡，因为监督信号总是指定哪个输出对于每个输入是正确的。我们总是知道标签是最好的输出，没有必要尝试不同的输出来确定是否优于模型当前的输出。

除了权衡探索和利用之外，强化学习背景下出现的另一个困难是难以评估和比较不同的策略。强化学习包括学习者和环境之间的相互作用。这个反馈回路意味着使用固定的测试集输入评估学习者的表现不是直接的。策略本身确定将看到哪些输入。Dudik *et al.* (2011) 提出了评估contextual bandit的技术。

### 12.5.2 知识表示、推理和回答

因为使用符号 (Rumelhart *et al.*, 1986a) 和词嵌入 (Deerwester *et al.*, 1990; Bengio *et al.*, 2001b)，深度学习方法在语言模型、机器翻译和自然语言处理方面非常成功。这些嵌入表示关于单个词或概念的语义知识。研究前沿是为短语或词和事实之间的关系开发嵌入。搜索引擎已经使用机器学习来实现这一目的，但是要改进这些更高级的表示还有许多工作要做。

#### 12.5.2.1 知识、联系和回答

一个有趣的研究方向是确定如何训练分布式表示才能捕获两个实体之间的关系 (relation)。

数学中，二元关系是一组有序的对象对。集合中的对具有这种关系，而那些不在集合中的对则没有。例如，我们可以在实体集  $\{1, 2, 3\}$  上定义关系“小于”来定义有序对的集合  $\mathbb{S} = \{(1, 2), (1, 3), (2, 3)\}$ 。一旦这个关系被定义，我们可以像动词一样使用它。因为  $(1, 2) \in \mathbb{S}$ ，我们说 1 小于 2。因为  $(2, 1) \notin \mathbb{S}$ ，我们不能说 2 小于 1。当然，彼此相关的实体不必是数字。我们可以定义关系 `is_a_type_of` 包含如（狗，

哺乳动物) 的元组。

在 AI 的背景下，我们将关系看作句法上简单且高度结构化的语言。关系起到动词的作用，而关系的两个参数发挥着主体和客体的作用。这些句子是一个三元组标记的形式：

$$(\text{subject}, \text{verb}, \text{object}) \quad (12.21)$$

其值是

$$(\text{entity}_i, \text{relation}_j, \text{entity}_k). \quad (12.22)$$

我们还可以定义 属性 (attribute)，类似于关系的概念，但只需要一个参数：

$$(\text{entity}_i, \text{attribute}_j). \quad (12.23)$$

例如，我们可以定义 `has_fur` 属性，并将其应用于像狗这样的实体。

许多应用中需要表示关系和推理。我们如何在神经网络中做到这一点？

机器学习模型当然需要训练数据。我们可以推断非结构化自然语言组成的训练数据集中实体之间的关系，也可以使用明确定义关系的结构化数据库。这些数据库的共同结构是关系型数据库，它存储这种相同类型的信息，虽然没有格式化为三元标记的句子。当数据库旨在将日常生活中常识或关于应用领域的专业知识传达给人工智能系统时，我们将这种数据库称为知识库。知识库包括一般的像 Freebase、OpenCyc、WordNet、Wikibase<sup>2</sup> 等等，和专业的知识库，如 GeneOntology<sup>3</sup>。实体和关系的表示可以将知识库中的每个三元组作为训练样本来学习，并且以最大化捕获它们的联合分布为训练目标 (Bordes *et al.*, 2013a)。

除了训练数据，我们还需定义训练的模型族。一种常见的方法是将神经语言模型扩展到模型实体和关系。神经语言模型学习提供每个词分布式表示的向量。他们还通过学习这些向量的函数来学习词之间的相互作用，例如哪些词可能出现在词序列之后。我们可以学习每个关系的嵌入向量将这种方法扩展到实体和关系。事实上，建模语言和通过关系编码建模知识的联系非常接近，研究人员可以同时使用知识库和自然语言句子训练这样的实体表示 (Bordes *et al.*, 2011, 2012; Wang *et al.*, 2014a)，或组合来自多个关系型数据库的数据 (Bordes *et al.*, 2013b)。可能与这种模型相关联的特定参数化有许多种。早期关于学习实体间关系的工作 (Paccanaro and Hinton,

<sup>2</sup> 分别可以在如下网址获取: [freebase.com](http://freebase.com), [cyc.com/opencyc](http://cyc.com/opencyc), [wordnet.princeton.edu](http://wordnet.princeton.edu), [wikiba.se](http://wikiba.se)

<sup>3</sup> [geneontology.org](http://geneontology.org)

2000) 假定高度受限的参数形式 (“线性关系嵌入”), 通常对关系使用与实体形式不同的表示。例如, Paccanaro and Hinton (2000) 和 Bordes *et al.* (2011) 用向量表示实体而矩阵表示关系, 其思想是关系在实体上相当于运算符。或者, 关系可以被认为是指任何其他实体 (Bordes *et al.*, 2012), 允许我们关于关系作声明, 但是更灵活的是将它们结合在一起并建模联合分布的机制。

这种模型的实际短期应用是 **链接预测** (link prediction): 预测知识图谱中缺失的弧。这是基于旧事实推广新事实的一种形式。目前存在的大多数知识库都是通过人力劳动构建的, 这往往使知识库缺失许多并且可能是大多数真正的关系。请查看 Wang *et al.* (2014b)、Lin *et al.* (2015) 和 Garcia-Duran *et al.* (2015) 中这样应用的例子。

我们很难评估链接预测任务上模型的性能, 因为我们的数据集只有正样本 (已知是真实的事)。如果模型提出了不在数据集中的事, 我们不确定模型是犯了错误还是发现了一个新的以前未知的事。度量基于测试模型如何将已知真实事的留存集合与不太可能为真的其他事相比较, 因此有些不精确。构造感兴趣的负样本 (可能为假的事) 的常见方式是从真实事开始, 并创建该事的损坏版本, 例如用随机选择的不同实体替换关系中的一个实体。通用的测试精度 (10% 度量) 计算模型在该事的所有损坏版本的前 10% 中选择 “正确” 事的次数。

知识库和分布式表示的另一个应用是 **词义消歧** (word-sense disambiguation) (Navigli and Velardi, 2005; Bordes *et al.*, 2012), 这个任务决定在某些语境中哪个词的意义是恰当。

最后, 知识的关系结合一个推理过程和对自然语言的理解可以让我们建立一个一般的问答系统。一般的问答系统必须能处理输入信息并记住重要的事, 并以之后能检索和推理的方式组织。这仍然是一个困难的开放性问题, 只能在受限的 “玩具” 环境下解决。目前, 记住和检索特定声明性事的最佳方法是使用显式记忆机制, 如第 10.12 节所述。记忆网络最开始是被用来解决一个玩具问答任务 (Weston *et al.*, 2014)。Kumar *et al.* (2015b) 提出了一种扩展, 使用 GRU 循环网络将输入读入存储器并且在给定存储器的内容后产生回答。

深度学习已经应用于其他许多应用 (除了这里描述的应用以外), 并且肯定会在之后应用于更多的场景。我们不可能全面描述与此主题相关的所有应用。本项调查尽可能地提供了在本文写作之时的代表性样本

第二部分介绍了涉及深度学习的现代实践, 包括了所有非常成功的方法。一般

而言，这些方法使用代价函数的梯度寻找模型（近似于某些所期望的函数）的参数。当具有足够的训练数据时，这种方法是非常强大的。我们现在转到第三部分，开始进入研究领域，旨在使用较少的训练数据或执行更多样的任务。而且相比目前为止所描述的情况，其中的挑战更困难并且远远没有解决。

# 第三部分

## 深度学习研究

本书这一部分描述目前研究社群所追求的、更有远见和更先进的深度学习方法。

在本书的前两部分，我们已经展示了如何解决监督学习问题，即在给定足够的映射样本的情况下，学习将一个向量映射到另一个。

我们想要解决的问题并不全都属于这个类别。我们可能希望生成新的样本、或确定一个点的似然性、或处理缺失值以及利用一组大量的未标记样本或相关任务的样本。当前应用于工业的最先进技术的缺点是我们的学习算法需要大量的监督数据才能实现良好的精度。在本书这一部分，我们讨论一些推测性的方法，来减少现有模型工作所需的标注数据量，并适用于更广泛的任务。实现这些目标通常需要某种形式的无监督或半监督学习。

许多深度学习算法被设计为处理无监督学习问题，但不像深度学习已经在很大程度上解决了各种任务的监督学习问题，没有一个算法能以同样的方式真正解决无监督学习问题。在本书这一部分，我们描述无监督学习的现有方法和一些如何在这一领域取得进展的流行思想。

无监督学习困难的核心原因是被建模的随机变量的高维度。这带来了两个不同的挑战：统计挑战和计算挑战。统计挑战与泛化相关：我们可能想要区分的配置数会随着感兴趣的维度数指数增长，并且这快速变得比可能具有的（或者在有限计算资源下使用的）样本数大得多。与高维分布相关联的计算挑战之所以会出现，是因为用于学习或使用训练模型的许多算法（特别是基于估计显式概率函数的算法）涉及难处理的计算量，并且随维数呈指数增长。

使用概率模型，这种计算挑战来自执行难解的推断或归一化分布。

- 难解的推断：推断主要在第十九章讨论。推断关于捕获  $a$ ,  $b$  和  $c$  上联合分布的模型，给定其他变量  $b$  的情况下，猜测一些变量  $a$  的可能值。为了计算这样的条件概率，我们需要对变量  $c$  的值求和，以及计算对  $a$  和  $c$  的值求和的归一化常数。
- 难解的归一化常数（配分函数）：配分函数主要在第十八章讨论。归一化概率函数的常数在推断（上文）以及学习中出现。许多概率模型涉及这样的归一化常数。不幸的是，学习这样的模型通常需要相对于模型参数计算配分函数对数的梯度。该计算通常与计算配分函数本身一样难解。马尔可夫链蒙特卡罗（MCMC）（第十七章）通常用于处理配分函数。不幸的是，当模型分布的模式众多且分离良好时，MCMC方法会出现问题，特别是在高维空间中

(第17.5节)。

面对这些难以处理的计算的一种方法是近似它们，如在本书的第三部分中讨论的，研究者已经提出了许多方法。这里还讨论另一种有趣的方式是通过设计模型，完全避免这些难以处理的计算，因此不需要这些计算的方法是非常有吸引力的。近年来，研究者已经提出了数种具有该动机的生成模型。其中第二十章讨论了各种各样的现代生成式建模方法。

第三部分对于研究者来说是最重要的，研究者想要了解深度学习领域的广度，并将领域推向真正的人工智能。

# 第十三章 线性因子模型

许多深度学习的研究前沿均涉及构建输入的概率模型  $p_{\text{model}}(\mathbf{x})$ 。原则上说，给定任何其他变量的情况下，这样的模型可以使用概率推断来预测其环境中的任何变量。许多这样的模型还具有潜变量  $\mathbf{h}$ ，其中  $p_{\text{model}}(\mathbf{x}) = \mathbb{E}_{\mathbf{h}} p_{\text{model}}(\mathbf{x} | \mathbf{h})$ 。这些潜变量提供了表示数据的另一种方式。我们在深度前馈网络和循环网络中已经发现，基于潜变量的分布式表示继承了表示学习的所有优点。

在本章中，我们描述了一些基于潜变量的最简单的概率模型：**线性因子模型** (linear factor model)。这些模型有时被用来作为混合模型的组成模块 (Hinton *et al.*, 1995a; Ghahramani and Hinton, 1996; Roweis *et al.*, 2002) 或者更大的深度概率模型 (Tang *et al.*, 2012)。同时，也介绍了构建生成模型所需的许多基本方法，在此基础上更先进的深度模型也将得到进一步扩展。

线性因子模型通过随机线性解码器函数来定义，该函数通过对  $\mathbf{h}$  的线性变换以及添加噪声来生成  $\mathbf{x}$ 。

有趣的是，通过这些模型我们能够发现一些符合简单联合分布的解释性因子。线性解码器的简单性使得它们成为了最早被广泛研究的潜变量模型。

线性因子模型描述如下的数据生成过程。首先，我们从一个分布中抽取解释性因子  $\mathbf{h}$

$$\mathbf{h} \sim p(\mathbf{h}), \quad (13.1)$$

其中  $p(\mathbf{h})$  是一个因子分布，满足  $p(\mathbf{h}) = \prod_i p(h_i)$ ，所以易于从中采样。接下来，在给定因子的情况下，我们对实值的可观察变量进行采样

$$\mathbf{x} = \mathbf{W}\mathbf{h} + \mathbf{b} + \text{noise}, \quad (13.2)$$

其中噪声通常是对角化的（在维度上是独立的）且服从高斯分布。这在图 13.1 有具

体说明。

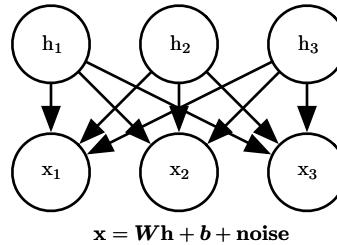


图 13.1: 描述线性因子模型族的有向图模型，其中我们假设观察到的数据向量  $\mathbf{x}$  是通过独立的潜在因子  $\mathbf{h}$  的线性组合再加上一定噪声获得的。不同的模型，比如概率 PCA，因子分析或者是 ICA，都是选择了不同形式的噪声以及先验  $p(\mathbf{h})$ 。

## 13.1 概率 PCA 和因子分析

**概率 PCA** (probabilistic PCA)、因子分析和其他线性因子模型是上述等式 (式(13.1)和式(13.2)) 的特殊情况，并且仅在对观测到  $\mathbf{x}$  之前的噪声分布和潜变量  $\mathbf{h}$  先验的选择上有所不同。

在**因子分析** (factor analysis) (Bartholomew, 1987; Basilevsky, 1994) 中，潜变量的先验是一个方差为单位矩阵的高斯分布

$$\mathbf{h} \sim \mathcal{N}(\mathbf{h}; \mathbf{0}, \mathbf{I}), \quad (13.3)$$

同时，假定在给定  $\mathbf{h}$  的条件下观察值  $x_i$  是**条件独立** (conditionally independent) 的。具体来说，我们可以假设噪声是从对角协方差矩阵的高斯分布中抽出的，协方差矩阵为  $\psi = \text{diag}(\sigma^2)$ ，其中  $\sigma^2 = [\sigma_1^2, \sigma_2^2, \dots, \sigma_n^2]^\top$  表示一个向量，每个元素表示一个变量的方差。

因此，潜变量的作用是捕获不同观测变量  $x_i$  之间的依赖关系。实际上，可以容易地看出  $\mathbf{x}$  服从多维正态分布，并满足

$$\mathbf{x} \sim \mathcal{N}(\mathbf{x}; \mathbf{b}, \mathbf{WW}^\top + \psi). \quad (13.4)$$

为了将 PCA 引入到概率框架中，我们可以对因子分析模型作轻微修改，使条件方差  $\sigma_i^2$  等于同一个值。在这种情况下， $\mathbf{x}$  的协方差简化为  $\mathbf{WW}^\top + \sigma^2 \mathbf{I}$ ，这里的  $\sigma^2$

是一个标量。由此可以得到条件分布，如下：

$$\mathbf{x} \sim \mathcal{N}(\mathbf{x}; \mathbf{b}, \mathbf{W}\mathbf{W}^\top + \sigma^2 \mathbf{I}), \quad (13.5)$$

或者等价地

$$\mathbf{x} = \mathbf{W}\mathbf{h} + \mathbf{b} + \sigma\mathbf{z}, \quad (13.6)$$

其中  $\mathbf{z} \sim \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I})$  是高斯噪声。之后 Tipping and Bishop (1999) 提出了一种迭代的 EM 算法来估计参数  $\mathbf{W}$  和  $\sigma^2$ 。

这个概率 PCA (probabilistic PCA) 模型利用了这样一种观察现象：除了一些微小残余的重构误差 (reconstruction error) (至多为  $\sigma^2$ )，数据中的大多数变化可以由潜变量  $\mathbf{h}$  描述。通过Tipping and Bishop (1999) 的研究我们可以发现，当  $\sigma \rightarrow 0$  时，概率 PCA 退化为 PCA。在这种情况下，给定  $\mathbf{x}$  情况下  $\mathbf{h}$  的条件期望等于将  $\mathbf{x} - \mathbf{b}$  投影到  $\mathbf{W}$  的  $d$  列所生成的空间上，与 PCA 一样。

当  $\sigma \rightarrow 0$  时，概率 PCA 所定义的密度函数在  $d$  维的  $\mathbf{W}$  的列生成空间周围非常尖锐。这导致模型会为没有在一个超平面附近聚集的数据分配非常低的概率。

## 13.2 独立成分分析

独立成分分析 (independent component analysis, ICA) 是最古老的表示学习算法之一 (Herault and Ans, 1984; Jutten and Herault, 1991; Comon, 1994; Hyvärinen, 1999; Hyvärinen *et al.*, 2001a; Hinton *et al.*, 2001; Teh *et al.*, 2003)。它是一种建模线性因子的方法，旨在将观察到的信号分离成许多潜在信号，这些潜在信号通过缩放和叠加可以恢复成观察数据。这些信号是完全独立的，而不是仅仅彼此不相关<sup>1</sup>。

许多不同的具体方法被称为 ICA。与我们本书中描述的其他生成模型最相似的 ICA 变种 (Pham *et al.*, 1992) 训练了完全参数化的生成模型。潜在因子  $\mathbf{h}$  的先验  $p(\mathbf{h})$ ，必须由用户提前给出并固定。接着模型确定性地生成  $\mathbf{x} = \mathbf{W}\mathbf{h}$ 。我们可以通过非线性变化 (使用式 (3.47)) 来确定  $p(\mathbf{x})$ 。然后通过一般的方法比如最大化似然进行学习。

这种方法的动机是，通过选择一个独立的  $p(\mathbf{h})$ ，我们可以尽可能恢复接近独立的潜在因子。这是一种常用的方法，它并不是用来捕捉高级别的抽象因果因子，而是

---

<sup>1</sup>第3.8节讨论了不相关变量和独立变量之间的差异。

恢复已经混合在一起的低级别信号。在该设置中，每个训练样本对应一个时刻，每个  $x_i$  是一个传感器对混合信号的观察值，并且每个  $h_i$  是单个原始信号的一个估计。例如，我们可能有  $n$  个人同时说话。如果我们在不同位置放置  $n$  个不同的麦克风，则 ICA 可以检测每个麦克风的音量变化，并且分离信号，使得每个  $h_i$  仅包含一个人清楚地说话。这通常用于脑电图的神经科学，这种技术可用于记录源自大脑的电信号。放置在受试者头部上的许多电极传感器用于测量来自身体的多种电信号。实验者通常仅对来自大脑的信号感兴趣，但是来自受试者心脏和眼睛的信号强到足以混淆在受试者头皮处的测量结果。信号到达电极，并且混合在一起，因此为了分离源于心脏与源于大脑的信号，并且将不同脑区域中的信号彼此分离，ICA 是必要的。

如前所述，ICA 存在许多变种。一些版本在  $\mathbf{x}$  的生成中添加一些噪声，而不是使用确定性的解码器。大多数方法不使用最大似然准则，而是旨在使  $\mathbf{h} = \mathbf{W}^{-1}\mathbf{x}$  的元素彼此独立。许多准则能够达成这个目标。式(3.47)需要用到  $\mathbf{W}$  的行列式，这可能是代价很高且数值不稳定的操作。ICA 的一些变种通过将  $\mathbf{W}$  约束为正交来避免这个有问题的操作。

ICA 的所有变种均要求  $p(\mathbf{h})$  是非高斯的。这是因为如果  $p(\mathbf{h})$  是具有高斯分量的独立先验，则  $\mathbf{W}$  是不可识别的。对于许多  $\mathbf{W}$  值，我们可以在  $p(\mathbf{x})$  上获得相同的分布。这与其他线性因子模型有很大的区别，例如概率 PCA 和因子分析通常要求  $p(\mathbf{h})$  是高斯的，以便使模型上的许多操作具有闭式解。在用户明确指定分布的最大似然方法中，一个典型的选择是使用  $p(h_i) = \frac{d}{dh_i}\sigma(h_i)$ 。这些非高斯分布的典型选择在 0 附近具有比高斯分布更高的峰值，因此我们也可以看到独立成分分析经常用于学习稀疏特征。

按照我们对生成模型这个术语的定义，ICA 的许多变种不是生成模型。在本书中，生成模型可以直接表示  $p(\mathbf{x})$ ，也可以认为是从  $p(\mathbf{x})$  中抽取样本。ICA 的许多变种仅知道如何在  $\mathbf{x}$  和  $\mathbf{h}$  之间变换，而没有任何表示  $p(\mathbf{h})$  的方式，因此也无法在  $p(\mathbf{x})$  上施加分布。例如，许多 ICA 变量旨在增加  $\mathbf{h} = \mathbf{W}^{-1}\mathbf{x}$  的样本峰度，因为高峰度说明了  $p(\mathbf{h})$  是非高斯的，但这是在没有显式表示  $p(\mathbf{h})$  的情况下完成的。这就是为什么 ICA 多被用作分离信号的分析工具，而不是用于生成数据或估计其密度。

正如 PCA 可以推广到第十四章中描述的非线性自编码器，ICA 也可以推广到非线性生成模型，其中我们使用非线性函数  $f$  来生成观测数据。关于非线性 ICA 最初的工作可以参考 Hyvärinen and Pajunen (1999)，它和集成学习的成功结合可以参见 Roberts and Everson (2001); Lappalainen *et al.* (2000)。ICA 的另一个非线性扩展是非线性独立成分估计 (nonlinear independent components estimation, NICE)

方法 (Dinh *et al.*, 2014)，这个方法堆叠了一系列可逆变换（在编码器阶段），其特性是能高效地计算每个变换的 Jacobian 行列式。这使得我们能够精确地计算似然，并且像 ICA 一样，NICE 尝试将数据变换到具有因子的边缘分布的空间。由于非线性编码器的使用，这种方法更可能成功。因为编码器和一个能进行完美逆变换的解码器相关联，所以可以直接从模型生成样本（首先从  $p(\mathbf{h})$  采样，然后使用解码器）。

ICA 的另一个推广是通过鼓励组内统计依赖关系、抑制组间依赖关系来学习特征组 (Hyvärinen and Hoyer, 1999; Hyvärinen *et al.*, 2001b)。当相关单元的组被选为不重叠时，这被称为 **独立子空间分析** (independent subspace analysis)。我们还可以向每个隐藏单元分配空间坐标，并且空间上相邻的单元组形成一定程度的重叠。这能够鼓励相邻的单元学习类似的特征。当应用于自然图像时，这种 **地质 ICA** (topographic ICA) 方法可以学习 Gabor 滤波器，从而使得相邻特征具有相似的方向、位置或频率。在每个区域内出现类似 Gabor 函数的许多不同相位存在抵消作用，使得在小区域上的池化产生了平移不变性。

### 13.3 慢特征分析

**慢特征分析** (slow feature analysis, SFA) 是使用来自时间信号的信息学习不变特征的线性因子模型 (Wiskott and Sejnowski, 2002)。

慢特征分析的想法源于所谓的 **慢性原则** (slowness principle)。其基本思想是，与场景中起描述作用的单个量度相比，场景的重要特性通常变化得非常缓慢。例如，在计算机视觉中，单个像素值可以非常快速地改变。如果斑马从左到右移动穿过图像并且它的条纹穿过对应的像素时，该像素将迅速从黑色变为白色，并再次恢复成黑色。通过比较，指示斑马是否在图像中的特征将不发生改变，并且描述斑马位置的特征将缓慢地改变。因此，我们可能希望将模型正则化，从而能够学习到那些随时间变化较为缓慢的特征。

慢性原则早于慢特征分析，并已被应用于各种模型 (Hinton, 1989; Földiák, 1989; Mobahi *et al.*, 2009; Bergstra and Bengio, 2009)。一般来说，我们可以将慢性原则应用于可以使用梯度下降训练的任何可微分模型。为了引入慢性原则，我们可以向代价函数添加以下项

$$\lambda \sum_t L(f(\mathbf{x}^{(t+1)}), f(\mathbf{x}^{(t)})), \quad (13.7)$$

其中  $\lambda$  是确定慢度正则化强度的超参数项,  $t$  是样本时间序列的索引,  $f$  是需要正则化的特征提取器,  $L$  是测量  $f(\mathbf{x}^{(t)})$  和  $f(\mathbf{x}^{(t+1)})$  之间的距离的损失函数。 $L$  的一个常见选择是均方误差。

慢特征分析是慢性原则中一个特别高效的应用。由于它被应用于线性特征提取器, 并且可以通过闭式解训练, 所以它是高效的。像 ICA 的一些变种一样, SFA 本身并不是生成模型, 只是在输入空间和特征空间之间定义了一个线性映射, 但是没有定义特征空间的先验, 因此没有在输入空间上施加分布  $p(\mathbf{x})$ 。

SFA 算法 (Wiskott and Sejnowski, 2002) 先将  $f(\mathbf{x}; \theta)$  定义为线性变换, 然后求解如下优化问题

$$\min_{\theta} \mathbb{E}_t (f(\mathbf{x}^{(t+1)})_i - f(\mathbf{x}^{(t)})_i)^2 \quad (13.8)$$

并且满足下面的约束:

$$\mathbb{E}_t f(\mathbf{x}^{(t)})_i = 0 \quad (13.9)$$

以及

$$\mathbb{E}_t [f(\mathbf{x}^{(t)})_i^2] = 1. \quad (13.10)$$

学习特征具有零均值的约束对于使问题具有唯一解是必要的; 否则我们可以向所有特征值添加一个常数, 并获得具有相等慢度目标值的不同解。特征具有单位方差的约束对于防止所有特征趋近于 0 的病态解是必要的。与 PCA 类似, SFA 特征是有序的, 其中学习第一特征是最慢的。要学习多个特征, 我们还必须添加约束

$$\forall i < j, \quad \mathbb{E}_t [f(\mathbf{x}^{(t)})_i f(\mathbf{x}^{(t)})_j] = 0. \quad (13.11)$$

这要求学习的特征必须彼此线性去相关。没有这个约束, 所有学到的特征将简单地捕获一个最慢的信号。可以想象使用其他机制, 如最小化重构误差, 也可以迫使特征多样化。但是由于 SFA 特征的线性, 这种去相关机制只能得到一种简单的解。SFA 问题可以通过线性代数软件获得闭式解。

在运行 SFA 之前, SFA 通常通过对  $\mathbf{x}$  使用非线性的基扩充来学习非线性特征。例如, 通常用  $\mathbf{x}$  的二次基扩充来代替原来的  $\mathbf{x}$ , 得到一个包含所有  $x_i x_j$  的向量。由此, 我们可以通过反复地学习一个线性 SFA 特征提取器, 对其输出应用非线性基扩展, 然后在该扩展之上学习另一个线性 SFA 特征提取器的方式来组合线性 SFA 模块从而学习深度非线性慢特征提取器。

当在自然场景视频的小块空间部分上训练时，使用二次基扩展的 SFA 所学习到的特征与 V1 皮层中那些复杂细胞的特征有许多共同特性 (Berkes and Wiskott, 2005)。当在计算机渲染的 3D 环境内随机运动的视频上训练时，深度 SFA 模型能够学习的特征与大鼠脑中用于导航的神经元学到的特征有许多共同特性 (Franzius *et al.*, 2007)。因此从生物学角度上来说 SFA 是一个合理的有依据的模型。

SFA 的一个主要优点是，即使在深度非线性条件下，它依然能够在理论上预测 SFA 能够学习哪些特征。为了做出这样的理论预测，必须知道关于配置空间的环境动力（例如，在 3D 渲染环境中随机运动的例子中，理论分析是从相机位置、速度的概率分布中入手的）。已知潜在因子如何改变的情况下，我们能够通过理论分析解出表达这些因子的最佳函数。在实践中，基于模拟数据的实验上，使用深度 SFA 似乎能够恢复理论预测的函数。相比之下，在其他学习算法中，代价函数高度依赖于特定像素值，使得难以确定模型将学习到什么特征。

深度 SFA 也已经被用于学习用在对象识别和姿态估计的特征 (Franzius *et al.*, 2008)。到目前为止，慢性原则尚未成为任何最先进应用的基础。究竟是什么因素限制了其性能仍有待研究。我们推测，或许慢度先验太过强势，并且，最好添加这样一个先验使得当前时间步到下一个时间步的预测更加容易，而不是加一个先验使得特征近似为一个常数。对象的位置是一个有用的特征，无论对象的速度是高还是低。但慢性原则鼓励模型忽略具有高速度的对象的位置。

## 13.4 稀疏编码

稀疏编码 (sparse coding) (Olshausen and Field, 1996) 是一个线性因子模型，已作为一种无监督特征学习和特征提取机制得到了广泛研究。严格来说，术语“稀疏编码”是指在该模型中推断  $\mathbf{h}$  值的过程，而“稀疏建模”是指设计和学习模型的过程，但是通常这两个概念都可以用术语“稀疏编码”描述。

像大多数其他线性因子模型一样，它使用了线性的解码器加上噪声的方式获得一个  $\mathbf{x}$  的重构，就像式(13.2)描述的一样。更具体地说，稀疏编码模型通常假设线性因子有一个各向同性精度为  $\beta$  的高斯噪声：

$$p(\mathbf{x} \mid \mathbf{h}) = \mathcal{N}(\mathbf{x}; \mathbf{W}\mathbf{h} + \mathbf{b}, \frac{1}{\beta}\mathbf{I}). \quad (13.12)$$

分布  $p(\mathbf{h})$  通常选取为一个峰值很尖锐且接近 0 的分布 (Olshausen and Field,

1996)。常见的选择包括可分解的 Laplace、Cauchy 或者可分解的 Student-t 分布。例如，以稀疏惩罚系数  $\lambda$  为参数的 Laplace 先验可以表示为

$$p(h_i) = \text{Laplace}(h_i; 0, \frac{2}{\lambda}) = \frac{\lambda}{4} e^{-\frac{1}{2}\lambda|h_i|}, \quad (13.13)$$

相应的，Student-t 先验分布可以表示为

$$p(h_i) \propto \frac{1}{(1 + \frac{h_i^2}{\nu})^{\frac{\nu+1}{2}}}. \quad (13.14)$$

使用最大似然的方法来训练稀疏编码模型是不可行的。相反，为了在给定编码的情况下更好地重构数据，训练过程在编码数据和训练解码器之间交替进行。稍后在第 19.3 节中，这种方法将被进一步证明为是解决最大似然问题的一种通用的近似方法。

对于诸如 PCA 的模型，我们已经看到使用了预测  $\mathbf{h}$  的参数化的编码器函数，并且该函数仅包括乘以权重矩阵。稀疏编码中的编码器不是参数化的编码器。相反，编码器是一个优化算法，在这个优化问题中，我们寻找单个最可能的编码值：

$$\mathbf{h}^* = f(\mathbf{x}) = \arg \max_{\mathbf{h}} p(\mathbf{h} | \mathbf{x}). \quad (13.15)$$

结合式 (13.13) 和式 (13.12)，我们得到如下的优化问题：

$$\arg \max_{\mathbf{h}} p(\mathbf{h} | \mathbf{x}) \quad (13.16)$$

$$= \arg \max_{\mathbf{h}} \log p(\mathbf{h} | \mathbf{x}) \quad (13.17)$$

$$= \arg \min_{\mathbf{h}} \lambda \|\mathbf{h}\|_1 + \beta \|\mathbf{x} - \mathbf{W}\mathbf{h}\|_2^2, \quad (13.18)$$

其中，我们扔掉了与  $\mathbf{h}$  无关的项，并除以一个正的缩放因子来简化表达。

由于在  $\mathbf{h}$  上施加  $L^1$  范数，这个过程将产生稀疏的  $\mathbf{h}^*$  (详见第 7.1.2 节)。

为了训练模型而不仅仅是进行推断，我们交替迭代关于  $\mathbf{h}$  和  $\mathbf{W}$  的最小化过程。在本文中，我们将  $\beta$  视为超参数。我们通常将其设置为 1，因为它在此优化问题的作用与  $\lambda$  类似，没有必要使用两个超参数。原则上，我们还可以将  $\beta$  作为模型的参数，并学习它。我们在这里已经放弃了一些不依赖于  $\mathbf{h}$  但依赖于  $\beta$  的项。要学习  $\beta$ ，必须包含这些项，否则  $\beta$  将退化为 0。

不是所有的稀疏编码方法都显式地构建了一个  $p(\mathbf{h})$  和一个  $p(\mathbf{x} | \mathbf{h})$ 。通常我们只是对学习一个带有激活值的特征的字典感兴趣，当特征是由这个推断过程提取时，这个激活值通常为 0。

如果我们从 Laplace 先验中采样  $\mathbf{h}$ ,  $\mathbf{h}$  的元素实际上为 0 是一个零概率事件。生成模型本身并不稀疏，只有特征提取器是稀疏的。Goodfellow *et al.* (2013f) 描述了不同模型族中的近似推断，如尖峰和平板稀疏编码模型，其中先验的样本通常包含许多真正的 0。

与非参数编码器结合的稀疏编码方法原则上可以比任何特定的参数化编码器更好地最小化重构误差和对数先验的组合。另一个优点是编码器没有泛化误差。参数化的编码器必须泛化地学习如何将  $\mathbf{x}$  映射到  $\mathbf{h}$ 。对于与训练数据差异很大的异常  $\mathbf{x}$ ，所学习的参数化编码器可能无法找到对应精确重构或稀疏的编码  $\mathbf{h}$ 。对于稀疏编码模型的绝大多数形式，推断问题是凸的，优化过程总能找到最优编码（除非出现退化的情况，例如重复的权重向量）。显然，稀疏和重构成本仍然可以在不熟悉的点上升，但这归因于解码器权重中的泛化误差，而不是编码器中的泛化误差。当稀疏编码用作分类器的特征提取器，而不是使用参数化的函数来预测编码值时，基于优化的稀疏编码模型的编码过程中较小的泛化误差可以得到更好的泛化能力。Coates and Ng (2011) 证明了在对象识别任务中稀疏编码特征比基于参数化的编码器（线性-sigmoid 自编码器）的特征拥有更好的泛化能力。受他们的工作启发，Goodfellow *et al.* (2013f) 表明一种稀疏编码的变体在标签极少（每类 20 个或更少标签）的情况下比相同情况下的其他特征提取器拥有更好的泛化能力。

非参数编码器的主要缺点是在给定  $\mathbf{x}$  的情况下需要大量的时间来计算  $\mathbf{h}$ ，因为非参数方法需要运行迭代算法。在第十四章中讲到的参数化自编码器方法仅使用固定数量的层，通常只有一层。另一个缺点是它不直接通过非参数编码器进行反向传播，这使得我们很难采用先使用无监督方式预训练稀疏编码模型然后使用监督方式对其进行精调的方法。允许近似导数的稀疏编码模型的修改版本确实存在但未被广泛使用 (Bagnell and Bradley, 2009)。

像其他线性因子模型一样，稀疏编码经常产生糟糕的样本，如图 13.2 所示。即使当模型能够很好地重构数据并为分类器提供有用的特征时，也会发生这种情况。这种现象发生的原因是每个单独的特征可以很好地被学习到，但是隐藏编码值的因子先验会导致模型包括每个生成样本中所有特征的随机子集。这促使人们开发更深的模型，可以在其中最深的编码层施加一个非因子分布，与此同时也在开发一些复杂的浅度模型。

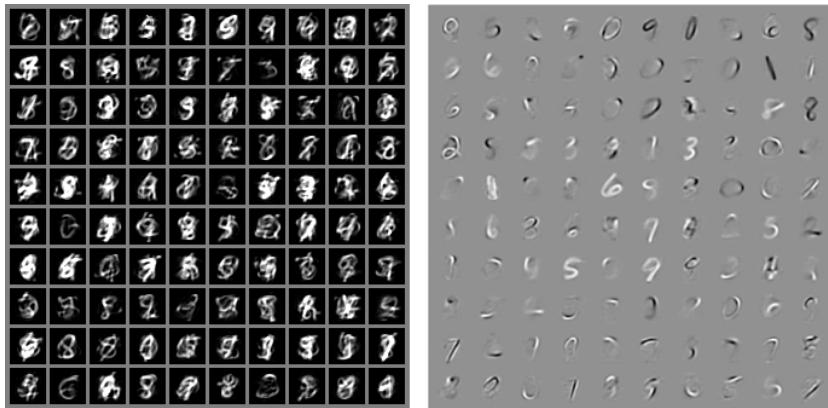


图 13.2: 尖峰和平板稀疏编码模型上在 MNIST 数据集训练的样例和权重。(左) 这个模型中的样本和训练样本相差很大。第一眼看来, 我们可能认为模型拟合得很差。(右) 这个模型的权重向量已经学习到了如何表示笔迹, 有时候还能写完整的数字。因此这个模型也学习到了有用的特征。问题在于特征的因子先验会导致特征子集合随机的组合。一些这样的子集能够合成可识别的 MNIST 集上的数字。这也促进了拥有更强大潜在编码分布的生成模型的发展。此图经 Goodfellow *et al.* (2013f) 允许转载。

## 13.5 PCA 的流形解释

线性因子模型, 包括 PCA 和因子分析, 可以理解为学习一个流形 (Hinton *et al.*, 1997)。我们可以将概率 PCA 定义为高概率的薄饼状区域, 即一个高斯分布, 沿着某些轴非常窄, 就像薄饼沿着其垂直轴非常平坦, 但沿着其他轴是细长的, 正如薄饼在其水平轴方向是很宽的一样。图 13.3 解释了这种现象。PCA 可以理解为将该薄饼与更高维空间中的线性流形对准。这种解释不仅适用于传统 PCA, 而且适用于学习矩阵  $\mathbf{W}$  和  $\mathbf{V}$  的任何线性自编码器, 其目的是使重构的  $\hat{\mathbf{x}}$  尽可能接近于原始的  $\mathbf{x}$ 。

编码器表示为

$$\mathbf{h} = f(\mathbf{x}) = \mathbf{W}^\top (\mathbf{x} - \boldsymbol{\mu}). \quad (13.19)$$

编码器计算  $\mathbf{h}$  的低维表示。从自编码器的角度来看, 解码器负责计算重构:

$$\hat{\mathbf{x}} = g(\mathbf{h}) = \mathbf{b} + \mathbf{V}\mathbf{h}. \quad (13.20)$$

能够最小化重构误差

$$\mathbb{E}[\|\mathbf{x} - \hat{\mathbf{x}}\|^2] \quad (13.21)$$

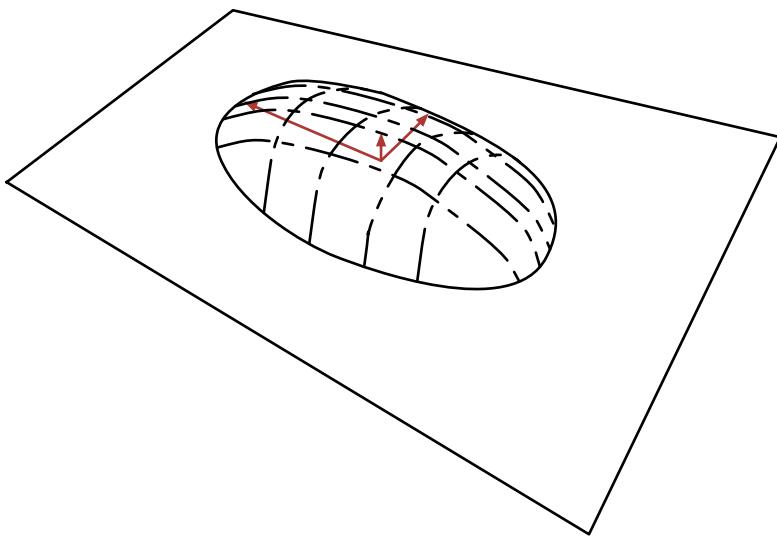


图 13.3: 平坦的高斯能够描述一个低维流形附近的概率密度。此图表示了“流形平面”上“馅饼”的上半部分，并且这个平面穿过了馅饼的中心。正交于流形方向（指向平面外的箭头方向）的方差非常小，可以被视作是“噪声”，其他方向（平面内的箭头）的方差则很大，对应了“信号”以及降维数据的坐标系统。

的线性编码器和解码器的选择对应着  $\mathbf{V} = \mathbf{W}$ ,  $\boldsymbol{\mu} = \mathbf{b} = \mathbb{E}[\mathbf{x}]$ ,  $\mathbf{W}$  的列形成一组标准正交基，这组基生成的子空间与协方差矩阵  $\mathbf{C}$

$$\mathbf{C} = \mathbb{E}[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^\top] \quad (13.22)$$

的主特征向量所生成的子空间相同。在 PCA 中， $\mathbf{W}$  的列是按照对应特征值（其全部是实数和非负数）幅度大小排序所对应的特征向量。

我们还可以发现  $\mathbf{C}$  的特征值  $\lambda_i$  对应了  $\mathbf{x}$  在特征向量  $\mathbf{v}^{(i)}$  方向上的方差。如果  $\mathbf{x} \in \mathbb{R}^D$ ,  $\mathbf{h} \in \mathbb{R}^d$  并且满足  $d < D$ ，则（给定上述的  $\boldsymbol{\mu}, \mathbf{b}, \mathbf{V}, \mathbf{W}$  的情况下）最佳的重构误差是

$$\min \mathbb{E}[\|\mathbf{x} - \hat{\mathbf{x}}\|^2] = \sum_{i=d+1}^D \lambda_i. \quad (13.23)$$

因此，如果协方差矩阵的秩为  $d$ ，则特征值  $\lambda_{d+1}$  到  $\lambda_D$  都为 0，并且重构误差为 0。

此外，我们还可以证明上述解可以通过在给定正交矩阵  $\mathbf{W}$  的情况下最大化  $\mathbf{h}$  元素的方差而不是最小化重构误差来获得。

某种程度上说，线性因子模型是最简单的生成模型和学习数据表示的最简单模型。许多模型如线性分类器和线性回归模型可以扩展到深度前馈网络，而这些线性因子模型可以扩展到自编码器网络和深度概率模型，它们可以执行相同任务但具有更强大和更灵活的模型族。

# 第十四章 自编码器

自编码器（autoencoder）是神经网络的一种，经过训练后能尝试将输入复制到输出。自编码器（autoencoder）内部有一个隐藏层  $\mathbf{h}$ ，可以产生 编码（code）表示输入。该网络可以看作由两部分组成：一个由函数  $\mathbf{h} = f(\mathbf{x})$  表示的编码器和一个生成重构的解码器  $\mathbf{r} = g(\mathbf{h})$ 。图 14.1 展示了这种架构。如果一个自编码器只是简单地学会将处处设置为  $g(f(\mathbf{x})) = \mathbf{x}$ ，那么这个自编码器就没什么特别的用处。相反，我们不应该将自编码器设计成输入到输出完全相等。这通常需要向自编码器强加一些约束，使它只能近似地复制，并只能复制与训练数据相似的输入。这些约束强制模型考虑输入数据的哪些部分需要被优先复制，因此它往往能学习到数据的有用特性。

现代自编码器将编码器和解码器的概念推而广之，将其中的确定函数推广为随机映射  $p_{\text{encoder}}(\mathbf{h} | \mathbf{x})$  和  $p_{\text{decoder}}(\mathbf{x} | \mathbf{h})$ 。

数十年间，自编码器的想法一直是神经网络历史景象的一部分 (LeCun, 1987; Bourlard and Kamp, 1988; Hinton and Zemel, 1994)。传统自编码器被用于降维或特征学习。近年来，自编码器与潜变量模型理论的联系将自编码器带到了生成式建模的前沿，我们将在第二十章揭示更多细节。自编码器可以被看作是前馈网络的一个特例，并且可以使用完全相同的技术进行训练，通常使用小批量梯度下降法（其中梯度基于反向传播计算）。不同于一般的前馈网络，自编码器也可以使用再循环（recirculation）训练 (Hinton and McClelland, 1988)，这种学习算法基于比较原始输入的激活和重构输入的激活。相比反向传播算法，再循环算法更具生物学意义，但很少用于机器学习应用。

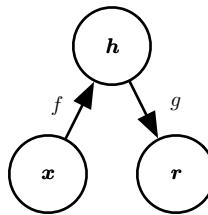


图 14.1: 自编码器的一般结构, 通过内部表示或编码  $h$  将输入  $x$  映射到输出 (称为重构)  $r$ 。自编码器具有两个组件: 编码器  $f$  (将  $x$  映射到  $h$ ) 和解码器  $g$  (将  $h$  映射到  $r$ )。

## 14.1 欠完备自编码器

将输入复制到输出听起来没什么用, 但我们通常不关心解码器的输出。相反, 我们希望通过训练自编码器对输入进行复制而使  $h$  获得有用的特性。

从自编码器获得有用特征的一种方法是限制  $h$  的维度比  $x$  小, 这种编码维度小于输入维度的自编码器称为 **欠完备** (undercomplete) 自编码器。学习欠完备的表示将强制自编码器捕捉训练数据中最显著的特征。

学习过程可以简单地描述为最小化一个损失函数

$$L(\mathbf{x}, g(f(\mathbf{x}))), \quad (14.1)$$

其中  $L$  是一个损失函数, 惩罚  $g(f(\mathbf{x}))$  与  $\mathbf{x}$  的差异, 如均方误差。

当解码器是线性的且  $L$  是均方误差, 欠完备的自编码器会学习出与 PCA 相同的生成子空间。这种情况下, 自编码器在训练来执行复制任务的同时学到了训练数据的主元子空间。

因此, 拥有非线性编码器函数  $f$  和非线性解码器函数  $g$  的自编码器能够学习出更强大的 PCA 非线性推广。不幸的是, 如果编码器和解码器被赋予过大的容量, 自编码器会执行复制任务而捕捉不到任何有关数据分布的有用信息。从理论上说, 我们可以设想这样一个自编码器, 它只有一维编码, 但它具有一个非常强大的非线性编码器, 能够将每个训练数据  $\mathbf{x}^{(i)}$  表示为编码  $i$ 。而解码器可以学习将这些整数索引映射回特定训练样本的值。这种特定情形不会在实际情况中发生, 但它清楚地说明, 如果自编码器的容量太大, 那训练来执行复制任务的自编码器可能无法学习到数据集的任何有用信息。

## 14.2 正则自编码器

编码维数小于输入维数的欠完备自编码器可以学习数据分布最显著的特征。我们已经知道，如果赋予这类自编码器过大的容量，它就不能学到任何有用的信息。

如果隐藏编码的维数允许与输入相等，或隐藏编码维数大于输入的过完备 (overcomplete) 情况下，会发生类似的问题。在这些情况下，即使是线性编码器和线性解码器也可以学会将输入复制到输出，而学不到任何有关数据分布的有用信息。

理想情况下，根据要建模的数据分布的复杂性，选择合适的编码维数和编码器、解码器容量，就可以成功训练任意架构的自编码器。正则自编码器提供这样的能力。正则自编码器使用的损失函数可以鼓励模型学习其他特性（除了将输入复制到输出），而不必限制使用浅层的编码器和解码器以及小的编码维数来限制模型的容量。这些特性包括稀疏表示、表示的小导数、以及对噪声或输入缺失的鲁棒性。即使模型容量大到足以学习一个无意义的恒等函数，非线性且过完备的正则自编码器仍然能够从数据中学到一些关于数据分布的有用信息。

除了这里所描述的方法（正则化自编码器最自然的解释），几乎任何带有潜变量并配有一个推断过程（计算给定输入的潜在表示）的生成模型，都可以看作是自编码器的一种特殊形式。强调与自编码器联系的两个生成式建模方法是 Helmholtz 机 (Hinton *et al.*, 1995b) 的衍生模型，如变分自编码器（第 20.10.3 节）和生成随机网络（第 20.12 节）。这些变种（或衍生）自编码器能够学习出高容量且过完备的模型，进而发现输入数据中有用的结构信息，并且也无需对模型进行正则化。这些编码显然是有用的，因为这些模型被训练为近似训练数据的概率分布而不是将输入复制到输出。

### 14.2.1 稀疏自编码器

稀疏自编码器简单地在训练时结合编码层的稀疏惩罚  $\Omega(\mathbf{h})$  和重构误差：

$$L(\mathbf{x}, g(f(\mathbf{x}))) + \Omega(\mathbf{h}), \quad (14.2)$$

其中  $g(\mathbf{h})$  是解码器的输出，通常  $\mathbf{h}$  是编码器的输出，即  $\mathbf{h} = f(\mathbf{x})$ 。

稀疏自编码器一般用来学习特征，以便用于像分类这样的任务。稀疏正则化的自编码器必须反映训练数据集的独特统计特征，而不是简单地充当恒等函数。以这种方式训练，执行附带稀疏惩罚的复制任务可以得到能学习有用特征的模型。

我们可以简单地将惩罚项  $\Omega(\mathbf{h})$  视为加到前馈网络的正则项，这个前馈网络的主要任务是将输入复制到输出（无监督学习的目标），并尽可能地根据这些稀疏特征执行一些监督学习任务（根据监督学习的目标）。不像其它正则项如权重衰减——没有直观的贝叶斯解释。如第 5.6.1 节描述，权重衰减和其他正则惩罚可以被解释为一个 MAP 近似贝叶斯推断，正则化的惩罚对应于模型参数的先验概率分布。这种观点认为，正则化的最大似然对应最大化  $p(\boldsymbol{\theta} | \mathbf{x})$ ，相当于最大化  $\log p(\mathbf{x} | \boldsymbol{\theta}) + \log p(\boldsymbol{\theta})$ 。 $\log p(\mathbf{x} | \boldsymbol{\theta})$  即通常的数据似然项，参数的对数先验项  $\log p(\boldsymbol{\theta})$  则包含了对  $\boldsymbol{\theta}$  特定值的偏好。这种观点在第 5.6 节有所描述。正则自编码器不适用这样的解释是因为正则项取决于数据，因此根据定义上（从文字的正式意义）来说，它不是一个先验。虽然如此，我们仍可以认为这些正则项隐式地表达了对函数的偏好。

我们可以认为整个稀疏自编码器框架是对带有潜变量的生成模型的近似最大似然训练，而不将稀疏惩罚视为复制任务的正则化。假如我们有一个带有可见变量  $\mathbf{x}$  和潜变量  $\mathbf{h}$  的模型，且具有明确的联合分布  $p_{\text{model}}(\mathbf{x}, \mathbf{h}) = p_{\text{model}}(\mathbf{h})p_{\text{model}}(\mathbf{x} | \mathbf{h})$ 。我们将  $p_{\text{model}}(\mathbf{h})$  视为模型关于潜变量的先验分布，表示模型看到  $\mathbf{x}$  的信念先验。这与我们之前使用“先验”的方式不同，之前指分布  $p(\boldsymbol{\theta})$  在我们看到数据前就对模型参数的先验进行编码。对数似然函数可分解为

$$\log p_{\text{model}}(\mathbf{x}) = \log \sum_{\mathbf{h}} p_{\text{model}}(\mathbf{h}, \mathbf{x}). \quad (14.3)$$

我们可以认为自编码器使用一个高似然值  $\mathbf{h}$  的点估计近似这个总和。这类似于稀疏编码生成模型（第 13.4 节），但  $\mathbf{h}$  是参数编码器的输出，而不是从优化结果推断出的最可能的  $\mathbf{h}$ 。从这个角度看，我们根据这个选择的  $\mathbf{h}$ ，最大化如下

$$\log p_{\text{model}}(\mathbf{h}, \mathbf{x}) = \log p_{\text{model}}(\mathbf{h}) + \log p_{\text{model}}(\mathbf{x} | \mathbf{h}). \quad (14.4)$$

$\log p_{\text{model}}(\mathbf{h})$  项能被稀疏诱导。如 Laplace 先验，

$$p_{\text{model}}(h_i) = \frac{\lambda}{2} e^{-\lambda|h_i|}, \quad (14.5)$$

对应于绝对值稀疏惩罚。将对数先验表示为绝对值惩罚，我们得到

$$\Omega(\mathbf{h}) = \lambda \sum_i |h_i|, \quad (14.6)$$

$$-\log p_{\text{model}}(\mathbf{h}) = \sum_i (\lambda|h_i| - \log \frac{\lambda}{2}) = \Omega(\mathbf{h}) + \text{const}, \quad (14.7)$$

这里的常数项只跟  $\lambda$  有关。通常我们将  $\lambda$  视为超参数，因此可以丢弃不影响参数学习的常数项。其他如 Student-t 先验也能诱导稀疏性。从稀疏性导致  $p_{\text{model}}(\mathbf{h})$  学习成近似最大似然的结果看，稀疏惩罚完全不是一个正则项。这仅仅影响模型关于潜变量的分布。这个观点提供了训练自编码器的另一个动机：这是近似训练生成模型的一种途径。这也给出了为什么自编码器学到的特征是有用的另一个解释：它们描述的潜变量可以解释输入。

稀疏自编码器的早期工作 (Ranzato *et al.*, 2007a, 2008) 探讨了各种形式的稀疏性，并提出了稀疏惩罚和  $\log Z$  项（将最大似然应用到无向概率模型  $p(\mathbf{x}) = \frac{1}{Z} \tilde{p}(\mathbf{x})$  时产生）之间的联系。这个想法是最小化  $\log Z$  防止概率模型处处具有高概率，同理强制稀疏可以防止自编码器处处具有低的重构误差。这种情况下，这种联系是对通用机制的直观理解而不是数学上的对应。在数学上更容易解释稀疏惩罚对应于有向模型  $p_{\text{model}}(\mathbf{h})p_{\text{model}}(\mathbf{x} | \mathbf{h})$  中的  $\log p_{\text{model}}(\mathbf{h})$ 。

Glorot *et al.* (2011b) 提出了一种在稀疏（和去噪）自编码器的  $\mathbf{h}$  中实现真正为零的方式。该想法是使用整流线性单元产生编码层。基于将表示真正推向零（如绝对值惩罚）的先验，可以间接控制表示中零的平均数量。

### 14.2.2 去噪自编码器

除了向代价函数增加一个惩罚项，我们也可以通过改变重构误差项来获得一个能学到有用信息的自编码器。

传统的自编码器最小化以下目标

$$L(\mathbf{x}, g(f(\mathbf{x}))), \quad (14.8)$$

其中  $L$  是一个损失函数，惩罚  $g(f(\mathbf{x}))$  与  $\mathbf{x}$  的差异，如它们彼此差异的  $L^2$  范数。如果模型被赋予过大的容量， $L$  仅仅使得  $g \circ f$  学成一个恒等函数。

相反，去噪自编码器 (denoising autoencoder, DAE) 最小化

$$L(\mathbf{x}, g(f(\tilde{\mathbf{x}}))), \quad (14.9)$$

其中  $\tilde{\mathbf{x}}$  是被某种噪声损坏的  $\mathbf{x}$  的副本。因此去噪自编码器必须撤消这些损坏，而不是简单地复制输入。

Alain and Bengio (2013) 和 Bengio *et al.* (2013c) 指出去噪训练过程强制  $f$  和  $g$  隐式地学习  $p_{\text{data}}(\mathbf{x})$  的结构。因此去噪自编码器也是一个通过最小化重构误差获

取有用特性的例子。这也是将过完备、高容量的模型用作自编码器的一个例子——只要小心防止这些模型仅仅学习一个恒等函数。去噪自编码器将在第 14.5 节给出更多细节。

### 14.2.3 惩罚导数作为正则

另一正则化自编码器的策略是使用一个类似稀疏自编码器中的惩罚项  $\Omega$ ,

$$L(\mathbf{x}, g(f(\mathbf{x}))) + \Omega(\mathbf{h}, \mathbf{x}), \quad (14.10)$$

但  $\Omega$  的形式不同:

$$\Omega(\mathbf{h}, \mathbf{x}) = \lambda \sum_i \|\nabla_{\mathbf{x}} h_i\|^2. \quad (14.11)$$

这迫使模型学习一个在  $\mathbf{x}$  变化小时目标也没有太大变化的函数。因为这个惩罚只对训练数据适用，它迫使自编码器学习可以反映训练数据分布信息的特征。

这样正则化的自编码器被称为 **收缩自编码器** (contractive autoencoder, CAE)。这种方法与去噪自编码器、流形学习和概率模型存在一定理论联系。收缩自编码器将在第 14.7 节更详细地描述。

## 14.3 表示能力、层的大小和深度

自编码器通常只有单层的编码器和解码器，但这不是必然的。实际上深度编码器和解码器能提供更多优势。

回忆第 6.4.1 节，其中提到加深前馈网络有很多优势。这些优势也同样适用于自编码器，因为它也属于前馈网络。此外，编码器和解码器各自都是一个前馈网络，因此这两个部分也能各自从深度结构中获得好处。

万能近似定理保证至少有一层隐藏层且隐藏单元足够多的前馈神经网络能以任意精度近似任意函数（在很大范围里），这是非平凡深度（至少有一层隐藏层）的一个主要优点。这意味着具有单隐藏层的自编码器在数据域内能表示任意近似数据的恒等函数。但是，从输入到编码的映射是浅层的。这意味着我们不能任意添加约束，比如约束编码稀疏。深度自编码器（编码器至少包含一层额外隐藏层）在给定足够多的隐藏单元的情况下，能以任意精度近似任何从输入到编码的映射。

深度可以指数地降低表示某些函数的计算成本。深度也能指数地减少学习一些函数所需的训练数据量。读者可以参考第 6.4.1 节巩固深度在前馈网络中的优势。

实验中，深度自编码器能比相应的浅层或线性自编码器产生更好的压缩效率 (Hinton and Salakhutdinov, 2006)。

训练深度自编码器的普遍策略是训练一堆浅层的自编码器来贪心地预训练相应的深度架构。所以即使最终目标是训练深度自编码器，我们也经常会遇到浅层自编码器。

## 14.4 随机编码器和解码器

自编码器本质上是一个前馈网络，可以使用与传统前馈网络相同的损失函数和输出单元。

如第 6.2.2.4 节中描述，设计前馈网络的输出单元和损失函数普遍策略是定义一个输出分布  $p(\mathbf{y} | \mathbf{x})$  并最小化负对数似然  $-\log p(\mathbf{y} | \mathbf{x})$ 。在这种情况下， $\mathbf{y}$  是关于目标的向量（如类标）。

在自编码器中， $\mathbf{x}$  既是输入也是目标。然而，我们仍然可以使用与之前相同的架构。给定一个隐藏编码  $\mathbf{h}$ ，我们可以认为解码器提供了一个条件分布  $p_{\text{model}}(\mathbf{x} | \mathbf{h})$ 。接着我们根据最小化  $-\log p_{\text{decoder}}(\mathbf{x} | \mathbf{h})$  来训练自编码器。损失函数的具体形式视  $p_{\text{decoder}}$  的形式而定。就传统的前馈网络来说，如果  $\mathbf{x}$  是实值的，那么我们通常使用线性输出单元参数化高斯分布的均值。在这种情况下，负对数似然对应均方误差准则。类似地，二值  $\mathbf{x}$  对应于一个 Bernoulli 分布，其参数由 sigmoid 输出单元确定的。而离散的  $\mathbf{x}$  对应 softmax 分布，以此类推。在给定  $\mathbf{h}$  的情况下，为了便于计算概率分布，输出变量通常被视为是条件独立的，但一些技术（如混合密度输出）可以解决输出相关的建模。

为了更彻底地与我们之前了解到的前馈网络相区别，我们也可以将**编码函数** (encoding function)  $f(\mathbf{x})$  的概念推广为**编码分布** (encoding distribution)  $p_{\text{encoder}}(\mathbf{h} | \mathbf{x})$ ，如图 14.2 中所示。

任何潜变量模型  $p_{\text{model}}(\mathbf{h}, \mathbf{x})$  定义一个随机编码器

$$p_{\text{encoder}}(\mathbf{h} | \mathbf{x}) = p_{\text{model}}(\mathbf{h} | \mathbf{x}) \quad (14.12)$$

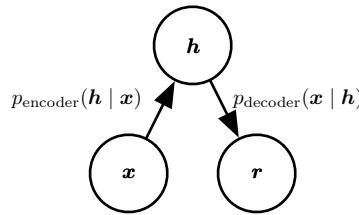


图 14.2: 随机自编码器的结构, 其中编码器和解码器包括一些噪声注入, 而不是简单的函数。这意味着可以将它们的输出视为来自分布的采样 (对于编码器是  $p_{\text{encoder}}(\mathbf{h} | \mathbf{x})$ , 对于解码器是  $p_{\text{decoder}}(\mathbf{x} | \mathbf{h})$ )。

以及一个随机解码器

$$p_{\text{decoder}}(\mathbf{x} | \mathbf{h}) = p_{\text{model}}(\mathbf{x} | \mathbf{h}). \quad (14.13)$$

通常情况下, 编码器和解码器的分布没有必要是与唯一一个联合分布  $p_{\text{model}}(\mathbf{x}, \mathbf{h})$  相容的条件分布。Alain *et al.* (2015) 指出, 在保证足够的容量和样本的情况下, 将编码器和解码器作为去噪自编码器训练, 能使它们渐近地相容。

## 14.5 去噪自编码器

**去噪自编码器** (denoising autoencoder, DAE) 是一类接受损坏数据作为输入, 并训练来预测原始未被损坏数据作为输出的自编码器。

DAE 的训练过程如图 14.3 中所示。我们引入一个损坏过程  $C(\tilde{\mathbf{x}} | \mathbf{x})$ , 这个条件分布代表给定数据样本  $\mathbf{x}$  产生损坏样本  $\tilde{\mathbf{x}}$  的概率。自编码器则根据以下过程, 从训练数据对  $(\mathbf{x}, \tilde{\mathbf{x}})$  中学习重构分布 (reconstruction distribution)  $p_{\text{reconstruct}}(\mathbf{x} | \tilde{\mathbf{x}})$ :

1. 从训练数据中采一个训练样本  $\mathbf{x}$ 。
2. 从  $C(\tilde{\mathbf{x}} | \mathbf{x} = \mathbf{x})$  采一个损坏样本  $\tilde{\mathbf{x}}$ 。
3. 将  $(\mathbf{x}, \tilde{\mathbf{x}})$  作为训练样本来估计自编码器的重构分布  $p_{\text{reconstruct}}(\mathbf{x} | \tilde{\mathbf{x}}) = p_{\text{decoder}}(\mathbf{x} | \mathbf{h})$ , 其中  $\mathbf{h}$  是编码器  $f(\tilde{\mathbf{x}})$  的输出,  $p_{\text{decoder}}$  根据解码函数  $g(\mathbf{h})$  定义。

通常我们可以简单地对负对数似然  $-\log p_{\text{decoder}}(\mathbf{x} | \mathbf{h})$  进行基于梯度法 (如小批

量梯度下降) 的近似最小化。只要编码器是确定性的, 去噪自编码器就是一个前馈网络, 并且可以使用与其他前馈网络完全相同的方式进行训练。

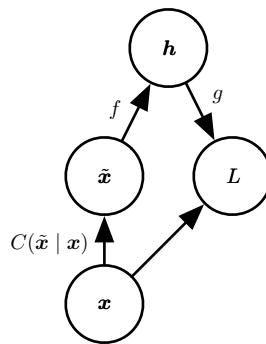


图 14.3: 去噪自编码器代价函数的计算图。去噪自编码器被训练为从损坏的版本  $\tilde{x}$  重构干净数据点  $x$ 。这可以通过最小化损失  $L = -\log p_{\text{decoder}}(x \mid h = f(\tilde{x}))$  实现, 其中  $\tilde{x}$  是样本  $x$  经过损坏过程  $C(\tilde{x} \mid x)$  后得到的损坏版本。通常, 分布  $p_{\text{decoder}}$  是因子的分布 (平均参数由前馈网络  $g$  给出)。

因此我们可以认为 DAE 是在以下期望下进行随机梯度下降:

$$-\mathbb{E}_{x \sim \hat{p}_{\text{data}}(x)} \mathbb{E}_{\tilde{x} \sim C(\tilde{x} \mid x)} \log p_{\text{decoder}}(x \mid h = f(\tilde{x})), \quad (14.14)$$

其中  $\hat{p}_{\text{data}}(x)$  是训练数据的分布。

### 14.5.1 得分估计

得分匹配 (Hyvärinen, 2005a) 是最大似然的代替。它提供了概率分布的一致估计, 促使模型在各个数据点  $x$  上获得与数据分布相同的得分 (score)。在这种情况下, 得分是一个特定的梯度场:

$$\nabla_x \log p(x). \quad (14.15)$$

我们将在第 18.4 节中更详细地讨论得分匹配。对于现在讨论的自编码器, 理解学习  $\log p_{\text{data}}$  的梯度场是学习  $p_{\text{data}}$  结构的一种方式就足够了。

DAE 的训练准则 (条件高斯  $p(x \mid h)$ ) 能让自编码器学到能估计数据分布得分的向量场 ( $g(f(x)) - x$ ), 这是 DAE 的一个重要特性。具体如图 14.4 所示。

对一类采用高斯噪声和均方误差作为重构误差的特定去噪自编码器 (具有 sigmoid 隐藏单元和线性重构单元) 的去噪训练过程, 与训练一类特定的被称为 RBM 的

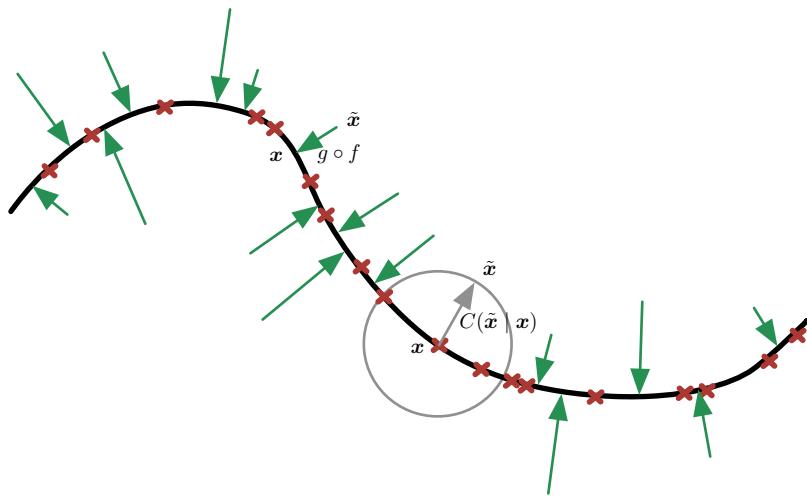


图 14.4: 去噪自编码器被训练为将损坏的数据点  $\tilde{x}$  映射回原始数据点  $x$ 。我们将训练样本  $x$  表示为位于低维流形（粗黑线）附近的红叉。我们用灰色圆圈表示等概率的损坏过程  $C(\tilde{x} | x)$ 。灰色箭头演示了如何将一个训练样本转换为经过此损坏过程的样本。当训练去噪自编码器最小化平方误差  $\|g(f(\tilde{x})) - x\|^2$  的平均值时，重构  $g(f(\tilde{x}))$  估计  $\mathbb{E}_{\mathbf{x}, \tilde{\mathbf{x}} \sim p_{\text{data}}(\mathbf{x})C(\tilde{\mathbf{x}}|\mathbf{x})} [\mathbf{x} | \tilde{\mathbf{x}}]$ 。 $g(f(\tilde{x}))$  对可能产生  $\tilde{x}$  的原始点  $x$  的质心进行估计，所以向量  $g(f(\tilde{x})) - \tilde{x}$  近似指向流形上最近的点。因此自编码器可以学习由绿色箭头表示的向量场  $g(f(x)) - x$ 。该向量场将得分  $\nabla_x \log p_{\text{data}}(x)$  估计为一个乘性因子，即重构误差均方根的平均。

无向概率模型是等价的 (Vincent, 2011)。这类模型将在第 20.5.1 节给出更详细的介绍；对于现在的讨论，我们只需知道这个模型能显式的给出  $p_{\text{model}}(\mathbf{x}; \boldsymbol{\theta})$ 。当 RBM 使用去噪得分匹配 (denoising score matching) 算法 (Kingma and LeCun, 2010a) 训练时，它的学习算法与训练对应的去噪自编码器是等价的。在一个确定的噪声水平下，正则化的得分匹配不是一致估计量；相反它会恢复分布的一个模糊版本。然而，当噪声水平趋向于 0 且训练样本数趋向于无穷时，一致性就会恢复。我们将会在第 18.5 节更详细地讨论去噪得分匹配。

自编码器和 RBM 还存在其他联系。在 RBM 上应用得分匹配后，其代价函数将等价于重构误差结合类似 CAE 惩罚的正则项 (Swersky *et al.*, 2011)。Bengio and Delalleau (2009) 指出自编码器的梯度是对 RBM 对比散度训练的近似。

对于连续的  $\mathbf{x}$ ，高斯损坏和重构分布的去噪准则得到的得分估计适用于一般编

码器和解码器的参数化 (Alain and Bengio, 2013)。这意味着一个使用平方误差准则

$$\|g(f(\tilde{\mathbf{x}})) - \mathbf{x}\|^2 \quad (14.16)$$

和噪声方差为  $\sigma^2$  的损坏

$$C(\tilde{\mathbf{x}} = \tilde{\mathbf{x}} | \mathbf{x}) = N(\tilde{\mathbf{x}}; \mu = \mathbf{x}, \Sigma = \sigma^2 I) \quad (14.17)$$

的通用编码器-解码器架构可以用来训练估计得分。图 14.5 展示其中的工作原理。

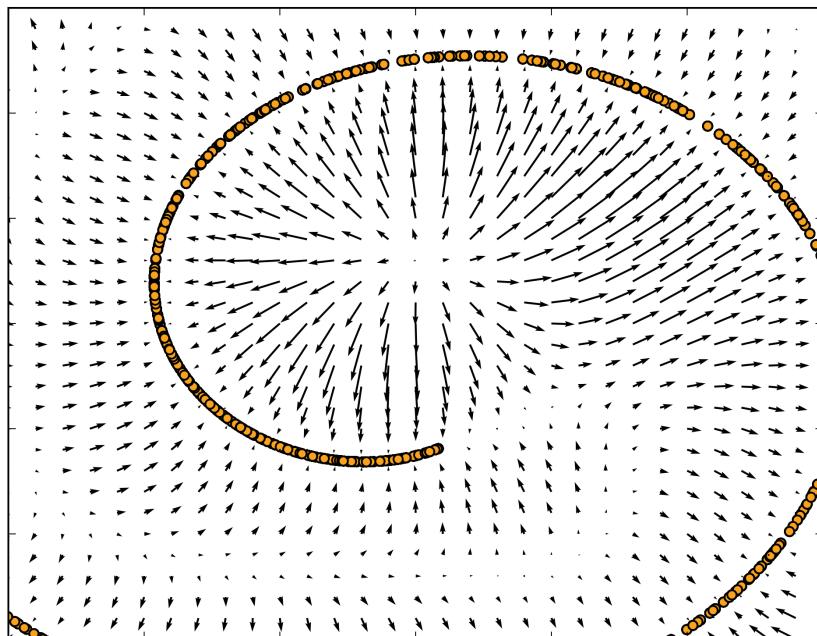


图 14.5: 由去噪自编码器围绕 1 维弯曲流形学习的向量场，其中数据集中在 2 维空间中。每个箭头与重构向量减去自编码器的输入向量后的向量成比例，并且根据隐式估计的概率分布指向较高的概率。向量场在估计的密度函数的最大值处（在数据流形上）和密度函数的最小值处都为零。例如，螺旋臂形成局部最大值彼此连接的 1 维流形。局部最小值出现在两个臂间隙的中间附近。当重构误差的范数（由箭头的长度示出）很大时，在箭头的方向上移动可以显著增加概率，并且在低概率的地方大多也是如此。自编码器将这些低概率点映射到较高的概率重构。在概率最大的情况下，重构变得更准确，因此箭头会收缩。经 Alain and Bengio (2013) 许可转载此图。

一般情况下，不能保证重构函数  $g(f(\mathbf{x}))$  减去输入  $\mathbf{x}$  后对应于某个函数的梯度，更不用说得分。这是早期工作 (Vincent, 2011) 专用于特定参数化的原因（其中  $g(f(\mathbf{x})) - \mathbf{x}$  能通过另一个函数的导数获得）。Kamyshanska and Memisevic (2015)

通过标识一类特殊的浅层自编码器家族，使  $g(f(\mathbf{x})) - \mathbf{x}$  对应于这个家族所有成员的一个得分，以此推广 Vincent (2011) 的结果。

目前为止我们所讨论的仅限于去噪自编码器如何学习表示一个概率分布。更一般的，我们可能希望使用自编码器作为生成模型，并从其分布中进行采样。这将在第 20.11 节中讨论。

### 14.5.2 历史展望

采用 MLP 去噪的想法可以追溯到 LeCun (1987) 和 Gallinari *et al.* (1987) 的工作。Behnke (2001) 也曾使用循环网络对图像去噪。在某种意义上，去噪自编码器仅仅是被训练去噪的 MLP。然而，“去噪自编码器”的命名指的不仅仅是学习去噪，而且可以学到一个好的内部表示（作为学习去噪的副效用）。这个想法提出较晚 (Vincent *et al.*, 2008b, 2010)。学到的表示可以被用来预训练更深的无监督网络或监督网络。与稀疏自编码器、稀疏编码、收缩自编码器等正则化的自编码器类似，DAE 的动机是允许学习容量很高的编码器，同时防止在编码器和解码器学习一个无用的恒等函数。

在引入现代 DAE 之前，Inayoshi and Kurita (2005) 探索了其中一些相同的方法和目标。他们除了在监督目标的情况下最小化重构误差之外，还在监督 MLP 的隐藏层注入噪声，通过引入重构误差和注入噪声提升泛化能力。然而，他们的方法基于线性编码器，因此无法学习到现代 DAE 能学习的强大函数族。

## 14.6 使用自编码器学习流形

如第 5.11.3 节描述，自编码器跟其他很多机器学习算法一样，也利用了数据集中在一个低维流形或者一小组这样的流形的思想。其中一些机器学习算法仅能学习到在流形上表现良好但给定不在流形上的输入会导致异常的函数。自编码器进一步借此想法，旨在学习流形的结构。

要了解自编码器如何做到这一点，我们必须介绍流形的一些重要特性。

流形的一个重要特征是切平面 (tangent plane) 的集合。 $d$  维流形上的一点  $\mathbf{x}$ ，切平面由能张成流形上允许变动的局部方向的  $d$  维基向量给出。如图 14.6 所示，这些局部方向决定了我们能如何微小地变动  $\mathbf{x}$  而保持于流形上。

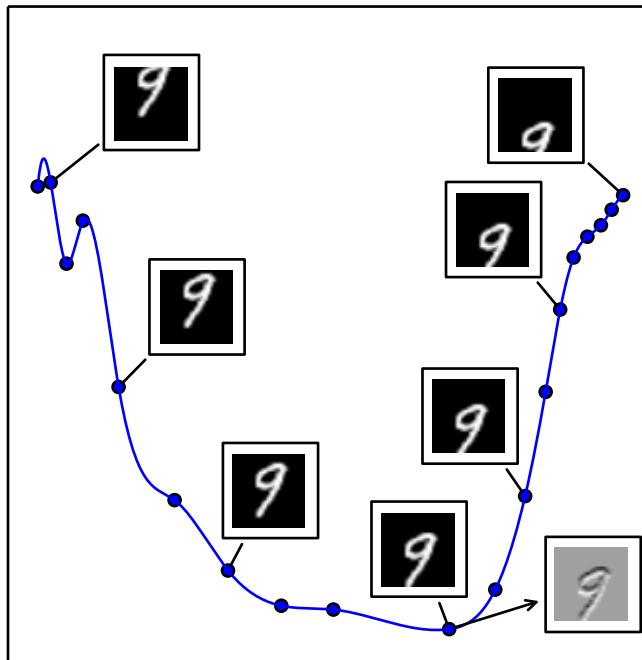


图 14.6: 正切超平面概念的图示。我们在 784 维空间中创建了 1 维流形。我们使用一张 784 像素的 MNIST 图像，并通过垂直平移来转换它。垂直平移的量定义沿着 1 维流形的坐标，轨迹为通过图像空间的弯曲路径。该图显示了沿着该流形的几个点。为了可视化，我们使用 PCA 将流形投影到 2 维空间中。 $n$  维流形在每个点处都具有  $n$  维切平面。该切平面恰好在该点接触流形，并且在该点处平行于流形表面。它定义了为保持在流形上可以移动的方向空间。该 1 维流形具有单个切线。我们在图中示出了一个点处的示例切线，其中图像表示该切线方向在图像空间中是怎样的。灰色像素表示沿着切线移动时不改变的像素，白色像素表示变亮的像素，黑色像素表示变暗的像素。

所有自编码器的训练过程涉及两种推动力的折衷：

1. 学习训练样本  $x$  的表示  $h$  使得  $x$  能通过解码器近似地从  $h$  中恢复。 $x$  是从训练数据挑出的这一事实很关键，因为这意味着自编码器不需要成功重构不属于数据生成分布下的输入。
2. 满足约束或正则惩罚。这可以是限制自编码器容量的架构约束，也可以是加入

到重构代价的一个正则项。这些技术一般倾向那些对输入较不敏感的解。

显然，单一的推动力是无用的——从它本身将输入复制到输出是无用的，同样忽略输入也是没用的。相反，两种推动力结合是有用的，因为它们驱使隐藏的表示能捕获有关数据分布结构的信息。重要的原则是，自编码器必须有能力表示重构训练实例所需的变化。如果该数据生成分布集中靠近一个低维流形，自编码器能隐式产生捕捉这个流形局部坐标系的表示：仅在  $x$  周围关于流形的相切变化需要对应于  $h = f(x)$  中的变化。因此，编码器学习从输入空间  $x$  到表示空间的映射，映射仅对沿着流形方向的变化敏感，并且对流形正交方向的变化不敏感。

图 14.7 中一维的例子说明，我们可以通过构建对数据点周围的输入扰动不敏感的重构函数，使得自编码器恢复流形结构。

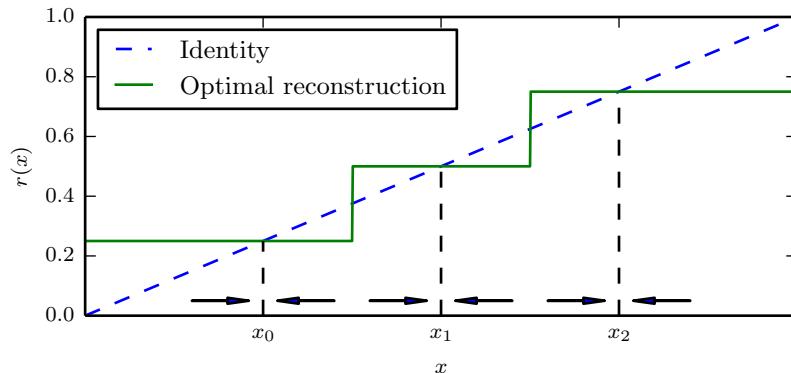


图 14.7：如果自编码器学习到对数据点附近的小扰动不变的重构函数，它就能捕获数据的流形结构。这里，流形结构是 0 维流形的集合。虚线对角线表示重构的恒等函数目标。最佳重构函数会在存在数据点的任意处穿过恒等函数。图底部的水平箭头表示在输入空间中基于箭头的  $r(x) - x$  重建方向向量，总是指向最近的“流形”（1 维情况下的单个数据点）。在数据点周围，去噪自编码器明确地尝试将重构函数  $r(x)$  的导数限制为很小。收缩自编码器的编码器执行相同操作。虽然在数据点周围， $r(x)$  的导数被要求很小，但在数据点之间它可能会很大。数据点之间的空间对应于流形之间的区域，为将损坏点映射回流形，重构函数必须具有大的导数。

为了理解自编码器可用于流形学习的原因，我们可以将自编码器和其他方法进行对比。学习表征流形最常见的是流形上（或附近）数据点的表示（representation）。对于特定的实例，这样的表示也被称为嵌入。它通常由一个低维向量给出，具有比这个流形的“外围”空间更少的维数。有些算法（下面讨论的非参数流形学习算法）直

接学习每个训练样例的嵌入，而其他算法学习更一般的映射（有时被称为编码器或表示函数），将周围空间（输入空间）的任意点映射到它的嵌入。

流形学习大多专注于试图捕捉到这些流形的无监督学习过程。最初的学习非线性流形的机器学习研究专注基于最近邻图（nearest neighbor graph）的非参数（non-parametric）方法。该图中每个训练样例对应一个节点，它的边连接近邻点对。如图 14.8 所示，这些方法（Schölkopf *et al.*, 1998b; Roweis and Saul, 2000; Tenenbaum *et al.*, 2000; Brand, 2003b; Belkin and Niyogi, 2003a; Donoho and Grimes, 2003; Weinberger and Saul, 2004b; Hinton and Roweis, 2003; van der Maaten and Hinton, 2008）将每个节点与张成实例和近邻之间的差向量变化方向的切平面相关联。

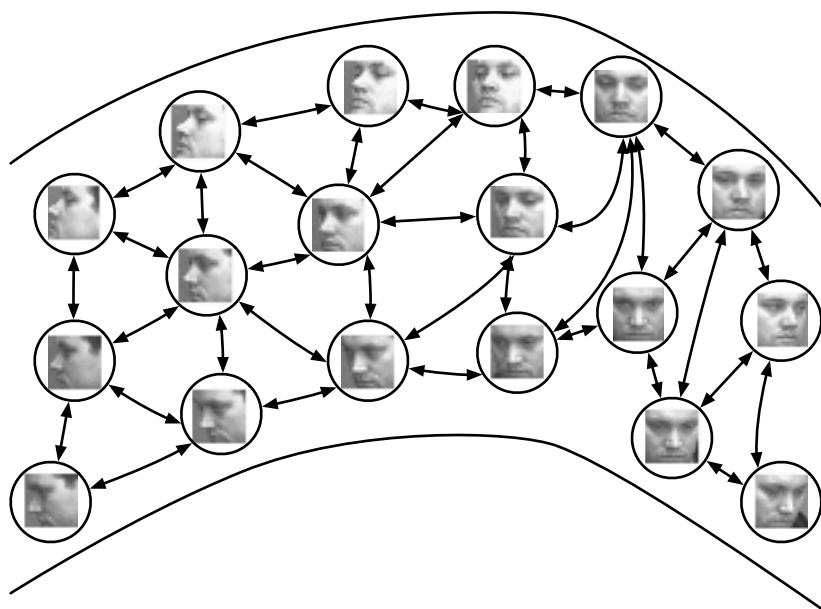


图 14.8：非参数流形学习过程构建的最近邻图，其中节点表示训练样本，有向边指示最近邻关系。因此，各种过程可以获得与图的邻域相关联的切平面以及将每个训练样本与实值向量位置或嵌入（embedding）相关联的坐标系。我们可以通过插值将这种表示概括为新的样本。只要样本的数量大到足以覆盖流形的弯曲和扭转，这些方法工作良好。图片来自 QMUL 多角度人脸数据集（Gong *et al.*, 2000）。

全局坐标系则可以通过优化或求解线性系统获得。图 14.9 展示了如何通过大量局部线性的类高斯样平铺（或“薄煎饼”，因为高斯块在切平面方向是扁平的）得到一个流形。

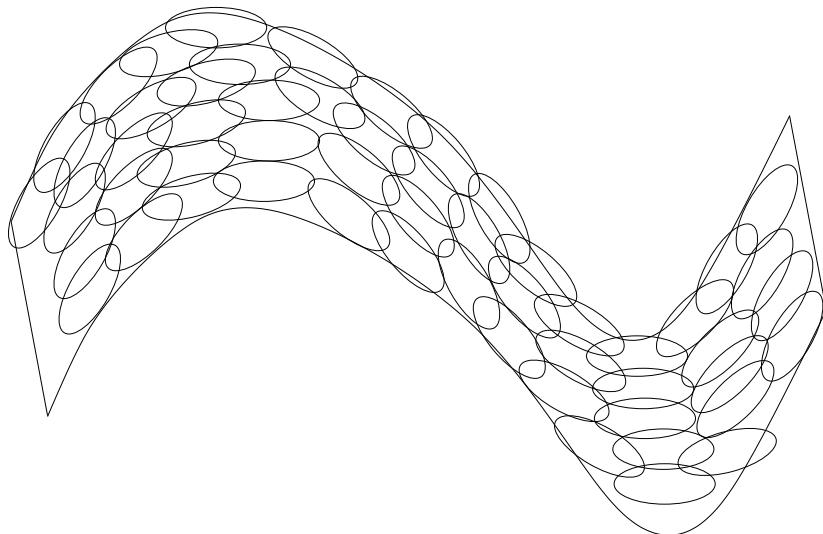


图 14.9: 如果每个位置处的切平面 (见图 14.6) 是已知的, 则它们可以平铺后形成全局坐标系或密度函数。每个局部块可以被认为是局部欧几里德坐标系或者是局部平面高斯或“薄饼”, 在与薄饼正交的方向上具有非常小的方差而在定义坐标系的方向上具有非常大的方差。这些高斯的混合提供了估计的密度函数, 如流形中的 Parzen 窗口算法 (Vincent and Bengio, 2003) 或其非局部的基于神经网络的变体 (Bengio *et al.*, 2006b)。

然而, Bengio and Monperrus (2005) 指出了这些局部非参数方法应用于流形学习的根本困难: 如果流形不是很光滑 (它们有许多波峰、波谷和曲折), 为覆盖其中的每一个变化, 我们可能需要非常多的训练样本, 导致没有能力泛化到没见过的变化。实际上, 这些方法只能通过内插, 概括相邻实例之间流形的形状。不幸的是, AI 问题中涉及的流形可能具有非常复杂的结构, 难以仅从局部插值捕获特征。考虑图 14.6 转换所得的流形样例。如果我们只观察输入向量内的一个坐标  $x_i$ , 当平移图像, 我们可以观察到当这个坐标遇到波峰或波谷时, 图像的亮度也会经历一个波峰或波谷。换句话说, 底层图像模板亮度的模式复杂性决定执行简单的图像变换所产生的流形的复杂性。这是采用分布式表示和深度学习捕获流形结构的动机。

## 14.7 收缩自编码器

收缩自编码器 (Rifai *et al.*, 2011a,b) 在编码  $\mathbf{h} = f(\mathbf{x})$  的基础上添加了显式的正则项，鼓励  $f$  的导数尽可能小：

$$\Omega(\mathbf{h}) = \lambda \left\| \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right\|_F^2. \quad (14.18)$$

惩罚项  $\Omega(\mathbf{h})$  为平方 Frobenius 范数（元素平方之和），作用于与编码器的函数相关偏导数的 Jacobian 矩阵。

去噪自编码器和收缩自编码器之间存在一定联系：Alain and Bengio (2013) 指出在小高斯噪声的限制下，当重构函数将  $\mathbf{x}$  映射到  $\mathbf{r} = g(f(\mathbf{x}))$  时，去噪重构误差与收缩惩罚项是等价的。换句话说，去噪自编码器能抵抗小且有限的输入扰动，而收缩自编码器使特征提取函数能抵抗极小的输入扰动。

分类任务中，基于 Jacobian 的收缩惩罚预训练特征函数  $f(\mathbf{x})$ ，将收缩惩罚应用在  $f(\mathbf{x})$  而不是  $g(f(\mathbf{x}))$  可以产生最好的分类精度。如第 14.5.1 节所讨论，应用于  $f(\mathbf{x})$  的收缩惩罚与得分匹配也有紧密的联系。

**收缩** (contractive) 源于 CAE 弯曲空间的方式。具体来说，由于 CAE 训练为抵抗输入扰动，鼓励将输入点邻域映射到输出点处更小的邻域。我们能认为这是将输入的邻域收缩到更小的输出邻域。

说得更清楚一点，CAE 只在局部收缩——一个训练样本  $\mathbf{x}$  的所有扰动都映射到  $f(\mathbf{x})$  的附近。全局来看，两个不同的点  $\mathbf{x}$  和  $\mathbf{x}'$  会分别被映射到远离原点的两个点  $f(\mathbf{x})$  和  $f(\mathbf{x}')$ 。 $f$  扩展到数据流形的中间或远处是合理的（见图 14.7 中小例子的情况）。当  $\Omega(\mathbf{h})$  惩罚应用于 sigmoid 单元时，收缩 Jacobian 的简单方式是令 sigmoid 趋向饱和的 0 或 1。这鼓励 CAE 使用 sigmoid 的极值编码输入点，或许可以解释为二进制编码。它也保证了 CAE 可以穿过大部分 sigmoid 隐藏单元能张成的超立方体，进而扩散其编码值。

我们可以认为点  $\mathbf{x}$  处的 Jacobian 矩阵  $\mathbf{J}$  能将非线性编码器近似为线性算子。这允许我们更形式地使用“收缩”这个词。在线性理论中，当  $\mathbf{J}\mathbf{x}$  的范数对于所有单位  $\mathbf{x}$  都小于等于 1 时， $\mathbf{J}$  被称为收缩的。换句话说，如果  $\mathbf{J}$  收缩了单位球，他就是收缩的。我们可以认为 CAE 为鼓励每个局部线性算子具有收缩性，而在每个训练数据点处将 Frobenius 范数作为  $f(\mathbf{x})$  的局部线性近似的惩罚。

如第 14.6 节中描述，正则自编码器基于两种相反的推动力学习流形。在 CAE 的

情况下，这两种推动力是重构误差和收缩惩罚  $\Omega(\mathbf{h})$ 。单独的重构误差鼓励 CAE 学习一个恒等函数。单独的收缩惩罚将鼓励 CAE 学习关于  $\mathbf{x}$  是恒定的特征。这两种推动力的折衷产生导数  $\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}}$  大多是微小的自编码器。只有少数隐藏单元，对应于一小部分输入数据的方向，可能有显著的导数。

CAE 的目标是学习数据的流形结构。使  $\mathbf{J}\mathbf{x}$  很大的方向  $\mathbf{x}$ ，会快速改变  $\mathbf{h}$ ，因此很可能是近似流形切平面的方向。Rifai *et al.* (2011a,b) 的实验显示训练 CAE 会导致  $\mathbf{J}$  中大部分奇异值（幅值）比 1 小，因此是收缩的。然而，有些奇异值仍然比 1 大，因为重构误差的惩罚鼓励 CAE 对最大局部变化的方向进行编码。对应于最大奇异值的方向被解释为收缩自编码器学到的切方向。理想情况下，这些切方向对应于数据的真实变化。比如，一个应用于图像的 CAE 应该能学到显示图像改变的切向量，如图 14.6 图中物体渐渐改变状态。如图 14.10 所示，实验获得的奇异向量的可视化似乎真的对应于输入图象有意义的变换。

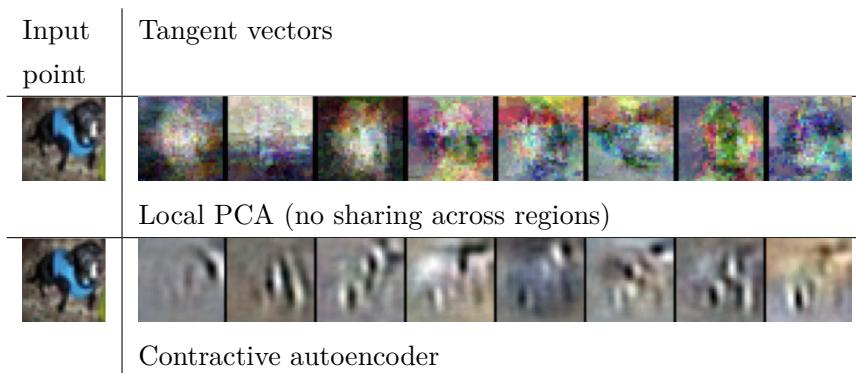


图 14.10：通过局部 PCA 和收缩自编码器估计的流形切向量的图示。流形的位置由来自 CIFAR-10 数据集中狗的输入图像定义。切向量通过输入到代码映射的 Jacobian 矩阵  $\frac{\partial \mathbf{h}}{\partial \mathbf{x}}$  的前导奇异向量估计。虽然局部 PCA 和 CAE 都可以捕获局部切方向，但 CAE 能够从有限训练数据形成更准确的估计，因为它利用了不同位置的参数共享（共享激活的隐藏单元子集）。CAE 切方向通常对应于物体的移动或改变部分（例如头或腿）。经 Rifai *et al.* (2011c) 许可转载此图。

收缩自编码器正则化准则的一个实际问题是，尽管它在单一隐藏层的自编码器情况下是容易计算的，但在更深的自编码器情况下会变的难以计算。根据 Rifai *et al.* (2011a) 的策略，分别训练一系列单层的自编码器，并且每个被训练为重构前一个自编码器的隐藏层。这些自编码器的组合就组成了一个深度自编码器。因为每个层分别训练成局部收缩，深度自编码器自然也是收缩的。这个结果与联合训练深度模型完整架构（带有关于 Jacobian 的惩罚项）获得的结果是不同的，但它抓住了

许多理想的定性特征。

另一个实际问题是，如果我们不对解码器强加一些约束，收缩惩罚可能导致无用的结果。例如，编码器将输入乘一个小常数  $\epsilon$ ，解码器将编码除以一个小常数  $\epsilon$ 。随着  $\epsilon$  趋向于 0，编码器会使收缩惩罚项  $\Omega(\mathbf{h})$  趋向于 0 而学不到任何关于分布的信息。同时，解码器保持完美的重构。Rifai *et al.* (2011a) 通过绑定  $f$  和  $g$  的权重来防止这种情况。 $f$  和  $g$  都是由线性仿射变换后进行逐元素非线性变换的标准神经网络层组成，因此将  $g$  的权重矩阵设成  $f$  权重矩阵的转置是很直观的。

## 14.8 预测稀疏分解

预测稀疏分解 (predictive sparse decomposition, PSD) 是稀疏编码和参数化自编码器 (Kavukcuoglu *et al.*, 2008) 的混合模型。参数化编码器被训练为能预测迭代推断的输出。PSD 被应用于图片和视频中对象识别的无监督特征学习 (Kavukcuoglu *et al.*, 2009, 2010; Jarrett *et al.*, 2009b; Farabet *et al.*, 2011)，在音频中也有所应用 (Henaff *et al.*, 2011)。这个模型由一个编码器  $f(\mathbf{x})$  和一个解码器  $g(\mathbf{h})$  组成，并且都是参数化的。在训练过程中， $\mathbf{h}$  由优化算法控制。优化过程是最小化

$$\|\mathbf{x} - g(\mathbf{h})\|^2 + \lambda |\mathbf{h}|_1 + \gamma \|\mathbf{h} - f(\mathbf{x})\|^2. \quad (14.19)$$

就像稀疏编码，训练算法交替地相对  $\mathbf{h}$  和模型的参数最小化上述目标。相对  $\mathbf{h}$  最小化较快，因为  $f(\mathbf{x})$  提供  $\mathbf{h}$  的良好初始值以及损失函数将  $\mathbf{h}$  约束在  $f(\mathbf{x})$  附近。简单的梯度下降算法只需 10 步左右就能获得理想的  $\mathbf{h}$ 。

PSD 所使用的训练程序不是先训练稀疏编码模型，然后训练  $f(\mathbf{x})$  来预测稀疏编码的特征。PSD 训练过程正则化解码器，使用  $f(\mathbf{x})$  可以推断出良好编码的参数。

预测稀疏分解是 **学习近似推断** (learned approximate inference) 的一个例子。在第 19.5 节中，这个话题将会进一步展开。第十九章中展示的工具能让我们了解到，PSD 能够被解释为通过最大化模型的对数似然下界训练有向稀疏编码的概率模型。

在 PSD 的实际应用中，迭代优化仅在训练过程中使用。模型被部署后，参数编码器  $f$  用于计算已经习得的特征。相比通过梯度下降推断  $\mathbf{h}$ ，计算  $f$  是很容易的。因为  $f$  是一个可微带参函数，PSD 模型可堆叠，并用于初始化其他训练准则的深度网络。

## 14.9 自编码器的应用

自编码器已成功应用于降维和信息检索任务。降维是表示学习和深度学习的第一批应用之一。它是研究自编码器早期驱动力之一。例如，Hinton and Salakhutdinov (2006) 训练了一个栈式 RBM，然后利用它们的权重初始化一个隐藏层逐渐减小的深度自编码器，终结于 30 个单元的瓶颈。生成的编码比 30 维的 PCA 产生更少的重构误差，所学到的表示更容易定性解释，并能联系基础类别，这些类别表现为分离良好的集群。

低维表示可以提高许多任务的性能，例如分类。小空间的模型消耗更少的内存和运行时间。据 Salakhutdinov and Hinton (2007b) 和 Torralba *et al.* (2008) 观察，许多降维的形式会将语义上相关的样本置于彼此邻近的位置。映射到低维空间所提供的线索有助于泛化。

相比普通任务，**信息检索** (information retrieval) 从降维中获益更多，此任务需要找到数据库中类似查询的条目。此任务不仅和其他任务一样从降维中获得一般益处，还使某些低维空间中的搜索变得极为高效。特别的，如果我们训练降维算法生成一个低维且二值的编码，那么我们就可以将所有数据库条目在哈希表映射为二值编码向量。这个哈希表允许我们返回具有相同二值编码的数据库条目作为查询结果进行信息检索。我们也可以非常高效地搜索稍有不同条目，只需反转查询编码的各个位。这种通过降维和二值化的信息检索方法被称为**语义哈希** (semantic hashing) (Salakhutdinov and Hinton, 2007b, 2009b)，已经被用于文本输入 (Salakhutdinov and Hinton, 2007b, 2009b) 和图像 (Torralba *et al.*, 2008; Weiss *et al.*, 2008; Krizhevsky and Hinton, 2011)。

通常在最终层上使用 sigmoid 编码函数产生语义哈希的二值编码。sigmoid 单元必须被训练为到达饱和，对所有输入值都接近 0 或接近 1。能做到这一点的窍门就是训练时在 sigmoid 非线性单元前简单地注入加性噪声。噪声的大小应该随时间增加。要对抗这种噪音并且保存尽可能多的信息，网络必须加大输入到 sigmoid 函数的幅度，直到饱和。

学习哈希函数的思想已在其他多个方向进一步探讨，包括改变损失训练表示的想法，其中所需优化的损失与哈希表中查找附近样本的任务有更直接的联系 (Norouzi and Fleet, 2011)。

# 第十五章 表示学习

在本章中，首先我们会讨论学习表示是什么意思，以及表示的概念如何有助于深度框架的设计。我们探讨学习算法如何在不同任务中共享统计信息，包括使用无监督任务中的信息来完成监督任务。共享表示有助于处理多模式或多领域，或是将已学到的知识迁移到样本很少或没有、但任务表示依然存在的任务上。最后，我们回过头探讨表示学习成功的原因，从分布式表示 (Hinton *et al.*, 1986) 和深度表示的理论优势，最后会讲到数据生成过程潜在假设的更一般概念，特别是观测数据的基本成因。

很多信息处理任务可能非常容易，也可能非常困难，这取决于信息是如何表示的。这是一个广泛适用于日常生活、计算机科学及机器学习的基本原则。例如，对于人而言，可以直接使用长除法计算 210 除以 6。但如果使用罗马数字表示，这个问题就没那么直接了。大部分现代人在使用罗马数字计算 CCX 除以 VI 时，都会将其转化成阿拉伯数字，从而使用位值系统的长除法。更具体地，我们可以使用合适或不合适的表示来量化不同操作的渐近运行时间。例如，插入一个数字到有序表中的正确位置，如果该数列表示为链表，那么所需时间是  $O(n)$ ；如果该列表表示为红黑树，那么只需要  $O(\log n)$  的时间。

在机器学习中，到底是什么因素决定了一种表示比另一种表示更好呢？一般而言，一个好的表示可以使后续的学习任务更容易。选择什么表示通常取决于后续的学习任务。

我们可以将监督学习训练的前馈网络视为表示学习的一种形式。具体地，网络的最后一层通常是线性分类器，如 softmax 回归分类器。网络的其余部分学习出该分类器的表示。监督学习训练模型，一般会使得模型的各个隐藏层（特别是接近顶层的隐藏层）的表示能够更加容易地完成训练任务。例如，输入特征线性不可分的类别可能在最后一个隐藏层变成线性可分离的。原则上，最后一层可以是另一种模

型，如最近邻分类器 (Salakhutdinov and Hinton, 2007a)。倒数第二层的特征应该根据最后一层的类型学习不同的性质。

前馈网络的监督训练并没有给学成的中间特征明确强加任何条件。其他的表示学习算法往往会以某种特定的方式明确设计表示。例如，我们想要学习一种使得密度估计更容易的表示。具有更多独立性的分布会更容易建模，因此，我们可以设计鼓励表示向量  $\mathbf{h}$  中元素之间相互独立的目标函数。就像监督网络，无监督深度学习算法有一个主要的训练目标，但也额外地学习出了表示。不论该表示是如何得到的，它都可以用于其他任务。或者，多个任务（有些是监督的，有些是无监督的）可以通过共享的内部表示一起学习。

大多数表示学习算法都会在尽可能多地保留与输入相关的信息和追求良好的性质（如独立性）之间作出权衡。

表示学习特别有趣，因为它提供了进行无监督学习和半监督学习的一种方法。我们通常会有巨量的未标注训练数据和相对较少的标注训练数据。在非常有限的标注数据集上监督学习通常会导致严重的过拟合。半监督学习通过进一步学习未标注数据，来解决过拟合的问题。具体地，我们可以从未标注数据上学习出很好的表示，然后用这些表示来解决监督学习问题。

人类和动物能够从非常少的标注样本中学习。我们至今仍不知道这是如何做到的。有许多假说解释人类的卓越学习能力——例如，大脑可能使用了大量的分类器或者贝叶斯推断技术的集成。一种流行的假说是，大脑能够利用无监督学习和半监督学习。利用未标注数据有多种方式。在本章中，我们主要使用的假说是未标注数据可以学习出良好的表示。

## 15.1 贪心逐层无监督预训练

无监督学习在深度神经网络的复兴上起到了关键的、历史性的作用，它使研究者首次可以训练不含诸如卷积或者循环这类特殊结构的深度监督网络。我们将这一过程称为 **无监督预训练** (unsupervised pretraining)，或者更精确地，**贪心逐层无监督预训练** (greedy layer-wise unsupervised pretraining)。此过程是一个任务（无监督学习，尝试获取输入分布的形状）的表示如何有助于另一个任务（具有相同输入域的监督学习）的典型示例。

贪心逐层无监督预训练依赖于单层表示学习算法，例如 RBM、单层自编码器、

稀疏编码模型或其他学习潜在表示的模型。每一层使用无监督学习预训练，将前一层的输出作为输入，输出数据的新的表示。这个新的表示的分布（或者是和其他变量比如要预测类别的关系）有可能是更简单的。如算法 15.1 所示的正式表述。

---

### 算法 15.1 贪心逐层无监督预训练的协定

给定如下：无监督特征学习算法  $\mathcal{L}$ ， $\mathcal{L}$  使用训练集样本并返回编码器或特征函数  $f$ 。原始输入数据是  $\mathbf{X}$ ，每行一个样本，并且  $f^{(1)}(\mathbf{X})$  是第一阶段编码器关于  $\mathbf{X}$  的输出。在执行精调的情况下，我们使用学习者  $\mathcal{T}$ ，并使用初始函数  $f$ ，输入样本  $\mathbf{X}$ （以及在监督精调情况下关联的目标  $\mathbf{Y}$ ），并返回细调好函数。阶段数为  $m$ 。

---

```

 $f \leftarrow$  恒等函数
 $\tilde{\mathbf{X}} = \mathbf{X}$ 
for  $k = 1, \dots, m$  do
     $f^{(k)} = \mathcal{L}(\tilde{\mathbf{X}})$ 
     $f \leftarrow f^{(k)} \circ f$ 
     $\tilde{\mathbf{X}} \leftarrow f^{(k)}(\tilde{\mathbf{X}})$ 
end for
if fine-tuning then
     $f \leftarrow \mathcal{T}(f, \mathbf{X}, \mathbf{Y})$ 
end if
Return  $f$ 
```

---

基于无监督标准的贪心逐层训练过程，早已被用来规避监督问题中深度神经网络难以联合训练多层的问题。这种方法至少可以追溯神经认知机 (Fukushima, 1975)。深度学习的复兴始于 2006 年，源于发现这种贪心学习过程能够为多层联合训练过程找到一个好的初始值，甚至可以成功训练全连接的结构 (Hinton *et al.*, 2006b; Hinton and Salakhutdinov, 2006; Hinton, 2006; Bengio *et al.*, 2007d; Ranzato *et al.*, 2007a)。在此发现之前，只有深度卷积网络或深度循环网络这类特殊结构的深度网络被认为是有希望训练的。现在我们知道训练具有全连接的深度结构时，不再需要使用贪心逐层无监督预训练，但无监督预训练是第一个成功的方法。

贪心逐层无监督预训练被称为 **贪心** (greedy) 的，是因为它是一个 **贪心算法** (greedy algorithm)，这意味着它独立地优化解决方案的每一个部分，每一步解决一个部分，而不是联合优化所有部分。它被称为 **逐层的** (layer-wise)，是因为这些独立的解决方案是网络层。具体地，贪心逐层无监督预训练每次处理一层网络，训练第  $k$

层时保持前面的网络层不变。特别地，低层网络（最先训练的）不会在引入高层网络后进行调整。它被称为 **无监督** (unsupervised) 的，是因为每一层用无监督表示学习算法训练。然而，它也被称为 **预训练** (pretraining)，是因为它只是在联合训练算法 **精调** (fine-tune) 所有层之前的第一步。在监督学习任务中，它可以被看作是正则化项（在一些实验中，预训练不能降低训练误差，但能降低测试误差）和参数初始化的一种形式。

通常而言，“**预训练**”不仅单指预训练阶段，也指结合预训练和监督学习的两阶段学习过程。监督学习阶段可能会使用预训练阶段得到的顶层特征训练一个简单分类器，或者可能会对预训练阶段得到的整个网络进行监督精调。不管采用什么类型的监督学习算法和模型，在大多数情况下，整个训练过程几乎是相同的。虽然无监督学习算法的选择将明显影响到细节，但是大多数无监督预训练应用都遵循这一基本方法。

贪心逐层无监督预训练也能用作其他无监督学习算法的初始化，比如深度自编码器 (Hinton and Salakhutdinov, 2006) 和具有很多潜变量层的概率模型。这些模型包括深度信念网络 (Hinton *et al.*, 2006b) 和深度玻尔兹曼机 (Salakhutdinov and Hinton, 2009a)。这些深度生成模型会在第二十章中讨论。

正如第 8.7.4 节所探讨的，我们也可以进行贪心逐层监督预训练。这是建立在训练浅层模型比深度模型更容易的前提下，而该前提似乎在一些情况下已被证实 (Erhan *et al.*, 2010)。

### 15.1.1 何时以及为何无监督预训练有效？

在很多分类任务中，贪心逐层无监督预训练能够在测试误差上获得重大提升。这一观察结果始于 2006 年对深度神经网络的重新关注 (Hinton *et al.*, 2006b; Bengio *et al.*, 2007d; Ranzato *et al.*, 2007a)。然而，在很多其他问题上，无监督预训练不能带来改善，甚至还会带来明显的负面影响。Ma *et al.* (2015) 研究了预训练对机器学习模型在化学活性预测上的影响。结果发现，平均而言预训练是有轻微负面影响的，但在有些问题上会有显著帮助。由于无监督预训练有时有效，但经常也会带来负面效果，因此很有必要了解它何时有效以及有效的因素，以确定它是否适合用于特定的任务。

首先，要注意的是这个讨论大部分都是针对贪心无监督预训练而言。还有很多其他完全不同的方法使用半监督学习来训练神经网络，比如第 7.13 节介绍的虚拟对抗

训练。我们还可以在训练监督模型的同时训练自编码器或生成模型。这种单阶段方法的例子包括判别 RBM (Larochelle and Bengio, 2008b) 和梯形网络 (Rasmus *et al.*, 2015)，其中整体目标是两项之和（一个使用标签，另一个仅仅使用输入）。

无监督预训练结合了两种不同的想法。第一，它利用了深度神经网络对初始参数的选择，可以对模型有着显著的正则化效果（在较小程度上，可以改进优化）的想法。第二，它利用了更一般的想法——学习输入分布有助于学习从输入到输出的映射。

这两个想法都涉及到机器学习算法中多个未能完全理解的部分之间复杂的相互作用。

第一个想法，即深度神经网络初始参数的选择对其性能具有很强的正则化效果，很少有关于这个想法的理解。在预训练变得流行时，在一个位置初始化模型被认为会使其接近某一个局部极小点，而不是另一个局部极小点。如今，局部极小值不再被认为是神经网络优化中的严重问题。现在我们知道标准的神经网络训练过程通常不会到达任何形式的临界点。仍然可能的是，预训练会初始化模型到一个可能不会到达的位置——例如，某种区域，其中代价函数从一个样本点到另一个样本点变化很大，而小批量只能提供噪声严重的梯度估计，或是某种区域中的 Hessian 矩阵条件数是病态的，梯度下降必须使用非常小的步长。然而，我们很难准确判断监督学习期间预训练参数的哪些部分应该保留。这是现代方法通常同时使用无监督学习和监督学习，而不是依序使用两个学习阶段的原因之一。除了这些复杂的方法可以让监督学习阶段保持无监督学习阶段提取的信息之外，还有一种简单的方法，固定特征提取器的参数，仅仅将监督学习作为顶层学成特征的分类器。

另一个想法有更好的理解，即学习算法可以使用无监督阶段学习的信息，在监督学习的阶段表现得更好。其基本想法是对于无监督任务有用的一些特征对于监督学习任务也可能是有用的。例如，如果我们训练汽车和摩托车图像的生成模型，它需要知道轮子的概念，以及一张图中应该有多少个轮子。如果我们幸运的话，无监督阶段学习的轮子表示会适合于监督学习。然而我们还未能从数学、理论层面上证明，因此并不总是能够预测哪种任务能以这种形式从无监督学习中受益。这种方法的许多方面高度依赖于具体使用的模型。例如，如果我们希望在预训练特征的顶层添加线性分类器，那么（学到的）特征必须使潜在的类别是线性可分离的。这些性质通常会在无监督学习阶段自然发生，但也并非总是如此。这是另一个监督和无监督学习同时训练更可取的原因——输出层施加的约束很自然地从一开始就包括在内。

从无监督预训练作为学习一个表示的角度来看，我们可以期望无监督预训练在初始表示较差的情况下更有效。一个重要的例子是词嵌入。使用 one-hot 向量表示的词并不具有很多信息，因为任意两个不同的 one-hot 向量之间的距离（平方  $L^2$  距离都是 2）都是相同的。学成的词嵌入自然会用它们彼此之间的距离来编码词之间的相似性。因此，无监督预训练在处理单词时特别有用。然而在处理图像时是不太有用的，可能是因为图像已经在一个很丰富的向量空间中，其中的距离只能提供低质量的相似性度量。

从无监督预训练作为正则化项的角度来看，我们可以期望无监督预训练在标注样本数量非常小时很有帮助。因为无监督预训练添加的信息来源于未标注数据，所以当未标注样本的数量非常大时，我们也可以期望无监督预训练的效果最好。无监督预训练的大量未标注样本和少量标注样本构成的半监督学习的优势特别明显。在 2011 年，无监督预训练赢得了两个国际迁移学习比赛 (Mesnil *et al.*, 2011; Goodfellow *et al.*, 2011)。在该情景中，目标任务中标注样本的数目很少（每类几个到几十个）。这些效果也出现在被 Paine *et al.* (2014) 严格控制的实验中。

还可能涉及到一些其他的因素。例如，当我们要学习的函数非常复杂时，无监督预训练可能会非常有用。无监督学习不同于权重衰减这样的正则化项，它不偏向于学习一个简单的函数，而是学习对无监督学习任务有用的特征函数。如果真实的潜在函数是复杂的，并且由输入分布的规律塑造，那么无监督学习更适合作为正则化项。

除了这些注意事项外，我们现在分析一些无监督预训练改善性能的成功示例，并解释这种改进发生的已知原因。无监督预训练通常用来改进分类器，并且从减少测试集误差的观点来看是很有意思的。然而，无监督预训练还有助于分类以外的任务，并且可以用于改进优化，而不仅仅只是作为正则化项。例如，它可以提高去噪自编码器的训练和测试重构误差 (Hinton and Salakhutdinov, 2006)。

Erhan *et al.* (2010) 进行了许多实验来解释无监督预训练的几个成功原因。对训练误差和测试误差的改进都可以解释为，无监督预训练将参数引入到了其他方法可能探索不到的区域。神经网络训练是非确定性的，并且每次运行都会收敛到不同的函数。训练可以停止在梯度很小的点；也可以提前终止结束训练，以防过拟合；还可以停止在梯度很大，但由于诸如随机性或 Hessian 矩阵病态条件等问题难以找到合适下降方向的点。经过无监督预训练的神经网络会一致地停止在一片相同的函数空间区域，但未经过预训练的神经网络会一致地停在另一个区域。图 15.1 可视化了这种现象。经过预训练的网络到达的区域是较小的，这表明预训练减少了估计过程的

方差，这进而又可以降低严重过拟合的风险。换言之，无监督预训练将神经网络参数初始化到它们不易逃逸的区域，并且遵循这种初始化的结果更加一致，和没有这种初始化相比，结果很差的可能性更低。

Erhan *et al.* (2010) 也回答了何时预训练效果最好——预训练的网络越深，测试误差的均值和方差下降得越多。值得注意的是，这些实验是在训练非常深层网络的现代方法发明和流行（整流线性单元，Dropout 和批标准化）之前进行的，因此对于无监督预训练与当前方法的结合，我们所知甚少。

一个重要的问题是无监督预训练是如何起到正则化项作用的。一个假设是，预训练鼓励学习算法发现那些与生成观察数据的潜在原因相关的特征。这也是启发除无监督预训练之外许多其他算法的重要思想，将会在第 15.3 节中进一步讨论。

与无监督学习的其他形式相比，无监督预训练的缺点是其使用了两个单独的训练阶段。很多正则化技术都具有一个优点，允许用户通过调整单一超参数的值来控制正则化的强度。无监督预训练没有一种明确的方法来调整无监督阶段正则化的强度。相反，无监督预训练有许多超参数，但其效果只能之后度量，通常难以提前预测。当我们同时执行无监督和监督学习而不使用预训练策略时，会有单个超参数（通常是附加到无监督代价的系数）控制无监督目标正则化监督模型的强度。减少该系数，总是能够可预测地获得较少正则化强度。在无监督预训练的情况下，没有一种灵活调整正则化强度的方式——要么监督模型初始化为预训练的参数，要么不是。

具有两个单独的训练阶段的另一个缺点是每个阶段都具有各自的超参数。第二阶段的性能通常不能在第一阶段期间预测，因此在第一阶段提出超参数和第二阶段根据反馈来更新之间存在较长的延迟。最通用的方法是在监督阶段使用验证集上的误差来挑选预训练阶段的超参数，如 Larochelle *et al.* (2009) 中讨论的。在实际中，有些超参数，如预训练迭代的次数，很方便在预训练阶段设定，通过无监督目标上使用提前终止策略完成。这个策略并不理想，但是在计算上比使用监督目标代价小得多。

如今，大部分算法已经不使用无监督预训练了，除了在自然语言处理领域中单词作为 one-hot 向量的自然表示不能传达相似性信息，并且有非常多的未标注数据集可用。在这种情况下，预训练的优点是可以对一个巨大的未标注集合（例如用包含数十亿单词的语料库）进行预训练，学习良好的表示（通常是单词，但也可以是句子），然后使用该表示或精调它，使其适合于训练集样本大幅减少的监督任务。这种方法由 Collobert and Weston (2008b)、Turian *et al.* (2010) 和 Collobert *et al.* (2011a)

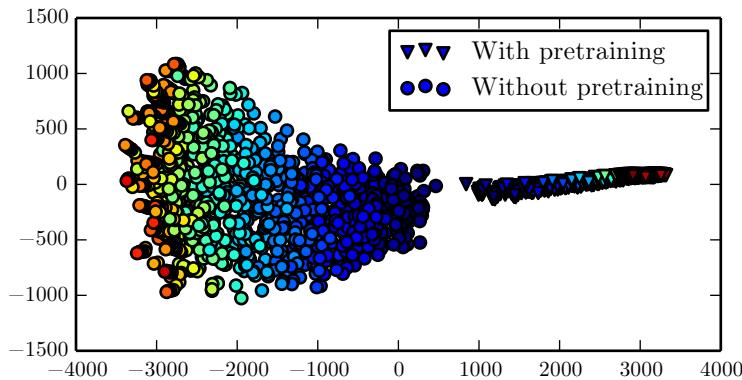


图 15.1: 在函数空间（并非参数空间，避免从参数向量到函数的多对一映射）不同神经网络的学习轨迹的非线性映射的可视化。不同网络采用不同的随机初始化，并且有的使用了无监督预训练，有的没有。每个点对应着训练过程中一个特定时间的神经网络。经Erhan *et al.* (2010) 许可改编此图。函数空间中的坐标是关于每组输入  $x$  和它的一个输出  $y$  的无限维向量。Erhan *et al.* (2010) 将很多特定  $x$  的  $y$  连接起来，线性投影到高维空间中。然后他们使用 Isomap (Tenenbaum *et al.*, 2000) 进行进一步的非线性投影并投到二维空间。颜色表示时间。所有的网络初始化在上图的中心点附近（对应的函数区域在不多数输入上具有近似均匀分布的类别  $y$ ）。随着时间推移，学习将函数向外移动到预测得更好的点。当使用预训练时，训练会一致地收敛到同一个区域；而不使用预训练时，训练会收敛到另一个不重叠的区域。Isomap 试图维持全局相对距离（体积因此也保持不变），因此使用预训练的模型对应的较小区域意味着，基于预训练的估计具有较小的方差。

开创，至今仍在使用。

基于监督学习的深度学习技术，通过 Dropout 或批标准化来正则化，能够在很多任务上达到人类级别的性能，但仅仅是在极大的标注数据集上。在中等大小的数据集（例如 CIFAR-10 和 MNIST，每个类大约有 5,000 个标注样本）上，这些技术的效果比无监督预训练更好。在极小的数据集，例如选择性剪接数据集，贝叶斯方法要优于基于无监督预训练的方法 (Srivastava, 2013)。由于这些原因，无监督预训练已经不如以前流行。然而，无监督预训练仍然是深度学习研究历史上的一个重要里程碑，并将继续影响当代方法。预训练的想法已经推广到 **监督预训练** (supervised pretraining)，这将在第 8.7.4 节中讨论，在迁移学习中这是非常常用的方法。迁移学习中的监督预训练流行 (Oquab *et al.*, 2014; Yosinski *et al.*, 2014) 于在 ImageNet 数据集上使用卷积网络预训练。由于这个原因，实践者们公布了这些网络训练出的参数，就像自然语言任务公布预训练的单词向量一样 (Collobert *et al.*, 2011a; Mikolov

*et al.*, 2013a)。

## 15.2 迁移学习和领域自适应

迁移学习和领域自适应指的是利用一个情景（例如，分布  $P_1$ ）中已经学到的内容去改善另一个情景（比如分布  $P_2$ ）中的泛化情况。这点概括了上一节提出的想法，即在无监督学习任务和监督学习任务之间转移表示。

在 **迁移学习** (transfer learning) 中，学习器必须执行两个或更多个不同的任务，但是我们假设能够解释  $P_1$  变化的许多因素和学习  $P_2$  需要抓住的变化相关。这通常能够在监督学习中解释，输入是相同的，但是输出不同的性质。例如，我们可能在第一种情景中学习了一组视觉类别，比如猫和狗，然后在第二种情景中学习一组不同的视觉类别，比如蚂蚁和黄蜂。如果第一种情景（从  $P_1$  采样）中具有非常多的数据，那么这有助于学习到能够使得从  $P_2$  抽取的非常少样本中快速泛化的表示。许多视觉类别共享一些低级概念，比如边缘、视觉形状、几何变化、光照变化的影响等等。一般而言，当存在对不同情景或任务有用特征时，并且这些特征对应多个情景出现的潜在因素，迁移学习、多任务学习（第 7.7 节）和领域自适应可以使用表示学习来实现。如图 7.2 所示，这是具有共享底层和任务相关上层的学习框架。

然而，有时不同任务之间共享的不是输入的语义，而是输出的语义。例如，语音识别系统需要在输出层产生有效的句子，但是输入附近的较低层可能需要识别相同音素或子音素发音的非常不同的版本（这取决于说话人）。在这样的情况下，共享神经网络的上层（输出附近）和进行任务特定的预处理是有意义的，如图 15.2 所示。

在 **领域自适应** (domain adaption) 的相关情况下，在每个情景之间任务（和最优的输入到输出的映射）都是相同的，但是输入分布稍有不同。例如，考虑情感分析的任务，如判断一条评论是表达积极的还是消极的情绪。网上的评论有许多类别。在书、视频和音乐等媒体内容上训练的顾客评论情感预测器，被用于分析诸如电视机或智能电话的消费电子产品的评论时，领域自适应情景可能会出现。可以想象，存在一个潜在的函数可以判断任何语句是正面的、中性的还是负面的，但是词汇和风格可能会因领域而有差异，使得跨域的泛化训练变得更加困难。简单的无监督预训练（去噪自编码器）已经能够非常成功地用于领域自适应的情感分析 (Glorot *et al.*, 2011c)。

一个相关的问题是 **概念漂移** (concept drift)，我们可以将其视为一种迁移学习，

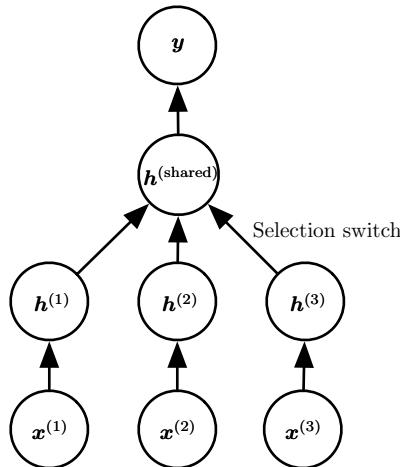


图 15.2: 多任务学习或者迁移学习的架构示例。输出变量  $y$  在所有的任务上具有相同的语义；输入变量  $x$  在每个任务（或者，比如每个用户）上具有不同的意义（甚至可能具有不同的维度），图上三个任务为  $x^{(1)}$ ,  $x^{(2)}$ ,  $x^{(3)}$ 。底层结构（决定了选择方向）是面向任务的，上层结构是共享的。底层结构学习将面向特定任务的输入转化为通用特征。

因为数据分布随时间而逐渐变化。概念漂移和迁移学习都可以被视为多任务学习的特定形式。“多任务学习”这个术语通常指监督学习任务，而更广义的迁移学习的概念也适用于无监督学习和强化学习。

在所有这些情况下，我们的目标是利用第一个情景下的数据，提取那些在第二种情景中学习时或直接进行预测时可能有用的信息。表示学习的核心思想是相同的表示可能在两种情景中都是有用的。两个情景使用相同的表示，使得表示可以受益于两个任务的训练数据。

如前所述，迁移学习中无监督深度学习已经在一些机器学习比赛中取得了成功 (Mesnil *et al.*, 2011; Goodfellow *et al.*, 2011)。这些比赛中的某一个实验配置如下。首先每个参与者获得一个第一种情景（来自分布  $P_1$ ）的数据集，其中含有一些类别的样本。参与者必须使用这个来学习一个良好的特征空间（将原始输入映射到某种表示），使得当我们将这个学成变换用于来自迁移情景（分布  $P_2$ ）的输入时，线性分类器可以在很少标注样本上训练、并泛化得很好。这个比赛中最引人注目的结果之一是，学习表示的网络架构越深（在第一个情景  $P_1$  中的数据使用纯无监督方式学习），在第二个情景（迁移） $P_2$  的新类别上学习到的曲线就越好。对于深度表示而言，迁移任务只需要少量标注样本就能显著地提升泛化性能。

迁移学习的两种极端形式是一次学习 (one-shot learning) 和零次学习 (zero-shot learning)，有时也被称为零数据学习 (zero-data learning)。只有一个标注样本的迁移任务被称为一次学习；没有标注样本的迁移任务被称为零次学习。

因为第一阶段学习出的表示就可以清楚地分离出潜在的类别，所以一次学习 (Fei-Fei *et al.*, 2006) 是可能的。在迁移学习阶段，仅需要一个标注样本来推断表示空间中聚集在相同点周围许多可能测试样本的标签。这使得在学成的表示空间中，对应于不变性的变化因子已经与其他因子完全分离，在区分某些类别的对象时，我们可以学习到哪些因素具有决定意义。

考虑一个零次学习情景的例子，学习器已经读取了大量文本，然后要解决对象识别的问题。如果文本足够好地描述了对象，那么即使没有看到某对象的图像，也能识别出该对象的类别。例如，已知猫有四条腿和尖尖的耳朵，那么学习器可以在没有见过猫的情况下猜测该图像中是猫。

只有在训练时使用了额外信息，零数据学习 (Larochelle *et al.*, 2008) 和零次学习 (Palatucci *et al.*, 2009; Socher *et al.*, 2013b) 才是有可能的。我们可以认为零数据学习场景包含三个随机变量：传统输入  $\mathbf{x}$ ，传统输出或目标  $\mathbf{y}$ ，以及描述任务的附加随机变量  $T$ 。该模型被训练来估计条件分布  $p(\mathbf{y} | \mathbf{x}, T)$ ，其中  $T$  是我们希望执行的任务的描述。在我们的例子中，读取猫的文本信息然后识别猫，输出是二元变量  $y$ ， $y = 1$  表示“是”， $y = 0$  表示“不是”。任务变量  $T$  表示要回答的问题，例如“这个图像中是否有猫？”如果训练集包含和  $T$  在相同空间的无监督对象样本，我们也许能够推断未知的  $T$  实例的含义。在我们的例子中，没有提前看到猫的图像而去识别猫，所以拥有一些未标注文本数据包含句子诸如“猫有四条腿”或“猫有尖耳朵”，对于学习非常有帮助。

零次学习要求  $T$  被表示为某种形式的泛化。例如， $T$  不能仅是指示对象类别的one-hot编码。通过使用每个类别词的词嵌入表示，Socher *et al.* (2013b) 提出了对象类别的分布式表示。

我们还可以在机器翻译中发现一种类似的现象 (Klementiev *et al.*, 2012; Mikolov *et al.*, 2013b; Gouws *et al.*, 2014)：我们已经知道一种语言中的单词，还可以学到单一语言料库中词与词之间的关系；另一方面，我们已经翻译了一种语言中的单词与另一种语言中的单词相关的句子。即使我们可能没有将语言  $X$  中的单词  $A$  翻译成语言  $Y$  中的单词  $B$  的标注样本，我们也可以泛化并猜出单词  $A$  的翻译，这是由于我们已经学习了语言  $X$  和  $Y$  单词的分布式表示，并且通过两种语言句子的匹配

对组成的训练样本，产生了关联于两个空间的链接（可能是双向的）。如果联合学习三种成分（两种表示形式和它们之间的关系），那么这种迁移将会非常成功。

零次学习是迁移学习的一种特殊形式。同样的原理可以解释如何能执行**多模态学习**（multimodal learning），学习两种模态的表示，和一种模态中的观察结果 $x$ 与另一种模态中的观察结果 $y$ 组成的对 $(x, y)$ 之间的关系（通常是一个联合分布）(Srivastava and Salakhutdinov, 2012)。通过学习所有的三组参数（从 $x$ 到它的表示、从 $y$ 到它的表示，以及两个表示之间的关系），一个表示中的概念被锚定在另一个表示中，反之亦然，从而可以有效地推广到新的对组。这个过程如图 15.3 所示。

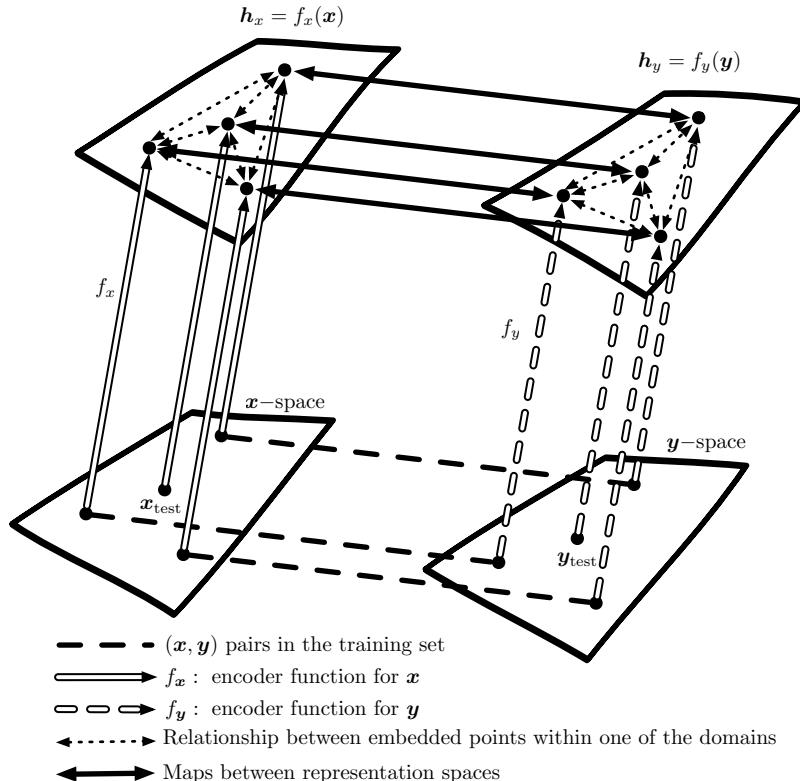


图 15.3: 两个域  $x$  和  $y$  之间的迁移学习能够进行零次学习。标注或未标注样本  $x$  可以学习表示函数  $f_x$ 。同样地，样本  $y$  也可以学习表示函数  $f_y$ 。上图中  $f_x$  和  $f_y$  旁都有一个向上的箭头，不同的箭头表示不同的作用函数。并且箭头的类型表示使用了哪一种函数。 $h_x$  空间中的相似性度量表示  $x$  空间中任意点对之间的距离，这种度量方式比直接度量  $x$  空间的距离更好。同样地， $h_y$  空间中的相似性度量表示  $y$  空间中任意点对之间的距离。这两种相似函数都使用带点的双向箭头表示。标注样本（水平虚线） $(x, y)$  能够学习表示  $f_x(x)$  和表示  $f_y(y)$  之间的单向或双向映射（实双向箭头），以及这些表示之间如何锚定。零数据学习可以通过以下方法实现。像  $x_{\text{test}}$  可以和单词  $y_{\text{test}}$  关联起来，即使该单词没有像，仅仅是因为单词表示  $f_y(y_{\text{test}})$  和像表示  $f_x(x_{\text{test}})$  可以通过表示空间的映射彼此关联。这种方法有效的原因是，尽管像和单词没有匹配成对，但是它们各自的特征向量  $f_x(x_{\text{test}})$  和  $f_y(y_{\text{test}})$  互相关联。上图受 Hrant Khachatrian 的建议启发。

### 15.3 半监督解释因果关系

表示学习的一个重要问题是“什么原因能够使一个表示比另一个表示更好？”一种假设是，理想表示中的特征对应到观测数据的潜在成因，特征空间中不同的特征

或方向对应着不同的原因，从而表示能够区分这些原因。这个假设促使我们去寻找表示  $p(\mathbf{x})$  的更好方法。如果  $\mathbf{y}$  是  $\mathbf{x}$  的重要成因之一，那么这种表示也可能是计算  $p(\mathbf{y} | \mathbf{x})$  的一种良好表示。从 20 世纪 90 年代以来，这个想法已经指导了大量的深度学习研究工作 (Becker and Hinton, 1992; Hinton and Sejnowski, 1999)。关于半监督学习可以超过纯监督学习的其他论点，请读者参考Chapelle *et al.* (2006) 的第 1.2 节。

在表示学习的其他方法中，我们大多关注易于建模的表示——例如，数据稀疏或是各项之间相互独立的情况。能够清楚地分离出潜在因素的表示可能并不一定易于建模。然而，该假设促使半监督学习使用无监督表示学习的一个更深层原因是，对于很多人工智能任务而言，有两个相随的特点：一旦我们能够获得观察结果基本成因的解释，那么将会很容易分离出个体属性。具体来说，如果表示向量  $\mathbf{h}$  表示观察值  $\mathbf{x}$  的很多潜在因素，并且输出向量  $\mathbf{y}$  是最为重要的原因之一，那么从  $\mathbf{h}$  预测  $\mathbf{y}$  会很容易。

首先，让我们看看  $p(\mathbf{x})$  的无监督学习无助于学习  $p(\mathbf{y} | \mathbf{x})$  时，半监督学习为何失败。例如，考虑一种情况， $p(\mathbf{x})$  是均匀分布的，我们希望学习  $f(\mathbf{x}) = \mathbb{E}[\mathbf{y} | \mathbf{x}]$ 。显然，仅仅观察训练集的值  $\mathbf{x}$  不能给我们关于  $p(\mathbf{y} | \mathbf{x})$  的任何信息。

接下来，让我们看看半监督学习成功的一个简单例子。考虑这样的情况， $\mathbf{x}$  来自一个混合分布，每个  $\mathbf{y}$  值具有一个混合分量，如图 15.4 所示。如果混合分量很好地分出来了，那么建模  $p(\mathbf{x})$  可以精确地指出每个分量的位置，每个类一个标注样本的训练集足以精确学习  $p(\mathbf{y} | \mathbf{x})$ 。但是更一般地，什么能将  $p(\mathbf{y} | \mathbf{x})$  和  $p(\mathbf{x})$  关联在一起呢？

如果  $\mathbf{y}$  与  $\mathbf{x}$  的成因之一非常相关，那么  $p(\mathbf{x})$  和  $p(\mathbf{y} | \mathbf{x})$  也会紧密关联，试图找到变化潜在因素的无监督表示学习可能像半监督学习一样有用。

假设  $\mathbf{y}$  是  $\mathbf{x}$  的成因之一，让  $\mathbf{h}$  代表所有这些成因。真实的生成过程可以被认为是根据这个有向图模型结构化出来的，其中  $\mathbf{h}$  是  $\mathbf{x}$  的父节点：

$$p(\mathbf{h}, \mathbf{x}) = p(\mathbf{x} | \mathbf{h})p(\mathbf{h}). \quad (15.1)$$

因此，数据的边缘概率是

$$p(\mathbf{x}) = \mathbb{E}_{\mathbf{h}} p(\mathbf{x} | \mathbf{h}). \quad (15.2)$$

从这个直观的观察中，我们得出结论， $\mathbf{x}$  最好可能的模型（从广义的观点）是会表示上述“真实”结构的，其中  $\mathbf{h}$  作为潜变量解释  $\mathbf{x}$  中可观察的变化。上文讨论的“理

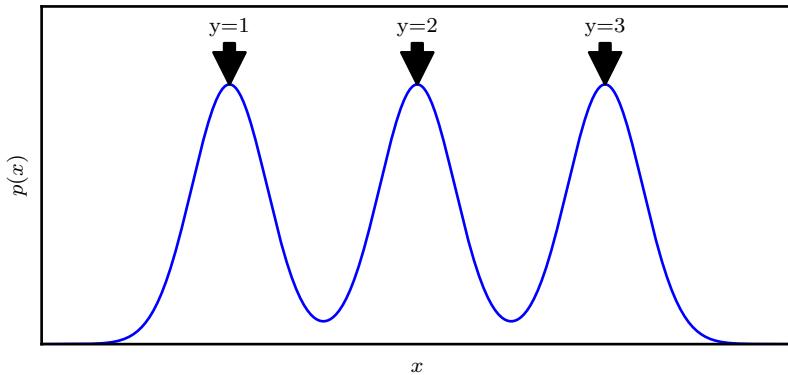


图 15.4: 混合模型。具有三个混合分量的  $x$  上混合密度示例。混合分量的内在本质是潜在解释因子  $y$ 。因为混合分量（例如，图像数据中的自然对象类别）在统计学上是显著的，所以仅仅使用未标注样本无监督建模  $p(x)$  也能揭示解释因子  $y$ 。

想”的表示学习应该能够反映出这些潜在因子。如果  $y$  是其中之一（或是紧密关联于其中之一），那么将很容易从这种表示中预测  $y$ 。我们会看到给定  $x$  下  $y$  的条件分布通过贝叶斯规则关联到上式中的分量：

$$p(y | x) = \frac{p(x | y)p(y)}{p(x)}. \quad (15.3)$$

因此边缘概率  $p(x)$  和条件概率  $p(y | x)$  密切相关，前者的结构信息应该有助于学习后者。因此，在这些假设情况下，半监督学习应该能提高性能。

关于这个事实的一个重要的研究问题是，大多数观察是由极其大量的潜在成因形成的。假设  $y = h_i$ ，但是无监督学习器并不知道是哪一个  $h_i$ 。对于一个无监督学习器暴力求解就是学习一种表示，这种表示能够捕获所有合理的重要生成因子  $h_j$ ，并将它们彼此区分开来，因此不管  $h_i$  是否关联于  $y$ ，从  $h$  预测  $y$  都是容易的。

在实践中，暴力求解是不可行的，因为不可能捕获影响观察的所有或大多数变化因素。例如，在视觉场景中，表示是否应该对背景中的所有最小对象进行编码？根据一个有据可查的心理学现象，人们不会察觉到环境中和他们所在进行的任务并不立刻相关的变化，具体例子可以参考 Simons and Levin (1998)。半监督学习的一个重要研究前沿是确定每种情况下要编码什么。目前，处理大量潜在原因的两个主要策略是，同时使用无监督学习和监督学习信号，从而使得模型捕获最相关的变动因素，或是使用纯无监督学习学习更大规模的表示。

无监督学习的另一个思路是选择一个更好的确定哪些潜在因素最为关键的定义。之前，自编码器和生成模型被训练来优化一个类似于均方误差的固定标准。这些固定标准确定了哪些因素是重要的。例如，图像像素的均方误差隐式地指定，一个潜在因素只有在其显著地改变大量像素的亮度时，才是重要影响因素。如果我们希望解决的问题涉及到小对象之间的相互作用，那么这将有可能遇到问题。如图 15.5 所示，在机器人任务中，自编码器未能学习到编码小乒乓球。同样是这个机器人，它可以成功地与更大的对象进行交互（例如棒球，均方误差在这种情况下很显著）。

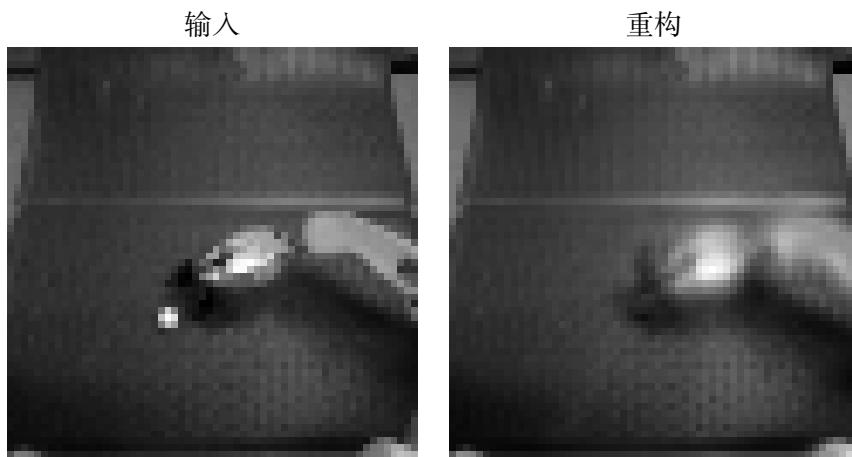


图 15.5：机器人任务上，基于均方误差训练的自编码器不能重构乒乓球。乒乓球的存在及其所有空间坐标，是生成图像且与机器人任务相关的重要潜在因素。不幸的是，自编码器具有有限的容量，基于均方误差的训练没能将乒乓球作为显著物体识别出来编码。以上图像由 Chelsea Finn 提供。

还有一些其他的显著性的定义。例如，如果一组像素具有高度可识别的模式，那么即使该模式不涉及到极端的亮度或暗度，该模式还是会被认为非常显著。实现这样一种定义显著的方法是使用最近提出的**生成式对抗网络** (generative adversarial network) (Goodfellow *et al.*, 2014c)。在这种方法中，生成模型被训练来愚弄前馈分类器。前馈分类器尝试将来自生成模型的所有样本识别为假的，并将来自训练集的所有样本识别为真的。在这个框架中，前馈网络能够识别出的任何结构化模式都是非常显著的。生成式对抗网络会在第 20.10.4 节中更详细地介绍。为了叙述方便，知道它能学习出如何决定什么是显著的就可以了。Lotter *et al.* (2015) 表明，生成人类头部头像的模型在使用均方误差训练时往往会忽视耳朵，但是对抗式框架学习能够成功地生成耳朵。因为耳朵与周围的皮肤相比不是非常明亮或黑暗，所以根据均方

误差损失它们不是特别突出，但是它们高度可识别的形状和一致的位置意味着前馈网络能够轻易地学习出如何检测它们，从而使得它们在生成式对抗框架下是高度突出的。图 15.6 给了一些样例图片。生成式对抗网络只是确定应该表示哪些因素的一小步。我们期望未来的研究能够发现更好的方式来确定表示哪些因素，并且根据任务来开发表示不同因素的机制。

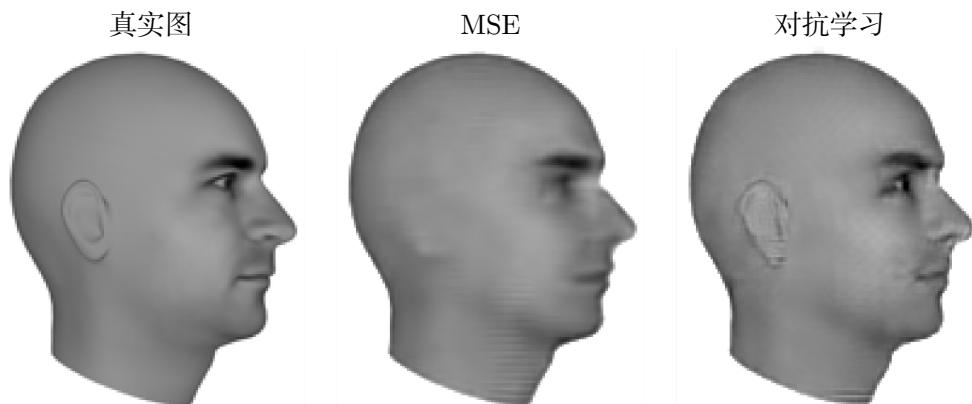


图 15.6：预测生成网络是一个学习哪些特征显著的例子。在这个例子中，预测生成网络已被训练成在特定视角预测人头的 3D 模型。(左) 真实情况。这是一张网络应该生成的正确图片。(中) 由具有均方误差的预测生成网络生成的图片。因为与相邻皮肤相比，耳朵不会引起亮度的极大差异，所以它们的显著性不足以让模型学习表示它们。(右) 由具有均方误差和对抗损失的模型生成的图片。使用这个学成的代价函数，由于耳朵遵循可预测的模式，因此耳朵是显著重要的。学习哪些原因对于模型而言是足够重要和相关的，是一个重要的活跃研究领域。以上图片由 Lotter *et al.* (2015) 提供。

正如 Schölkopf *et al.* (2012) 指出，学习潜在因素的好处是，如果真实的生成过程中  $\mathbf{x}$  是结果， $\mathbf{y}$  是原因，那么建模  $p(\mathbf{x} | \mathbf{y})$  对于  $p(\mathbf{y})$  的变化是鲁棒的。如果因果关系被逆转，这是不对的，因为根据贝叶斯规则， $p(\mathbf{x} | \mathbf{y})$  将会对  $p(\mathbf{y})$  的变化十分敏感。很多时候，我们考虑分布的变化（由于不同领域、时间不稳定性或任务性质的变化）时，因果机制是保持不变的（“宇宙定律不变”），而潜在因素的边缘分布是会变化的。因此，通过学习试图恢复成因向量  $\mathbf{h}$  和  $p(\mathbf{x} | \mathbf{h})$  的生成模型，我们可以期望最后的模型对所有种类的变化有更好的泛化和鲁棒性。

## 15.4 分布式表示

分布式表示的概念（由很多元素组合的表示，这些元素之间可以设置成可分离的）是表示学习最重要的工具之一。分布式表示非常强大，因为他们能用具有  $k$  个值的  $n$  个特征去描述  $k^n$  个不同的概念。正如我们在本书中看到的，具有多个隐藏单元的神经网络和具有多个潜变量的概率模型都利用了分布式表示的策略。我们现在再介绍一个观察结果。许多深度学习算法基于的假设是，隐藏单元能够学习表示出解释数据的潜在因果因子，就像第 15.3 节中讨论的一样。这种方法在分布式表示上是自然的，因为表示空间中的每个方向都对应着一个不同的潜在配置变量的值。

$n$  维二元向量是一个分布式表示的示例，有  $2^n$  种配置，每一种都对应输入空间中的一个不同区域，如图 15.7 所示。这可以与符号表示相比较，其中输入关联到单一符号或类别。如果字典中有  $n$  个符号，那么可以想象有  $n$  个特征监测器，每个特征探测器监测相关类别的存在。在这种情况下，只有表示空间中  $n$  个不同配置才有可能在输入空间中刻画  $n$  个不同的区域，如图 15.8 所示。这样的符号表示也被称为 one-hot 表示，因为它可以表示成相互排斥的  $n$  维二元向量（其中只有一位是激活的）。符号表示是更广泛的非分布式表示类中的一个具体示例，它可以包含很多条目，但是每个条目没有显著意义的单独控制作用。

以下是基于非分布式表示的学习算法的示例：

- 聚类算法，包含  $k$ -means 算法：每个输入点恰好分配到一个类别。
- $k$ -最近邻算法：给定一个输入，一个或几个模板或原型样本与之关联。在  $k > 1$  的情况下，每个输入都使用多个值来描述，但是它们不能彼此分开控制，因此这不能算真正的分布式表示。
- 决策树：给定输入时，只有一个叶节点（和从根到该叶节点路径上的点）是被激活的。
- 高斯混合体和专家混合体：模板（聚类中心）或专家关联一个激活的程度。和  $k$ -最近邻算法一样，每个输入用多个值表示，但是这些值不能轻易地彼此分开控制。
- 具有高斯核（或其他类似的局部核）的核机器：尽管每个“支持向量”或模板样本的激活程度是连续值，但仍然会出现和高斯混合体相同的问题。

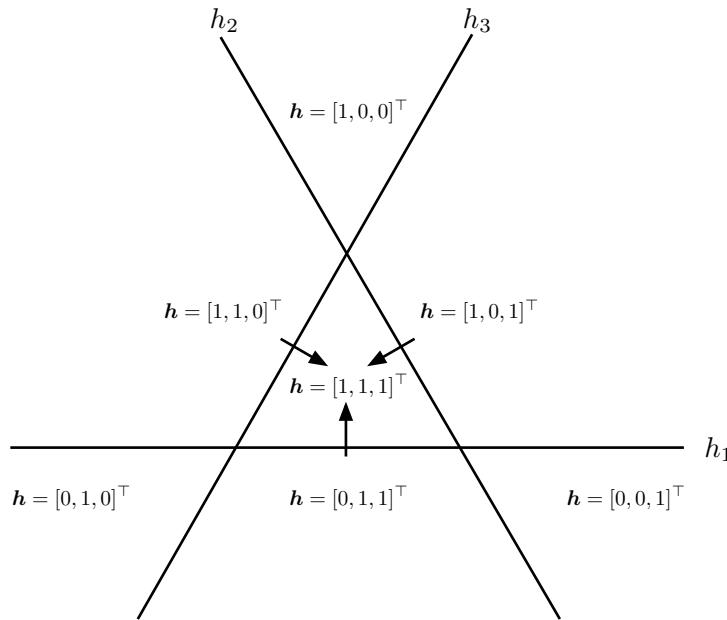


图 15.7: 基于分布式表示的学习算法如何将输入空间分割成多个区域的图示。这个例子具有二元变量  $h_1, h_2, h_3$ 。每个特征通过为学成的线性变换设定输出阀值而定义。每个特征将  $\mathbb{R}^2$  分成两个半平面。令  $h_i^+$  表示输入点  $h_i = 1$  的集合； $h_i^-$  表示输入点  $h_i = 0$  的集合。在这个图示中，每条线代表着一个  $h_i$  的决策边界，对应的箭头指向边界的  $h_i^+$  区域。整个表示在这些半平面的每个相交区域都指定一个唯一值。例如，表示值为  $[1, 1, 1]^\top$  对应着区域  $h_1^+ \cap h_2^+ \cap h_3^+$ 。可以将以上表示和图 15.8 中的非分布式表示进行比较。在输入维度是  $d$  的一般情况下，分布式表示通过半空间（而不是半平面）的交叉分割  $\mathbb{R}^d$ 。具有  $n$  个特征的分布式表示给  $O(n^d)$  个不同区域分配唯一的编码，而具有  $n$  个样本的最近邻算法只能给  $n$  个不同区域分配唯一的编码。因此，分布式表示能够比非分布式表示多分配指数级的区域。注意并非所有的  $h$  值都是可取的（这个例子中没有  $h = \mathbf{0}$ ），在分布式表示上的线性分类器不能向每个相邻区域分配不同的类别标识；甚至深度线性阀值网络的 VC 维只有  $O(w \log w)$ （其中  $w$  是权重数目）(Sontag, 1998)。强表示层和弱分类器层的组合是一个强正则化项。试图学习“人”和“非人”概念的分类器不需要给表示为“戴眼镜的女人”和“没有戴眼镜的男人”的输入分配不同的类别。容量限制鼓励每个分类器关注少数几个  $h_i$ ，鼓励  $h$  以线性可分的方式学习表示这些类别。

- 基于  $n$ -gram 的语言或翻译模型：根据后缀的树结构划分上下文集合（符号序列）。例如，一个叶节点可能对应于最后两个单词  $w_1$  和  $w_2$ 。树上的每个叶节点分别估计单独的参数（有些共享也是可能的）。

对于部分非分布式算法而言，有些输出并非是恒定的，而是在相邻区域之间内

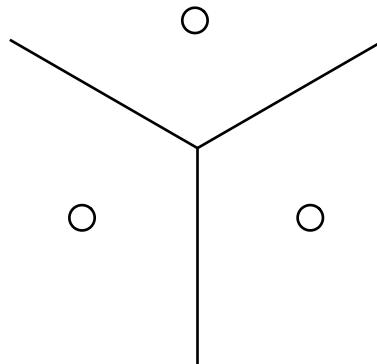


图 15.8: 最近邻算法如何将输入空间分成不同区域的图示。最近邻算法是一个基于非分布式表示的学习算法的示例。不同的非分布式算法可以具有不同的几何形状，但是它们通常将输入空间分成区域，每个区域具有不同的参数。非分布式方法的优点是，给定足够的参数，它能够拟合一个训练集，而不需要复杂的优化算法。因为它直接为每个区域独立地设置不同的参数。缺点是，非分布式表示的模型只能通过平滑先验来局部地泛化，因此学习波峰波谷多于样本的复杂函数时，该方法是不可行的。和分布式表示的对比，可以参照图 15.7。

插。参数（或样本）的数量和它们能够定义区域的数量之间仍保持线性关系。

将分布式表示和符号表示区分开来的一个重要概念是，由不同概念之间的共享属性而产生的泛化。作为纯符号，“猫”和“狗”之间的距离和任意其他两种符号的距离一样。然而，如果将它们与有意义的分布式表示相关联，那么关于猫的很多特点可以推广到狗，反之亦然。例如，我们的分布式表示可能会包含诸如“具有皮毛”或“腿的数目”这类在“猫”和“狗”的嵌入上具有相同值的项。正如第 12.4.2 节所讨论的，作用于单词分布式表示的神经语言模型比其他直接对单词 one-hot 表示进行操作的模型泛化得更好。分布式表示具有丰富的相似性空间，语义上相近的概念（或输入）在距离上接近，这是纯粹的符号表示所缺少的特点。

在学习算法中使用分布式表示何时以及为什么具有统计优势？当一个明显复杂的结构可以用较少参数紧致地表示时，分布式表示具有统计上的优点。一些传统的非分布式学习算法仅仅在平滑假设的情况下能够泛化，也就是说如果  $u \approx v$ ，那么学到的目标函数  $f$  通常具有  $f(u) \approx f(v)$  的性质。有许多方法来形式化这样一个假设，但其结果是如果我们有一个样本  $(x, y)$ ，并且我们知道  $f(x) \approx y$ ，那么我们可以选取一个估计  $\hat{f}$  近似地满足这些限制，并且当我们移动到附近的输入  $x + \epsilon$  时， $\hat{f}$  尽可能少地发生改变。显然这个假设是非常有用的，但是它会遭受维数灾难：学习出一个

能够在很多不同区域上增加或减少很多次的目标函数<sup>1</sup>，我们可能需要至少和可区分区域数量一样多的样本。我们可以将每一个区域视为一个类别或符号：通过让每个符号（或区域）具有单独的自由度，我们可以学习出从符号映射到值的任意解码器。然而，这不能推广到新区域的新符号上。

如果我们幸运的话，除了平滑之外，目标函数可能还有一些其他规律。例如，具有最大池化的卷积网络可以在不考虑对象在图像中位置（即使对象的空间变换不对应输入空间的平滑变换）的情况下识别出对象。

让我们检查分布式表示学习算法的一个特殊情况，它通过对输入的线性函数进行阀值处理来提取二元特征。该表示中的每个二元特征将  $\mathbb{R}^d$  分成一对半空间，如图 15.7 所示。 $n$  个相应半空间的指数级数量的交集确定了该分布式表示学习器能够区分多少区域。空间  $\mathbb{R}^d$  中的  $n$  个超平面的排列组合能够生成多少区间？通过应用关于超平面交集的一般结果 (Zaslavsky, 1975)，我们发现 (Pascanu *et al.*, 2014b) 这个二元特征表示能够区分的空间数量是

$$\sum_{j=0}^d \binom{n}{j} = O(n^d). \quad (15.4)$$

因此，我们会发现关于输入大小呈指数级增长，关于隐藏单元的数量呈多项式级增长。

这提供了分布式表示泛化能力的一种几何解释： $O(nd)$  个参数（空间  $\mathbb{R}^d$  中的  $n$  个线性阀值特征）能够明确表示输入空间中  $O(n^d)$  个不同区域。如果我们没有对数据做任何假设，并且每个区域使用唯一的符号来表示，每个符号使用单独的参数去识别  $\mathbb{R}^d$  中的对应区域，那么指定  $O(n^d)$  个区域需要  $O(n^d)$  个样本。更一般地，分布式表示的优势还可以体现在我们对分布式表示中的每个特征使用非线性的、可能连续的特征提取器，而不是线性阀值单元的情况。在这种情况下，如果具有  $k$  个参数的参数变换可以学习输入空间中的  $r$  个区域 ( $k \ll r$ )，并且如果学习这样的表示有助于关注的任务，那么这种方式会比非分布式情景（我们需要  $O(r)$  个样本来获得相同的特征，将输入空间相关联地划分成  $r$  个区域。）泛化得更好。使用较少的参数来表示模型意味着我们只需拟合较少的参数，因此只需要更少的训练样本去获得良好的泛化。

另一个解释基于分布式表示的模型泛化能力更好的说法是，尽管能够明确地编

---

<sup>1</sup>一般来说，我们可能会想要学习一个函数，这个函数在指数级数量区域的表现都是不同的：在  $d$ -维空间中，为了区分每一维，至少有两个不同的值。我们想要函数  $f$  区分这  $2^d$  个不同的区域，需要  $O(2^d)$  量级的训练样本

码这么多不同的区域，但它们的容量仍然是很有限的。例如，线性阀值单元神经网络的 VC 维仅为  $O(w \log w)$ ，其中  $w$  是权重的数目 (Sontag, 1998)。这种限制出现的原因是，虽然我们可以为表示空间分配非常多的唯一码，但是我们不能完全使用所有的码空间，也不能使用线性分类器学习出从表示空间  $\mathbf{h}$  到输出  $\mathbf{y}$  的任意函数映射。因此使用与线性分类器相结合的分布式表示传达了一种先验信念，待识别的类在  $\mathbf{h}$  代表的潜在因果因子的函数下是线性可分的。我们通常想要学习类别，例如所有绿色对象的图像集合，或是所有汽车图像集合，但不会是需要非线性 XOR 逻辑的类别。例如，我们通常不会将数据划分成所有红色汽车和绿色卡车作为一个集合，所有绿色汽车和红色卡车作为另一个集合。

到目前为止讨论的想法都是抽象的，但是它们可以通过实验证。Zhou *et al.* (2015) 发现，在 ImageNet 和 Places 基准数据集上训练的深度卷积网络中的隐藏单元学成的特征通常是可以解释的，对应人类自然分配的标签。在实践中，隐藏单元并不能总是学习出具有简单语言学名称的事物，但有趣的是，这些事物会在那些最好的计算机视觉深度网络的顶层附近出现。这些特征的共同之处在于，我们可以设想学习其中的每个特征不需要知道所有其他特征的所有配置。Radford *et al.* (2015) 发现生成模型可以学习人脸图像的表示，在表示空间中的不同方向捕获不同的潜在变差因素。图 15.9 展示表示空间中的一个方向对应着该人是男性还是女性，而另一个方向对应着该人是否戴着眼镜。这些特征都是自动发现的，而非先验固定的。我们没有必要为隐藏单元分类器提供标签：只要该任务需要这样的特征，梯度下降就能在感兴趣的目标函数上自然地学习出语义上有趣的特征。我们可以学习出男性和女性之间的区别，或者是眼镜的存在与否，而不必通过涵盖所有这些值组合的样本来表征其他  $n - 1$  个特征的所有配置。这种形式的统计可分离性质能够泛化到训练期间从未见过的新特征上。

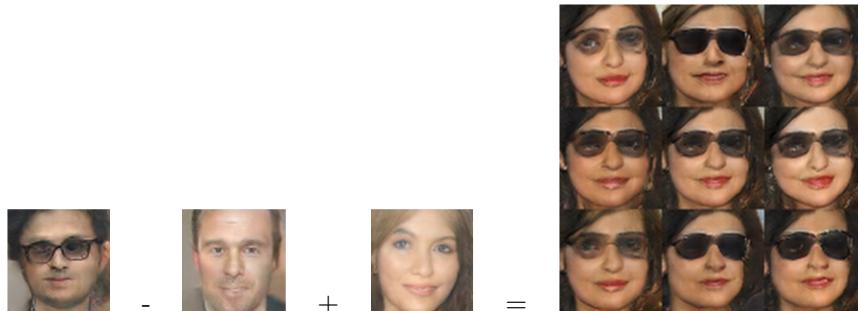


图 15.9: 生成模型学到了分布式表示, 能够从戴眼镜的概念中区分性别的概念。如果我们从一个戴眼镜的男人的概念表示向量开始, 然后减去一个没戴眼镜的男人的概念表示向量, 最后加上一个没戴眼镜的女人的概念表示向量, 那么我们会得到一个戴眼镜的女人的概念表示向量。生成模型将所有这些表示向量正确地解码为可被识别为正确类别的图像。图片转载许可自 Radford *et al.* (2015)。

## 15.5 得益于深度的指数增益

我们已经在第 6.4.1 节中看到, 多层感知机是万能近似器, 相比于浅层网络, 一些函数能够用指数级小的深度网络表示。缩小模型规模能够提高统计效率。在本节中, 我们描述如何将类似结果更一般地应用于其他具有分布式隐藏表示的模型。

在第 15.4 节中, 我们看到了一个生成模型的示例, 能够学习人脸图像的潜在解释因子, 包括性别以及是否佩戴眼镜。完成这个任务的生成模型是基于一个深度神经网络的。浅层网络例如线性网络不能学习出这些抽象解释因子和图像像素之间的复杂关系。在这个任务和其他 AI 任务中, 这些因子几乎彼此独立地被抽取, 但仍然对应到有意义输入的因素, 很有可能是高度抽象的, 并且和输入呈高度非线性的关系。我们认为这需要深度分布式表示, 需要许多非线性组合来获得较高级的特征(被视为输入的函数)或因子(被视为生成原因)。

在许多不同情景中已经证明, 非线性和重用特征层次结构的组合来组织计算, 可以使分布式表示获得指数级加速之外, 还可以获得统计效率的指数级提升。许多种类的只有一个隐藏层的网络(例如, 具有饱和非线性, 布尔门, 和/积, 或 RBF 单元的网络)都可以被视为万能近似器。在给定足够多隐藏单元的情况下, 这个模型族是一个万能近似器, 可以在任意非零允错级别近似一大类函数(包括所有连续函数)。然而, 隐藏单元所需的数量可能会非常大。关于深层架构表达能力的理论结果表明, 有些函数族可以高效地通过深度  $k$  层的网络架构表示, 但是深度不够(深度

为 2 或  $k - 1$  ) 时会需要指数级 ( 相对于输入大小而言 ) 的隐藏单元。

在第 6.4.1 节中，我们看到确定性前馈网络是函数的万能近似器。许多具有单个隐藏层 ( 潜变量 ) 的结构化概率模型 ( 包括受限玻尔兹曼机，深度信念网络 ) 是概率分布的万能近似器 ( Le Roux and Bengio, 2008, 2010; Montúfar and Ay, 2011; Montúfar, 2014; Krause *et al.*, 2013)。

在第 6.4.1 节中，我们看到足够深的前馈网络会比深度不够的网络具有指数级优势。这样的结果也能从诸如概率模型的其他模型中获得。和-积网络 ( sum-product network, SPN ) ( Poon and Domingos, 2011 ) 是这样的一种概率模型。这些模型使用多项式回路来计算一组随机变量的概率分布。Delalleau and Bengio ( 2011 ) 表明存在一种概率分布，对 SPN 的最小深度有要求，以避免模型规模呈指数级增长。后来，Martens and Medabalimi ( 2014 ) 表明，任意两个有限深度的 SPN 之间都会存在显著差异，并且一些使 SPN 易于处理的约束可能会限制其表示能力。

另一个有趣的进展是，一系列和卷积网络相关的深度回路族表达能力的理论结果，即使让浅度回路只去近似深度回路计算的函数，也能突出反映深度回路的指数级优势 ( Cohen *et al.*, 2015)。相比之下，以前的理论工作只研究了浅度回路必须精确复制特定函数的情况。

## 15.6 提供发现潜在原因的线索

我们回到最初的问题之一来结束本章：什么原因能够使一个表示比另一个表示更好？首先在第 15.3 节中介绍的一个答案是，一个理想的表示能够区分生成数据变化的潜在因果因子，特别是那些与我们的应用相关的因素。表示学习的大多数策略都会引入一些有助于学习潜在变差因素的线索。这些线索可以帮助学习器将这些观察到的因素与其他因素分开。监督学习提供了非常强的线索：每个观察向量  $\mathbf{x}$  的标签  $\mathbf{y}$ ，它通常直接指定了至少一个变差因素。更一般地，为了利用丰富的未标注数据，表示学习会使用关于潜在因素的其他不太直接的提示。这些提示包含一些我们 ( 学习算法的设计者 ) 为了引导学习器而强加的隐式先验信息。诸如没有免费午餐定理的这些结果表明，正则化策略对于获得良好泛化是很有必要的。当不可能找到一个普遍良好的正则化策略时，深度学习的一个目标是找到一套相当通用的正则化策略，使其能够适用于各种各样的 AI 任务 ( 类似于人和动物能够解决的任务 )。

在此，我们提供了一些通用正则化策略的列表。该列表显然是不详尽的，但是

给出了一些学习算法是如何发现对应潜在因素的特征的具体示例。该列表在 Bengio *et al.* (2013d) 的第 3.1 节中提出，这里进行了部分拓展。

- **平滑：**假设对于单位  $\mathbf{d}$  和小量  $\epsilon$  有  $f(\mathbf{x} + \epsilon\mathbf{d}) \approx f(\mathbf{x})$ 。这个假设允许学习器从训练样本泛化到输入空间中附近的点。许多机器学习算法都利用了这个想法，但它不能克服维数灾难难题。
- **线性：**很多学习算法假定一些变量之间的关系是线性的。这使得算法能够预测远离观测数据的点，但有时可能会导致一些极端的预测。大多数简单的学习算法不会做平滑假设，而会做线性假设。这些假设实际上是不同的，具有很大权重的线性函数在高维空间中可能不是非常平滑的。参看 Goodfellow *et al.* (2014b) 了解关于线性假设局限性的进一步讨论。
- **多个解释因子：**许多表示学习算法受以下假设的启发，数据是由多个潜在解释因子生成的，并且给定每一个因子的状态，大多数任务都能轻易解决。第 15.3 节描述了这种观点如何通过表示学习来启发半监督学习的。学习  $p(\mathbf{x})$  的结构要求学习出一些对建模  $p(\mathbf{y} | \mathbf{x})$  同样有用的特征，因为它们都涉及到相同的潜在解释因子。第 15.4 节介绍了这种观点如何启发分布式表示的使用，表示空间中分离的方向对应着分离的变差因素。
- **因果因子：**该模型认为学成表示所描述的变差因素是观察数据  $\mathbf{x}$  的成因，而非反过来。正如第 15.3 节中讨论的，这对于半监督学习是有利的，当潜在成因上的分布发生改变，或者我们应用模型到一个新的任务上时，学成的模型都会更加鲁棒。
- **深度，或者解释因子的层次组织：**高级抽象概念能够通过将简单概念层次化来定义。从另一个角度来看，深度架构表达了我们认为任务应该由多个程序步骤完成的观念，其中每一个步骤回溯到先前步骤处理之后的输出。
- **任务间共享因素：**当多个对应到不同变量  $y_i$  的任务共享相同的输入  $\mathbf{x}$  时，或者当每个任务关联到全局输入  $\mathbf{x}$  的子集或者函数  $f^{(i)}(\mathbf{x})$  时，我们会假设每个变量  $y_i$  关联到来自相关因素  $\mathbf{h}$  公共池的不同子集。因为这些子集有重叠，所以通过共享的中间表示  $P(\mathbf{h} | \mathbf{x})$  来学习所有的  $P(y_i | \mathbf{x})$  能够使任务间共享统计强度。
- **流形：**概率质量集中，并且集中区域是局部连通的，且占据很小的体积。在连续情况下，这些区域可以用比数据所在原始空间低很多维的低维流形来近似。

很多机器学习算法只在这些流形上有效 (Goodfellow *et al.*, 2014b)。一些机器学习算法，特别是自编码器，会试图显式地学习流形的结构。

- **自然聚类：**很多机器学习算法假设输入空间中每个连通流形可以被分配一个单独的类。数据分布在许多个不连通的流形上，但相同流形上数据的类别是相同的。这个假设激励了各种学习算法，包括正切传播、双反向传播、流形正切分类器和对抗训练。
- **时间和空间相干性：**慢特征分析和相关的算法假设，最重要的解释因子随时间变化很缓慢，或者至少假设预测真实的潜在解释因子比预测诸如像素值这类原始观察会更容易些。读者可以参考第 13.3 节，进一步了解这个方法。
- **稀疏性：**假设大部分特征和大部分输入不相关，如在表示猫的图像时，没有必要使用象鼻的特征。因此，我们可以强加一个先验，任何可以解释为“存在”或“不存在”的特征在大多数时间都是不存在的。
- **简化因子依赖：**在良好的高级表示中，因子会通过简单的依赖相互关联。最简单的可能是边缘独立，即  $P(\mathbf{h}) = \prod_i P(\mathbf{h}_i)$ 。但是线性依赖或浅层自编码器所能表示的依赖关系也是合理的假设。这可以从许多物理定律中看出来，并且假设在学成表示的顶层插入线性预测器或分解的先验。

表示学习的概念将许多深度学习形式联系在了一起。前馈网络和循环网络，自编码器和深度概率模型都在学习和使用表示。学习最佳表示仍然是一个令人兴奋的研究方向。

# 第十六章 深度学习中的结构化概率模型

深度学习为研究者们提供了许多建模方式，用以设计以及描述算法。其中一种形式是 **结构化概率模型** (structured probabilistic model) 的思想。我们曾经在第 3.14 节中简要讨论过结构化概率模型。此前简要的介绍已经足够使我们充分了解如何使用结构化概率模型作为描述第二部分中某些算法的语言。现在在第三部分，我们可以看到结构化概率模型是许多深度学习重要研究方向的关键组成部分。作为讨论这些研究方向的预备知识，本章将更加详细地描述结构化概率模型。本章内容是自洽的，所以在阅读本章之前读者不需要回顾之前的介绍。

结构化概率模型使用图来描述概率分布中随机变量之间的直接相互作用，从而描述一个概率分布。在这里我们使用了图论（一系列结点通过一系列边来连接）中“图”的概念，由于模型结构是由图定义的，所以这些模型也通常被称为 **图模型** (graphical model)。

图模型的研究社群是巨大的，并提出过大量的模型、训练算法和推断算法。在本章中，我们将介绍图模型中几个核心方法的基本背景，并且重点描述已被证明对深度学习社群最有用的观点。如果你已经熟知图模型，那么你可以跳过本章的绝大部分。然而，我们相信即使是资深的图模型方向的研究者也会从本章的最后一节中获益匪浅，详见第 16.7 节，其中我们强调了在深度学习算法中使用图模型的独特方式。相比于其他图模型研究领域的是，深度学习的研究者们通常会使用完全不同的模型结构、学习算法和推断过程。在本章中，我们将指明这种区别并解释其中的原因。

我们首先介绍了构建大规模概率模型时面临的挑战。之后，我们介绍如何使用一个图来描述概率分布的结构。尽管这个方法能够帮助我们解决许多挑战和问题，它本身仍有很多缺陷。图模型中的一个主要难点就是判断哪些变量之间存在直接的相

互作用关系，也就是对于给定的问题哪一种图结构是最适合的。在第 16.5 节中，我们通过了解 **依赖** (dependency)，简要概括了解决这个难点的两种方法。最后，作为本章的收尾，我们在第 16.7 节中讨论深度学习研究者使用图模型特定方式的独特之处。

## 16.1 非结构化建模的挑战

深度学习的目标是使得机器学习能够解决许多人工智能中亟需解决的挑战。这也意味着它们能够理解具有丰富结构的高维数据。举个例子，我们希望AI的算法能够理解自然图片<sup>1</sup>，表示语音的声音信号和包含许多词和标点的文档。

分类问题可以把这样一个来自高维分布的数据作为输入，然后使用一个类别的标签来概括它——这个标签可以是照片中是什么物品，一段语音中说的是哪个单词，也可以是一段文档描述的是哪个话题。这个分类过程丢弃了输入数据中的大部分信息，然后产生单个值的输出（或者是关于单个输出值的概率分布）。这个分类器通常可以忽略输入数据的很多部分。例如，当我们识别一张照片中的一个物体时，我们通常可以忽略图片的背景。

我们也可以使用概率模型完成许多其他的任务。这些任务通常相比于分类成本更高。其中的一些任务需要产生多个输出。大部分任务需要对输入数据整个结构的完整理解，所以并不能舍弃数据的一部分。这些任务包括以下几个：

- **估计密度函数：**给定一个输入  $\mathbf{x}$ ，机器学习系统返回一个对数据生成分布的真实密度函数  $p(\mathbf{x})$  的估计。这只需要一个输出，但它需要完全理解整个输入。即使向量中只有一个元素不太正常，系统也会给它赋予很低的概率。
- **去噪：**给定一个受损的或者观察有误的输入数据  $\tilde{\mathbf{x}}$ ，机器学习系统返回一个对原始的真实  $\mathbf{x}$  的估计。举个例子，有时候机器学习系统需要从一张老相片中去除灰尘或者抓痕。这个系统会产生多个输出值（对应着估计的干净样本  $\mathbf{x}$  的每一个元素），并且需要我们有一个对输入的整体理解（因为即使只有一个损坏的区域，仍然会显示最终估计被损坏）。
- **缺失值的填补：**给定  $\mathbf{x}$  的某些元素作为观察值，模型被要求返回一个  $\mathbf{x}$  一些或者全部未观察值的估计或者概率分布。这个模型返回的也是多个输出。由于这个模型需要恢复  $\mathbf{x}$  的每一个元素，所以它必须理解整个输入。

---

<sup>1</sup>自然图片指的是能够在正常的环境下被照相机拍摄的图片，不同于合成的图片，或者一个网页的截图等等。

- 采样：**模型从分布  $p(\mathbf{x})$  中抽取新的样本。其应用包括语音合成，即产生一个听起来很像人说话的声音。这个模型也需要多个输出以及对输入整体的良好建模。即使样本只有一个从错误分布中产生的元素，那么采样的过程也是错误的。

图 16.1 中描述了一个使用较小的自然图片的采样任务。

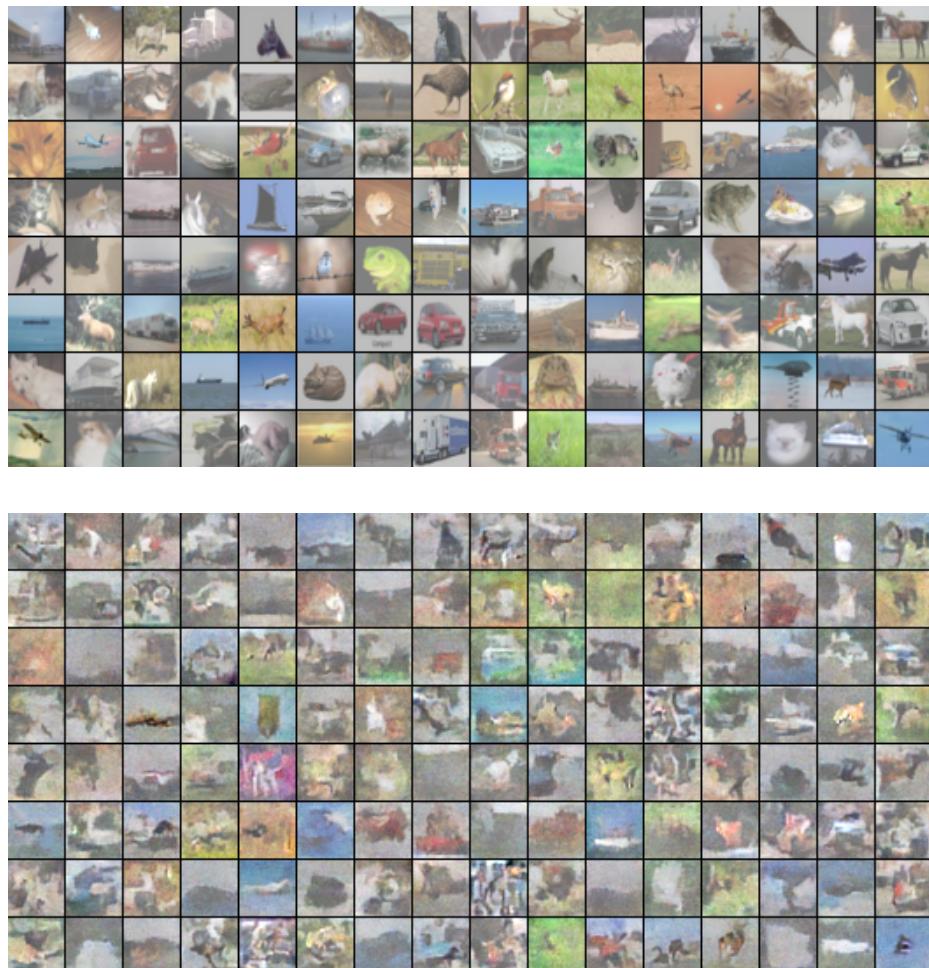


图 16.1：自然图片的概率建模。(上) CIFAR-10 数据集 (Krizhevsky and Hinton, 2009) 中的 32×32 像素的样例图片。(下) 从这个数据集上训练的结构化概率模型中抽出的样本。每一个样本都出现在与其欧氏距离最近的训练样本的格点中。这种比较使得我们发现这个模型确实能够生成新的图片，而不是记住训练样本。为了方便展示，两个集合的图片都经过了微调。图片经 Courville *et al.* (2011a) 许可转载。

对上千甚至是上百万随机变量的分布建模，无论从计算上还是从统计意义上说，都是一个极具挑战性的任务。假设我们只想对二值的随机变量建模。这是一个最简单的例子，但是我们仍然无能为力。对一个只有  $32 \times 32$  像素的彩色（RGB）图片来说，存在  $2^{3072}$  种可能的二值图片。这个数量已经超过了  $10^{800}$ ，比宇宙中的原子总数还要多。

通常意义上讲，如果我们希望对一个包含  $n$  个离散变量并且每个变量都能取  $k$  个值的  $\mathbf{x}$  的分布建模，那么最简单的表示  $P(\mathbf{x})$  的方法需要存储一个可以查询的表格。这个表格记录了每一种可能值的概率，则需要  $k^n$  个参数。

基于下述几个原因，这种方式是不可行的：

- **内存：**存储参数的开销。除了极小的  $n$  和  $k$  的值，用表格的形式来表示这样一个分布需要太多的存储空间。
- **统计的高效性：**当模型中的参数个数增加时，使用统计估计器估计这些参数所需要的训练数据数量也需要相应地增加。因为基于查表的模型拥有天文数字级别的参数，为了准确地拟合，相应的训练集的大小也是相同级别的。任何这样的模型都会导致严重的过拟合，除非我们添加一些额外的假设来联系表格中的不同元素（正如第 12.4.1 节中所举的回退或者平滑  $n$ -gram 模型）。
- **运行时间：**推断的开销。假设我们需要完成这样一个推断的任务，其中我们需要使用联合分布  $P(\mathbf{x})$  来计算某些其他的分布，比如说边缘分布  $P(x_1)$  或者是条件分布  $P(x_2 | x_1)$ 。计算这样的分布需要对整个表格的某些项进行求和操作，因此这样的操作的运行时间和上述高昂的内存开销是一个级别的。
- **运行时间：**采样的开销。类似的，假设我们想要从这样的模型中采样。最简单的方法就是从均匀分布中采样， $u \sim U(0, 1)$ ，然后把表格中的元素累加起来，直到和大于  $u$ ，然后返回最后一个加上的元素。最差情况下，这个操作需要读取整个表格，所以和其他操作一样，它也需要指数级别的时间。

基于表格操作的方法的主要问题是显式地对每一种可能的变量子集所产生的每一种可能类型的相互作用建模。在实际问题中我们遇到的概率分布远比这个简单。通常，许多变量只是间接地相互作用。

例如，我们想要对接力跑步比赛中一个队伍完成比赛的时间进行建模。假设这个队伍有三名成员：Alice，Bob 和 Carol。在比赛开始时，Alice 拿着接力棒，开始

跑第一段距离。在跑完她的路程以后，她把棒递给了 Bob。然后 Bob 开始跑，再把棒给 Carol，Carol 跑最后一棒。我们可以用连续变量来建模他们每个人完成的时间。因为 Alice 第一个跑，所以她的完成时间并不依赖于其他的人。Bob 的完成时间依赖于 Alice 的完成时间，因为 Bob 只能在 Alice 跑完以后才能开始跑。如果 Alice 跑得更快，那么 Bob 也会完成得更快。所有其他关系都可以被类似地推出。最后，Carol 的完成时间依赖于她的两个队友。如果 Alice 跑得很慢，那么 Bob 也会完成得更慢。结果，Carol 将会更晚开始跑步，因此她的完成时间也更有可能要晚。然而，在给定 Bob 完成时间的情况下，Carol 的完成时间只是间接地依赖于 Alice 的完成时间。如果我们已经知道了 Bob 的完成时间，知道 Alice 的完成时间对估计 Carol 的完成时间并无任何帮助。这意味着我们可以通过仅仅两个相互作用来建模这个接力赛。这两个相互作用分别是 Alice 的完成时间对 Bob 的完成时间的影响和 Bob 的完成时间对 Carol 的完成时间的影响。在这个模型中，我们可以忽略第三种间接的相互作用，即 Alice 的完成时间对 Carol 的完成时间的影响。

结构化概率模型为随机变量之间的直接作用提供了一个正式的建模框架。这种方式大大减少了模型的参数个数以致于模型只需要更少的数据来进行有效的估计。这些更小的模型大大减小了在模型存储、模型推断以及从模型中采样时的计算开销。

## 16.2 使用图描述模型结构

结构化概率模型使用图（在图论中“结点”是通过“边”来连接的）来表示随机变量之间的相互作用。每一个结点代表一个随机变量。每一条边代表一个直接相互作用。这些直接相互作用隐含着其他的间接相互作用，但是只有直接的相互作用会被显式地建模。

使用图来描述概率分布中相互作用的方法不止一种。在下文中我们会介绍几种最为流行和有用的方法。图模型可以被大致分为两类：基于有向无环图的模型和基于无向图的模型。

### 16.2.1 有向模型

**有向图模型** (directed graphical model) 是一种结构化概率模型，也被称为**信念网络** (belief network) 或者**贝叶斯网络** (Bayesian network)<sup>2</sup> (Pearl, 1985)。

之所以命名为有向图模型是因为所有的边都是有方向的，即从一个结点指向另一个结点。这个方向可以通过画一个箭头来表示。箭头所指的方向表示了这个随机变量的概率分布是由其他变量的概率分布所定义的。画一个从结点 a 到结点 b 的箭头表示了我们用一个条件分布来定义 b，而 a 是作为这个条件分布符号右边的一个变量。换句话说，b 的概率分布依赖于 a 的取值。

我们继续第 16.1 节所讲的接力赛的例子，我们假设 Alice 的完成时间为  $t_0$ ，Bob 的完成时间为  $t_1$ ，Carol 的完成时间为  $t_2$ 。就像我们之前看到的一样， $t_1$  的估计是依赖于  $t_0$  的， $t_2$  的估计是直接依赖于  $t_1$  的，但是仅仅间接地依赖于  $t_0$ 。我们用一个有向图模型来建模这种关系，如图 16.2 所示。

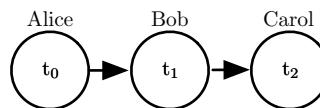


图 16.2：描述接力赛例子的有向图模型。Alice 的完成时间  $t_0$  影响了 Bob 的完成时间  $t_1$ ，因为 Bob 只能在 Alice 完成比赛后才开始。类似的，Carol 也只会在 Bob 完成之后才开始，所以 Bob 的完成时间  $t_1$  直接影响了 Carol 的完成时间  $t_2$ 。

正式地说，变量  $\mathbf{x}$  的有向概率模型是通过有向无环图  $\mathcal{G}$ （每个结点都是模型中的随机变量）和一系列**局部条件概率分布** (local conditional probability distribution)  $p(\mathbf{x}_i | Pa_{\mathcal{G}}(\mathbf{x}_i))$  来定义的，其中  $Pa_{\mathcal{G}}(\mathbf{x}_i)$  表示结点  $\mathbf{x}_i$  的所有父结点。 $\mathbf{x}$  的概率分布可以表示为

$$p(\mathbf{x}) = \prod_i p(\mathbf{x}_i | Pa_{\mathcal{G}}(\mathbf{x}_i)). \quad (16.1)$$

在之前所述的接力赛的例子中，参考图 16.2，这意味着概率分布可以被表示为

$$p(t_0, t_1, t_2) = p(t_0)p(t_1 | t_0)p(t_2 | t_1). \quad (16.2)$$

<sup>2</sup>当我们希望“强调”从网络中计算出的值的“推断”本质，即强调这些值代表的是置信程度大小而不是事件的频率时，Judea Pearl 建议使用“贝叶斯网络”这个术语。

这是我们看到的第一个结构化概率模型的实际例子。我们能够检查这样建模的计算开销，为了验证相比于非结构化建模，结构化建模为什么有那么多的优势。

假设我们采用从第 0 分钟到第 10 分钟每 6 秒一块的方式离散化地表示时间。这使得  $t_0$ ,  $t_1$  和  $t_2$  都是一个有 100 个取值可能的离散变量。如果我们尝试着用一个表来表示  $p(t_0, t_1, t_2)$ , 那么我们需要存储 999,999 个值 ( $100$  个  $t_0$  的可能取值  $\times$   $100$  个  $t_1$  的可能取值  $\times$   $100$  个  $t_2$  的可能取值减去 1, 由于存在所有的概率之和为 1 的限制, 所以其中有 1 个值的存储是多余的)。反之, 如果我们用一个表来记录每一种条件概率分布, 那么表中记录  $t_0$  的分布需要存储 99 个值, 给定  $t_0$  情况下  $t_1$  的分布需要存储 9900 个值, 给定  $t_1$  情况下  $t_2$  的分布也需要存储 9900 个值。加起来总共需要存储 19,899 个值。这意味着使用有向图模型将参数的个数减少了超过 50 倍!

通常意义上说, 对每个变量都能取  $k$  个值的  $n$  个变量建模, 基于建表的方法需要的复杂度是  $O(k^n)$ , 就像我们之前观察到的一样。现在假设我们用一个有向图模型来对这些变量建模。如果  $m$  代表图模型的单个条件概率分布中最大的变量数目(在条件符号的左右皆可), 那么对这个有向模型建表的复杂度大致为  $O(k^m)$ 。只要我们在设计模型时使其满足  $m \ll n$ , 那么复杂度就会被大大地减小。

换一句话说, 只要图中的每个变量都只有少量的父结点, 那么这个分布就可以用较少的参数来表示。图结构上的一些限制条件, 比如说要求这个图为一棵树, 也可以保证一些操作(例如求一小部分变量的边缘或者条件分布)更加地高效。

决定哪些信息需要被包含在图中而哪些不需要是很重要的。如果变量之间可以被假设为是条件独立的, 那么这个图可以包含这种简化假设。当然也存在其他类型的简化图模型的假设。例如, 我们可以假设无论 Alice 的表现如何, Bob 总是跑得一样快(实际上, Alice 的表现很大概率会影响 Bob 的表现, 这取决于 Bob 的性格, 如果在之前的比赛中 Alice 跑得特别快, 这有可能鼓励 Bob 更加努力并取得更好的成绩, 当然这也有可能使得 Bob 过分自信或者变得懒惰)。那么 Alice 对 Bob 的唯一影响就是在计算 Bob 的完成时间时需要加上 Alice 的时间。这个假设使得我们所需要的参数量从  $O(k^2)$  降到了  $O(k)$ 。然而, 值得注意的是在这个假设下  $t_0$  和  $t_1$  仍然是直接相关的, 因为  $t_1$  表示的是 Bob 完成时的时间, 并不是他跑的总时间。这也意味着图中会有一个从  $t_0$  指向  $t_1$  的箭头。“Bob 的个人跑步时间相对于其他因素是独立的”这个假设无法在  $t_0$ ,  $t_1$ ,  $t_2$  的图中被表示出来。反之, 我们只能将这个关系表示在条件分布的定义中。这个条件分布不再是一个大小为  $k \times k - 1$  的分别对应着  $t_0$ ,  $t_1$  的表格, 而是一个包含了  $k - 1$  个参数的略微复杂的公式。有向图模型的语法并不能对我们如何定义条件分布作出任何限制。它只定义了哪些变量可以作为其中

的参数。

### 16.2.2 无向模型

有向图模型为我们提供了一种描述结构化概率模型的语言。而另一种常见的语言则是 **无向模型** (undirected Model)，也被称为 **马尔可夫随机场** (Markov random field, MRF) 或者是 **马尔可夫网络** (Markov network) (Kindermann, 1980)。就像它们的名字所说的那样，无向模型中所有的边都是没有方向的。

当存在很明显的理由画出每一个指向特定方向的箭头时，有向模型显然最适用。有向模型中，经常存在我们理解的具有因果关系以及因果关系有明确方向的情况。接力赛的例子就是一个这样的情况。之前运动员的表现会影响后面运动员的完成时间，而后面运动员却不会影响前面运动员的完成时间。

然而并不是所有情况的相互作用都有一个明确的方向关系。当相互的作用并没有本质性的指向，或者是明确的双向相互作用时，使用无向模型更加合适。

作为一个这种情况的例子，假设我们希望对三个二值随机变量建模：你是否生病，你的同事是否生病以及你的室友是否生病。就像在接力赛的例子中所作的简化假设一样，我们可以在这里做一些关于相互作用的简化假设。假设你的室友和同事并不认识，所以他们不太可能直接相互传染一些疾病，比如说感冒。这个事件太过罕见，所以我们不对此事件建模。然而，很有可能其中之一将感冒传染给你，然后通过你再传染给了另一个人。我们通过对你的同事传染给你以及你传染给你的室友建模来对这种间接的从你的同事到你的室友的感冒传染建模。

在这种情况下，你传染给你的室友和你的室友传染给你都是非常容易的，所以模型不存在一个明确的单向箭头。这启发我们使用无向模型。其中随机变量对应着图中的相互作用的结点。与有向模型相同的是，如果在无向模型中的两个结点通过一条边相连接，那么对应这些结点的随机变量相互之间是直接作用的。不同于有向模型，在无向模型中的边是没有方向的，并不与一个条件分布相关联。

我们把对应你健康状况的随机变量记作  $h_y$ ，对应你的室友健康状况的随机变量记作  $h_r$ ，你的同事健康的变量记作  $h_c$ 。图 16.3 表示这种关系。

正式地说，一个无向模型是一个定义在无向模型  $\mathcal{G}$  上的结构化概率模型。对于图中的每一个团<sup>3</sup>  $\mathcal{C}$ ，一个因子 (factor)  $\phi(\mathcal{C})$ (也称为 团势能 (clique potential) )，

---

<sup>3</sup>图的一个团是图中结点的一个子集，并且其中的点是全连接的

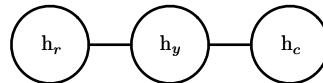


图 16.3: 表示你室友健康状况的  $h_r$ 、你健康状况的  $h_y$  和你同事健康状况的  $h_c$  之间如何相互影响的一个无向图。你和你的室友可能会相互传染感冒，你和你的同事之间也是如此，但是假设你室友和同事之间相互不认识，他们只能通过你来间接传染。

		$h_y = 0$	$h_y = 1$
		2	1
$h_c = 0$	2	1	
$h_c = 1$	1	10	

衡量了团中变量每一种可能的联合状态所对应的密切程度。这些因子都被限制为是非负的。它们一起定义了 **未归一化概率函数** (unnormalized probability function) :

$$\tilde{p}(\mathbf{x}) = \prod_{\mathcal{C} \in \mathcal{G}} \phi(\mathcal{C}). \quad (16.3)$$

只要所有团中的结点数都不大，那么我们就能够高效地处理这些未归一化概率函数。它包含了这样的思想，密切度越高的状态有越大的概率。然而，不像贝叶斯网络，几乎不存在团定义的结构，所以不能保证把它们乘在一起能够得到一个有效的概率分布。图 16.4 展示了一个从无向模型中读取分解信息的例子。

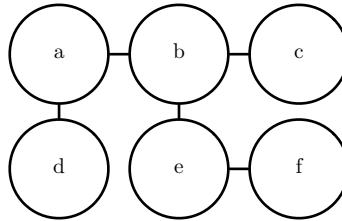


图 16.4: 这个图说明通过选择适当的  $\phi$ ，函数  $p(a, b, c, d, e, f)$  可以写作  $\frac{1}{Z} \phi_{a,b}(a, b) \phi_{b,c}(b, c) \phi_{a,d}(a, d) \phi_{b,e}(b, e) \phi_{e,f}(e, f)$ 。

在你、你的室友和同事之间感冒传染的例子中包含了两个团。一个团包含了  $h_y$  和  $h_c$ 。这个团的因子可以通过一个表来定义，可能取到下面的值：

状态为 1 代表了健康的状态，相对的状态为 0 则表示不好的健康状态（即感染了感冒）。你们两个通常都是健康的，所以对应的状态拥有最高的密切程度。两个人中只有一个人是生病的密切程度是最低的，因为这是一个很罕见的状态。两个人都

生病的状态（通过一个人来传染给了另一个人）有一个稍高的密切程度，尽管仍然不及两个人都健康的密切程度。

为了完整地定义这个模型，我们需要对包含  $h_y$  和  $h_r$  的团定义类似的因子。

### 16.2.3 配分函数

尽管这个未归一化概率函数处处不为零，我们仍然无法保证它的概率之和或者积分为 1。为了得到一个有效的概率分布，我们需要使用对应的归一化的概率分布<sup>4</sup>：

$$p(\mathbf{x}) = \frac{1}{Z} \tilde{p}(\mathbf{x}), \quad (16.4)$$

其中， $Z$  是使得所有的概率之和或者积分为 1 的常数，并且满足：

$$Z = \int \tilde{p}(\mathbf{x}) d\mathbf{x}. \quad (16.5)$$

当函数  $\phi$  固定时，我们可以把  $Z$  当成是一个常数。值得注意的是如果函数  $\phi$  带有参数时，那么  $Z$  是这些参数的一个函数。在相关文献中为了节省空间忽略控制  $Z$  的变量而直接写  $Z$  是一个常用的方式。归一化常数  $Z$  被称作是配分函数，这是一个从统计物理学中借鉴的术语。

由于  $Z$  通常是由对所有可能的  $\mathbf{x}$  状态的联合分布空间求和或者求积分得到的，它通常是很困难计算的。为了获得一个无向模型的归一化概率分布，模型的结构和函数  $\phi$  的定义通常需要设计为有助于高效地计算  $Z$ 。在深度学习中， $Z$  通常是难以处理的。由于  $Z$  难以精确地计算出，我们只能使用一些近似的方法。这样的近似方法是第十八章的主要内容。

在设计无向模型时，我们必须牢记在心的一个要点是设定一些使得  $Z$  不存在的因子也是有可能的。当模型中的一些变量是连续的，且  $\tilde{p}$  在其定义域上的积分发散时这种情况就会发生。例如，当我们需要对一个单独的标量变量  $x \in \mathbb{R}$  建模，并且单个团势能定义为  $\phi(x) = x^2$  时。在这种情况下，

$$Z = \int x^2 dx. \quad (16.6)$$

由于这个积分是发散的，所以不存在一个对应着这个势能函数  $\phi(x)$  的概率分布。有时候  $\phi$  函数某些参数的选择可以决定相应的概率分布是否能够被定义。例如，对  $\phi$

---

<sup>4</sup>一个通过归一化团势能乘积定义的分布也被称作是吉布斯分布 (Gibbs distribution)

函数  $\phi(x; \beta) = \exp(-\beta x^2)$  来说，参数  $\beta$  决定了归一化常数  $Z$  是否存在。正的  $\beta$  使得  $\phi$  函数是一个关于  $x$  的高斯分布，但是非正的参数  $\beta$  则使得  $\phi$  不可能被归一化。

有向建模和无向建模之间一个重要的区别就是有向模型是通过从起始点的概率分布直接定义的，反之无向模型的定义显得更加宽松，通过  $\phi$  函数转化为概率分布而定义。这改变了我们处理这些建模问题的直觉。当我们处理无向模型时需要牢记一点，每一个变量的定义域对于一系列给定的  $\phi$  函数所对应的概率分布有着重要的影响。举个例子，我们考虑一个  $n$  维向量的随机变量  $\mathbf{x}$  以及一个由偏置向量  $\mathbf{b}$  参数化的无向模型。假设  $\mathbf{x}$  的每一个元素对应着一个团，并且满足  $\phi^{(i)}(\mathbf{x}_i) = \exp(b_i \mathbf{x}_i)$ 。在这种情况下概率分布是怎样的呢？答案是我们无法确定，因为我们并没有指定  $\mathbf{x}$  的定义域。如果  $\mathbf{x}$  满足  $\mathbf{x} \in \mathbb{R}^n$ ，那么有关归一化常数  $Z$  的积分是发散的，这导致了对应的概率分布是不存在的。如果  $\mathbf{x} \in \{0, 1\}^n$ ，那么  $p(\mathbf{x})$  可以被分解成  $n$  个独立的分布，并且满足  $p(\mathbf{x}_i = 1) = \text{sigmoid}(b_i)$ 。如果  $\mathbf{x}$  的定义域是基本单位向量 ( $\{[1, 0, \dots, 0], [0, 1, \dots, 0], \dots, [0, 0, \dots, 1]\}$ ) 的集合，那么  $p(\mathbf{x}) = \text{softmax}(\mathbf{b})$ ，因此对于  $j \neq i$ ，一个较大的  $b_j$  的值会降低所有  $p(\mathbf{x}_j = 1)$  的概率。通常情况下，通过仔细选择变量的定义域，能够从一个相对简单的  $\phi$  函数的集合可以获得一个相对复杂的表达。我们会在第 20.6 节中讨论这个想法的实际应用。

#### 16.2.4 基于能量的模型

无向模型中许多有趣的理论结果都依赖于  $\forall \mathbf{x}, \tilde{p}(\mathbf{x}) > 0$  这个假设。使这个条件满足的一种简单方式是使用 **基于能量的模型** (Energy-based model, EBM)，其中

$$\tilde{p}(\mathbf{x}) = \exp(-E(\mathbf{x})), \quad (16.7)$$

$E(\mathbf{x})$  被称作是 **能量函数** (energy function)。对所有的  $z$ ,  $\exp(z)$  都是正的，这保证了没有一个能量函数会使得某一个状态  $\mathbf{x}$  的概率为 0。我们可以完全自由地选择那些能够简化学习过程的能量函数。如果我们直接学习各个团势能，我们需要利用约束优化方法来任意地指定一些特定的最小概率值。学习能量函数的过程中，我们可以采用无约束的优化方法<sup>5</sup>。基于能量的模型中的概率可以无限趋近于 0 但是永远达不到 0。

服从式(16.7)形式的任意分布都是**玻尔兹曼分布** (Boltzmann distribution) 的一个实例。正是基于这个原因，我们把许多基于能量的模型称为**玻尔兹曼机**

---

<sup>5</sup>对于某些模型，我们可以仍然使用约束优化方法来确保  $Z$  存在。

( Boltzmann Machine ) (Fahlman *et al.*, 1983; Ackley *et al.*, 1985; Hinton *et al.*, 1984a; Hinton and Sejnowski, 1986)。关于什么时候称之为基于能量的模型，什么时候称之为玻尔兹曼机不存在一个公认的判别标准。一开始玻尔兹曼机这个术语是用来描述一个只有二值变量的模型，但是如今许多模型，比如均值-协方差 RBM，也涉及到了实值变量。虽然玻尔兹曼机最初的定义既可以包含潜变量也可以不包含潜变量，但是时至今日玻尔兹曼机这个术语通常用于指拥有潜变量的模型，而没有潜变量的玻尔兹曼机则经常被称为马尔可夫随机场或对数线性模型。

无向模型中的团对应于未归一化概率函数中的因子。通过  $\exp(a + b) = \exp(a)\exp(b)$ ，我们发现无向模型中的不同团对应于能量函数的不同项。换句话说，基于能量的模型只是一种特殊的马尔可夫网络：求幂使能量函数中的每个项对应于不同团的一个因子。关于如何从无向模型结构中获得能量函数形式的示例可以参考图 16.5。人们可以将能量函数中带有多个项的基于能量的模型视作是**专家之积** (product of expert) (Hinton, 1999)。能量函数中的每一项对应的是概率分布中的一个因子。能量函数中的每一项都可以看作决定一个特定的软约束是否能够满足的“专家”。每个专家只执行一个约束，而这个约束仅仅涉及随机变量的一个低维投影，但是当其结合概率的乘法时，专家们一同构造了复杂的高维约束。

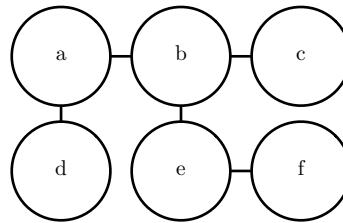


图 16.5: 这个图说明通过为每个团选择适当的能量函数  $E(a, b, c, d, e, f)$  可以写作  $E_{a,b}(a, b) + E_{b,c}(b, c) + E_{a,d}(a, d) + E_{b,e}(b, e) + E_{e,f}(e, f)$ 。值得注意的是，我们令  $\phi$  等于对应负能量的指数，可以获得图 16.4 中的  $\phi$  函数，比如， $\phi_{a,b}(a, b) = \exp(-E(a, b))$ 。

基于能量的模型定义的一部分无法用机器学习观点来解释：即式 (16.7) 中的“-”符号。这个“-”符号可以被包含在  $E$  的定义之中。对于很多  $E$  函数的选择来说，学习算法可以自由地决定能量的符号。这个负号的存在主要是为了保持机器学习文献和物理学文献之间的兼容性。概率建模的许多研究最初都是由统计物理学家做出的，其中  $E$  是指实际的、物理概念的能量，没有任何符号。诸如“能量”和“配分函数”这类术语仍然与这些技术相关联，尽管它们的数学适用性比在物理中更宽。一些机器学习研究者（例如，Smolensky (1986) 将负能量称为 **harmony**）发出了不同的声

音，但这些都不是标准惯例。

许多对概率模型进行操作的算法不需要计算  $p_{\text{model}}(\mathbf{x})$ ，而只需要计算  $\log \tilde{p}_{\text{model}}(\mathbf{x})$ 。对于具有潜变量  $\mathbf{h}$  的基于能量的模型，这些算法有时会将该量的负数称为自由能（free energy）：

$$\mathcal{F}(\mathbf{x}) = -\log \sum_{\mathbf{h}} \exp(-E(\mathbf{x}, \mathbf{h})). \quad (16.8)$$

在本书中，我们更倾向于更为通用的基于  $\log \tilde{p}_{\text{model}}(\mathbf{x})$  的定义。

### 16.2.5 分离和 d-分离

图模型中的边告诉我们哪些变量直接相互作用。我们经常需要知道哪些变量间接相互作用。某些间接相互作用可以通过观察其他变量来启用或禁用。更正式地，我们想知道在给定其他变量子集的值时，哪些变量子集彼此条件独立。

在无向模型中，识别图中的条件独立性是非常简单的。在这种情况下，图中隐含的条件独立性称为 **分离**（separation）。如果图结构显示给定变量集  $S$  的情况下变量集  $A$  与变量集  $B$  无关，那么我们声称给定变量集  $S$  时，变量集  $A$  与另一组变量集  $B$  是分离的。如果连接两个变量  $a$  和  $b$  的连接路径仅涉及未观察变量，那么这些变量不是分离的。如果它们之间没有路径，或者所有路径都包含可观测的变量，那么它们是分离的。我们认为仅涉及未观察到的变量的路径是“活跃”的，而包括可观察变量的路径称为“非活跃”的。

当我们画图时，我们可以通过加阴影来表示观察到的变量。图 16.6 用于描述当以这种方式绘图时无向模型中的活跃和非活跃路径的样子。图 16.7 描述了一个从无向模型中读取分离信息的例子。

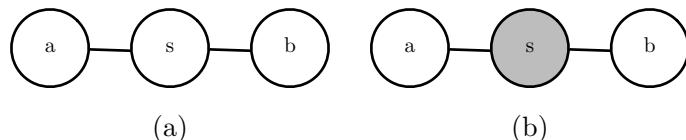


图 16.6: (a) 随机变量  $a$  和随机变量  $b$  之间穿过  $s$  的路径是活跃的，因为  $s$  是观察不到的。这意味着  $a$ ,  $b$  之间不是分离的。(b) 图中  $s$  用阴影填充，表示它是可观察的。因为  $a$  和  $b$  之间的唯一路径通过  $s$ ，并且这条路径是不活跃的，我们可以得出结论，在给定  $s$  的条件下  $a$  和  $b$  是分离的。

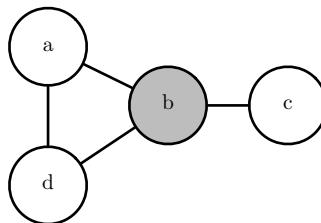


图 16.7: 从一个无向图中读取分离性质的一个例子。这里 b 用阴影填充, 表示它是可观察的。由于 b 挡住了从 a 到 c 的唯一路径, 我们说在给定 b 的情况下 a 和 c 是相互分离的。观察值 b 同样挡住了从 a 到 d 的一条路径, 但是它们之间有另一条活跃路径。因此给定 b 的情况下 a 和 d 不是分离的。

类似的概念适用于有向模型, 只是在有向模型中, 这些概念被称为 **d-分离** (*d-separation*)。“d”代表“依赖”的意思。有向图中 d-分离的定义与无向模型中分离的定义相同: 如果图结构显示给定变量集  $S$  时, 变量集  $A$  与变量集  $B$  无关, 那么我们认为给定变量集  $S$  时, 变量集  $A$  d-分离于变量集  $B$ 。

与无向模型一样, 我们可以通过查看图中存在的活跃路径来检查图中隐含的独立性。如前所述, 如果两个变量之间存在活跃路径, 则两个变量是依赖的, 如果没有活跃路径, 则为d-分离。在有向网络中, 确定路径是否活跃有点复杂。关于在有向模型中识别活跃路径的方法可以参考图 16.8。图 16.9是从一个图中读取一些属性的例子。

尤其重要的是要记住分离和d-分离只能告诉我们图中隐含的条件独立性。图并不需要表示所有存在的独立性。进一步的, 使用完全图(具有所有可能的边的图)来表示任何分布总是合法的。事实上, 一些分布包含不可能用现有图形符号表示的独立性。**特定环境下的独立** (*context-specific independences*) 指的是取决于网络中一些变量值的独立性。例如, 考虑三个二值变量的模型:  $a$ ,  $b$  和  $c$ 。假设当  $a$  是 0 时,  $b$  和  $c$  是独立的, 但是当  $a$  是 1 时,  $b$  确定地等于  $c$ 。当  $a = 1$  时图模型需要连接  $b$  和  $c$  的边。但是图不能说明当  $a = 0$  时  $b$  和  $c$  不是独立的。

一般来说, 当独立性不存在时, 图不会显示独立性。然而, 图可能无法编码独立性。

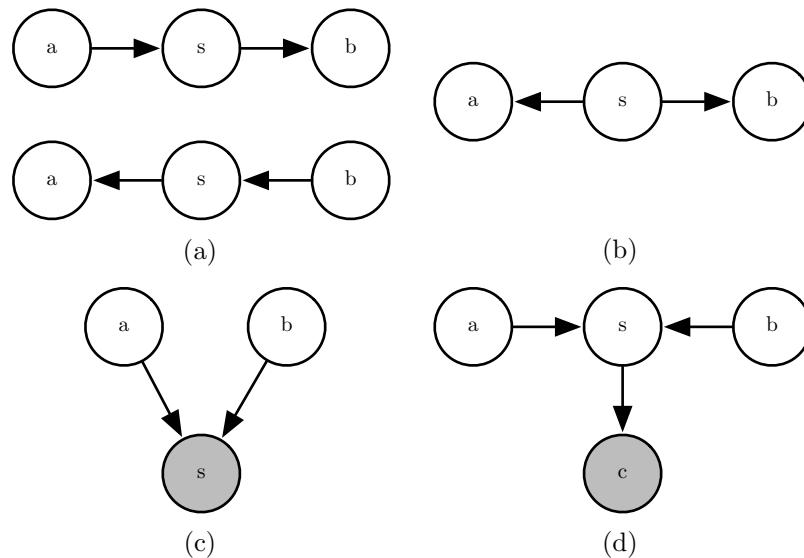


图 16.8: 两个随机变量  $a$ ,  $b$  之间存在的长度为 2 的所有种类的活跃路径。*(a)* 箭头方向从  $a$  指向  $b$  的任何路径, 反过来也一样。如果  $s$  可以被观察到, 这种路径就是阻塞的。在接力赛的例子中, 我们已经看到过这种类型的路径。*(b)* 变量  $a$  和  $b$  通过共因  $s$  相连。举个例子, 假设  $s$  是一个表示是否存在飓风的变量,  $a$  和  $b$  表示两个相邻气象监控区域的风速。如果我们在  $a$  处观察到很高的风速, 我们可以期望在  $b$  处也观察到高速的风。如果观察到  $s$ , 那么这条路径就被阻塞了。如果我们已经知道存在飓风, 那么无论  $a$  处观察到什么, 我们都能期望  $b$  处有较高的风速。在  $a$  处观察到一个低于预期的风速(对飓风而言)并不会改变我们对  $b$  处风速的期望(已知有飓风的情况下)。然而, 如果  $s$  不被观测到, 那么  $a$  和  $b$  是依赖的, 即路径是活跃的。*(c)* 变量  $a$  和  $b$  都是  $s$  的父节点。这称为**V-结构**(V-structure)或者**碰撞情况**(the collider case)。根据**相消解释作用**(explaining away effect), V-结构导致  $a$  和  $b$  是相关的。在这种情况下, 当  $s$  被观测到时路径是活跃的。举个例子, 假设  $s$  是一个表示你的同事不在工作的变量。变量  $a$  表示她生病了, 而变量  $b$  表示她在休假。如果你观察到了她不在工作, 你可以假设她很有可能是生病了或者是在度假, 但是这两件事同时发生是不太可能的。如果你发现她在休假, 那么这个事实足够解释她的缺席了。你可以推断她很可能没有生病。*(d)* 即使  $s$  的任意后代都被观测到, 相消解释作用也会起作用。举个例子, 假设  $c$  是一个表示你是否收到你同事的报告的一个变量。如果你注意到你还没有收到这个报告, 这会增加你估计的她今天不在工作的概率, 这反过来又会增加她今天生病或者度假的概率。阻塞 V-结构中路径的唯一方法就是共享子节点的后代一个都观察不到。

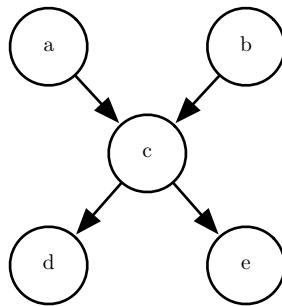


图 16.9: 从这张图中，我们可以发现一些 d-分离的性质。这包括了：

- 给定空集的情况下，a 和 b 是 d-分离的。
- 给定 c 的情况下，a 和 e 是 d-分离的。
- 给定 c 的情况下，d 和 e 是 d-分离的。

我们还可以发现当我们观察到一些变量时，一些变量不再是 d-分离的：

- 给定 c 的情况下，a 和 b 不是 d-分离的。
- 给定 d 的情况下，a 和 b 不是 d-分离的。

### 16.2.6 在有向模型和无向模型中转换

我们经常将特定的机器学习模型称为无向模型或有向模型。例如，我们通常将受限玻尔兹曼机称为无向模型，而稀疏编码则被称为有向模型。这种措辞的选择可能有点误导，因为没有概率模型本质上是有向或无向的。但是，一些模型很适合使用有向图描述，而另一些模型很适合使用无向模型描述。

有向模型和无向模型都有其优点和缺点。这两种方法都不是明显优越和普遍优选的。相反，我们根据具体的每个任务来决定使用哪一种模型。这个选择部分取决于我们希望描述的概率分布。根据哪种方法可以最大程度地捕捉到概率分布中的独立性，或者哪种方法使用最少的边来描述分布，我们可以决定使用有向建模还是无向建模。还有其他因素可以影响我们决定使用哪种建模方式。即使在使用单个概率分布时，我们有时也可以在不同的建模方式之间切换。有时，如果我们观察到变量的某个子集，或者如果我们希望执行不同的计算任务，换一种建模方式可能更合适。例如，有向模型通常提供了一种高效地从模型中抽取样本（在第 16.3 节中描述）的直接方法。而无向模型形式通常对于推导近似推断过程（我们将在第十九章中看到，式(19.56)强调了无向模型的作用）是很有用的。

每个概率分布可以由有向模型或由无向模型表示。在最坏的情况下，我们可以使用“完全图”来表示任何分布。在有向模型的情况下，完全图是任意有向无环图，其中我们对随机变量排序，并且每个变量在排序中位于其之前的所有其他变量作为其图中的祖先。对于无向模型，完全图只是包含所有变量的单个团。图 16.10 给出了一个实例。

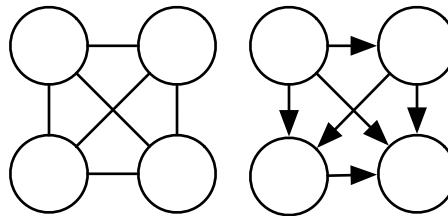


图 16.10：完全图的例子，完全图能够描述任何的概率分布。这里我们展示了一个带有四个随机变量的例子。(左) 完全无向图。在无向图中，完全图是唯一的。(右) 一个完全有向图。在有向图中，并不存在唯一的完全图。我们选择一种变量的排序，然后对每一个变量，从它本身开始，向每一个指向顺序在其后面的变量画一条弧。因此存在着关于变量数阶乘数量级的不同种完全图。在这个例子中，我们从左到右从上到下地排序变量。

当然，图模型的优势在于图能够包含一些变量不直接相互作用的信息。完全图并不是很有用，因为它并不隐含任何独立性。

当我们用图表示概率分布时，我们想要选择一个包含尽可能多独立性的图，但是并不会假设任何实际上不存在的独立性。

从这个角度来看，一些分布可以使用有向模型更高效地表示，而其他分布可以使用无向模型更高效地表示。换句话说，有向模型可以编码一些无向模型所不能编码的独立性，反之亦然。

有向模型能够使用一种无向模型无法完美表示的特定类型的子结构。这个子结构被称为**不道德** (immorality)。这种结构出现在当两个随机变量  $a$  和  $b$  都是第三个随机变量  $c$  的父结点，并且不存在任一方向上直接连接  $a$  和  $b$  的边时。（“不道德”的名字可能看起来很奇怪；它在图模型文献中使用源于一个关于未婚父母的笑话。）为了将有向模型图  $\mathcal{D}$  转换为无向模型，我们需要创建一个新图  $\mathcal{U}$ 。对于每对变量  $x$  和  $y$ ，如果存在连接  $\mathcal{D}$  中的  $x$  和  $y$  的有向边（在任一方向上），或者如果  $x$  和  $y$  都是图  $\mathcal{D}$  中另一个变量  $z$  的父节点，则在  $\mathcal{U}$  中添加连接  $x$  和  $y$  的无向边。得到的图  $\mathcal{U}$  被称为是**道德图** (moralized graph)。关于一个通过道德化将有向图模型转化为无向模型的例子可以参考图 16.11。

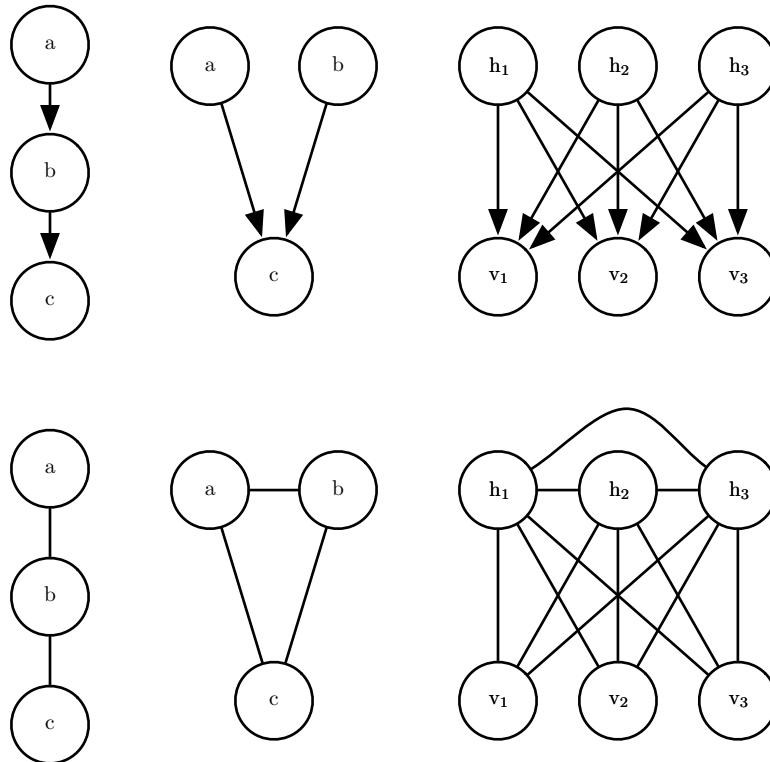


图 16.11: 通过构造道德图将有向模型（上一行）转化为无向模型（下一行）的例子。（左）只需要把有向边替换成无向边就可以把这个简单的链转化为一个道德图。得到的无向模型包含了完全相同的独立关系和条件独立关系。（中）这个图是在不丢失独立性的情况下是无法转化为无向模型的最简单的有向模型。这个图包含了单个完整的不道德结构。因为  $a$  和  $b$  都是  $c$  的父节点，当  $c$  被观察到时，它们之间通过活跃路径相连。为了捕捉这个依赖，无向模型必须包含一个含有所有三个变量的团。这个团无法编码  $a \perp b$  这个信息。（右）一般来说，道德化的过程会给图添加许多边，因此丢失了一些隐含的独立性。举个例子，这个稀疏编码图需要在每一对隐藏单元之间添加道德化的边，因此也引入了二次数量级的新的直接依赖。

同样的，无向模型可以包括有向模型不能完美表示的子结构。具体来说，如果  $\mathcal{U}$  包含长度大于 3 的环 (loop)，则有向图  $\mathcal{D}$  不能捕获无向模型  $\mathcal{U}$  所包含的所有条件独立性，除非该环还包含弦 (chord)。环指的是由无向边连接的变量序列，并且满足序列中的最后一个变量连接回序列中的第一个变量。弦是定义环序列中任意两个非连续变量之间的连接。如果  $\mathcal{U}$  具有长度为 4 或更大的环，并且这些环没有弦，我们必须在将它们转换为有向模型之前添加弦。添加这些弦会丢弃在  $\mathcal{U}$  中编码的一些

独立信息。通过将弦添加到  $\mathcal{U}$  形成的图被称为 **弦图** (chordal graph) 或者 **三角形化图** (triangulated graph)，因为我们现在可以用更小的、三角的环来描述所有的环。要从弦图构建有向图  $\mathcal{D}$ ，我们还需要为边指定方向。当这样做时，我们不能在  $\mathcal{D}$  中创建有向循环，否则将无法定义有效的有向概率模型。为  $\mathcal{D}$  中的边分配方向的一种方法是对随机变量排序，然后将每个边从排序较早的节点指向排序稍后的节点。一个简单的实例可以参考图 16.12。

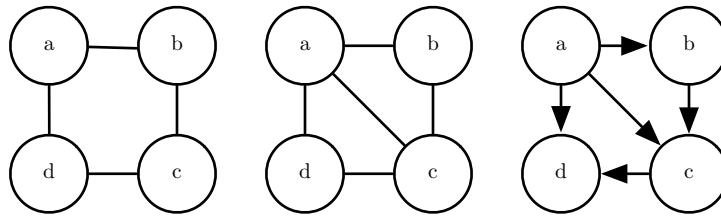


图 16.12：将一个无向模型转化为一个有向模型。(左) 这个无向模型无法转化为有向模型，因为它有一个长度为 4 且不带有弦的环。具体说来，这个无向模型包含了两种不同的独立性，并且不存在一个有向模型可以同时描述这两种性质： $a \perp c \mid \{b, d\}$  和  $b \perp d \mid \{a, c\}$ 。(中) 为了将无向图转化为有向图，我们必须通过保证所有长度大于 3 的环都有弦来三角形化图。为了实现这个目标，我们可以加一条连接  $a$  和  $c$  或者连接  $b$  和  $d$  的边。在这个例子中，我们选择添加一条连接  $a$  和  $c$  的边。(右) 为了完成转化的过程，我们必须给每条边分配一个方向。执行这个任务时，我们必须保证不产生任何有向环。避免出现有向环的一种方法是赋予节点一定的顺序，然后将每个边从排序较早的节点指向排序稍后的节点。在这个例子中，我们根据变量名的字母进行排序。

### 16.2.7 因子图

**因子图** (factor graph) 是从无向模型中抽样的另一种方法，它可以解决标准无向模型语法中图表达的模糊性。在无向模型中，每个  $\phi$  函数的范围必须是图中某个团的子集。我们无法确定每一个团是否含有一个作用域包含整个团的因子——比如说一个包含三个结点的团可能对应的是一个有三个结点的因子，也可能对应的是三个因子并且每个因子包含了一对结点，这通常会导致模糊性。通过显式地表示每一个  $\phi$  函数的作用域，因子图解决了这种模糊性。具体来说，因子图是一个包含无向二分图的无向模型的图形化表示。一些节点被绘制为圆形。就像在标准无向模型中一样，这些节点对应于随机变量。其余节点绘制为方块。这些节点对应于未归一化概率函数的因子  $\phi$ 。变量和因子可以通过无向边连接。当且仅当变量包含在未归一化概率函数的因子中时，变量和因子在图中存在连接。没有因子可以连接到图中的

另一个因子，也不能将变量连接到变量。图 16.2.7 给出了一个例子来说明因子图如何解决无向网络中的模糊性。

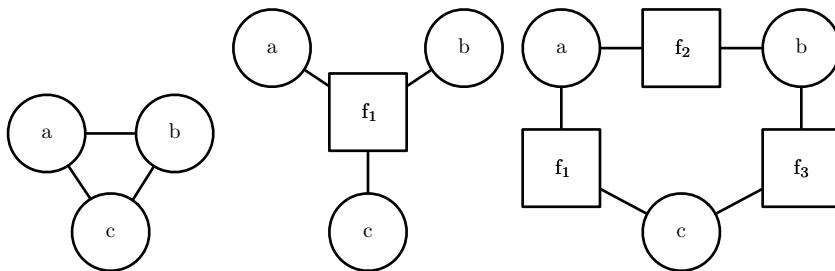


图 16.13: 因子图如何解决无向网络中的模糊性的一个例子。(左)一个包含三个变量 (a、b 和 c) 的团组成的无向网络。(中)对应这个无向模型的因子图。这个因子图有一个包含三个变量的因子。(右)对应这个无向模型的另一种有效的因子图。这个因子图包含了三个因子，每个因子只对应两个变量。即使它们表示的是同一个无向模型，这个因子图上进行的表示、推断和学习相比于中图描述的因子图都要渐近地廉价。

### 16.3 从图模型中采样

图模型同样简化了从模型中采样的过程。

有向图模型的一个优点是，可以通过一个简单高效的过程从模型所表示的联合分布中产生样本，这个过程被称为 **原始采样** (Ancestral Sampling)。

原始采样的基本思想是将图中的变量  $x_i$  使用拓扑排序，使得对于所有  $i$  和  $j$ ，如果  $x_i$  是  $x_j$  的一个父亲结点，则  $j$  大于  $i$ 。然后可以按此顺序对变量进行采样。换句话说，我们可以首先采  $x_1 \sim P(x_1)$ ，然后采  $x_2 \sim P(x_2 | Pa_G(x_2))$ ，以此类推，直到最后我们从  $P(x_n | Pa_G(x_n))$  中采样。只要不难从每个条件分布  $x_i \sim P(x_i | Pa_G(x_i))$  中采样，那么从整个模型中采样也是容易的。拓扑排序操作保证我们可以按照式 (16.1) 中条件分布的顺序依次采样。如果没有拓扑排序，我们可能会在其父节点可用之前试图对该变量进行抽样。

有些图可能存在多个拓扑排序。原始采样可以使用这些拓扑排序中的任何一个。

原始采样通常非常快（假设从每个条件分布中采样都是很容易的）并且非常简便。

原始采样的一个缺点是其仅适用于有向图模型。另一个缺点是它并不是每次采

样都是条件采样操作。当我们希望从有向图模型中变量的子集中采样时，给定一些其他变量，我们经常要求所有给定的条件变量在顺序图中比要采样的变量的顺序要早。在这种情况下，我们可以从模型分布指定的局部条件概率分布中采样。否则，我们需要采样的条件分布是给定观测变量的后验分布。这些后验分布在模型中通常没有明确指定和参数化。推断这些后验分布的代价可能是很高的。在这种情况下的模型中，原始采样不再有效。

不幸的是，原始采样仅适用于有向模型。我们可以通过将无向模型转换为有向模型来实现从无向模型中抽样，但是这通常需要解决棘手的推断问题（要确定新有向图的根节点上的边缘分布），或者需要引入许多边从而会使得到的有向模型变得难以处理。从无向模型采样，而不首先将其转换为有向模型的做法似乎需要解决循环依赖的问题。每个变量与每个其他变量相互作用，因此对于采样过程没有明确的起点。不幸的是，从无向模型中抽取样本是一个成本很高的多次迭代的过程。理论上最简单的方法是 **Gibbs 采样** (Gibbs Sampling)。假设我们在一个  $n$  维向量的随机变量  $\mathbf{x}$  上有一个图模型。我们迭代地访问每个变量  $x_i$ ，在给定其他变量的条件下从  $p(x_i | \mathbf{x}_{-i})$  中抽样。由于图模型的分离性质，抽取  $x_i$  时我们可以等价地仅对  $x_i$  的邻居条件化。不幸的是，在我们遍历图模型一次并采样所有  $n$  个变量之后，我们仍然无法得到一个来自  $p(\mathbf{x})$  的客观样本。相反，我们必须重复该过程并使用它们邻居的更新值对所有  $n$  个变量重新取样。在多次重复之后，该过程渐近地收敛到正确的目标分布。我们很难确定样本何时达到所期望分布的足够精确的近似。无向模型的采样技术是一个高级的研究方向，第十七章将对此进行更详细的讨论。

## 16.4 结构化建模的优势

使用结构化概率模型的主要优点是它们能够显著降低表示概率分布、学习和推断的成本。有向模型中采样还可以被加速，但是对于无向模型情况则较为复杂。选择不对某些变量的相互作用进行建模是允许所有这些操作使用较少的运行时间和内存的主要机制。图模型通过省略某些边来传达信息。在没有边的情况下，模型假设不对变量间直接的相互作用建模。

结构化概率模型允许我们明确地将给定的现有知识与知识的学习或者推断分开，这是一个不容易量化的益处。这使我们的模型更容易开发和调试。我们可以设计、分析和评估适用于更广范围的图的学习算法和推断算法。同时，我们可以设计能够捕捉到我们认为数据中存在的主要关系的模型。然后，我们可以组合这些不同的算

法和结构，并获得不同可能性的笛卡尔乘积。然而，为每种可能的情况设计端到端的算法会更加困难。

## 16.5 学习依赖关系

良好的生成模型需要准确地捕获所观察到的或“可见”变量  $\mathbf{v}$  上的分布。通常  $\mathbf{v}$  的不同元素彼此高度依赖。在深度学习中，最常用于建模这些依赖关系的方法是引入几个潜在或“隐藏”变量  $\mathbf{h}$ 。然后，该模型可以捕获任何对（变量  $v_i$  和  $v_j$  间接依赖可以通过  $v_i$  和  $\mathbf{h}$  之间直接依赖和  $\mathbf{h}$  和  $v_j$  直接依赖捕获）之间的依赖关系。

如果一个良好的关于  $\mathbf{v}$  的模型不包含任何潜变量，那么它在贝叶斯网络中的每个节点需要具有大量父节点或在马尔可夫网络中具有非常大的团。仅仅表示这些高阶相互作用的成本就很高了，首先从计算角度上考虑，存储在存储器中的参数数量是团中成员数量的指数级别，接着在统计学意义上，因为这些指数数量的参数需要大量的数据来准确估计。

当模型旨在描述直接连接的可见变量之间的依赖关系时，通常不可能连接所有变量，因此设计图模型时需要连接那些紧密相关的变量，并忽略其他变量之间的作用。机器学习中有一个称为 **结构学习** (structure learning) 的领域专门讨论这个问题。Koller and Friedman (2009) 是一个不错的结构学习参考资料。大多数结构学习技术基于一种贪婪搜索的形式。它们提出了一种结构，对具有该结构的模型进行训练，然后给出分数。该分数奖励训练集上的高精度并对模型的复杂度进行惩罚。然后提出添加或移除少量边的候选结构作为搜索的下一步。搜索向一个预计会增加分数的新结构发展。

使用潜变量而不是自适应结构避免了离散搜索和多轮训练的需要。可见变量和潜变量之间的固定结构可以使用可见单元和隐藏单元之间的直接作用，从而建模可见单元之间的间接作用。使用简单的参数学习技术，我们可以学习到一个具有固定结构的模型，这个模型在边缘分布  $p(\mathbf{v})$  上拥有正确的结构。

潜变量除了发挥本来的作用，即能够高效地描述  $p(\mathbf{v})$  以外，还具有另外的优势。新变量  $\mathbf{h}$  还提供了  $\mathbf{v}$  的替代表示。例如，如第 3.9.6 节所示，高斯混合模型学习了一个潜变量，这个潜变量对应于输入样本是从哪一个混合体中抽出。这意味着高斯混合模型中的潜变量可以用于做分类。我们可以看到第十四章中简单的概率模型如稀疏编码，是如何学习可以用作分类器输入特征或者作为流形上坐标的潜变量的。

其他模型也可以使用相同的方式，但是更深的模型和具有多种相互作用方式的模型可以获得更丰富的输入描述。许多方法通过学习潜变量来完成特征学习。通常，给定  $\mathbf{v}$  和  $\mathbf{h}$ ，实验观察显示  $\mathbb{E}[\mathbf{h} \mid \mathbf{v}]$  或  $\arg \max_h p(\mathbf{h}, \mathbf{v})$  都是  $\mathbf{v}$  的良好特征映射。

## 16.6 推断和近似推断

解决变量之间如何相互关联的问题是我们使用概率模型的一个主要方式。给定一组医学测试，我们可以询问患者可能患有什么疾病。在一个潜变量模型中，我们可能需要提取能够描述可观察变量  $\mathbf{v}$  的特征  $\mathbb{E}[\mathbf{h} \mid \mathbf{v}]$ 。有时我们需要解决这些问题来执行其他任务。我们经常使用最大似然的准则来训练我们的模型。由于

$$\log p(\mathbf{v}) = \mathbb{E}_{\mathbf{h} \sim p(\mathbf{h} \mid \mathbf{v})} [\log p(\mathbf{h}, \mathbf{v}) - \log p(\mathbf{h} \mid \mathbf{v})], \quad (16.9)$$

学习过程中，我们经常需要计算  $p(\mathbf{h} \mid \mathbf{v})$ 。所有这些都是 **推断** (inference) 问题的例子，其中我们必须预测给定其他变量的情况下一些变量的值，或者在给定其他变量值的情况下预测一些变量的概率分布。

不幸的是，对于大多数有趣的深度模型来说，即使我们使用结构化图模型来简化这些推断问题，它们仍然是难以处理的。图结构允许我们用合理数量的参数来表示复杂的高维分布，但是用于深度学习的图并不满足这样的条件，从而难以实现高效地推断。

我们可以直接看出，计算一般图模型的边缘概率是 #P-hard 的。复杂性类别 #P 是复杂性类别 NP 的泛化。NP 中的问题只需确定其中一个问题是否有解决方案，并找到一个解决方案（如果存在）就可以解决。#P 中的问题需要计算解决方案的数量。为了构建最坏情况的图模型，我们可以设想一下我们在 3-SAT 问题中定义二值变量的图模型。我们可以对这些变量施加均匀分布。然后我们可以为每个子句添加一个二值潜变量，来表示每个子句是否成立。然后，我们可以添加另一个潜变量，来表示所有子句是否成立。这可以通过构造一个潜变量的缩减树来完成，树中的每个结点表示其他两个变量是否成立，从而不需要构造一个大的团。该树的叶是每个子句的变量。树的根表示整个问题是否成立。由于子句的均匀分布，缩减树根结点的边缘分布表示子句有多少比例是成立的。虽然这是一个设计的最坏情况的例子，NP-hard 图确实会频繁地出现在现实世界的场景中。

这促使我们使用近似推断。在深度学习中，这通常涉及变分推断，其中通过寻求尽可能接近真实分布的近似分布  $q(\mathbf{h} \mid \mathbf{v})$  来逼近真实分布  $p(\mathbf{h} \mid \mathbf{v})$ 。这个技术将在

第十九章中深入讨论。

## 16.7 结构化概率模型的深度学习方法

深度学习从业者通常与其他从事结构化概率模型研究的机器学习研究者使用相同的基本计算工具。然而，在深度学习中，我们通常对如何组合这些工具作出不同的设计决定，导致总体算法、模型与更传统的图模型具有非常不同的风格。

深度学习并不总是涉及特别深的图模型。在图模型中，我们可以根据图模型的图而不是计算图来定义模型的深度。如果从潜变量  $h_i$  到可观察变量的最短路径是  $j$  步，我们可以认为潜变量  $h_j$  处于深度  $j$ 。我们通常将模型的深度描述为任何这样的  $h_j$  的最大深度。这种深度不同于由计算图定义的深度。用于深度学习的许多生成模型没有潜变量或只有一层潜变量，但使用深度计算图来定义模型中的条件分布。

深度学习基本上总是利用分布式表示的思想。即使是用于深度学习目的的浅层模型（例如预训练浅层模型，稍后将形成深层模型），也几乎总是具有单个大的潜变量层。深度学习模型通常具有比可观察变量更多的潜变量。变量之间复杂的非线性相互作用通过多个潜变量的间接连接来实现。

相比之下，传统的图模型通常包含至少是偶尔观察到的变量，即使一些训练样本中的许多变量随机地丢失。传统模型大多使用高阶项和结构学习来捕获变量之间复杂的非线性相互作用。如果有潜变量，它们的数量通常很少。

潜变量的设计方式在深度学习中也有所不同。深度学习从业者通常不希望潜变量提前包含了任何特定的含义——训练算法可以自由地开发对特定数据集建模所需要的概念。在事后解释潜变量通常是很困难的，但是可视化技术可以得到它们表示的一些粗略表征。当潜变量在传统图模型中使用时，它们通常被赋予一些特定含义——比如文档的主题、学生的智力、导致患者症状的疾病等。这些模型通常由研究者解释，并且通常具有更多的理论保证，但是不能扩展到复杂的问题，并且不能像深度模型一样在许多不同背景中重复使用。

另一个明显的区别是深度学习方法中经常使用的连接类型。深度图模型通常具有大的与其他单元组全连接的单元组，使得两个组之间的相互作用可以由单个矩阵描述。传统的图模型具有非常少的连接，并且每个变量的连接选择可以单独设计。模型结构的设计与推断算法的选择紧密相关。图模型的传统方法通常旨在保持精确推断的可解性。当这个约束太强时，我们可以采用一种流行的被称为 环状信念传播

(loopy belief propagation) 的近似推断算法。这两种方法通常在稀疏连接图上都有很好的效果。相比之下，在深度学习中使用的模型倾向于将每个可见单元  $v_i$  连接到非常多的隐藏单元  $h_j$  上，从而使得  $\mathbf{h}$  可以获得一个  $v_i$  的分布式表示（也可能是其他几个可观察变量）。分布式表示具有许多优点，但是从图模型和计算复杂性的观点来看，分布式表示有一个缺点就是很难产生对于精确推断和环状信念传播等传统技术来说足够稀疏的图。结果，大规模图模型和深度图模型最大的区别之一就是深度学习中几乎从来不会使用环状信念传播。相反的，许多深度学习模型可以设计来加速 Gibbs 采样或者变分推断。此外，深度学习模型包含了大量的潜变量，使得高效的数值计算代码显得格外重要。除了选择高级推断算法之外，这提供了另外的动机，用于将结点分组成层，相邻两层之间用一个矩阵来描述相互作用。这要求实现算法的单个步骤可以实现高效的矩阵乘积运算，或者专门适用于稀疏连接的操作，例如块对角矩阵乘积或卷积。

最后，图模型的深度学习方法的一个主要特征在于对未知量的较高容忍度。与简化模型直到它的每一个量都可以被精确计算不同的是，我们仅仅直接使用数据运行或者是训练，以增强模型的能力。我们一般使用边缘分布不能计算的模型，但可以从中简单地采近似样本。我们经常训练具有难以处理的目标函数的模型，我们甚至不能在合理的时间内近似，但是如果能够高效地获得这样一个函数的梯度估计，我们仍然能够近似训练模型。深度学习方法通常是找出我们绝对需要的最小量信息，然后找出如何尽快得到该信息的合理近似。

### 16.7.1 实例：受限玻尔兹曼机

**受限玻尔兹曼机** (Restricted Boltzmann Machine, RBM) (Smolensky, 1986) 或者 **簧风琴** (harmonium) 是图模型如何用于深度学习的典型例子。RBM 本身不是一个深层模型。相反，它有一层潜变量，可用于学习输入的表示。在第二十章中，我们将看到 RBM 如何被用来构建许多的深层模型。在这里，我们举例展示了 RBM 在许多深度图模型中使用的实践：它的单元被分成很大的组，这种组称作层，层之间的连接由矩阵描述，连通性相对密集。该模型被设计为能够进行高效的 Gibbs 采样，并且模型设计的重点在于以很高的自由度来学习潜变量，而潜变量的含义并不是设计者指定的。之后在第 20.2 节，我们将更详细地再次讨论 RBM。

标准的 RBM 是具有二值的可见和隐藏单元的基于能量的模型。其能量函数为

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{b}^\top \mathbf{v} - \mathbf{c}^\top \mathbf{h} - \mathbf{v}^\top \mathbf{W} \mathbf{h}, \quad (16.10)$$

其中  $\mathbf{b}$ ,  $\mathbf{c}$  和  $\mathbf{W}$  都是无约束、实值的可学习参数。我们可以看到，模型被分成两组单元： $\mathbf{v}$  和  $\mathbf{h}$ ，它们之间的相互作用由矩阵  $\mathbf{W}$  来描述。该模型在图 16.14 中以图的形式描绘。该图能够使我们更清楚地发现，该模型的一个重要方面是在任何两个可见单元之间或任何两个隐藏单元之间没有直接的相互作用（因此称为“受限”，一般的玻尔兹曼机可以具有任意连接）。

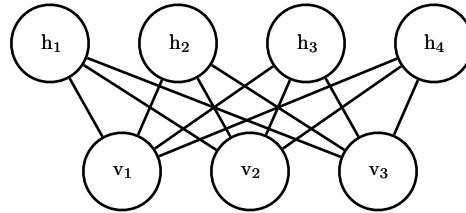


图 16.14: 一个画成马尔可夫网络形式的RBM。

对 RBM 结构的限制产生了良好的属性

$$p(\mathbf{h} \mid \mathbf{v}) = \prod_i p(h_i \mid \mathbf{v}) \quad (16.11)$$

以及

$$p(\mathbf{v} \mid \mathbf{h}) = \prod_i p(v_i \mid \mathbf{h}). \quad (16.12)$$

独立的条件分布很容易计算。对于二元的受限玻尔兹曼机，我们可以得到：

$$p(h_i = 1 \mid \mathbf{v}) = \sigma(\mathbf{v}^\top \mathbf{W}_{:,i} + b_i), \quad (16.13)$$

$$p(h_i = 0 \mid \mathbf{v}) = 1 - \sigma(\mathbf{v}^\top \mathbf{W}_{:,i} + b_i). \quad (16.14)$$

结合这些属性可以得到高效的 **块吉布斯采样** (block Gibbs Sampling)，它在同时采样所有  $\mathbf{h}$  和同时采样所有  $\mathbf{v}$  之间交替。RBM 模型通过 Gibbs 采样产生的样本展示在图 16.15 中。

由于能量函数本身只是参数的线性函数，很容易获取能量函数的导数。例如，

$$\frac{\partial}{\partial W_{i,j}} E(\mathbf{v}, \mathbf{h}) = -v_i h_j. \quad (16.15)$$

这两个属性，高效的 Gibbs 采样和导数计算，使训练过程变得非常方便。在第十八章中，我们将看到，可以通过计算应用于这种来自模型样本的导数来训练无向模型。



图 16.15: 训练好的 RBM 的样本及其权重。(左) 用 MNIST 训练模型, 然后用 Gibbs 采样进行采样。每一列是一个单独的 Gibbs 采样过程。每一行表示另一个 1000 步后 Gibbs 采样的输出。连续的样本之间彼此高度相关。(右) 对应的权重向量。将本图结果与图13.2中描述的线性因子模型的样本和权重相比。由于 RBM 的先验  $p(\mathbf{h})$  没有限制为因子, 这里的样本表现得好很多。采样时 RBM 能够学习到哪些特征需要一起出现。另一方面说, RBM 后验  $p(\mathbf{h} \mid \mathbf{v})$  是因子的, 而稀疏编码的后验并不是, 所以在特征提取上稀疏编码模型表现得更好。其他的模型可以使用非因子的  $p(\mathbf{h})$  和非因子的  $p(\mathbf{h} \mid \mathbf{h})$ 。图片经 LISA (2008) 允许转载。

训练模型可以得到数据  $\mathbf{v}$  的表示  $\mathbf{h}$ 。我们经常使用  $\mathbb{E}_{\mathbf{h} \sim p(\mathbf{h} \mid \mathbf{v})}[\mathbf{h}]$  作为一组描述  $\mathbf{v}$  的特征。

总的来说, RBM 展示了典型的图模型深度学习方法: 使用多层潜变量, 并由矩阵参数化层之间的高效相互作用来完成表示学习。

图模型为描述概率模型提供了一种优雅、灵活、清晰的语言。在未来的章节中, 我们将使用这种语言, 以其他视角来描述各种各样的深度概率模型。

# 第十七章 蒙特卡罗方法

随机算法可以粗略地分为两类：Las Vegas 算法和蒙特卡罗算法。Las Vegas 算法总是精确地返回一个正确答案（或者返回算法失败了）。这类方法通常需要占用随机量的计算资源（一般指内存或运行时间）。与此相对的，蒙特卡罗方法返回的答案具有随机大小的错误。花费更多的计算资源（通常包括内存和运行时间）可以减少这种错误。在任意固定的计算资源下，蒙特卡罗算法可以得到一个近似解。

对于机器学习中的许多问题来说，我们很难得到精确的答案。这类问题很难用精确的确定性算法如 Las Vegas 算法解决。取而代之的是确定性的近似算法或蒙特卡罗近似方法。这两种方法在机器学习中都非常普遍。本章主要关注蒙特卡罗方法。

## 17.1 采样和蒙特卡罗方法

机器学习中的许多重要工具都基于从某种分布中采样以及用这些样本对目标量做一个蒙特卡罗估计。

### 17.1.1 为什么需要采样？

有许多原因使我们希望从某个分布中采样。当我们需要以较小的代价近似许多项的和或某个积分时，采样是一种很灵活的选择。有时候，我们使用它加速一些很费时却易于处理的求和估计，就像我们使用小批量对整个训练代价进行子采样一样。在其他情况下，我们需要近似一个难以处理的求和或积分，例如估计一个无向模型中配分函数对数的梯度时。在许多其他情况下，抽样实际上是我们的目标，例如我们想训练一个可以从训练分布采样的模型。

### 17.1.2 蒙特卡罗采样的基础

当无法精确计算和或积分（例如，和具有指数数量个项，且无法被精确简化）时，通常可以使用蒙特卡罗采样来近似它。这种想法把和或者积分视作某分布下的期望，然后通过估计对应的平均值来近似这个期望。令

$$s = \sum_{\mathbf{x}} p(\mathbf{x}) f(\mathbf{x}) = E_p[f(\mathbf{x})] \quad (17.1)$$

或者

$$s = \int p(\mathbf{x}) f(\mathbf{x}) d\mathbf{x} = E_p[f(\mathbf{x})] \quad (17.2)$$

为我们所需要估计的和或者积分，写成期望的形式， $p$  是一个关于随机变量  $\mathbf{x}$  的概率分布（求和时）或者概率密度函数（求积分时）。

我们可以通过从  $p$  中抽取  $n$  个样本  $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}$  来近似  $s$  并得到一个经验平均值

$$\hat{s}_n = \frac{1}{n} \sum_{i=1}^n f(\mathbf{x}^{(i)}). \quad (17.3)$$

下面几个性质表明了这种近似的合理性。首先很容易观察到  $\hat{s}$  这个估计是无偏的，由于

$$\mathbb{E}[\hat{s}_n] = \frac{1}{n} \sum_{i=1}^n \mathbb{E}[f(\mathbf{x}^{(i)})] = \frac{1}{n} \sum_{i=1}^n s = s. \quad (17.4)$$

此外，根据 **大数定理** (Law of large number)，如果样本  $\mathbf{x}^{(i)}$  是独立同分布的，那么其平均值几乎必然收敛到期望值，即

$$\lim_{n \rightarrow \infty} \hat{s}_n = s, \quad (17.5)$$

只需要满足各个单项的方差  $\text{Var}[f(\mathbf{x}^{(i)})]$  有界。详细地说，我们考虑当  $n$  增大时  $\hat{s}_n$  的方差。只要满足  $\text{Var}[f(\mathbf{x}^{(i)})] < \infty$ ，方差  $\text{Var}[\hat{s}_n]$  就会减小并收敛到 0：

$$\text{Var}[\hat{s}_n] = \frac{1}{n^2} \sum_{i=1}^n \text{Var}[f(\mathbf{x})] \quad (17.6)$$

$$= \frac{\text{Var}[f(\mathbf{x})]}{n}. \quad (17.7)$$

这个简单有用的结果启迪我们如何估计蒙特卡罗均值中的不确定性，或者等价地说是蒙特卡罗估计的期望误差。我们计算了  $f(\mathbf{x}^{(i)})$  的经验均值和方差<sup>1</sup>，然后将估计的方差除以样本数  $n$  来得到  $\text{Var}[\hat{s}_n]$  的估计。中心极限定理 (central limit theorem) 告诉我们  $\hat{s}_n$  的分布收敛到以  $s$  为均值以  $\frac{\text{Var}[f(\mathbf{x})]}{n}$  为方差的正态分布。这使得我们可以利用正态分布的累积函数来估计  $\hat{s}_n$  的置信区间。

以上的所有结论都依赖于我们可以从基准分布  $p(\mathbf{x})$  中轻易地采样，但是这个假设并不是一直成立的。当我们无法从  $p$  中采样时，一个备选方案是用第 17.2 节讲到的重要采样。一种更加通用的方式是构建一个收敛到目标分布的估计序列。这就是马尔可夫链蒙特卡罗方法（见第 17.3 节）。

## 17.2 重要采样

如方程 (17.2) 所示，在蒙特卡罗方法中，对积分（或者和）分解，确定积分中哪一部分作为概率分布  $p(\mathbf{x})$  以及哪一部分作为被积的函数  $f(\mathbf{x})$ （我们感兴趣的是估计  $f(\mathbf{x})$  在概率分布  $p(\mathbf{x})$  下的期望）是很关键的一步。 $p(\mathbf{x})f(\mathbf{x})$  不存在唯一的分解，因为它总是可以被写成

$$p(\mathbf{x})f(\mathbf{x}) = q(\mathbf{x}) \frac{p(\mathbf{x})f(\mathbf{x})}{q(\mathbf{x})}, \quad (17.8)$$

在这里，我们从  $q$  分布中采样，然后估计  $\frac{pf}{q}$  在此分布下的均值。许多情况下，我们希望在给定  $p$  和  $f$  的情况下计算某个期望，这个问题既然是求期望，那么很自然地  $p$  和  $f$  是一种分解选择。然而，如果考虑达到某给定精度所需要的样本数量，这个问题最初的分解选择不是最优的选择。幸运的是，最优的选择  $q^*$  可以被简单地推导出来。这种最优的采样函数  $q^*$  对应所谓的最优重要采样。

从式 (17.8) 所示的关系中可以发现，任意蒙特卡罗估计

$$\hat{s}_p = \frac{1}{n} \sum_{i=1, \mathbf{x}^{(i)} \sim p}^n f(\mathbf{x}^{(i)}) \quad (17.9)$$

可以被转化为一个重要采样的估计

$$\hat{s}_q = \frac{1}{n} \sum_{i=1, \mathbf{x}^{(i)} \sim q}^n \frac{p(\mathbf{x}^{(i)})f(\mathbf{x}^{(i)})}{q(\mathbf{x}^{(i)})}. \quad (17.10)$$

---

<sup>1</sup>通常我们会倾向于计算方差的无偏估计，它由偏差的平方和除以  $n - 1$  而非  $n$  得到。

我们可以容易地发现估计的期望与  $q$  分布无关:

$$\mathbb{E}_q[\hat{s}_q] = \mathbb{E}_p[\hat{s}_p] = s. \quad (17.11)$$

然而, 重要采样的方差可能对  $q$  的选择非常敏感。这个方差可以表示为

$$\text{Var}[\hat{s}_q] = \text{Var}\left[\frac{p(\mathbf{x})f(\mathbf{x})}{q(\mathbf{x})}\right]/n. \quad (17.12)$$

方差想要取到最小值,  $q$  需要满足

$$q^*(\mathbf{x}) = \frac{p(\mathbf{x})|f(\mathbf{x})|}{Z}, \quad (17.13)$$

在这里  $Z$  表示归一化常数, 选择适当的  $Z$  使得  $q^*(\mathbf{x})$  之和或者积分为 1。一个更好的重要采样分布会把更多的权重放在被积函数较大的地方。事实上, 当  $f(\mathbf{x})$  的正负符号不变时,  $\text{Var}[\hat{s}_{q^*}] = 0$ , 这意味着当使用最优的  $q$  分布时, 只需要一个样本就足够了。当然, 这仅仅是因为计算  $q^*$  时已经解决了原问题。所以在实践中这种只需要采样一个样本的方法往往是无法实现的。

对于重要采样来说任意  $q$  分布都是可行的 (从得到一个期望上正确的值的角度来说),  $q^*$  指的是最优的  $q$  分布 (从得到最小方差的角度上考虑)。从  $q^*$  中采样往往是不可行的, 但是其他仍然能降低方差的  $q$  的选择还是可行的。

另一种方法是采用**有偏重要采样** (biased importance sampling), 这种方法有一个优势, 即不需要归一化的  $p$  或  $q$  分布。在处理离散变量时, 有偏重要采样估计可以表示为

$$\hat{s}_{\text{BIS}} = \frac{\sum_{i=1}^n \frac{p(\mathbf{x}^{(i)})}{q(\mathbf{x}^{(i)})} f(\mathbf{x}^{(i)})}{\sum_{i=1}^n \frac{p(\mathbf{x}^{(i)})}{q(\mathbf{x}^{(i)})}} \quad (17.14)$$

$$= \frac{\sum_{i=1}^n \frac{p(\mathbf{x}^{(i)})}{\tilde{q}(\mathbf{x}^{(i)})} f(\mathbf{x}^{(i)})}{\sum_{i=1}^n \frac{p(\mathbf{x}^{(i)})}{\tilde{q}(\mathbf{x}^{(i)})}} \quad (17.15)$$

$$= \frac{\sum_{i=1}^n \frac{\tilde{p}(\mathbf{x}^{(i)})}{\tilde{q}(\mathbf{x}^{(i)})} f(\mathbf{x}^{(i)})}{\sum_{i=1}^n \frac{\tilde{p}(\mathbf{x}^{(i)})}{\tilde{q}(\mathbf{x}^{(i)})}}, \quad (17.16)$$

其中  $\tilde{p}$  和  $\tilde{q}$  分别是分布  $p$  和  $q$  的未经归一化的形式,  $\mathbf{x}^{(i)}$  是从分布  $q$  中抽取的样本。这种估计是有偏的, 因为  $\mathbb{E}[\hat{s}_{\text{BIS}}] \neq s$ , 只有当  $n \rightarrow \infty$  且方程式 (17.14) 的分母收敛到 1 时, 等式才渐近地成立。所以这一估计也被称为渐近无偏的。

一个好的  $q$  分布的选择可以显著地提高蒙特卡罗估计的效率，而一个糟糕的  $q$  分布选择则会使效率更糟糕。我们回过头来看看方程 式(17.12)会发现，如果存在一个  $q$  使得  $\frac{p(\mathbf{x})f(\mathbf{x})}{q(\mathbf{x})}$  很大，那么这个估计的方差也会很大。当  $q(\mathbf{x})$  很小，而  $f(\mathbf{x})$  和  $p(\mathbf{x})$  都较大并且无法抵消  $q$  时，这种情况会非常明显。 $q$  分布经常会取一些简单常用的分布使得我们能够从  $q$  分布中容易地采样。当  $\mathbf{x}$  是高维数据时， $q$  分布的简单性使得它很难与  $p$  或者  $p|f|$  相匹配。当  $q(\mathbf{x}^{(i)}) \gg p(\mathbf{x}^{(i)})|f(\mathbf{x}^{(i)})|$  时，重要采样采到了很多无用的样本（很小的数或零相加）。另一种相对少见的情况是  $q(\mathbf{x}^{(i)}) \ll p(\mathbf{x}^{(i)})|f(\mathbf{x}^{(i)})|$ ，相应的比值会非常大。正因为后一个事件是很少发生的，这种样本很难被采到，通常使得对  $s$  的估计出现了典型的欠估计，很难被整体的过估计抵消。这样的不均匀情况在高维数据屡见不鲜，因为在高维度分布中联合分布的动态域可能非常大。

尽管存在上述的风险，但是重要采样及其变种在机器学习的应用中仍然扮演着重要的角色，包括深度学习算法。例如，重要采样被应用于加速训练具有大规模词表的神经网络语言模型的过程中（见第 12.4.3.3 节）或者其他有着大量输出结点的神经网络中。此外，还可以看到重要采样应用于估计配分函数（一个概率分布的归一化常数），详见第 18.7 节，以及在深度有向图模型比如变分自编码器中估计对数似然（详见第 20.10.3 节）。采用随机梯度下降训练模型参数时重要采样可以用来改进对代价函数梯度的估计，尤其是分类器这样的模型，其中代价函数的大部分代价来自于少量错误分类的样本。在这种情况下，更加频繁地抽取这些困难的样本可以减小梯度估计的方差 (Hinton *et al.*, 2006a)。

### 17.3 马尔可夫链蒙特卡罗方法

在许多实例中，我们希望采用蒙特卡罗方法，然而往往又不存在一种简单的方法可以直接从目标分布  $p_{\text{model}}(\mathbf{x})$  中精确采样或者一个好的（方差较小的）重要采样分布  $q(\mathbf{x})$ 。在深度学习中，当分布  $p_{\text{model}}(\mathbf{x})$  表示成无向模型时，这种情况往往会发生。在这种情况下，为了从分布  $p_{\text{model}}(\mathbf{x})$  中近似采样，我们引入了一种称为 **马尔可夫链** (Markov Chain) 的数学工具。利用马尔可夫链来进行蒙特卡罗估计的这一类算法被称为 **马尔可夫链蒙特卡罗** (Markov Chain Monte Carlo, MCMC) 方法。Koller and Friedman (2009) 花了大量篇幅来描述马尔可夫链蒙特卡罗算法在机器学习中的应用。MCMC 技术最标准、最一般的理论保证只适用于那些各状态概率均不为零的模型。因此，这些技术最方便的使用方法是用于从**基于能量的模型** (Energy-based model) 即  $p(\mathbf{x}) \propto \exp(-E(\mathbf{x}))$  中采样，见第 16.2.4 节。在 EBM 的公式表述中，每

一个状态所对应的概率都不为零。事实上，MCMC 方法可以被广泛地应用在包含 0 概率状态的许多概率分布中。然而，在这种情况下，关于 MCMC 方法性能的理论保证只能依据具体不同类型的分布具体分析证明。在深度学习中，我们通常依赖于那些一般的理论保证，其在所有基于能量的模型都能自然成立。

为了解释从基于能量的模型中采样困难的原因，我们考虑一个包含两个变量的 EBM 的例子，记  $p(a, b)$  为其分布。为了采  $a$ ，我们必须先从  $p(a | b)$  中采样；为了采  $b$ ，我们又必须从  $p(b | a)$  中采样。这似乎成了棘手的先有鸡还是先有蛋的问题。有向模型避免了这一问题因为它的图是有向无环的。为了完成 **原始采样** (Ancestral Sampling)，在给定每个变量的所有父结点的条件下，我们根据拓扑顺序采样每一个变量，这个变量是确定能够被采样的（详见第 16.3 节）。原始采样定义了一种高效的、单遍的方法来抽取一个样本。

在 EBM 中，我们通过使用马尔可夫链来采样，从而避免了先有鸡还是先有蛋的问题。马尔可夫链的核心思想是从某个可取任意值的状态  $\mathbf{x}$  出发。随着时间的推移，我们随机地反复更新状态  $\mathbf{x}$ 。最终  $\mathbf{x}$  成为了一个从  $p(\mathbf{x})$  中抽出的（非常接近）比较一般的样本。在正式的定义中，马尔可夫链由一个随机状态  $x$  和一个转移分布  $T(\mathbf{x}' | \mathbf{x})$  定义而成， $T(\mathbf{x}' | \mathbf{x})$  是一个概率分布，说明了给定状态  $\mathbf{x}$  的情况下随机地转移到  $\mathbf{x}'$  的概率。运行一个马尔可夫链意味着根据转移分布  $T(\mathbf{x}' | \mathbf{x})$  采出的值  $\mathbf{x}'$  来更新状态  $\mathbf{x}$ 。

为了给出 MCMC 方法为何有效的一些理论解释，重参数化这个问题是很有用的。首先我们关注一些简单的情况，其中随机变量  $\mathbf{x}$  有可数个状态。我们将这种状态简单地记作正整数  $x$ 。不同的整数  $x$  的大小对应着原始问题中  $\mathbf{x}$  的不同状态。

接下来我们考虑如果并行地运行无穷多个马尔可夫链的情况。不同马尔可夫链的所有状态都采样自某一个分布  $q^{(t)}(x)$ ，在这里  $t$  表示消耗的时间数。开始时，对每个马尔可夫链，我们采用一个分布  $q^0$  来任意地初始化  $x$ 。之后， $q^{(t)}$  与所有之前运行的马尔可夫链有关。我们的目标是  $q^{(t)}(x)$  收敛到  $p(x)$ 。

因为我们已经用正整数  $x$  重参数化了这个问题，我们可以用一个向量  $\mathbf{v}$  来描述这个概率分布  $q$ ，

$$q(\mathbf{x} = i) = v_i. \quad (17.17)$$

然后我们考虑更新单一的马尔可夫链，从状态  $x$  到新状态  $x'$ 。单一状态转移到

$x'$  的概率可以表示为

$$q^{(t+1)}(x') = \sum_x q^{(t)}(x)T(x' | x). \quad (17.18)$$

根据状态为整数的参数化设定，我们可以将转移算子  $T$  表示成一个矩阵  $\mathbf{A}$ 。矩阵  $\mathbf{A}$  的定义如下：

$$\mathbf{A}_{i,j} = T(\mathbf{x}' = i | \mathbf{x} = j). \quad (17.19)$$

使用这一定义，我们可以改写式 (17.18)。不同于之前使用  $q$  和  $T$  来理解单个状态的更新，我们现在可以使用  $\mathbf{v}$  和  $\mathbf{A}$  来描述当我们更新时（并行运行的）不同马尔可夫链上整个分布是如何变化的：

$$\mathbf{v}^{(t)} = \mathbf{A}\mathbf{v}^{(t-1)}. \quad (17.20)$$

重复地使用马尔可夫链更新相当于重复地与矩阵  $\mathbf{A}$  相乘。换言之，我们可以认为这一过程就是关于  $\mathbf{A}$  的幂乘：

$$\mathbf{v}^{(t)} = \mathbf{A}^t \mathbf{v}^{(0)}. \quad (17.21)$$

矩阵  $\mathbf{A}$  有一种特殊的结构，因为它的每一列都代表一个概率分布。这样的矩阵被称为 **随机矩阵** (Stochastic Matrix)。如果对于任意状态  $x$  到任意其他状态  $x'$  存在一个  $t$  使得转移概率不为 0，那么 Perron-Frobenius 定理 (Perron, 1907; Frobenius, 1908) 可以保证这个矩阵的最大特征值是实数且大小为 1。我们可以看到所有的特征值随着时间呈现指数变化：

$$\mathbf{v}^{(t)} = (\mathbf{V}\text{diag}(\boldsymbol{\lambda})\mathbf{V}^{-1})^t \mathbf{v}^{(0)} = \mathbf{V}\text{diag}(\boldsymbol{\lambda})^t \mathbf{V}^{-1} \mathbf{v}^{(0)}. \quad (17.22)$$

这个过程导致了所有不等于 1 的特征值都衰减到 0。在一些额外的较为宽松的假设下，我们可以保证矩阵  $\mathbf{A}$  只有一个对应特征值为 1 的特征向量。所以这个过程收敛到 **平稳分布** (Stationary Distribution)，有时也被称为 **均衡分布** (Equilibrium Distribution)。收敛时，我们得到

$$\mathbf{v}' = \mathbf{A}\mathbf{v} = \mathbf{v}, \quad (17.23)$$

这个条件也适用于收敛之后的每一步。这就是特征向量方程。作为收敛的稳定点， $\mathbf{v}$  一定是特征值为 1 所对应的特征向量。这个条件保证收敛到了平稳分布以后，再重

复转移采样过程不会改变所有不同马尔可夫链上状态的分布（尽管转移算子自然而然地会改变每个单独的状态）。

如果我们正确地选择了转移算子  $T$ ，那么最终的平稳分布  $q$  将会等于我们所希望采样的分布  $p$ 。我们会将第 17.4 节介绍如何选择  $T$ 。

可数状态马尔可夫链的大多数性质可以被推广到连续状态的马尔可夫链中。在这种情况下，一些研究者把这种马尔可夫链称为 **哈里斯链** (Harris Chain)，但是我们将这两种情况都称为马尔可夫链。通常在一些宽松的条件下，一个带有转移算子  $T$  的马尔可夫链都会收敛到一个不动点，这个不动点可以写成如下形式：

$$q'(\mathbf{x}') = \mathbb{E}_{\mathbf{x} \sim q} T(\mathbf{x}' | \mathbf{x}), \quad (17.24)$$

这个方程的离散版本就相当于重新改写方程 式 (17.23)。当  $\mathbf{x}$  是离散值时，这个期望对应着求和，而当  $\mathbf{x}$  是连续值时，这个期望对应的是积分。

无论状态是连续的还是离散的，所有的马尔可夫链方法都包括了重复、随机地更新直到最后状态开始从均衡分布中采样。运行马尔可夫链直到它达到均衡分布的过程通常被称为马尔可夫链的 **磨合** (Burning-in) 过程。在马尔可夫链达到均衡分布之后，我们可以从均衡分布中抽取一个无限多数量的样本序列。这些样本服从同一分布，但是两个连续的样本之间会高度相关。所以一个有限的序列无法完全表达均衡分布。一种解决这个问题的方法是每隔  $n$  个样本返回一个样本，从而使得我们对于均衡分布的统计量的估计不会被 MCMC 方法的样本之间的相关性所干扰。所以马尔可夫链的计算代价很高，主要源于达到均衡分布前需要磨合的时间以及在达到均衡分布之后从一个样本转移到另一个足够无关的样本所需要的时间。如果我们想要得到完全独立的样本，那么我们可以同时并行地运行多个马尔可夫链。这种方法使用了额外的并行计算来减少时延。使用一条马尔可夫链来生成所有样本的策略和（使用多条马尔可夫链）每条马尔可夫链只产生一个样本的策略是两种极端。深度学习的从业者们通常选取的马尔可夫链的数目和小批量中的样本数相近，然后从这些固定的马尔可夫链集合中抽取所需要的样本。马尔可夫链的数目通常选为 100。

另一个难点是我们无法预先知道马尔可夫链需要运行多少步才能到达均衡分布。这段时间通常被称为 **混合时间** (Mixing Time)。检测一个马尔可夫链是否达到平衡是很困难的。我们并没有足够完善的理论来解决这个问题。理论只能保证马尔可夫链会最终收敛，但是无法保证其他。如果我们从矩阵  $A$  作用在概率向量  $v$  上的角度来分析马尔可夫链，那么我们可以发现当  $A^t$  除了单个 1 以外的特征值都趋于 0 时，马尔可夫链混合成功（收敛到了均衡分布）。这也意味着矩阵  $A$  的第二大特征值决

定了马尔可夫链的混合时间。然而，在实践中，我们通常不能真的将马尔可夫链表示成矩阵的形式。我们的概率模型所能够达到的状态是变量数的指数级别，所以表达  $v$ ,  $A$  或者  $\mathbf{A}$  的特征值是不现实的。由于以上在内的诸多阻碍，我们通常无法知道马尔可夫链是否已经混合成功。作为替代，我们只能运行一定量时间马尔可夫链直到我们粗略估计这段时间是足够的，然后使用启发式的方法来判断马尔可夫链是否混合成功。这些启发性的算法包括了手动检查样本或者衡量前后样本之间的相关性。

## 17.4 Gibbs 采样

目前为止我们已经了解了如何通过反复更新  $x \leftarrow x' \sim T(x' | x)$  从一个分布  $q(x)$  中采样。然而我们还没有介绍过如何确定  $q(x)$  是否是一个有效的分布。本书中将会描述两种基本的方法。第一种方法是从已经学习到的分布  $p_{\text{model}}$  中推导出  $T$ ，下文描述了如何从基于能量的模型中采样。第二种方法是直接用参数描述  $T$ ，然后学习这些参数，其平稳分布隐式地定义了我们所感兴趣的模型  $p_{\text{model}}$ 。我们将在第 20.12 节和第 20.13 节中讨论第二种方法的例子。

在深度学习中，我们通常使用马尔可夫链从定义为基于能量的模型的分布  $p_{\text{model}}(x)$  中采样。在这种情况下，我们希望马尔可夫链的  $q(x)$  分布就是  $p_{\text{model}}(x)$ 。为了得到所期望的  $q(x)$  分布，我们必须选取合适的  $T(x' | x)$ 。

**Gibbs 采样** (Gibbs Sampling) 是一种概念简单而又有效的方法。它构造一个从  $p_{\text{model}}(x)$  中采样的马尔可夫链，其中在基于能量的模型中从  $T(x' | x)$  采样是通过选择一个变量  $x_i$ ，然后从  $p_{\text{model}}$  中该点关于在无向图  $\mathcal{G}$  (定义了基于能量的模型结构) 中邻接点的条件分布中采样。只要一些变量在给定相邻变量时是条件独立的，那么这些变量就可以被同时采样。正如在第 16.7.1 节中看到的 RBM 示例一样，RBM 中所有的隐藏单元可以被同时采样，因为在给定所有可见单元的条件下它们相互条件独立。同样地，所有的可见单元也可以被同时采样，因为在给定所有隐藏单元的情况下它们相互条件独立。以这种方式同时更新许多变量的 Gibbs 采样通常被称为 **块吉布斯采样** (block Gibbs Sampling)。

设计从  $p_{\text{model}}$  中采样的马尔可夫链还存在其他备选方法。比如说，Metropolis-Hastings 算法在其他领域中广泛使用。不过在深度学习的无向模型中，我们主要使用 Gibbs 采样，很少使用其他方法。改进采样技巧也是一个潜在的研究热点。

## 17.5 不同的峰值之间的混合挑战

使用 MCMC 方法的主要难点在于马尔可夫链的 混合（Mixing）通常不理想。在理想情况下，从设计好的马尔可夫链中采出的连续样本之间是完全独立的，而且在  $\mathbf{x}$  空间中，马尔可夫链会按概率大小访问许多不同区域。

然而，MCMC 方法采出的样本可能会具有很强的相关性，尤其是在高维的情况下。我们把这种现象称为慢混合甚至混合失败。具有缓慢混合的 MCMC 方法可以被视为对能量函数无意地执行类似于带噪声的梯度下降的操作，或者说等价于相对于链的状态（被采样的随机变量）依据概率进行噪声爬坡。（在马尔可夫链的状态空间中）从  $\mathbf{x}^{(t-1)}$  到  $\mathbf{x}^{(t)}$  该链倾向于选取很小的步长，其中能量  $E(\mathbf{x}^{(t)})$  通常低于或者近似等于能量  $E(\mathbf{x}^{(t-1)})$ ，倾向于向较低能量的区域移动。当从可能性较小的状态（比来自  $p(\mathbf{x})$  的典型样本拥有更高的能量）开始时，链趋向于逐渐减少状态的能量，并且仅仅偶尔移动到另一个峰值。一旦该链已经找到低能量的区域（例如，如果变量是图像中的像素，则低能量的区域可以是同一对象所对应图像的一个连通的流形），我们称之为峰值，链将倾向于围绕着这个峰值游走（按某一种形式随机游走）。它时不时会走出该峰值，但是结果通常会返回该峰值或者（如果找到一条离开的路线）移向另一个峰值。问题是对于很多有趣的分布来说成功的离开路线很少，所以马尔可夫链将在一个峰值附近抽取远超过需求的样本。

当我们考虑 Gibbs 采样算法（见第 17.4 节）时，这种现象格外明显。在这种情况下，我们考虑在一定步数内从一个峰值移动到一个临近峰值的概率。决定这个概率的是两个峰值之间的“能量障碍”的形状。隔着一个巨大“能量障碍”（低概率的区域）的两个峰值之间的转移概率是（随着能量障碍的高度）指数下降的，如图 17.1 所示。当目标分布有多个高概率峰值并且被低概率区域所分割，尤其当 Gibbs 采样的每一步都只是更新变量的一小部分而这一小部分变量又严重依赖其他的变量时，就会产生问题。

举一个简单的例子，考虑两个变量  $a, b$  的基于能量的模型，这两个变量都是二值的，取值  $+1$  或者  $-1$ 。如果对某个较大的正数  $w$ ,  $E(a, b) = -wab$ , 那么这个模型传达了一个强烈的信息， $a$  和  $b$  有相同的符号。当  $a = 1$  时用 Gibbs 采样更新  $b$ 。给定  $b$  时的条件分布满足  $p(b = 1 | a = 1) = \sigma(w)$ 。如果  $w$  的值很大，sigmoid 函数趋近于饱和，那么  $b$  也取到 1 的概率趋近于 1。同理，如果  $a = -1$ ，那么  $b$  取到  $-1$  的概率也趋于 1。根据模型  $p_{\text{model}}(a, b)$ ，两个变量取一样的符号的概率几乎相等。根据  $p_{\text{model}}(a | b)$ ，两个变量应该有相同的符号。这也意味着 Gibbs 采样很难会

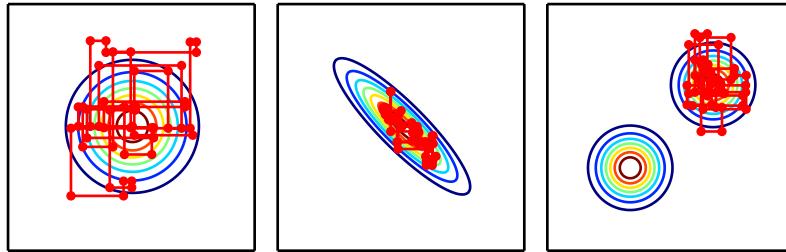


图 17.1: 对于三种分布使用 Gibbs 采样所产生的路径,所有的分布马尔可夫链初始值都设为峰值。(左)一个带有两个独立变量的多维正态分布。由于变量之间是相互独立的, Gibbs 采样混合得很好。(中) 变量之间存在高度相关性的一个多维正态分布。变量之间的相关性使得马尔可夫链很难混合。因为每一个变量的更新需要相对其他变量求条件分布, 相关性减慢了马尔可夫链远离初始点的速度。(右) 峰值之间间距很大且不在轴上对齐的混合高斯分布。Gibbs 采样混合得很慢, 因为每次更新仅仅一个变量很难跨越不同的峰值。

改变这些变量的符号。

在更实际的问题中,这种挑战更加艰巨因为在实际问题中我们不能仅仅关注在两个峰值之间的转移,更要关注在多个峰值之间的转移。如果由于峰值之间混合困难,而导致某几个这样的转移难以完成,那么得到一些可靠的覆盖大部分峰值的样本集合的计算代价是很高的,同时马尔可夫链收敛到它的平稳分布的过程也会非常缓慢。

通过寻找一些高度依赖变量的组以及分块同时更新块(组)中的变量,这个问题有时候是可以被解决的。然而不幸的是,当依赖关系很复杂时,从这些组中采样的过程从计算角度上说是难以处理的。归根结底,马尔可夫链最初就是被提出来解决这个问题,即从大量变量中采样的问题。

在定义了一个联合分布  $p_{\text{model}}(\mathbf{x}, \mathbf{h})$  的潜变量模型中,我们经常通过交替地从  $p_{\text{model}}(\mathbf{x} | \mathbf{h})$  和  $p_{\text{model}}(\mathbf{h} | \mathbf{x})$  中采样来达到抽  $\mathbf{x}$  的目的。从快速混合的角度上说,我们更希望  $p_{\text{model}}(\mathbf{h} | \mathbf{x})$  有很大的熵。然而,从学习一个  $\mathbf{h}$  的有用表示的角度上考虑,我们还是希望  $\mathbf{h}$  能够包含  $\mathbf{x}$  的足够信息从而能够较完整地重构它,这意味着  $\mathbf{h}$  和  $\mathbf{x}$  要有非常高的互信息。这两个目标是相互矛盾的。我们经常学习到能够将  $\mathbf{x}$  精确地

编码为  $h$  的生成模型，但是无法很好混合。这种情况在玻尔兹曼机中经常出现，一个玻尔兹曼机学到的分布越尖锐，该分布的马尔可夫链采样越难混合得好。这个问题在图 17.2 中有所描述。

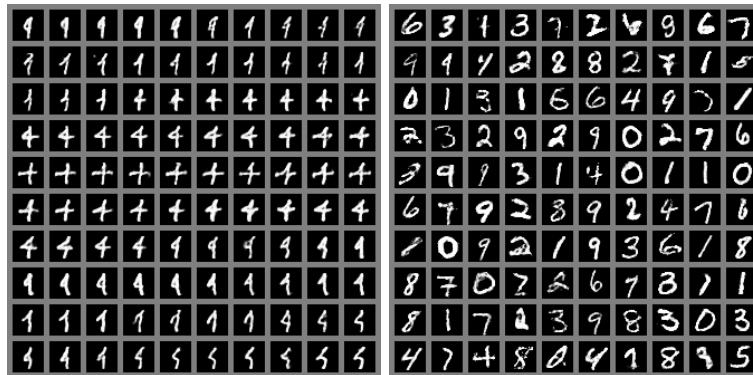


图 17.2: 深度概率模型中一个混合缓慢问题的例证。每张图都是按照从左到右从上到下的顺序的。  
(左) Gibbs 采样从 MNIST 数据集训练成的深度玻尔兹曼机中采出的连续样本。这些连续的样本之间非常相似。由于 Gibbs 采样作用于一个深度图模型，相似度更多地是基于语义而非原始视觉特征。但是对于吉布斯链来说从分布的一个峰值转移到另一个仍然是很困难的，比如说改变数字。  
(右) 从生成式对抗网络中抽出的连续原始样本。因为原始采样生成的样本之间互相独立，所以不存在混合问题。

当感兴趣的分布对于每个类具有单独的流形结构时，所有这些问题都使 MCMC 方法变得不那么有用：分布集中在许多峰值周围，并且这些峰值由大量高能量区域分割。我们在许多分类问题中遇到的是这种类型的分布，由于峰值之间混合缓慢，它将使得 MCMC 方法非常缓慢地收敛。

### 17.5.1 不同峰值之间通过回火来混合

当一个分布有一些陡峭的峰并且被低概率区域包围时，很难在分布的不同峰值之间混合。一些加速混合的方法是基于构造一个概率分布替代目标分布，这个概率分布的峰值没有那么高，峰值周围的低谷也没有那么低。基于能量的模型为这个想法提供一种简单的做法。目前为止，我们一直将基于能量的模型描述为定义一个概率分布：

$$p(\mathbf{x}) \propto \exp(-E(\mathbf{x})). \quad (17.25)$$

基于能量的模型可以通过添加一个额外的控制峰值尖锐程度的参数  $\beta$  来加强：

$$p_\beta(\mathbf{x}) \propto \exp(-\beta E(\mathbf{x})). \quad (17.26)$$

$\beta$  参数可以被理解为 温度 (temperature) 的倒数，反映了基于能量的模型的统计物理学起源。当温度趋近于 0 时， $\beta$  趋近于无穷大，此时的基于能量的模型是确定性的。当温度趋近于无穷大时， $\beta$  趋近于零，基于能量的模型（对离散的  $\mathbf{x}$ ）成了均匀分布。

通常情况下，在  $\beta = 1$  时训练一个模型。但我们也利用其他温度，尤其是  $\beta < 1$  的情况。回火 (tempering) 作为一种通用的策略，它通过从  $\beta < 1$  模型中采样来实现在  $p_1$  的不同峰值之间快速混合。

基于回火转移 (tempered transition) (Neal, 1994) 的马尔可夫链临时从高温度的分布中采样使其在不同峰值之间混合，然后继续从单位温度的分布中采样。这些技巧被应用在一些模型比如 RBM 中 (Salakhutdinov, 2010)。另一种方法是利用并行回火 (parallel tempering) (Iba, 2001)。其中马尔可夫链并行地模拟许多不同温度的不同状态。最高温度的状态混合较慢，相比之下最低温度的状态，即温度为 1 时，采出了精确的样本。转移算子包括了两个温度之间的随机跳转，所以一个高温度状态分布槽中的样本有足够的概率跳转到低温度分布的槽中。这个方法也被应用到了 RBM 中 (Desjardins *et al.*, 2010; Cho *et al.*, 2010a)。尽管回火这种方法前景可期，现今它仍然无法让我们在采样复杂的基于能量的模型中更进一步。一个可能的原因是在 临界温度 (critical temperatures) 时温度转移算子必须设置得非常慢（因为温度需要逐渐下降）来确保回火的有效性。

### 17.5.2 深度也许会有助于混合

当我们从潜变量模型  $p(\mathbf{h}, \mathbf{x})$  中采样时，我们可以发现如果  $p(\mathbf{h} | \mathbf{x})$  将  $\mathbf{x}$  编码得非常好，那么从  $p(\mathbf{x} | \mathbf{h})$  中采样时，并不会太大地改变  $\mathbf{x}$ ，那么混合结果会很糟糕。解决这个问题的一种方法是使得  $\mathbf{h}$  成为一种将  $\mathbf{x}$  编码为  $\mathbf{h}$  的深度表示，从而使得马尔可夫链在  $\mathbf{h}$  空间中更容易混合。在许多表示学习算法如自编码器和 RBM 中， $\mathbf{h}$  的边缘分布相比于  $\mathbf{x}$  上的原始数据分布，通常表现为更加均匀、更趋近于单峰值。或许可以说，这是因为利用了所有可用的表示空间并尽量减小重构误差。因为当训练集上的不同样本之间在  $\mathbf{h}$  空间能够被非常容易地区分时，我们也会很容易地最小化重构误差。Bengio *et al.* (2013a) 观察到这样的现象，堆叠越深的正则化自编码

器或者 RBM，顶端  $\mathbf{h}$  空间的边缘分布越趋向于均匀和发散，而且不同峰值（比如说实验中的类别）所对应区域之间的间距也会越小。在高层空间中训练 RBM 会使得 Gibbs 采样在峰值间混合得更快。然而，如何利用这种观察到的现象来辅助训练深度生成模型或者从中采样仍然有待探索。

尽管存在混合的难点，蒙特卡罗技术仍然是一个有用的工具，通常也是最好的可用工具。事实上，在遇到难以处理的无向模型中的配分函数时，蒙特卡罗方法仍然是最主要的工具，这将在下一章详细阐述。

# 第十八章 直面配分函数

在第 16.2.2 节中，我们看到许多概率模型（通常是无向图模型）由一个未归一化的概率分布  $\tilde{p}(\mathbf{x}; \boldsymbol{\theta})$  定义。我们必须通过除以配分函数  $Z(\boldsymbol{\theta})$  来归一化  $\tilde{p}$ ，以获得一个有效的概率分布：

$$p(\mathbf{x}; \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \tilde{p}(\mathbf{x}; \boldsymbol{\theta}). \quad (18.1)$$

配分函数是未归一化概率所有状态的积分（对于连续变量）或求和（对于离散变量）：

$$\int \tilde{p}(\mathbf{x}) d\mathbf{x} \quad (18.2)$$

或者

$$\sum_{\mathbf{x}} \tilde{p}(\mathbf{x}). \quad (18.3)$$

对于很多有趣的模型而言，以上积分或求和难以计算。

正如我们将在第二十章看到的，有些深度学习模型被设计成具有一个易于处理的归一化常数，或被设计成能够在不涉及计算  $p(\mathbf{x})$  的情况下使用。然而，其他一些模型会直接面对难以计算的配分函数的挑战。在本章中，我们会介绍用于训练和评估那些具有难以处理的配分函数的模型的技术。

## 18.1 对数似然梯度

通过最大似然学习无向模型特别困难的原因在于配分函数依赖于参数。对数似然相对于参数的梯度具有一项对应于配分函数的梯度：

$$\nabla_{\boldsymbol{\theta}} \log p(\mathbf{x}; \boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\mathbf{x}; \boldsymbol{\theta}) - \nabla_{\boldsymbol{\theta}} \log Z(\boldsymbol{\theta}). \quad (18.4)$$

这是机器学习中非常著名的 **正相** (positive phase) 和 **负相** (negative phase) 的分解。

对于大多数感兴趣的无向模型而言，负相是困难的。没有潜变量或潜变量之间很少相互作用的模型通常会有一个易于计算的正相。RBM 的隐藏单元在给定可见单元的情况下彼此条件独立，是一个典型的具有简单正相和困难负相的模型。正相计算困难，潜变量之间具有复杂相互作用的情况将主要在第十九章中讨论。本章主要探讨负相计算中的难点。

让我们进一步分析  $\log Z$  的梯度：

$$\nabla_{\theta} \log Z \quad (18.5)$$

$$= \frac{\nabla_{\theta} Z}{Z} \quad (18.6)$$

$$= \frac{\nabla_{\theta} \sum_{\mathbf{x}} \tilde{p}(\mathbf{x})}{Z} \quad (18.7)$$

$$= \frac{\sum_{\mathbf{x}} \nabla_{\theta} \tilde{p}(\mathbf{x})}{Z}. \quad (18.8)$$

对于保证所有的  $\mathbf{x}$  都有  $p(\mathbf{x}) > 0$  的模型，我们可以用  $\exp(\log \tilde{p}(\mathbf{x}))$  代替  $\tilde{p}(\mathbf{x})$ ：

$$\frac{\sum_{\mathbf{x}} \nabla_{\theta} \exp(\log \tilde{p}(\mathbf{x}))}{Z} \quad (18.9)$$

$$= \frac{\sum_{\mathbf{x}} \exp(\log \tilde{p}(\mathbf{x})) \nabla_{\theta} \log \tilde{p}(\mathbf{x})}{Z} \quad (18.10)$$

$$= \frac{\sum_{\mathbf{x}} \tilde{p}(\mathbf{x}) \nabla_{\theta} \log \tilde{p}(\mathbf{x})}{Z} \quad (18.11)$$

$$= \sum_{\mathbf{x}} p(\mathbf{x}) \nabla_{\theta} \log \tilde{p}(\mathbf{x}) \quad (18.12)$$

$$= \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \nabla_{\theta} \log \tilde{p}(\mathbf{x}). \quad (18.13)$$

上述推导对离散的  $\mathbf{x}$  进行求和，对连续的  $\mathbf{x}$  进行积分也可以得到类似结果。在连续版本的推导中，使用在积分符号内取微分的莱布尼兹法则可以得到等式

$$\nabla_{\theta} \int \tilde{p}(\mathbf{x}) d\mathbf{x} = \int \nabla_{\theta} \tilde{p}(\mathbf{x}) d\mathbf{x}. \quad (18.14)$$

该等式只适用于  $\tilde{p}$  和  $\nabla_{\theta} \tilde{p}(\mathbf{x})$  上的一些特定规范条件。在测度论术语中，这些条件是：(1) 对每一个  $\theta$  而言，未归一化分布  $\tilde{p}$  必须是  $\mathbf{x}$  的勒贝格可积函数。(2) 对于所

有的  $\theta$  和几乎所有  $x$ , 梯度  $\nabla_{\theta}\tilde{p}(x)$  必须存在。(3) 对于所有的  $\theta$  和几乎所有的  $x$ , 必须存在一个可积函数  $R(x)$  使得  $\max_i |\frac{\partial}{\partial \theta_i} \tilde{p}(x)| \leq R(x)$ 。幸运的是, 大多数感兴趣的机器学习模型都具有这些性质。

等式

$$\nabla_{\theta} \log Z = \mathbb{E}_{x \sim p(x)} \nabla_{\theta} \log \tilde{p}(x) \quad (18.15)$$

是使用各种蒙特卡罗方法近似最大化(具有难计算配分函数模型的)似然的基础。

蒙特卡罗方法为学习无向模型提供了直观的框架, 我们能够在其中考虑正相和负相。在正相中, 我们增大从数据中采样得到的  $\log \tilde{p}(x)$ 。在负相中, 我们通过降低从模型分布中采样的  $\log \tilde{p}(x)$  来降低配分函数。

在深度学习文献中, 经常会看到用能量函数(式(16.7))来参数化  $\log \tilde{p}$ 。在这种情况下, 正相可以解释为压低训练样本的能量, 负相可以解释为提高模型抽出的样本的能量, 如图 18.1 所示。

## 18.2 随机最大似然和对比散度

实现式(18.15)的一个朴素方法是, 每次需要计算梯度时, 磨合随机初始化的一组马尔可夫链。当使用随机梯度下降进行学习时, 这意味着马尔可夫链必须在每次梯度步骤中磨合。这种方法引导下的训练过程如算法 18.1 所示。内循环中磨合马尔可夫链的计算代价过高, 导致这个过程在实际中是不可行的, 但是这个过程是其他更加实际的近似算法的基础。

我们可以将最大化似然的 MCMC 方法视为在两种力之间平衡, 一种力拉高数据出现时的模型分布, 一种拉低模型采样出现时的模型分布。图 18.1 展示了这个过程。这两种力分别对应最大化  $\log \tilde{p}$  和最小化  $\log Z$ 。对于负相会有一些近似方法。这些近似都可以被理解为使负相更容易计算, 但是也可能将其推向错误的位置。

因为负相涉及到从模型分布中抽样, 所以我们可以认为它在找模型信任度很高的点。因为负相减少了这些点的概率, 它们一般被认为代表了模型不正确的信念。在文献中, 它们经常被称为“幻觉”或“幻想粒子”。事实上, 负相已经被作为人类和其他动物做梦的一种可能解释(Crick and Mitchison, 1983)。这个想法是说, 大脑维持着世界的概率模型, 并且在醒着经历真实事件时会遵循  $\log \tilde{p}$  的梯度, 在睡觉时会遵循  $\log \tilde{p}$  的负梯度最小化  $\log Z$ , 其经历的样本采样自当前的模型。这个视角解释了具有正相和负相的大多数算法, 但是它还没有被神经科学实验证明是正确的。在机

---

**算法 18.1** 一种朴素的 MCMC 算法，使用梯度上升最大化具有难以计算配分函数的对数似然。

---

设步长  $\epsilon$  为一个小正数。

设吉布斯步数  $k$  大到足以允许磨合。在小图像集上训练一个 RBM 大致设为 100。

**while** 不收敛 **do**

    从训练集中采包含  $m$  个样本  $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$  的小批量。

$$\mathbf{g} \leftarrow \frac{1}{m} \sum_{i=1}^m \nabla_{\theta} \log \tilde{p}(\mathbf{x}^{(i)}; \boldsymbol{\theta}).$$

    初始化  $m$  个样本  $\{\tilde{\mathbf{x}}^{(1)}, \dots, \tilde{\mathbf{x}}^{(m)}\}$  为随机值（例如，从均匀或正态分布中采，或大致与模型边缘分布匹配的分布）。

**for**  $i = 1$  to  $k$  **do**

**for**  $j = 1$  to  $m$  **do**

$$\tilde{\mathbf{x}}^{(j)} \leftarrow \text{gibbs\_update}(\tilde{\mathbf{x}}^{(j)}).$$

**end for**

**end for**

$$\mathbf{g} \leftarrow \mathbf{g} - \frac{1}{m} \sum_{i=1}^m \nabla_{\theta} \log \tilde{p}(\tilde{\mathbf{x}}^{(i)}; \boldsymbol{\theta}).$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \epsilon \mathbf{g}.$$

**end while**

---

器学习模型中，通常有必要同时使用正相和负相，而不是按不同时间阶段分为清醒和 REM 睡眠时期。正如我们将在第 19.5 节中看到的，一些其他机器学习算法出于其他原因从模型分布中采样，这些算法也能提供睡觉做梦的解释。

这样理解学习正相和负相的作用之后，我们设计了一个比算法 18.1 计算代价更低的替代算法。简单的 MCMC 算法的计算成本主要来自每一步的随机初始化磨合马尔可夫链。一个自然的解决方法是初始化马尔可夫链为一个非常接近模型分布的分布，从而大大减少磨合步骤。

**对比散度** (CD，或者是具有  $k$  个 Gibbs 步骤的 CD- $k$ ) 算法在每个步骤中初始化马尔可夫链为采样自数据分布中的样本 (Hinton, 2000, 2010)，如算法 18.2 所示。从数据分布中获取样本是计算代价最小的，因为它们已经在数据集中了。初始时，数据分布并不接近模型分布，因此负相不是非常准确。幸运的是，正相仍然可以准确地增加数据的模型概率。进行正相阶段一段时间之后，模型分布会更接近于数据分布，并且负相开始变得准确。

当然，CD 仍然是真实负相的一个近似。CD 未能定性地实现真实负相的主要原

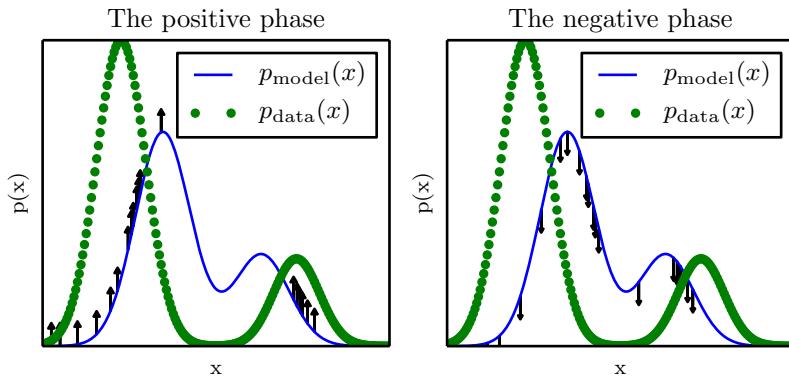


图 18.1: 算法 18.1 角度的“正相”和“负相”。(左) 在正相中, 我们从数据分布中采样, 然后推高它们未归一化的概率。这意味着概率越高的数据点未归一化的概率被推高得越多。(右) 在负相中, 我们从模型分布中采样, 然后压低它们未归一化的概率。这与正相的倾向相反, 给未归一化的概率处处添加了一个大常数。当数据分布和模型分布相等时, 正相推高数据点和负相压低数据点的机会相等。此时, 不再有任何的梯度 (期望上说), 训练也必须停止。

因是, 它不能抑制远离真实训练样本的高概率区域。这些区域在模型上具有高概率, 但是在数据生成区域上具有低概率, 被称为 **虚假模态** (spurious modes)。图 18.2 解释了这种现象发生的原因。基本上, 除非  $k$  非常大, 模型分布中远离数据分布的峰值不会被使用训练数据初始化的马尔可夫链访问到。

Carreira-Perpiñan and Hinton (2005) 实验上证明 CD 估计偏向于 RBM 和完全可见的玻尔兹曼机, 因为它会收敛到与最大似然估计不同的点。他们认为, 由于偏差较小, CD 可以作为一种计算代价低的方式来初始化模型, 之后可以通过计算代价高的 MCMC 方法进行精调。Bengio and Delalleau (2009) 表明, CD 可以被理解为去掉了正确 MCMC 梯度更新中的最小项, 这解释了偏差的由来。

在训练诸如 RBM 的浅层网络时 CD 是很有用的。反过来, 这些可以堆叠起来初始化更深的模型, 如 DBN 或 DBM。但是 CD 并不直接有助于训练更深的模型。这是因为在给定可见单元样本的情况下, 很难获得隐藏单元的样本。由于隐藏单元不包括在数据中, 所以使用训练点初始化无法解决这个问题。即使我们使用数据初始化可见单元, 我们仍然需要磨合在给定这些可见单元的隐藏单元条件分布上采样的马尔可夫链。

CD 算法可以被理解为惩罚某类模型, 这类模型的马尔可夫链会快速改变来自数

---

**算法 18.2 对比散度算法，使用梯度上升作为优化过程。**

---

设步长  $\epsilon$  为一个小正数。

设吉布斯步数  $k$  大到足以让从  $p_{\text{data}}$  初始化并从  $p(\mathbf{x}; \boldsymbol{\theta})$  采样的马尔可夫链混合。在小图像集上训练一个 RBM 大致设为 1-20。

**while** 不收敛 **do**

    从训练集中采包含  $m$  个样本  $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$  的小批量。

$$\mathbf{g} \leftarrow \frac{1}{m} \sum_{i=1}^m \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\mathbf{x}^{(i)}; \boldsymbol{\theta}).$$

**for**  $i = 1$  to  $m$  **do**

$$\tilde{\mathbf{x}}^{(i)} \leftarrow \mathbf{x}^{(i)}.$$

**end for**

**for**  $i = 1$  to  $k$  **do**

**for**  $j = 1$  to  $m$  **do**

$$\tilde{\mathbf{x}}^{(j)} \leftarrow \text{gibbs\_update}(\tilde{\mathbf{x}}^{(j)}).$$

**end for**

**end for**

$$\mathbf{g} \leftarrow \mathbf{g} - \frac{1}{m} \sum_{i=1}^m \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\tilde{\mathbf{x}}^{(i)}; \boldsymbol{\theta}).$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \epsilon \mathbf{g}.$$

**end while**

---

据的输入。这意味着使用 CD 训练从某种程度上说类似于训练自编码器。即使 CD 估计比一些其他训练方法具有更大偏差，但是它有助于预训练之后会堆叠起来的浅层模型。这是因为堆栈中最早的模型会受激励复制更多的信息到其潜变量，使其可用于随后的模型。这应该更多地被认为是 CD 训练中经常可利用的副产品，而不是主要的设计优势。

Sutskever and Tieleman (2010) 表明，CD 的更新方向不是任何函数的梯度。这使得 CD 可能存在永久循环的情况，但在实践中这并不是一个严重的问题。

另一个解决 CD 中许多问题的不同策略是，在每个梯度步骤中初始化马尔可夫链为先前梯度步骤的状态值。这个方法首先被应用数学和统计学社群发现，命名为随机最大似然 (SML) (Younes, 1998)，后来又在深度学习社群中以名称持续性对比散度 (PCD，或者每个更新中具有  $k$  个 Gibbs 步骤的 PCD-k) 独立地被重新发现 (Tieleman, 2008)。具体可以参考算法 18.3。这种方法的基本思想是，只要随机梯度算法得到的步长很小，那么前一步骤的模型将类似于当前步骤的模型。因此，来

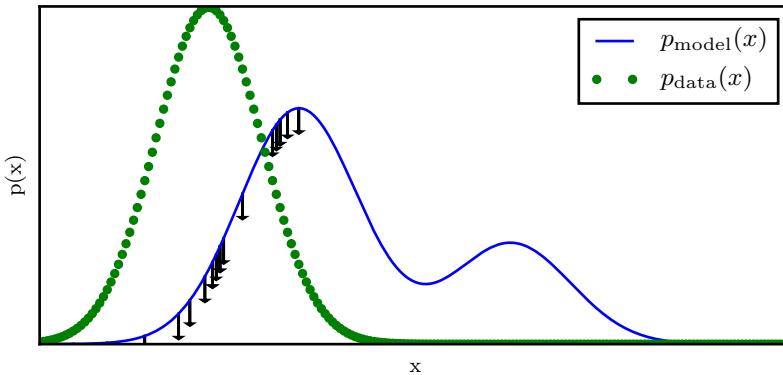


图 18.2: 一个虚假模态。说明对比散度（算法 18.2）的负相为何无法抑制虚假模态的例子。一个虚假模态指的是一个在模型分布中出现数据分布中却不存在的模式。由于对比散度从数据点中初始化它的马尔可夫链然后仅仅运行了几步马尔可夫链，不太可能到达模型中离数据点较远的模式。这意味着从模型中采样时，我们有时候会得到一些与数据并不相似的样本。这也意味着由于在这些模式上浪费了一些概率质量，模型很难把较高的概率质量集中于正确的模式上。出于可视化的目的，这个图使用了某种程度上说更加简单的距离的概念——在  $\mathbb{R}$  的数轴上虚假模态与正确的模式有很大的距离。这对应着基于局部移动  $\mathbb{R}$  上的单个变量  $x$  的马尔可夫链。对于大部分深度概率模型来说，马尔可夫链是基于 Gibbs 采样的，并且对于单个变量产生非局部的移动但是无法同时移动所有的变量。对于这些问题来说，考虑编辑距离比欧式距离通常更好。然而，高维空间的编辑距离很难在二维空间作图展示。

自先前模型分布的样本将非常接近来自当前模型分布的客观样本，用这些样本初始化的马尔可夫链将不需要花费很多时间来完成混合。

因为每个马尔可夫链在整个学习过程中不断更新，而不是在每个梯度步骤中重新开始，马尔可夫链可以自由探索很远，以找到模型的所有峰值。因此，SML 比 CD 更不容易形成具有虚假模态的模型。此外，因为可以存储所有采样变量的状态，无论是可见的还是潜在的，SML 为隐藏单元和可见单元都提供了初始值。CD 只能为可见单元提供初始化，因此深度模型需要进行磨合步骤。SML 能够高效地训练深度模型。Marlin *et al.* (2010) 将 SML 与本章中提出的许多其他标准方法进行比较。他们发现，SML 在 RBM 上得到了最佳的测试集对数似然，并且如果 RBM 的隐藏单元被用作 SVM 分类器的特征，那么 SML 会得到最好的分类精度。

在  $k$  太小或  $\epsilon$  太大时，随机梯度算法移动模型的速率比马尔可夫链在迭代步中混合更快，此时 SML 容易变得不准确。不幸的是，这些值的容许范围高度依赖

---

**算法 18.3** 随机最大似然/持续性对比散度算法，使用梯度上升作为优化过程。

---

设步长  $\epsilon$  为一个小正数。

设吉布斯步数  $k$  大到足以让从  $p(\mathbf{x}; \boldsymbol{\theta} + \epsilon\mathbf{g})$  采样的马尔可夫链磨合（从采自  $p(\mathbf{x}; \boldsymbol{\theta})$  的样本开始）。在小图像集上训练一个RBM大致设为 1，对于更复杂的模型如深度玻尔兹曼机可能要设为 5 到 50。

初始化  $m$  个样本  $\{\tilde{\mathbf{x}}^{(1)}, \dots, \tilde{\mathbf{x}}^{(m)}\}$  为随机值（例如，从均匀或正态分布中采，或大致与模型边缘分布匹配的分布）。

**while** 不收敛 **do**

    从训练集中采包含  $m$  个样本  $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$  的小批量。

$$\mathbf{g} \leftarrow \frac{1}{m} \sum_{i=1}^m \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\mathbf{x}^{(i)}; \boldsymbol{\theta}).$$

**for**  $i = 1$  to  $k$  **do**

**for**  $j = 1$  to  $m$  **do**

$$\tilde{\mathbf{x}}^{(j)} \leftarrow \text{gibbs\_update}(\tilde{\mathbf{x}}^{(j)}).$$

**end for**

**end for**

$$\mathbf{g} \leftarrow \mathbf{g} - \frac{1}{m} \sum_{i=1}^m \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\tilde{\mathbf{x}}^{(i)}; \boldsymbol{\theta}).$$

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \epsilon \mathbf{g}.$$

**end while**

---

于具体问题。现在还没有方法能够正式地测试马尔可夫链是否能够在迭代步骤之间成功混合。主观地，如果对于 Gibbs 步骤数目而言学习率太大的话，那么梯度步骤中负相采样的方差会比不同马尔可夫链中负相采样的方差更大。例如，一个 MNIST 模型在一个步骤中只采样得到了 7。然后学习过程将会极大降低 7 对应的峰值，在下一个步骤中，模型可能会只采样得到 9。

从使用 SML 训练的模型中评估采样必须非常小心。在模型训练完之后，有必要从一个随机起点初始化的新马尔可夫链抽取样本。用于训练的连续负相链中的样本受到了模型最近几个版本的影响，会使模型看起来具有比其实际更大的容量。

Berglund and Raiko (2013) 进行了实验来检验由 CD 和 SML 进行梯度估计带来的偏差和方差。结果证明 CD 比基于精确采样的估计具有更低的方差。而 SML 有更高的方差。CD 方差低的原因是，其在正相和负相中使用了相同的训练点。如果从不同的训练点来初始化负相，那么方差会比基于精确采样的估计的方差更大。

所有基于 MCMC 从模型中抽取样本的方法在原则上几乎可以与 MCMC 的任何变体一起使用。这意味着诸如 SML 这样的技术可以使用第十七章中描述的任何增强 MCMC 的技术（例如并行回火）来加以改进 (Desjardins *et al.*, 2010; Cho *et al.*, 2010b)。

一种在学习期间加速混合的方法是，不改变蒙特卡罗采样技术，而是改变模型的参数化和代价函数。**快速持续性对比散度** (fast persistent contrastive divergence)，或者 FPCD (Tieleman and Hinton, 2009) 使用如下表达式去替换传统模型的参数  $\theta$

$$\theta = \theta^{(\text{slow})} + \theta^{(\text{fast})}. \quad (18.16)$$

现在的参数是以前的两倍多，将其逐个相加以定义原始模型的参数。快速复制参数可以使用更大的学习率来训练，从而使其快速响应学习的负相，并促使马尔可夫链探索新的区域。这能够使马尔可夫链快速混合，尽管这种效应只会在学习期间快速权重可以自由改变的时候。通常，在短时间地将快速权重设为大值并保持足够长时间，使马尔可夫链改变峰值之后，我们会对快速权重使用显著的权重衰减，促使它们收敛到较小的值。

本节介绍的基于 MCMC 的方法的一个关键优点是它们提供了  $\log Z$  梯度的估计，因此我们可以从本质上将问题分解为  $\log \tilde{p}$  和  $\log Z$  两块。然后我们可以使用任何其他的方法来处理  $\log \tilde{p}(\mathbf{x})$ ，只需将我们的负相梯度加到其他方法的梯度中。特别地，这意味着正相可以使用那些仅提供  $\tilde{p}$  下限的方法。然而，本章介绍处理  $\log Z$  的大多数其他方法都和基于边界的正相方法是不兼容的。

### 18.3 伪似然

蒙特卡罗近似配分函数及其梯度需要直接处理配分函数。有些其他方法通过训练不需要计算配分函数的模型来绕开这个问题。这些方法大多数都基于以下观察：无向概率模型中很容易计算概率的比率。这是因为配分函数同时出现在比率的分子和分母中，互相抵消：

$$\frac{p(\mathbf{x})}{p(\mathbf{y})} = \frac{\frac{1}{Z}\tilde{p}(\mathbf{x})}{\frac{1}{Z}\tilde{p}(\mathbf{y})} = \frac{\tilde{p}(\mathbf{x})}{\tilde{p}(\mathbf{y})}. \quad (18.17)$$

伪似然正是基于条件概率可以采用这种基于比率的形式，因此可以在没有配分函数的情况下进行计算。假设我们将  $\mathbf{x}$  分为  $\mathbf{a}$ ,  $\mathbf{b}$  和  $\mathbf{c}$ ，其中  $\mathbf{a}$  包含我们想要的条

件分布的变量， $\mathbf{b}$  包含我们想要条件化的变量， $\mathbf{c}$  包含除此之外的变量：

$$p(\mathbf{a} \mid \mathbf{b}) = \frac{p(\mathbf{a}, \mathbf{b})}{p(\mathbf{b})} = \frac{p(\mathbf{a}, \mathbf{b})}{\sum_{\mathbf{a}, \mathbf{c}} p(\mathbf{a}, \mathbf{b}, \mathbf{c})} = \frac{\tilde{p}(\mathbf{a}, \mathbf{b})}{\sum_{\mathbf{a}, \mathbf{c}} \tilde{p}(\mathbf{a}, \mathbf{b}, \mathbf{c})}. \quad (18.18)$$

以上计算需要边缘化  $\mathbf{a}$ ，假设  $\mathbf{a}$  和  $\mathbf{c}$  包含的变量并不多，那么这将是非常高效的操作。在极端情况下， $\mathbf{a}$  可以是单个变量， $\mathbf{c}$  可以为空，那么该计算仅需要估计与单个随机变量值一样多的  $\tilde{p}$ 。

不幸的是，为了计算对数似然，我们需要边缘化很多变量。如果总共有  $n$  个变量，那么我们必须边缘化  $n - 1$  个变量。根据概率的链式法则，我们有

$$\log p(\mathbf{x}) = \log p(x_1) + \log p(x_2 \mid x_1) + \cdots + \log p(x_n \mid \mathbf{x}_{1:n-1}). \quad (18.19)$$

在这种情况下，我们已经使  $\mathbf{a}$  尽可能小，但是  $\mathbf{c}$  可以大到  $\mathbf{x}_{2:n}$ 。如果我们简单地将  $\mathbf{c}$  移到  $\mathbf{b}$  中以减少计算代价，那么会发生什么呢？这便产生了**伪似然** (pseudolikelihood) (Besag, 1975) 目标函数，给定所有其他特征  $\mathbf{x}_{-i}$ ，预测特征  $x_i$  的值：

$$\sum_{i=1}^n \log p(x_i \mid \mathbf{x}_{-i}). \quad (18.20)$$

如果每个随机变量有  $k$  个不同的值，那么计算  $\tilde{p}$  需要  $k \times n$  次估计，而计算配分函数需要  $k^n$  次估计。

这看起来似乎是一个没有道理的策略，但可以证明最大化伪似然的估计是渐近一致的 (Mase, 1995)。当然，在数据集不趋近于大采样极限的情况下，伪似然可能表现出与最大似然估计不同的结果。

我们可以使用**广义伪似然估计** (generalized pseudolikelihood estimator) 来权衡计算复杂度和最大似然表现的偏差 (Huang and Ogata, 2002)。广义伪似然估计使用  $m$  个不同的集合  $\mathbb{S}^{(i)}$ ， $i = 1, \dots, m$  作为变量的指标出现在条件棒的左侧。在  $m = 1$  和  $\mathbb{S}^{(1)} = 1, \dots, n$  的极端情况下，广义伪似然估计会变为对数似然。在  $m = n$  和  $\mathbb{S}^{(i)} = \{i\}$  的极端情况下，广义伪似然会恢复为伪似然。广义伪似然估计目标函数如下所示

$$\sum_{i=1}^m \log p(\mathbf{x}_{\mathbb{S}^{(i)}} \mid \mathbf{x}_{-\mathbb{S}^{(i)}}). \quad (18.21)$$

基于伪似然的方法的性能在很大程度上取决于模型是如何使用的。对于完全联合分布  $p(\mathbf{x})$  模型的任务（例如密度估计和采样），伪似然通常效果不好。对于在训

练期间只需要使用条件分布的任务而言，它的效果比最大似然更好，例如填充少量的缺失值。如果数据具有规则结构，使得  $\mathbb{S}$  索引集可以被设计为表现最重要的相关性质，同时略去相关性可忽略的变量，那么广义伪似然策略将会非常有效。例如，在自然图像中，空间中相隔很远的像素也具有弱相关性，因此广义伪似然可以应用于每个  $\mathbb{S}$  集是小的局部空间窗口的情况。

伪似然估计的一个弱点是它不能与仅在  $\tilde{p}(\mathbf{x})$  上提供下界的其他近似一起使用，例如第十九章中介绍的变分推断。这是因为  $\tilde{p}$  出现在了分母中。分母的下界仅提供了整个表达式的上界，然而最大化上界没有什么意义。这使得我们难以将伪似然方法应用于诸如深度玻尔兹曼机的深度模型，因为变分方法是近似边缘化互相作用的多层隐藏变量的主要方法之一。尽管如此，伪似然仍然可以用在深度学习中，它可以用于单层模型，或使用不基于下界的近似推断方法的深度模型中。

伪似然比 SML 在每个梯度步骤中的计算代价要大得多，这是由于其对所有条件进行显式计算。但是，如果每个样本只计算一个随机选择的条件，那么广义伪似然和类似标准仍然可以很好地运行，从而使计算代价降低到和 SML 差不多的程度 (Goodfellow *et al.*, 2013d)。

虽然伪似然估计没有显式地最小化  $\log Z$ ，但是我们仍然认为它具有类似负相的效果。每个条件分布的分母会使得学习算法降低所有仅具有一个变量不同于训练样本的状态的概率。

读者可以参考 Marlin and de Freitas (2011) 了解伪似然渐近效率的理论分析，。

## 18.4 得分匹配和比率匹配

得分匹配 (Hyvärinen, 2005b) 提供了另一种训练模型而不需要估计  $Z$  或其导数的一致性方法。对数密度关于参数的导数  $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ ，被称为其 **得分** (score)，得分匹配这个名称正是来自这样的术语。得分匹配采用的策略是，最小化模型对数密度和数据对数密度关于输入的导数之间的平方差期望：

$$L(\mathbf{x}, \boldsymbol{\theta}) = \frac{1}{2} \|\nabla_{\mathbf{x}} \log p_{\text{model}}(\mathbf{x}; \boldsymbol{\theta}) - \nabla_{\mathbf{x}} \log p_{\text{data}}(\mathbf{x})\|_2^2, \quad (18.22)$$

$$J(\boldsymbol{\theta}) = \frac{1}{2} \mathbb{E}_{p_{\text{data}}(\mathbf{x})} L(\mathbf{x}, \boldsymbol{\theta}), \quad (18.23)$$

$$\boldsymbol{\theta}^* = \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta}). \quad (18.24)$$

该目标函数避免了微分配分函数  $Z$  带来的难题，因为  $Z$  不是  $\mathbf{x}$  的函数，所以  $\nabla_{\mathbf{x}}Z = 0$ 。最初，得分匹配似乎有一个新的困难：计算数据分布的得分需要知道生成训练数据的真实分布  $p_{\text{data}}$ 。幸运的是，最小化  $L(\mathbf{x}; \boldsymbol{\theta})$  的期望等价于最小化下式的期望

$$\tilde{L}(\mathbf{x}, \boldsymbol{\theta}) = \sum_{j=1}^n \left( \frac{\partial^2}{\partial x_j^2} \log p_{\text{model}}(\mathbf{x}; \boldsymbol{\theta}) + \frac{1}{2} \left( \frac{\partial}{\partial x_j} \log p_{\text{model}}(\mathbf{x}; \boldsymbol{\theta}) \right)^2 \right), \quad (18.25)$$

其中  $n$  是  $\mathbf{x}$  的维度。

因为得分匹配需要关于  $\mathbf{x}$  的导数，所以它不适用于具有离散数据的模型，但是模型中的潜变量可以是离散的。

类似于伪似然，得分匹配只有在我们能够直接估计  $\log \tilde{p}(\mathbf{x})$  及其导数的时候才有效。它与对  $\log \tilde{p}(\mathbf{x})$  仅提供下界的方法不兼容，因为得分匹配需要  $\log \tilde{p}(\mathbf{x})$  的导数和二阶导数，而下限不能传达关于导数的任何信息。这意味着得分匹配不能应用于隐藏单元之间具有复杂相互作用的模型估计，例如稀疏编码模型或深度玻尔兹曼机。虽然得分匹配可以用于预训练较大模型的第一个隐藏层，但是它没有被用于预训练较大模型的较深层网络。这可能是因为这些模型的隐藏层通常包含一些离散变量。

虽然得分匹配没有明确显示具有负相信息，但是它可以被视为使用特定类型马尔可夫链的对比散度的变种 (Hyvärinen, 2007a)。在这种情况下，马尔可夫链并没有采用 Gibbs 采样，而是采用一种由梯度引导局部更新的不同方法。当局部更新的大小接近于零时，得分匹配等价于具有这种马尔可夫链的对比散度。

Lyu (2009) 将得分匹配推广到离散的情况（但是推导有误，后由 Marlin *et al.* (2010) 修正）。Marlin *et al.* (2010) 发现，广义得分匹配 (**generalized score matching**, GSM) 在许多样本观测概率为 0 的高维离散空间中不起作用。

一种更成功地将得分匹配的基本想法扩展到离散数据的方法是 **比率匹配** (ratio matching) (Hyvärinen, 2007b)。比率匹配特别适用于二值数据。比率匹配最小化以下目标函数在样本上的均值：

$$L^{(\text{RM})}(\mathbf{x}, \boldsymbol{\theta}) = \sum_{j=1}^n \left( \frac{1}{1 + \frac{p_{\text{model}}(\mathbf{x}; \boldsymbol{\theta})}{p_{\text{model}}(f(\mathbf{x}), j; \boldsymbol{\theta})}} \right)^2. \quad (18.26)$$

其中  $f(\mathbf{x}, j)$  返回  $j$  处位值取反的  $\mathbf{x}$ 。比率匹配使用了与伪似然估计相同的策略来绕开配分函数：配分函数会在两个概率的比率中抵消掉。Marlin *et al.* (2010) 发现，训

练模型给测试集图像去噪时，比率匹配的效果要优于 SML、伪似然和 GSM。

类似于伪似然估计，比率匹配对每个数据点都需要  $n$  个  $\tilde{p}$  的估计，因此每次更新的计算代价大约比 SML 的计算代价高出  $n$  倍。

与伪似然估计一样，我们可以认为比率匹配减小了所有只有一个变量不同于训练样本的状态的概率。由于比率匹配特别适用于二值数据，这意味着在与数据的汉明距离为 1 内的所有状态上，比率匹配都是有效的。

比率匹配还可以作为处理高维稀疏数据（例如词计数向量）的基础。这类稀疏数据对基于 MCMC 的方法提出了挑战，因为以密集格式表示数据是非常消耗计算资源的，而只有在模型学会表示数据分布的稀疏性之后，MCMC 采样才会产生稀疏值。Dauphin and Bengio (2013) 设计了比率匹配的无偏随机近似来解决这个问题。该近似只估计随机选择的目标子集，不需要模型生成完整的样本。

读者可以参考 Marlin and de Freitas (2011) 了解比率匹配渐近效率的理论分析，。

## 18.5 去噪得分匹配

某些情况下，我们希望拟合以下分布来正则化得分匹配

$$p_{\text{smoothed}}(\mathbf{x}) = \int p_{\text{data}}(\mathbf{y})q(\mathbf{x} \mid \mathbf{y})d\mathbf{y} \quad (18.27)$$

而不是拟合真实分布  $p_{\text{data}}$ 。分布  $q(\mathbf{x} \mid \mathbf{y})$  是一个损坏过程，通常在形成  $\mathbf{x}$  的过程中会向  $\mathbf{y}$  中添加少量噪声。

去噪得分匹配非常有用，因为在实践中，通常我们不能获取真实的  $p_{\text{data}}$ ，而只能得到其样本确定的经验分布。给定足够容量，任何一致估计都会使  $p_{\text{model}}$  成为一组以训练点为中心的 Dirac 分布。考虑在第 5.4.5 节介绍的渐近一致性上的损失，通过  $q$  来平滑有助于缓解这个问题。Kingma and LeCun (2010b) 介绍了平滑分布  $q$  为正态分布噪声的正则化得分匹配。

回顾第 14.5.1 节，有一些自编码器训练算法等价于得分匹配或去噪得分匹配。因此，这些自编码器训练算法也是解决配分函数问题的一种方式。

## 18.6 噪声对比估计

具有难求解的配分函数的大多数模型估计都没有估计配分函数。SML 和 CD 只估计对数配分函数的梯度，而不是估计配分函数本身。得分匹配和伪似然避免了和配分函数相关的计算。

**噪声对比估计** (**noise-contrastive estimation**, NCE) (Gutmann and Hyvonen, 2010) 采取了一种不同的策略。在这种方法中，模型估计的概率分布被明确表示为

$$\log p_{\text{model}}(\mathbf{x}) = \log \tilde{p}_{\text{model}}(\mathbf{x}; \boldsymbol{\theta}) + c, \quad (18.28)$$

其中  $c$  是  $-\log Z(\boldsymbol{\theta})$  的近似。噪声对比估计过程将  $c$  视为另一参数，使用相同的算法同时估计  $\boldsymbol{\theta}$  和  $c$ ，而不是仅仅估计  $\boldsymbol{\theta}$ 。因此，所得到的  $\log p_{\text{model}}(\mathbf{x})$  可能并不完全对应有效的概率分布，但随着  $c$  估计的改进，它将变得越来越接近有效值<sup>1</sup>。

这种方法不可能使用最大似然作为估计的标准。最大似然标准可以设置  $c$  为任意大的值，而不是设置  $c$  以创建一个有效的概率分布。

NCE 将估计  $p(\mathbf{x})$  的无监督学习问题转化为学习一个概率二元分类器，其中一个类别对应模型生成的数据。该监督学习问题中的最大似然估计定义了原始问题的渐近一致估计。

具体地说，我们引入第二个分布，**噪声分布** (noise distribution)  $p_{\text{noise}}(\mathbf{x})$ 。噪声分布应该易于估计和从中采样。我们现在可以构造一个联合  $\mathbf{x}$  和新二值变量  $y$  的模型。在新的联合模型中，我们指定

$$p_{\text{joint}}(y = 1) = \frac{1}{2}, \quad (18.29)$$

$$p_{\text{joint}}(\mathbf{x} \mid y = 1) = p_{\text{model}}(\mathbf{x}), \quad (18.30)$$

和

$$p_{\text{joint}}(\mathbf{x} \mid y = 0) = p_{\text{noise}}(\mathbf{x}). \quad (18.31)$$

换言之， $y$  是一个决定我们从模型还是从噪声分布中生成  $\mathbf{x}$  的开关变量。

我们可以在训练数据上构造一个类似的联合模型。在这种情况下，开关变量决定是从**数据**还是从噪声分布中抽取  $\mathbf{x}$ 。正式地， $p_{\text{train}}(y = 1) = \frac{1}{2}$ ， $p_{\text{train}}(\mathbf{x} \mid y = 1) = p_{\text{data}}(\mathbf{x})$ ，和  $p_{\text{train}}(\mathbf{x} \mid y = 0) = p_{\text{noise}}(\mathbf{x})$ 。

---

<sup>1</sup>NCE 也适用于具有易于处理的，不需要引入额外参数  $c$  的配分函数的问题。它已经是最令人感兴趣的，估计具有复杂配分函数模型的方法。

现在我们可以应用标准的最大似然学习拟合  $p_{\text{joint}}$  到  $p_{\text{train}}$  的监督学习问题：

$$\theta, c = \arg \max_{\theta, c} \mathbb{E}_{\mathbf{x}, \mathbf{y} \sim p_{\text{train}}} \log p_{\text{joint}}(\mathbf{y} | \mathbf{x}). \quad (18.32)$$

分布  $p_{\text{joint}}$  本质上是将逻辑回归模型应用于模型和噪声分布之间的对数概率之差：

$$p_{\text{joint}}(\mathbf{y} = 1 | \mathbf{x}) = \frac{p_{\text{model}}(\mathbf{x})}{p_{\text{model}}(\mathbf{x}) + p_{\text{noise}}(\mathbf{x})} \quad (18.33)$$

$$= \frac{1}{1 + \frac{p_{\text{noise}}(\mathbf{x})}{p_{\text{model}}(\mathbf{x})}} \quad (18.34)$$

$$= \frac{1}{1 + \exp\left(\log \frac{p_{\text{noise}}(\mathbf{x})}{p_{\text{model}}(\mathbf{x})}\right)} \quad (18.35)$$

$$= \sigma\left(-\log \frac{p_{\text{noise}}(\mathbf{x})}{p_{\text{model}}(\mathbf{x})}\right) \quad (18.36)$$

$$= \sigma(\log p_{\text{model}}(\mathbf{x}) - \log p_{\text{noise}}(\mathbf{x})). \quad (18.37)$$

因此，只要  $\log \tilde{p}_{\text{model}}$  易于反向传播，并且如上所述， $p_{\text{noise}}$  应易于估计（以便评估  $p_{\text{joint}}$ ）和采样（以生成训练数据），那么 NCE 就易于使用。

NCE 能够非常成功地应用于随机变量较少的问题，但即使随机变量有很多可以取的值时，它也很有效。例如，它已经成功地应用于给定单词上下文建模单词的条件分布 (Mnih and Kavukcuoglu, 2013)。虽然单词可以采样自一个很大的词汇表，但是只能采样一个单词。

当 NCE 应用于具有许多随机变量的问题时，其效率会变得较低。当逻辑回归分类器发现某个变量的取值不大可能时，它会拒绝这个噪声样本。这意味着在  $p_{\text{model}}$  学习了基本的边缘统计之后，学习进程会大大减慢。想象一个使用非结构化高斯噪声作为  $p_{\text{noise}}$  来学习面部图像的模型。如果  $p_{\text{model}}$  学会了眼睛，就算没有学习任何其他面部特征，比如嘴，它也会拒绝几乎所有的非结构化噪声样本。

噪声分布  $p_{\text{noise}}$  必须是易于估计和采样的约束可能是过于严格的限制。当  $p_{\text{noise}}$  比较简单时，大多数采样可能与数据有着明显不同，而不会迫使  $p_{\text{model}}$  进行显著改进。

类似于得分匹配和伪似然，如果  $\tilde{p}$  只有下界，那么 NCE 不会有效。这样的下界能够用于构建  $p_{\text{joint}}(\mathbf{y} = 1 | \mathbf{x})$  的下界，但是它只能用于构建  $p_{\text{joint}}(\mathbf{y} = 0 | \mathbf{x})$ （出现

在一半的 NCE 对象中) 的上界。同样地,  $p_{\text{noise}}$  的下界也没有用, 因为它只提供了  $p_{\text{joint}}(y=1 \mid \mathbf{x})$  的上界。

在每个梯度步骤之前, 模型分布被复制来定义新的噪声分布时, NCE 定义了一个被称为 **自对比估计** (self-contrastive estimation) 的过程, 其梯度期望等价于最大似然的梯度期望 (Goodfellow, 2014)。特殊情况的 NCE (噪声采样由模型生成) 表明最大似然可以被解释为使模型不断学习以将现实与自身发展的信念区分的过程, 而噪声对比估计通过让模型区分现实和固定的基准 (噪声模型), 我们降低了计算成本。

在训练样本和生成样本 (使用模型能量函数定义分类器) 之间进行分类以得到模型的梯度的方法, 已经在更早的时候以各种形式提出来 (Welling *et al.*, 2003b; Bengio, 2009)。

噪声对比估计是基于良好生成模型应该能够区分数据和噪声的想法。一个密切相关的想法是, 良好的生成模型能够生成分类器无法将其与数据区分的样本。这个想法诞生了生成式对抗网络 (第 20.10.4 节)。

## 18.7 估计配分函数

尽管本章中的大部分内容都在避免计算与无向图模型相关的难以计算的配分函数  $Z(\boldsymbol{\theta})$ , 但在本节中我们将会讨论几种直接估计配分函数的方法。

估计配分函数可能会很重要, 当我们希望计算数据的归一化似然时, 我们会需要它。在评估模型, 监控训练性能, 和比较模型时, 这通常是很重要的。

例如, 假设我们有两个模型: 概率分布为  $p_A(\mathbf{x}; \boldsymbol{\theta}_A) = \frac{1}{Z_A} \tilde{p}_A(\mathbf{x}; \boldsymbol{\theta}_A)$  的模型  $\mathcal{M}_A$  和概率分布为  $p_B(\mathbf{x}; \boldsymbol{\theta}_B) = \frac{1}{Z_B} \tilde{p}_B(\mathbf{x}; \boldsymbol{\theta}_B)$  的模型  $\mathcal{M}_B$ 。比较模型的常用方法是评估和比较两个模型分配给独立同分布测试数据集的似然。假设测试集含  $m$  个样本  $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ 。如果  $\prod_i p_A(\mathbf{x}^{(i)}; \boldsymbol{\theta}_A) > \prod_i p_B(\mathbf{x}^{(i)}; \boldsymbol{\theta}_B)$ , 或等价地, 如果

$$\sum_i \log p_A(\mathbf{x}^{(i)}; \boldsymbol{\theta}_A) - \sum_i \log p_B(\mathbf{x}^{(i)}; \boldsymbol{\theta}_B) > 0, \quad (18.38)$$

那么我们说  $\mathcal{M}_A$  是一个比  $\mathcal{M}_B$  更好的模型 (或者, 至少可以说, 它在测试集上是一个更好的模型), 这是指它有一个更好的测试对数似然。不幸的是, 测试这个条件是否成立需要知道配分函数。式 (18.38) 看起来需要估计模型分配给每个点的对数概率, 因而需要估计配分函数。我们可以通过将式 (18.38) 重新转化为另一种形式来简

化情况，在该形式中我们只需要知道两个模型的配分函数的比率：

$$\sum_i \log p_A(\mathbf{x}^{(i)}; \boldsymbol{\theta}_A) - \sum_i \log p_B(\mathbf{x}^{(i)}; \boldsymbol{\theta}_B) = \sum_i \left( \log \frac{\tilde{p}_A(\mathbf{x}^{(i)}; \boldsymbol{\theta}_A)}{\tilde{p}_B(\mathbf{x}^{(i)}; \boldsymbol{\theta}_B)} \right) - m \log \frac{Z(\boldsymbol{\theta}_A)}{Z(\boldsymbol{\theta}_B)}. \quad (18.39)$$

因此，我们可以在不知道任一模型的配分函数，而只知道它们比率的情况下，判断模型  $\mathcal{M}_A$  是否比模型  $\mathcal{M}_B$  更优。正如我们将很快看到的，在两个模型相似的情况下，我们可以使用重要采样来估计比率。

然而，如果我们想要计算测试数据在  $\mathcal{M}_A$  或  $\mathcal{M}_B$  上的真实概率，我们需要计算配分函数的真实值。如果我们知道两个配分函数的比率， $r = \frac{Z(\boldsymbol{\theta}_B)}{Z(\boldsymbol{\theta}_A)}$ ，并且我们知道两者中一个的实际值，比如说  $Z(\boldsymbol{\theta}_A)$ ，那么我们可以计算另一个的值：

$$Z(\boldsymbol{\theta}_B) = r Z(\boldsymbol{\theta}_A) = \frac{Z(\boldsymbol{\theta}_B)}{Z(\boldsymbol{\theta}_A)} Z(\boldsymbol{\theta}_A). \quad (18.40)$$

一种估计配分函数的简单方法是使用蒙特卡罗方法，例如简单重要采样。以下用连续变量积分来表示该方法，也可以替换积分为求和，很容易将其应用到离散变量的情况。我们使用提议分布  $p_0(\mathbf{x}) = \frac{1}{Z_0} \tilde{p}_0(\mathbf{x})$ ，其在配分函数  $Z_0$  和未归一化分布  $\tilde{p}_0(\mathbf{x})$  上易于采样和估计。

$$Z_1 = \int \tilde{p}_1(\mathbf{x}) d\mathbf{x} \quad (18.41)$$

$$= \int \frac{p_0(\mathbf{x})}{p_0(\mathbf{x})} \tilde{p}_1(\mathbf{x}) d\mathbf{x} \quad (18.42)$$

$$= Z_0 \int p_0(\mathbf{x}) \frac{\tilde{p}_1(\mathbf{x})}{\tilde{p}_0(\mathbf{x})} d\mathbf{x} \quad (18.43)$$

$$\hat{Z}_1 = \frac{Z_0}{K} \sum_{k=1}^K \frac{\tilde{p}_1(\mathbf{x}^{(k)})}{\tilde{p}_0(\mathbf{x}^{(k)})} \quad \text{s.t. : } \mathbf{x}^{(k)} \sim p_0 \quad (18.44)$$

在最后一行，我们使用蒙特卡罗估计，使用从  $p_0(\mathbf{x})$  中抽取的采样计算积分  $\hat{Z}_1$ ，然后用未归一化的  $\tilde{p}_1$  和提议分布  $p_0$  的比率对每个采样加权。

这种方法使得我们可以估计配分函数之间的比率：

$$\frac{1}{K} \sum_{k=1}^K \frac{\tilde{p}_1(\mathbf{x}^{(k)})}{\tilde{p}_0(\mathbf{x}^{(k)})} \quad \text{s.t. : } \mathbf{x}^{(k)} \sim p_0. \quad (18.45)$$

然后该值可以直接比较式 (18.39) 中的两个模型。

如果分布  $p_0$  接近  $p_1$ , 那么式(18.44)能够有效地估计配分函数(Minka, 2005)。不幸的是, 大多数时候  $p_1$  都很复杂(通常是多峰值的), 并且定义在高维空间中。很难找到一个易求解的  $p_0$ , 既能易于评估, 又能充分接近  $p_1$  以保持高质量的近似。如果  $p_0$  和  $p_1$  不接近, 那么  $p_0$  的大多数采样将在  $p_1$  中具有较低的概率, 从而在式(18.44)的求和中产生(相对的)可忽略的贡献。

如果求和中只有少数几个具有显著权重的样本, 那么将会由于高方差而导致估计的效果很差。这可以通过估计  $\hat{Z}_1$  的方差来定量地理解:

$$\text{Var}(\hat{Z}_1) = \frac{Z_0}{K^2} \sum_{k=1}^K \left( \frac{\tilde{p}_1(\mathbf{x}^{(k)})}{\tilde{p}_0(\mathbf{x}^{(k)})} - \hat{Z}_1 \right)^2. \quad (18.46)$$

当重要性权重  $\frac{\tilde{p}_1(\mathbf{x}^{(k)})}{\tilde{p}_0(\mathbf{x}^{(k)})}$  存在显著偏差时, 上式的值是最大的。

我们现在关注两个解决高维空间复杂分布上估计配分函数的方法: 退火重要采样和桥式采样。两者都始于上面介绍的简单重要采样方法, 并且都试图通过引入缩小  $p_0$  和  $p_1$  之间差距的中间分布, 来解决  $p_0$  远离  $p_1$  的问题。

### 18.7.1 退火重要采样

在  $D_{KL}(p_0||p_1)$  很大的情况下(即  $p_0$  和  $p_1$  之间几乎没有重叠), 一种称为**退火重要采样**(annealed importance sampling, AIS)的方法试图通过引入中间分布来缩小这种差距(Jarzynski, 1997; Neal, 2001)。考虑分布序列  $p_{\eta_0}, \dots, p_{\eta_n}$ , 其中  $0 = \eta_0 < \eta_1 < \dots < \eta_{n-1} < \eta_n = 1$ , 分布序列中的第一个和最后一个分别是  $p_0$  和  $p_1$ 。

这种方法使我们能够估计定义在高维空间多峰分布(例如训练 RBM 时定义的分布)上的配分函数。我们从一个已知配分函数的简单模型(例如, 权重为零的RBM)开始, 估计两个模型配分函数之间的比率。该比率的估计基于许多个相似分布的比率估计, 例如在零和学习到的权重之间插值一组权重不同的RBM。

现在我们可以将比率  $\frac{Z_1}{Z_0}$  写作

$$\frac{Z_1}{Z_0} = \frac{Z_1}{Z_0} \frac{Z_{\eta_1}}{Z_{\eta_1}} \cdots \frac{Z_{\eta_{n-1}}}{Z_{\eta_{n-1}}} \quad (18.47)$$

$$= \frac{Z_{\eta_1}}{Z_0} \frac{Z_{\eta_2}}{Z_{\eta_1}} \cdots \frac{Z_{\eta_{n-1}}}{Z_{\eta_{n-2}}} \frac{Z_1}{Z_{\eta_{n-1}}} \quad (18.48)$$

$$= \prod_{j=0}^{n-1} \frac{Z_{\eta_{j+1}}}{Z_{\eta_j}}. \quad (18.49)$$

如果对于所有的  $0 \leq j \leq n-1$ , 分布  $p_{\eta_j}$  和  $p_{\eta_{j+1}}$  足够接近, 那么我们能够使用简单的重要采样来估计每个因子  $\frac{Z_{\eta_{j+1}}}{Z_{\eta_j}}$ , 然后使用这些得到  $\frac{Z_1}{Z_0}$  的估计。

这些中间分布是从哪里来的呢? 正如最先的提议分布  $p_0$  是一种设计选择, 分布序列  $p_{\eta_1} \dots p_{\eta_{n-1}}$  也是如此。也就是说, 它们可以被特别设计为特定的问题领域。中间分布的一个通用和流行选择是使用目标分布  $p_1$  的加权几何平均, 起始分布 (其配分函数是已知的) 为  $p_0$ :

$$p_{\eta_j} \propto p_1^{\eta_j} p_0^{1-\eta_j}. \quad (18.50)$$

为了从这些中间分布中采样, 我们定义了一组马尔可夫链转移函数  $T_{\eta_j}(\mathbf{x}' | \mathbf{x})$ , 定义了给定  $\mathbf{x}$  转移到  $\mathbf{x}'$  的条件概率分布。转移算子  $T_{\eta_j}(\mathbf{x}' | \mathbf{x})$  定义如下, 保持  $p_{\eta_j}(\mathbf{x})$  不变:

$$p_{\eta_j}(\mathbf{x}) = \int p_{\eta_j}(\mathbf{x}') T_{\eta_j}(\mathbf{x} | \mathbf{x}') d\mathbf{x}'. \quad (18.51)$$

这些转移可以被构造为任何马尔可夫链蒙特卡罗方法 (例如, Metropolis-Hastings, Gibbs), 包括涉及多次遍历所有随机变量或其他迭代的方法。

然后, AIS 采样方法从  $p_0$  开始生成样本, 并使用转移算子从中间分布顺序地生成采样, 直到我们得到目标分布  $p_1$  的采样:

- 对于  $k = 1 \dots K$ 
  - 采样  $\mathbf{x}_{\eta_1}^{(k)} \sim p_0(\mathbf{x})$
  - 采样  $\mathbf{x}_{\eta_2}^{(k)} \sim T_{\eta_1}(\mathbf{x}_{\eta_2}^{(k)} | \mathbf{x}_{\eta_1}^{(k)})$
  - ...
  - 采样  $\mathbf{x}_{\eta_{n-1}}^{(k)} \sim T_{\eta_{n-2}}(\mathbf{x}_{\eta_{n-1}}^{(k)} | \mathbf{x}_{\eta_{n-2}}^{(k)})$
  - 采样  $\mathbf{x}_{\eta_n}^{(k)} \sim T_{\eta_{n-1}}(\mathbf{x}_{\eta_n}^{(k)} | \mathbf{x}_{\eta_{n-1}}^{(k)})$
- 结束

对于采样  $k$ , 通过连接式(18.49)给出的中间分布之间的重要性权重, 我们可以导出目标重要性权重:

$$w^{(k)} = \frac{\tilde{p}_{\eta_1}(\mathbf{x}_{\eta_1}^{(k)})}{\tilde{p}_0(\mathbf{x}_{\eta_1}^{(k)})} \frac{\tilde{p}_{\eta_2}(\mathbf{x}_{\eta_2}^{(k)})}{\tilde{p}_{\eta_1}(\mathbf{x}_{\eta_2}^{(k)})} \cdots \frac{\tilde{p}_1(\mathbf{x}_1^{(k)})}{\tilde{p}_{\eta_{n-1}}(\mathbf{x}_{\eta_n}^{(k)})}. \quad (18.52)$$

为了避免诸如上溢的数值问题, 最佳方法可能是通过加法或减法计算  $\log w^{(k)}$ , 而不是通过概率乘法和除法计算  $w^{(k)}$ 。

利用由此定义的采样过程和式(18.52)中给出的重要性权重, 配分函数的比率估计如下所示:

$$\frac{Z_1}{Z_0} \approx \frac{1}{K} \sum_{k=1}^K w^{(k)} \quad (18.53)$$

为了验证该过程定义的重要采样方案是否有效, 我们可以展示(Neal, 2001) AIS 过程对应着扩展状态空间上的简单重要采样, 其中数据点采样自乘积空间  $[\mathbf{x}_{\eta_1}, \dots, \mathbf{x}_{\eta_{n-1}}, \mathbf{x}_1]$ 。为此, 我们将扩展空间上的分布定义为

$$\tilde{p}(\mathbf{x}_{\eta_1}, \dots, \mathbf{x}_{\eta_{n-1}}, \mathbf{x}_1) \quad (18.54)$$

$$= \tilde{p}_1(\mathbf{x}_1) \tilde{T}_{\eta_{n-1}}(\mathbf{x}_{\eta_{n-1}} \mid \mathbf{x}_1) \tilde{T}_{\eta_{n-2}}(\mathbf{x}_{\eta_{n-2}} \mid \mathbf{x}_{\eta_{n-1}}) \cdots \tilde{T}_{\eta_1}(\mathbf{x}_{\eta_1} \mid \mathbf{x}_{\eta_2}), \quad (18.55)$$

其中  $\tilde{T}_a$  是由  $T_a$  定义的转移算子的逆(应用贝叶斯规则):

$$\tilde{T}_a(\mathbf{x}' \mid \mathbf{x}) = \frac{p_a(\mathbf{x}')}{p_a(\mathbf{x})} T_a(\mathbf{x} \mid \mathbf{x}') = \frac{\tilde{p}_a(\mathbf{x}')}{\tilde{p}_a(\mathbf{x})} T_a(\mathbf{x} \mid \mathbf{x}'). \quad (18.56)$$

将以上代入到式(18.55)给出的扩展状态空间上的联合分布中, 我们得到:

$$\tilde{p}(\mathbf{x}_{\eta_1}, \dots, \mathbf{x}_{\eta_{n-1}}, \mathbf{x}_1) \quad (18.57)$$

$$= \tilde{p}_1(\mathbf{x}_1) \frac{\tilde{p}_{\eta_{n-1}}(\mathbf{x}_{\eta_{n-1}})}{\tilde{p}_{\eta_{n-1}}(\mathbf{x}_1)} T_{\eta_{n-1}}(\mathbf{x}_1 \mid \mathbf{x}_{\eta_{n-1}}) \prod_{i=1}^{n-2} \frac{\tilde{p}_{\eta_i}(\mathbf{x}_{\eta_i})}{\tilde{p}_{\eta_i}(\mathbf{x}_{\eta_{i+1}})} T_{\eta_i}(\mathbf{x}_{\eta_{i+1}} \mid \mathbf{x}_{\eta_i}) \quad (18.58)$$

$$= \frac{\tilde{p}_1(\mathbf{x}_1)}{\tilde{p}_{\eta_{n-1}}(\mathbf{x}_1)} T_{\eta_{n-1}}(\mathbf{x}_1 \mid \mathbf{x}_{\eta_{n-1}}) \tilde{p}_{\eta_1}(\mathbf{x}_{\eta_1}) \prod_{i=1}^{n-2} \frac{\tilde{p}_{\eta_{i+1}}(\mathbf{x}_{\eta_{i+1}})}{\tilde{p}_{\eta_i}(\mathbf{x}_{\eta_{i+1}})} T_{\eta_i}(\mathbf{x}_{\eta_{i+1}} \mid \mathbf{x}_{\eta_i}). \quad (18.59)$$

通过上面给定的采样方案, 现在我们可以从扩展样本上的联合提议分布  $q$  上生成采样, 联合分布如下

$$q(\mathbf{x}_{\eta_1}, \dots, \mathbf{x}_{\eta_{n-1}}, \mathbf{x}_1) = p_0(\mathbf{x}_{\eta_1}) T_{\eta_1}(\mathbf{x}_{\eta_2} \mid \mathbf{x}_{\eta_1}) \cdots T_{\eta_{n-1}}(\mathbf{x}_1 \mid \mathbf{x}_{\eta_{n-1}}). \quad (18.60)$$

式(18.59)给出了扩展空间上的联合分布。将  $q(\mathbf{x}_{\eta_1}, \dots, \mathbf{x}_{\eta_{n-1}}, \mathbf{x}_1)$  作为扩展状态空间上的提议分布(我们会从中抽样), 重要性权重如下

$$w^{(k)} = \frac{\tilde{p}(\mathbf{x}_{\eta_1}, \dots, \mathbf{x}_{\eta_{n-1}}, \mathbf{x}_1)}{q(\mathbf{x}_{\eta_1}, \dots, \mathbf{x}_{\eta_{n-1}}, \mathbf{x}_1)} = \frac{\tilde{p}_1(\mathbf{x}_1^{(k)})}{\tilde{p}_{\eta_{n-1}}(\mathbf{x}_{\eta_{n-1}}^{(k)})} \cdots \frac{\tilde{p}_{\eta_2}(\mathbf{x}_{\eta_2}^{(k)})}{\tilde{p}_{\eta_1}(\mathbf{x}_{\eta_1}^{(k)})} \frac{\tilde{p}_{\eta_1}(\mathbf{x}_{\eta_1}^{(k)})}{\tilde{p}_0(\mathbf{x}_0^{(k)})}. \quad (18.61)$$

这些权重和 AIS 上的权重相同。因此, 我们可以将 AIS 解释为应用于扩展状态上的简单重要采样, 其有效性直接来源于重要采样的有效性。

退火重要采样首先由 Jarzynski (1997) 发现, 然后由 Neal (2001) 再次独立发现。目前它是估计无向概率模型的配分函数的最常用方法。其原因可能与一篇有影响力的论文 (Salakhutdinov and Murray, 2008) 有关, 该论文并没有讨论该方法相对于其他方法的优点, 而是介绍了将其应用于估计受限玻尔兹曼机和深度信念网络的配分函数。

关于 AIS 估计性质(例如, 方差和效率)的讨论, 请参看 Neal (2001)。

## 18.7.2 桥式采样

类似于 AIS, 桥式采样 (Bennett, 1976) 是另一种处理重要采样缺点的方法。并非将一系列中间分布连接在一起, 桥式采样依赖于单个分布  $p_*$ (被称为桥), 在已知配分函数的分布  $p_0$  和分布  $p_1$ (我们试图估计其配分函数  $Z_1$ ) 之间插值。

桥式采样估计比率  $Z_1/Z_0$ :  $\tilde{p}_0$  和  $\tilde{p}_*$  之间重要性权重期望与  $\tilde{p}_1$  和  $\tilde{p}_*$  之间重要性权重的比率,

$$\frac{Z_1}{Z_0} \approx \sum_{k=1}^K \frac{\tilde{p}_*(\mathbf{x}_0^{(k)})}{\tilde{p}_0(\mathbf{x}_0^{(k)})} \Big/ \sum_{k=1}^K \frac{\tilde{p}_*(\mathbf{x}_1^{(k)})}{\tilde{p}_1(\mathbf{x}_1^{(k)})}. \quad (18.62)$$

如果仔细选择桥式采样  $p_*$ , 使其与  $p_0$  和  $p_1$  都有很大重合的话, 那么桥式采样能够允许两个分布(或更正式地,  $D_{\text{KL}}(p_0 \| p_1)$ ) 之间有较大差距(相对标准重要采样而言)。

可以表明, 最优的桥式采样是  $p_*^{(\text{opt})}(\mathbf{x}) \propto \frac{\tilde{p}_0(\mathbf{x})\tilde{p}_1(\mathbf{x})}{r\tilde{p}_0(\mathbf{x}) + \tilde{p}_1(\mathbf{x})}$ , 其中  $r = Z_1/Z_0$ 。这似乎是一个不可行的解决方案, 因为它似乎需要我们估计数值  $Z_1/Z_0$ 。然而, 可以从粗糙的  $r$  开始估计, 然后使用得到的桥式采样逐步迭代以改进估计(Neal, 2005)。也就是说, 我们会迭代地重新估计比率, 并使用每次迭代更新  $r$  的值。

**链接重要采样** AIS 和桥式采样各有优点。如果  $D_{\text{KL}}(p_0 \| p_1)$  不太大（由于  $p_0$  和  $p_1$  足够接近）的话，那么桥式采样能比 AIS 更高效地估计配分函数比率。然而，如果对于单个分布  $p_*$  而言，两个分布相距太远难以桥接差距，那么 AIS 至少可以使用许多潜在中间分布来跨越  $p_0$  和  $p_1$  之间的差距。Neal (2005) 展示链接重要采样方法如何利用桥式采样的优点，桥接 AIS 中使用的中间分布，并且显著改进了整个配分函数的估计。

**在训练期间估计配分函数** 虽然 AIS 已经被认为是用于估计许多无向模型配分函数的标准方法，但是它在计算上代价很高，以致其在训练期间仍然不很实用。研究者探索了一些在训练过程中估计配分函数的替代方法。

使用桥式采样、短链 AIS 和并行回火的组合，Desjardins *et al.* (2011) 设计了一种在训练过程中追踪 RBM 配分函数的方法。该策略的基础是，在并行回火方法操作的每个温度下，RBM 配分函数的独立估计会一直保持。作者将相邻链（来自并行回火）的配分函数比率的桥式采样估计和跨越时间的 AIS 估计组合起来，提出一个在每次迭代学习时估计配分函数的（且方差较小的）方法。

本章中描述的工具提供了许多不同的方法，以解决难处理的配分函数问题，但是在训练和使用生成模型时，可能会存在一些其他问题。其中最重要的是我们接下来会遇到的难以推断的问题。

# 第十九章 近似推断

许多概率模型很难训练的原因是很难进行推断。在深度学习中，通常我们有一系列可见变量  $v$  和一系列潜变量  $h$ 。推断困难通常是指难以计算  $p(h | v)$  或其期望。而这样的操作在一些诸如最大似然学习的任务中往往是必需的。

许多仅含一个隐藏层的简单图模型会定义成易于计算  $p(h | v)$  或其期望的形式，例如受限玻尔兹曼机和概率 PCA。不幸的是，大多数具有多层隐藏变量的图模型的后验分布都很难处理。对于这些模型而言，精确推断算法需要指数量级的运行时间。即使一些只有单层的模型，如稀疏编码，也存在着这样的问题。

在本章中，我们将会介绍几个用来解决这些难以处理的推断问题的技巧。稍后，在第二十章中，我们还将描述如何将这些技巧应用到训练其他方法难以奏效的概率模型中，如深度信念网络、深度玻尔兹曼机。

在深度学习中难以处理的推断问题通常源于结构化图模型中潜变量之间的相互作用。读者可以参考图 19.1 的几个例子。这些相互作用可能是无向模型的直接相互作用，也可能是在有向模型中同一个可见变量的共同祖先之间的“相消解释”作用。

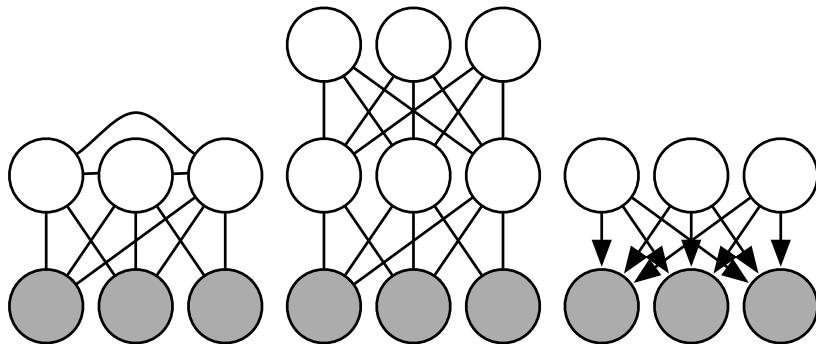


图 19.1: 深度学习中难以处理的推断问题通常是由于结构化图模型中潜变量的相互作用。这些相互作用产生于一个潜变量与另一个潜变量或者当V-结构的子节点可观察时与更长的激活路径相连。(左)一个隐藏单元存在连接的半受限玻尔兹曼机 (semi-restricted Boltzmann Machine) (Osindero and Hinton, 2008)。由于存在大量潜变量的团, 潜变量的直接连接使得后验分布难以处理。(中)一个深度玻尔兹曼机, 被分层从而使得不存在层内连接, 由于层之间的连接其后验分布仍然难以处理。(右)当可见变量可观察时这个有向模型的潜变量之间存在相互作用, 因为每两个潜变量都是共父。即使拥有上图中的某一种结构, 一些概率模型依然能够获得易于处理的关于潜变量的后验分布。如果我们选择条件概率分布来引入相对于图结构描述的额外的独立性这种情况也是可能出现的。举个例子, 概率 PCA 的图结构如右图所示, 然而由于其条件分布的特殊性质 (带有相互正交基向量的线性高斯条件分布) 依然能够进行简单的推断。

## 19.1 把推断视作优化问题

精确推断问题可以描述为一个优化问题, 有许多方法正是由此解决了推断的困难。通过近似这样一个潜在的优化问题, 我们往往可以推导出近似推断算法。

为了构造这样一个优化问题, 假设我们有一个包含可见变量  $v$  和潜变量  $h$  的概率模型。我们希望计算观察数据的对数概率  $\log p(v; \theta)$ 。有时候如果边缘化消去  $h$  的操作很费时, 我们会难以计算  $\log p(v; \theta)$ 。作为替代, 我们可以计算一个  $\log p(v; \theta)$  的下界  $\mathcal{L}(v, \theta, q)$ 。这个下界被称为 **证据下界** (evidence lower bound, ELBO)。这个下界的另一个常用名称是**负变分自由能** (variational free energy)。具体地, 这个证据下界是这样定义的:

$$\mathcal{L}(v, \theta, q) = \log p(v; \theta) - D_{\text{KL}}(q(h | v) \| p(h | v; \theta)), \quad (19.1)$$

其中  $q$  是关于  $h$  的一个任意概率分布。

因为  $\log p(v)$  和  $\mathcal{L}(v, \theta, q)$  之间的距离是由 KL 散度来衡量的, 且 KL 散度总是

非负的，我们可以发现  $\mathcal{L}$  总是小于等于所求的对数概率。当且仅当分布  $q$  完全相等于  $p(\mathbf{h} | \mathbf{v})$  时取到等号。

令人吃惊的是，对于某些分布  $q$ ，计算  $\mathcal{L}$  可以变得相当简单。通过简单的代数运算我们可以把  $\mathcal{L}$  重写成一个更加简单形式：

$$\mathcal{L}(\mathbf{v}; \boldsymbol{\theta}, q) = \log p(\mathbf{v}; \boldsymbol{\theta}) - D_{\text{KL}}(q(\mathbf{h} | \mathbf{v}) \| p(\mathbf{h} | \mathbf{v}; \boldsymbol{\theta})) \quad (19.2)$$

$$= \log p(\mathbf{v}; \boldsymbol{\theta}) - \mathbb{E}_{\mathbf{h} \sim q} \log \frac{q(\mathbf{h} | \mathbf{v})}{p(\mathbf{h} | \mathbf{v})} \quad (19.3)$$

$$= \log p(\mathbf{v}; \boldsymbol{\theta}) - \mathbb{E}_{\mathbf{h} \sim q} \log \frac{q(\mathbf{h} | \mathbf{v})}{\frac{p(\mathbf{h}, \mathbf{v}; \boldsymbol{\theta})}{p(\mathbf{v}; \boldsymbol{\theta})}} \quad (19.4)$$

$$= \log p(\mathbf{v}; \boldsymbol{\theta}) - \mathbb{E}_{\mathbf{h} \sim q} [\log q(\mathbf{h} | \mathbf{v}) - \log p(\mathbf{h}, \mathbf{v}; \boldsymbol{\theta}) + \log p(\mathbf{v}; \boldsymbol{\theta})] \quad (19.5)$$

$$= -\mathbb{E}_{\mathbf{h} \sim q} [\log q(\mathbf{h} | \mathbf{v}) - \log p(\mathbf{h}, \mathbf{v}; \boldsymbol{\theta})]. \quad (19.6)$$

这也给出了证据下界的标准定义：

$$\mathcal{L}(\mathbf{v}, \boldsymbol{\theta}, q) = \mathbb{E}_{\mathbf{h} \sim q} [\log p(\mathbf{h}, \mathbf{v})] + H(q). \quad (19.7)$$

对于一个选择的合适分布  $q$  来说， $\mathcal{L}$  是容易计算的。对任意分布  $q$  的选择来说， $\mathcal{L}$  提供了似然函数的一个下界。越好地近似  $p(\mathbf{h} | \mathbf{v})$  的分布  $q(\mathbf{h} | \mathbf{v})$ ，得到的下界就越紧，换言之，就是与  $\log p(\mathbf{v})$  更加接近。当  $q(\mathbf{h} | \mathbf{v}) = p(\mathbf{h} | \mathbf{v})$  时，这个近似是完美的，也意味着  $\mathcal{L}(\mathbf{v}, \boldsymbol{\theta}, q) = \log p(\mathbf{v}; \boldsymbol{\theta})$ 。

因此我们可以将推断问题看作是找一个分布  $q$  使得  $\mathcal{L}$  最大的过程。精确推断能够在包含分布  $p(\mathbf{h} | \mathbf{v})$  的函数族中搜索一个函数，完美地最大化  $\mathcal{L}$ 。在本章中，我们将会讲到如何通过近似优化寻找分布  $q$  的方法来推导出不同形式的近似推断。我们可以通过限定分布  $q$  的形式或者使用并不彻底的优化方法来使得优化的过程更加高效（却更粗略），但是优化的结果是不完美的，不求彻底地最大化  $\mathcal{L}$ ，而只要显著地提升  $\mathcal{L}$ 。

无论我们选择什么样的分布  $q$ ， $\mathcal{L}$  始终是一个下界。我们可以通过选择一个更简单或更复杂的计算过程来得到对应的更松或更紧的下界。通过一个不彻底的优化过程或者将分布  $q$  做很强的限定（并且使用一个彻底的优化过程）我们可以获得一个很差的分布  $q$ ，但是降低了计算开销。

## 19.2 期望最大化

我们介绍的第一个最大化下界  $\mathcal{L}$  的算法是 **期望最大化** (expectation maximization, EM) 算法。在潜变量模型中，这是一个非常常见的训练算法。在这里我们描述 Neal and Hinton (1999) 所提出的 EM 算法。与大多数我们在本章中介绍的其他算法不同的是，EM 并不是一个近似推断算法，而是一种能够学到近似后验的算法。

EM 算法由交替迭代，直到收敛的两步运算组成：

- **E 步** (expectation step)：令  $\boldsymbol{\theta}^{(0)}$  表示在这一步开始时的参数值。对任何我们想要训练的（对所有的或者小批量数据均成立）索引为  $i$  的训练样本  $\mathbf{v}^{(i)}$ ，令  $q(\mathbf{h}^{(i)} | \mathbf{v}) = p(\mathbf{h}^{(i)} | \mathbf{v}^{(i)}; \boldsymbol{\theta}^{(0)})$ 。通过这个定义，我们认为  $q$  在当前参数  $\boldsymbol{\theta}^{(0)}$  下定义。如果我们改变  $\boldsymbol{\theta}$ ，那么  $p(\mathbf{h} | \mathbf{v}; \boldsymbol{\theta})$  将会相应地变化，但是  $q(\mathbf{h} | \mathbf{v})$  还是不变并且等于  $p(\mathbf{h} | \mathbf{v}; \boldsymbol{\theta}^{(0)})$ 。
- **M 步** (maximization step)：使用选择的优化算法完全地或者部分地关于  $\boldsymbol{\theta}$  最大化

$$\sum_i \mathcal{L}(\mathbf{v}^{(i)}, \boldsymbol{\theta}, q). \quad (19.8)$$

这可以被看作通过坐标上升算法来最大化  $\mathcal{L}$ 。在第一步中，我们更新分布  $q$  来最大化  $\mathcal{L}$ ，而在第二步中，我们更新  $\boldsymbol{\theta}$  来最大化  $\mathcal{L}$ 。

基于潜变量模型的随机梯度上升可以被看作是一个 EM 算法的特例，其中 M 步包括了单次梯度操作。EM 算法的其他变种可以实现多次梯度操作。对一些模型族来说，M 步甚至可以直接推出解析解，不同于其他方法，在给定当前  $q$  的情况下直接求出最优解。

尽管 E 步采用的是精确推断，我们仍然可以将 EM 算法视作是某种程度上的近似推断。具体地说，M 步假设一个分布  $q$  可以被所有的  $\boldsymbol{\theta}$  值分享。当 M 步越来越远离 E 步中的  $\boldsymbol{\theta}^{(0)}$  时，这将会导致  $\mathcal{L}$  和真实的  $\log p(\mathbf{v})$  之间出现差距。幸运的是，在进入下一个循环时，E 步把这种差距又降到了 0。

EM 算法还包含一些不同的见解。首先，它包含了学习过程的一个基本框架，就是我们通过更新模型参数来提高整个数据集的似然，其中缺失变量的值是通过后验分布来估计的。这种特定的性质并非 EM 算法独有的。例如，使用梯度下降来最大化对数似然函数的方法也有相同的性质。计算对数似然函数的梯度需要对隐藏单元的

后验分布求期望。EM 算法另一个关键的性质是当我们移动到另一个  $\theta$  时候，我们仍然可以使用旧的分布  $q$ 。在传统机器学习中，这种特有的性质在推导大 M 步更新时候得到了广泛的应用。在深度学习中，大多数模型太过于复杂以至于在最优大 M 步更新中很难得到一个简单的解。所以 EM 算法的第二个特质，更多为其所独有，较少被使用。

### 19.3 最大后验推断和稀疏编码

我们通常使用 **推断** (inference) 这个术语来指代给定一些其他变量的情况下计算某些变量概率分布的过程。当训练带有潜变量的概率模型时，我们通常关注于计算  $p(\mathbf{h} | \mathbf{v})$ 。另一种可选的推断形式是计算一个缺失变量的最可能值来代替在所有可能值的完整分布上的推断。在潜变量模型中，这意味着计算

$$\mathbf{h}^* = \arg \max_{\mathbf{h}} p(\mathbf{h} | \mathbf{v}). \quad (19.9)$$

这被称作 **最大后验** (Maximum A Posteriori) 推断，简称 MAP 推断。

MAP 推断并不被视作是一种近似推断，它只是精确地计算了最有可能的一个  $\mathbf{h}^*$ 。然而，如果我们希望设计一个最大化  $\mathcal{L}(\mathbf{v}, \mathbf{h}, q)$  的学习过程，那么把 MAP 推断视作是输出一个  $q$  值的学习过程是很有帮助的。在这种情况下，我们可以将 MAP 推断视作是近似推断，因为它并不能提供一个最优的  $q$ 。

我们回过头来看看第 19.1 节中所描述的精确推断，它指的是关于一个在无限制的概率分布族中的分布  $q$  使用精确的优化算法来最大化

$$\mathcal{L}(\mathbf{v}, \theta, q) = \mathbb{E}_{\mathbf{h} \sim q} [\log p(\mathbf{h}, \mathbf{v})] + H(q). \quad (19.10)$$

我们通过限定分布  $q$  属于某个分布族，能够使得 MAP 推断成为一种形式的近似推断。具体地说，我们令分布  $q$  满足一个 Dirac 分布：

$$q(\mathbf{h} | \mathbf{v}) = \delta(\mathbf{h} - \boldsymbol{\mu}). \quad (19.11)$$

这也意味着现在我们可以通过  $\boldsymbol{\mu}$  来完全控制分布  $q$ 。将  $\mathcal{L}$  中不随  $\boldsymbol{\mu}$  变化的项丢弃，我们只需解决一个优化问题：

$$\boldsymbol{\mu}^* = \arg \max_{\boldsymbol{\mu}} \log p(\mathbf{h} = \boldsymbol{\mu}, \mathbf{v}), \quad (19.12)$$

这等价于 MAP 推断问题

$$\mathbf{h}^* = \arg \max_{\mathbf{h}} p(\mathbf{h} \mid \mathbf{v}). \quad (19.13)$$

因此我们能够证明一种类似于 EM 算法的学习算法，其中我们轮流迭代两步，一步是用 MAP 推断估计出  $\mathbf{h}^*$ ，另一步是更新  $\boldsymbol{\theta}$  来增大  $\log p(\mathbf{h}^*, \mathbf{v})$ 。从 EM 算法角度看，这也是对  $\mathcal{L}$  的一种形式的坐标上升，交替迭代时通过推断来优化关于  $q$  的  $\mathcal{L}$  以及通过参数更新来优化关于  $\boldsymbol{\theta}$  的  $\mathcal{L}$ 。作为一个整体，这个算法的正确性可以得到保证，因为  $\mathcal{L}$  是  $\log p(\mathbf{v})$  的下界。在 MAP 推断中，这个保证是无效的，因为 Dirac 分布的微分熵趋近于负无穷，使得这个界会无限地松。然而，人为加入一些  $\mu$  的噪声会使得这个界又有了意义。

MAP 推断作为特征提取器以及一种学习机制被广泛地应用在了深度学习中。它主要用于稀疏编码模型中。

我们回过头来看第 13.4 节中的稀疏编码，稀疏编码是一种在隐藏单元上加上了诱导稀疏性的先验知识的线性因子模型。一个常用的选择是可分解的 Laplace 先验，表示为

$$p(h_i) = \frac{\lambda}{2} \exp(-\lambda|h_i|). \quad (19.14)$$

可见的节点是由一个线性变化加上噪声生成的：

$$p(\mathbf{v} \mid \mathbf{h}) = \mathcal{N}(\mathbf{v}; \mathbf{W}\mathbf{h} + \mathbf{b}, \beta^{-1}\mathbf{I}). \quad (19.15)$$

分布  $p(\mathbf{h} \mid \mathbf{v})$  难以计算，甚至难以表达。每一对  $h_i, h_j$  变量都是  $\mathbf{v}$  的母节点。这也意味着当  $\mathbf{v}$  可被观察时，图模型包含了一条连接  $h_i$  和  $h_j$  的活跃路径。因此  $p(\mathbf{h} \mid \mathbf{v})$  中所有的隐藏单元都包含在了一个巨大的团中。如果是高斯模型，那么这些相互作用关系可以通过协方差矩阵来高效地建模。然而稀疏型先验使得这些相互作用关系并不服从高斯分布。

分布  $p(\mathbf{x} \mid \mathbf{h})$  的难处理性导致了对数似然及其梯度也很难得到。因此我们不能使用精确的最大似然估计来进行学习。取而代之的是，我们通过 MAP 推断以及最大化由以  $\mathbf{h}$  为中心的 Dirac 分布所定义而成的 ELBO 来学习模型参数。

如果我们将训练集中所有的向量  $\mathbf{h}$  拼成矩阵  $\mathbf{H}$ ，并将所有的向量  $\mathbf{v}$  拼起来组成矩阵  $\mathbf{V}$ ，那么稀疏编码问题意味着最小化

$$J(\mathbf{H}, \mathbf{W}) = \sum_{i,j} |H_{i,j}| + \sum_{i,j} \left( \mathbf{V} - \mathbf{HW}^\top \right)_{i,j}^2. \quad (19.16)$$

为了避免如极端小的  $\mathbf{H}$  和极端大的  $\mathbf{W}$  这样的病态的解，大多数稀疏编码的应用包含了权重衰减或者对  $\mathbf{H}$  列范数的限制。

我们可以通过交替迭代，分别关于  $\mathbf{H}$  和  $\mathbf{W}$  最小化  $J$  的方式来最小化  $J$ 。且两个子问题都是凸的。事实上，关于  $\mathbf{W}$  的最小化问题就是一个线性回归问题。然而关于这两个变量同时最小化  $J$  的问题通常并不是凸的。

关于  $\mathbf{H}$  的最小化问题需要某些特别设计的算法，例如特征符号搜索方法 (Lee *et al.*, 2007)。

## 19.4 变分推断和变分学习

我们已经说明过了为什么证据下界  $\mathcal{L}(\mathbf{v}; \boldsymbol{\theta}, q)$  是  $\log p(\mathbf{v}; \boldsymbol{\theta})$  的一个下界、如何将推断看作是关于分布  $q$  最大化  $\mathcal{L}$  的过程以及如何将学习看作是关于参数  $\boldsymbol{\theta}$  最大化  $\mathcal{L}$  的过程。我们也讲到了 EM 算法在给定了分布  $q$  的条件下能够进行大学习步骤，而基于 MAP 推断的学习算法则是学习一个  $p(\mathbf{h} | \mathbf{v})$  的点估计而非推断整个完整的分布。在这里我们介绍一些变分学习中更加通用的算法。

变分学习的核心思想就是在一个关于  $q$  的有约束的分布族上最大化  $\mathcal{L}$ 。选择这个分布族时应该考虑到计算  $\mathbb{E}_q \log p(\mathbf{h}, \mathbf{v})$  的难易度。一个典型的方法就是添加分布  $q$  如何分解的假设。

一种常用的变分学习的方法是加入一些限制使得  $q$  是一个因子分布：

$$q(\mathbf{h} | \mathbf{v}) = \prod_i q(h_i | \mathbf{v}). \quad (19.17)$$

这被称为 **均值场** (mean-field) 方法。更一般地说，我们可以通过选择分布  $q$  的形式来选择任何图模型的结构，通过选择变量之间相互作用的多少来灵活地决定近似程度的大小。这种完全通用的图模型方法被称为 **结构化变分推断** (structured variational inference) (Saul and Jordan, 1996)。

变分方法的优点是我们不需要为分布  $q$  设定一个特定的参数化形式。我们设定它如何分解，之后通过解决优化问题来找出在这些分解限制下最优的概率分布。对离散型潜变量来说，这意味着我们使用传统的优化技巧来优化描述分布  $q$  的有限个变量。对连续型潜变量来说，这意味着我们使用一个被称为变分法的数学分支工具来解决函数空间上的优化问题。然后决定哪一个函数来表示分布  $q$ 。变分法是“变分学习”或者“变分推断”这些名字的来因，尽管当潜变量是离散时变分法并没有用武

之地。当遇到连续型潜变量时，变分法不需要过多地人工选择模型，是一种很有用的工具。我们只需要设定分布  $q$  如何分解，而不需要去猜测一个特定的能够精确近似原后验分布的分布  $q$ 。

因为  $\mathcal{L}(\mathbf{v}; \boldsymbol{\theta}, q)$  被定义成  $\log p(\mathbf{v}; \boldsymbol{\theta}) - D_{\text{KL}}(q(\mathbf{h} | \mathbf{v}) \| p(\mathbf{h} | \mathbf{v}; \boldsymbol{\theta}))$ ，我们可以认为关于  $q$  最大化  $\mathcal{L}$  的问题等价于（关于  $q$ ）最小化  $D_{\text{KL}}(q(\mathbf{h} | \mathbf{v}) \| p(\mathbf{h} | \mathbf{v}))$ 。在这种情况下，我们要用  $q$  来拟合  $p$ 。然而，与以前方法不同，我们使用 KL 散度的相反方向来拟合一个近似。当我们使用最大似然估计来用模型拟合数据时，我们最小化  $D_{\text{KL}}(p_{\text{data}} \| p_{\text{model}})$ 。如图 3.6 所示，这意味着最大似然鼓励模型在每一个数据达到高概率的地方达到高概率，而基于优化的推断则鼓励了  $q$  在每一个真实后验分布概率低的地方概率较小。这两种基于 KL 散度的方法都有各自的优点与缺点。选择哪一种方法取决于在具体每一个应用中哪一种性质更受偏好。在基于优化的推断问题中，从计算角度考虑，我们选择使用  $D_{\text{KL}}(q(\mathbf{h} | \mathbf{v}) \| p(\mathbf{h} | \mathbf{v}))$ 。具体地说，计算  $D_{\text{KL}}(q(\mathbf{h} | \mathbf{v}) \| p(\mathbf{h} | \mathbf{v}))$  涉及到了计算分布  $q$  下的期望。所以通过将分布  $q$  设计得较为简单，我们可以简化求所需要的期望的计算过程。KL 散度的相反方向需要计算真实后验分布下的期望。因为真实后验分布的形式是由模型的选择决定的，所以我们不能设计出一种能够精确计算  $D_{\text{KL}}(p(\mathbf{h} | \mathbf{v}) \| q(\mathbf{h} | \mathbf{v}))$  的开销较小的方法。

### 19.4.1 离散型潜变量

关于离散型潜变量的变分推断相对来说比较直接。我们定义一个分布  $q$ ，通常分布  $q$  的每个因子都由一些离散状态的可查询表格定义。在最简单的情况下， $\mathbf{h}$  是二值的并且我们做了均值场假定，分布  $q$  可以根据每一个  $h_i$  分解。在这种情况下，我们可以用一个向量  $\hat{\mathbf{h}}$  来参数化分布  $q$ ， $\hat{\mathbf{h}}$  的每一个元素都代表一个概率，即  $q(h_i = 1 | \mathbf{v}) = \hat{h}_i$ 。

在确定了如何表示分布  $q$  以后，我们只需要优化它的参数。在离散型潜变量模型中，这是一个标准的优化问题。基本上分布  $q$  的选择可以通过任何优化算法解决，比如梯度下降算法。

因为它在许多学习算法的内循环中出现，所以这个优化问题必须可以很快求解。为了追求速度，我们通常使用特殊设计的优化算法。这些算法通常能够在极少的循环内解决一些小而简单的问题。一个常见的选择是使用不动点方程，换句话说，就是解关于  $\hat{h}_i$  的方程

$$\frac{\partial}{\partial \hat{h}_i} \mathcal{L} = 0. \quad (19.18)$$

我们反复地更新  $\hat{\mathbf{h}}$  不同的元素直到满足收敛准则。

为了具体化这些描述，我们接下来会讲如何将变分推断应用到二值稀疏编码 (binary sparse coding) 模型（这里我们所描述的模型是 Henniges *et al.* (2010) 提出的，但是我们采用了传统、通用的均值场方法，而原文作者采用了一种特殊设计的算法）中。数学推导过程非常详细，为希望完全了解我们描述过的变分推断和变分学习高级概念描述的读者所准备。而对于并不计划推导或者实现变分学习算法的读者来说，可以放心跳过，直接阅读下一节，这并不会遗漏新的高级概念。建议那些从事二值稀疏编码研究的读者可以重新看一下第 3.10 节中描述的一些经常在概率模型中出现的有用的函数性质。我们在推导过程中随意地使用了这些性质，并没有特别强调它们。

在二值稀疏编码模型中，输入  $\mathbf{v} \in \mathbb{R}^n$ ，是由模型通过添加高斯噪声到  $m$  个或有或无的不同成分的和而生成的。每一个成分可以是开或者关的，对应着隐藏单元  $\mathbf{h} \in \{0, 1\}^m$ :

$$p(h_i = 1) = \sigma(b_i), \quad (19.19)$$

$$p(\mathbf{v} | \mathbf{h}) = \mathcal{N}(\mathbf{v}; \mathbf{W}\mathbf{h}, \boldsymbol{\beta}^{-1}), \quad (19.20)$$

其中  $\mathbf{b}$  是一个可以学习的偏置集合， $\mathbf{W}$  是一个可以学习的权值矩阵， $\boldsymbol{\beta}$  是一个可以学习的对角精度矩阵。

使用最大似然来训练这样一个模型需要对参数进行求导。我们考虑对其中一

个偏置进行求导的过程：

$$\frac{\partial}{\partial b_i} \log p(\mathbf{v}) \quad (19.21)$$

$$= \frac{\frac{\partial}{\partial b_i} p(\mathbf{v})}{p(\mathbf{v})} \quad (19.22)$$

$$= \frac{\frac{\partial}{\partial b_i} \sum_{\mathbf{h}} p(\mathbf{h}, \mathbf{v})}{p(\mathbf{v})} \quad (19.23)$$

$$= \frac{\frac{\partial}{\partial b_i} \sum_{\mathbf{h}} p(\mathbf{h}) p(\mathbf{v} | \mathbf{h})}{p(\mathbf{v})} \quad (19.24)$$

$$= \frac{\sum_{\mathbf{h}} p(\mathbf{v} | \mathbf{h}) \frac{\partial}{\partial b_i} p(\mathbf{h})}{p(\mathbf{v})} \quad (19.25)$$

$$= \sum_{\mathbf{h}} p(\mathbf{h} | \mathbf{v}) \frac{\partial}{\partial b_i} p(\mathbf{h}) \quad (19.26)$$

$$= \mathbb{E}_{\mathbf{h} \sim p(\mathbf{h} | \mathbf{v})} \frac{\partial}{\partial b_i} \log p(\mathbf{h}). \quad (19.27)$$

这需要计算  $p(\mathbf{h} | \mathbf{v})$  下的期望。不幸的是， $p(\mathbf{h} | \mathbf{v})$  是一个很复杂的分布。关于  $p(\mathbf{h}, \mathbf{v})$  和  $p(\mathbf{h} | \mathbf{v})$  的图结构可以参考图 19.2。隐藏单元的后验分布对应的是关于隐藏单元的完全图，所以相对于暴力算法，变量消去算法并不能有助于提高计算期望的效率。

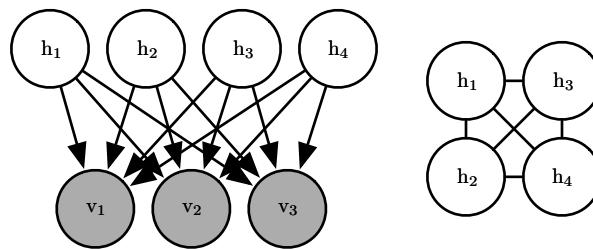


图 19.2：包含四个隐藏单元的二值稀疏编码的图结构。(左)  $p(\mathbf{h}, \mathbf{v})$  的图结构。要注意边是有向的，每两个隐藏单元都是每个可见单元的共父。(右)  $p(\mathbf{h}, \mathbf{v})$  的图结构。为了解释共父之间的活跃路径，后验分布所有隐藏单元之间都有边。

取而代之的是，我们可以应用变分推断和变分学习来解决这个难点。

我们可以做一个均值场近似：

$$q(\mathbf{h} | \mathbf{v}) = \prod_i q(h_i | \mathbf{v}). \quad (19.28)$$

二值稀疏编码中的潜变量是二值的，所以为了表示可分解的  $q$  我们假设对  $m$  个 Bernoulli 分布  $q(h_i | \mathbf{v})$  建模。表示 Bernoulli 分布的一种很自然的方法是使用一个概率向量  $\hat{\mathbf{h}}$ ，满足  $q(h_i | \mathbf{v}) = \hat{h}_i$ 。为了避免计算中的误差，比如说计算  $\log \hat{h}_i$  时，我们对  $\hat{h}_i$  添加一个约束，即  $\hat{h}_i$  不等于 0 或者 1。

我们将会看到变分推断方程理论上永远不会赋予  $\hat{h}_i$  0 或者 1。然而在软件实现过程中，机器的舍入误差会导致 0 或者 1 的值。在二值稀疏编码的软件实现中，我们希望使用一个没有限制的变分参数向量  $\mathbf{z}$  以及通过关系  $\hat{\mathbf{h}} = \sigma(\mathbf{z})$  来获得  $\mathbf{h}$ 。因此通过使用等式  $\log \sigma(z_i) = -\zeta(-z_i)$  来建立 sigmoid 函数和 softplus 函数的关系，我们可以放心地在计算机上计算  $\log \hat{h}_i$ 。

在开始二值稀疏编码模型中变分学习的推导时，我们首先说明了均值场近似的使用可以使得学习过程更加简单。

证据下界可以表示为

$$\mathcal{L}(\mathbf{v}, \boldsymbol{\theta}, q) \quad (19.29)$$

$$= \mathbb{E}_{\mathbf{h} \sim q} [\log p(\mathbf{h}, \mathbf{v})] + H(q) \quad (19.30)$$

$$= \mathbb{E}_{\mathbf{h} \sim q} [\log p(\mathbf{h}) + \log p(\mathbf{v} | \mathbf{h}) - \log q(\mathbf{h} | \mathbf{v})] \quad (19.31)$$

$$= \mathbb{E}_{\mathbf{h} \sim q} \left[ \sum_{i=1}^m \log p(h_i) + \sum_{i=1}^n \log p(v_i | \mathbf{h}) - \sum_{i=1}^m \log q(h_i | \mathbf{v}) \right] \quad (19.32)$$

$$= \sum_{i=1}^m \left[ \hat{h}_i (\log \sigma(b_i) - \log \hat{h}_i) + (1 - \hat{h}_i) (\log \sigma(-b_i) - \log(1 - \hat{h}_i)) \right] \quad (19.33)$$

$$+ \mathbb{E}_{\mathbf{h} \sim q} \left[ \sum_{i=1}^n \log \sqrt{\frac{\beta_i}{2\pi}} \exp\left(-\frac{\beta_i}{2}(v_i - \mathbf{W}_{i,:}\mathbf{h})^2\right) \right] \quad (19.34)$$

$$= \sum_{i=1}^m \left[ \hat{h}_i (\log \sigma(b_i) - \log \hat{h}_i) + (1 - \hat{h}_i) (\log \sigma(-b_i) - \log(1 - \hat{h}_i)) \right] \quad (19.35)$$

$$+ \frac{1}{2} \sum_{i=1}^n \left[ \log \frac{\beta_i}{2\pi} - \beta_i \left( v_i^2 - 2v_i \mathbf{W}_{i,:}\hat{\mathbf{h}} + \sum_j \left[ W_{i,j}^2 \hat{h}_j + \sum_{k \neq j} W_{i,j} W_{i,k} \hat{h}_j \hat{h}_k \right] \right) \right]. \quad (19.36)$$

尽管这些方程从美学观点来看有些不尽如人意。他们展示了  $\mathcal{L}$  可以被表示为少量简单的代数运算。因此证据下界  $\mathcal{L}$  是易于处理的。我们可以把  $\mathcal{L}$  看作是难以处理的对数似然函数的一个替代。

原则上说，我们可以使用关于  $\mathbf{v}$  和  $\mathbf{h}$  的梯度上升。这会成为一个推断和学习算

法的完美组合。但是，由于两个原因，我们往往不这么做。第一点，对每一个  $v$  我们需要存储  $\hat{h}$ 。我们通常更加偏向于那些不需要为每一个样本都准备内存的算法。如果我们需要为每一个样本都存储一个动态更新的向量，使得算法很难处理几十亿的样本。第二个原因就是为了能够识别  $v$  的内容，我们希望能够有能力快速提取特征  $\hat{h}$ 。在实际应用场景中，我们需要在有限时间内计算出  $\hat{h}$ 。

由于以上两个原因，我们通常不会采用梯度下降来计算均值场参数  $\hat{h}$ 。取而代之的是，我们使用不动点方程来快速估计。

不动点方程的核心思想是我们寻找一个关于  $h$  的局部极大点，满足  $\nabla_h \mathcal{L}(v, \theta, \hat{h}) = 0$ 。我们无法同时高效地计算所有  $\hat{h}$  的元素。然而，我们可以解决单个变量的问题：

$$\frac{\partial}{\partial \hat{h}_i} \mathcal{L}(v, \theta, \hat{h}) = 0. \quad (19.37)$$

我们可以迭代地将这个解应用到  $i = 1, \dots, m$ ，然后重复这个循环直到我们满足了收敛准则。常见的收敛准则包含了当整个循环所改进的  $\mathcal{L}$  不超过预设的容差量时停止，或者是循环中改变的  $\hat{h}$  不超过某个值时停止。

在很多不同的模型中，迭代的均值场不动点方程是一种能够提供快速变分推断的通用算法。为了使它更加具体，我们详细地讲一下如何推导出二值稀疏编码模型的更新过程。

首先，我们给出了对  $\hat{h}_i$  的导数表达式。为了得到这个表达式，我们将式(19.36)代

入到式(19.37)的左边：

$$\frac{\partial}{\partial \hat{h}_i} \mathcal{L}(\mathbf{v}, \boldsymbol{\theta}, \hat{\mathbf{h}}) \quad (19.38)$$

$$= \frac{\partial}{\partial \hat{h}_i} \left[ \sum_{j=1}^m \left[ \hat{h}_j (\log \sigma(b_j) - \log \hat{h}_j) + (1 - \hat{h}_j) (\log \sigma(-b_j) - \log(1 - \hat{h}_j)) \right] \right] \quad (19.39)$$

$$+ \frac{1}{2} \sum_{j=1}^n \left[ \log \frac{\beta_j}{2\pi} - \beta_j \left( v_j^2 - 2v_j \mathbf{W}_{j,:} \hat{\mathbf{h}} + \sum_k \left[ W_{j,k}^2 \hat{h}_k + \sum_{l \neq k} W_{j,k} W_{j,l} \hat{h}_k \hat{h}_l \right] \right) \right] \quad (19.40)$$

$$= \log \sigma(b_i) - \log \hat{h}_i - 1 + \log(1 - \hat{h}_i) + 1 - \log \sigma(-b_i) \quad (19.41)$$

$$+ \sum_{j=1}^n \left[ \beta_j \left( v_j W_{j,i} - \frac{1}{2} W_{j,i}^2 - \sum_{k \neq i} \mathbf{W}_{j,k} \mathbf{W}_{j,i} \hat{h}_k \right) \right] \quad (19.42)$$

$$= b_i - \log \hat{h}_i + \log(1 - \hat{h}_i) + \mathbf{v}^\top \boldsymbol{\beta} \mathbf{W}_{:,i} - \frac{1}{2} \mathbf{W}_{:,i}^\top \boldsymbol{\beta} \mathbf{W}_{:,i} - \sum_{j \neq i} \mathbf{W}_{:,j}^\top \boldsymbol{\beta} \mathbf{W}_{:,i} \hat{h}_j. \quad (19.43)$$

为了应用固定点更新的推断规则，我们通过令式(19.43)等于0来解 $\hat{h}_i$ ：

$$\hat{h}_i = \sigma \left( b_i + \mathbf{v}^\top \boldsymbol{\beta} \mathbf{W}_{:,i} - \frac{1}{2} \mathbf{W}_{:,i}^\top \boldsymbol{\beta} \mathbf{W}_{:,i} - \sum_{j \neq i} \mathbf{W}_{:,j}^\top \boldsymbol{\beta} \mathbf{W}_{:,i} \hat{h}_j \right). \quad (19.44)$$

此时，我们可以发现图模型中的推断和循环神经网络之间存在着紧密的联系。具体地说，均值场不动点方程定义了一个循环神经网络。这个神经网络的任务就是完成推断。我们已经从模型描述的角度介绍了如何推导这个网络，但是直接训练这个推断网络也是可行的。有关这种思路的一些想法在第二十章中有所描述。

在二值稀疏编码模型中，我们可以发现式(19.44)中描述的循环网络连接包含了根据相邻隐藏单元变化值来反复更新当前隐藏单元的操作。输入层通常给隐藏单元发送一个固定的信息 $\mathbf{v}^\top \boldsymbol{\beta} \mathbf{W}$ ，然而隐藏单元不断地更新互相传送的信息。具体地说，当 $\hat{h}_i$ 和 $\hat{h}_j$ 两个单元的权重向量平行时，它们会互相抑制。这也是一种形式的竞争——两个解释输入的隐藏单元之间，只有一个解释得更好的才被允许继续保持活跃。在二值稀疏编码的后验分布中，均值场近似试图捕获到更多的相消解释相互作用，从而产生了这种竞争。事实上，相消解释效应会产生一个多峰值的后验分布，以至于如果我们从后验分布中采样，一些样本在一个单元是活跃的，其他的样本在另一个单元活跃，只有很少的样本能够两者都处于活跃状态。不幸的是，相消解释作

用无法通过均值场中因子分布  $q$  来建模，因此建模时均值场近似只能选择一个峰值。这个现象的一个例子可以参考图 3.6。

我们将式 (19.44) 重写成等价的形式来揭示一些深层的含义：

$$\hat{h}_i = \sigma \left( b_i + \left( \mathbf{v} - \sum_{j \neq i} \mathbf{W}_{:,j} \hat{h}_j \right)^{\top} \boldsymbol{\beta} \mathbf{W}_{:,i} - \frac{1}{2} \mathbf{W}_{:,i}^{\top} \boldsymbol{\beta} \mathbf{W}_{:,i} \right). \quad (19.45)$$

在这种新的形式中，我们可以将  $\mathbf{v} - \sum_{j \neq i} \mathbf{W}_{:,j} \hat{h}_j$  看作是输入，而不是  $\mathbf{v}$ 。因此，我们可以把第  $i$  个单元视作给定其他单元编码时给  $\mathbf{v}$  中的剩余误差编码。由此我们可以将稀疏编码视作是一个迭代的自编码器，将输入反复地编码解码，试图在每一轮迭代后都能修复重构中的误差。

在这个例子中，我们已经推导出了每一次更新单个结点的更新规则。如果能够同时更新更多的结点，那会更令人满意。某些图模型，比如深度玻尔兹曼机，我们可以同时解出  $\hat{h}$  中的许多元素。不幸的是，二值稀疏编码并不适用这种块更新。取而代之的是，我们使用一种被称为衰减 (damping) 的启发式技巧来实现块更新。在衰减方法中，对  $\hat{h}$  中的每一个元素我们都可以解出最优值，然后对于所有的值都在这个方向上移动一小步。这个方法不能保证每一步都能增加  $\mathcal{L}$ ，但是对于许多模型都很有效。关于在信息传输算法中如何选择同步程度以及使用衰减策略可以参考 Koller and Friedman (2009)。

## 19.4.2 变分法

在继续介绍变分学习之前，我们有必要简单地介绍一种变分学习中重要的数学工具：变分法 (calculus of variations)。

许多机器学习的技巧是基于寻找一个输入向量  $\boldsymbol{\theta} \in \mathbb{R}^n$  来最小化函数  $J(\boldsymbol{\theta})$ ，使得它取到最小值。这个步骤可以利用多元微积分以及线性代数的知识找到满足  $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = 0$  的临界点来完成。在某些情况下，我们希望能够解一个函数  $f(\mathbf{x})$ ，比如当我们希望找到一些随机变量的概率密度函数时。正是变分法能够让我们完成这个目标。

函数  $f$  的函数被称为泛函 (functional)  $J[f]$ 。正如我们许多情况下对一个函数求关于以向量的元素为变量的偏导数一样，我们可以使用泛函导数 (functional derivative)，即在任意特定的  $\mathbf{x}$  值，对一个泛函  $J[f]$  求关于函数  $f(\mathbf{x})$  的导数，这也被称为变分导数 (variational derivative)。泛函  $J$  的关于函数  $f$  在点  $\mathbf{x}$  处的泛函

导数被记作  $\frac{\delta}{\delta f(\mathbf{x})} J$ 。

完整正式的泛函导数的推导不在本书的范围之内。对于我们的目标而言，了解可微分函数  $f(\mathbf{x})$  以及带有连续导数的可微分函数  $g(y, \mathbf{x})$  就足够了：

$$\frac{\delta}{\delta f(\mathbf{x})} \int g(f(\mathbf{x}), \mathbf{x}) d\mathbf{x} = \frac{\partial}{\partial y} g(f(\mathbf{x}), \mathbf{x}). \quad (19.46)$$

为了使上述等式更加直观，我们可以把  $f(\mathbf{x})$  看作是一个有着无穷不可数多元素的向量，由一个实数向量  $\mathbf{x}$  表示。在这里（看作是一个不完全的介绍），这种关系式中描述的泛函导数和向量  $\theta \in \mathbb{R}^n$  的导数相同：

$$\frac{\partial}{\partial \theta_i} \sum_j g(\theta_j, j) = \frac{\partial}{\partial \theta_i} g(\theta_i, i). \quad (19.47)$$

在其他机器学习文献中的许多结果则使用了更为通用的 欧拉-拉格朗日方程 (Euler-Lagrange Equation)，它能够使得  $g$  不仅依赖于  $f$  的值，还依赖于  $f$  的导数。但是在本书中我们不需要这个通用版本。

为了关于一个向量优化某个函数，我们求出了这个函数关于这个向量的梯度，然后找这个梯度中每一个元素都为 0 的点。类似地，我们可以通过寻找一个函数使得泛函导数的每个点都等于 0 从而来优化一个泛函。

下面介绍一个该过程如何运行的例子，我们考虑寻找一个定义在  $x \in \mathbb{R}$  上的有最大微分熵的概率密度函数。我们回过头来看一下一个概率分布  $p(x)$  的熵，定义如下：

$$H[p] = -\mathbb{E}_x \log p(x). \quad (19.48)$$

对于连续的值，这个期望可以被看作一个积分：

$$H[p] = - \int p(x) \log p(x) dx. \quad (19.49)$$

我们不能简单地仅仅关于函数  $p(x)$  最大化  $H[p]$ ，因为那样的话结果可能不是一个概率分布。为了解决这个问题，我们需要使用一个拉格朗日乘子来添加一个分布  $p(x)$  积分值为 1 的约束。同样地，当方差增大时，熵也会无限制地增加。因此，寻找哪一个分布有最大熵这个问题是没有意义的。但是，在给定固定的方差  $\sigma^2$  时，我们可以寻找一个最大熵的分布。最后，这个问题还是欠定的，因为在不改变熵的条件下一个分布可以被随意地改变。为了获得一个唯一的解，我们再加一个约束：分

布的均值必须为  $\mu$ 。那么这个问题的拉格朗日泛函如下：

$$\mathcal{L}[p] = \lambda_1 \left( \int p(x) dx - 1 \right) + \lambda_2 (\mathbb{E}[x] - \mu) + \lambda_3 (\mathbb{E}[(x - \mu)^2] - \sigma^2) + H[p] \quad (19.50)$$

$$= \int \left( \lambda_1 p(x) + \lambda_2 p(x)x + \lambda_3 p(x)(x - \mu)^2 - p(x) \log p(x) \right) dx - \lambda_1 - \mu \lambda_2 - \sigma^2 \lambda_3. \quad (19.51)$$

为了关于  $p$  最小化拉格朗日乘子，我们令泛函导数等于 0：

$$\forall x, \frac{\delta}{\delta p(x)} \mathcal{L} = \lambda_1 + \lambda_2 x + \lambda_3 (x - \mu)^2 - 1 - \log p(x) = 0. \quad (19.52)$$

这个条件告诉我们  $p(x)$  的泛函形式。通过代数运算重组上述方程，我们可以得到

$$p(x) = \exp(\lambda_1 + \lambda_2 x + \lambda_3 (x - \mu)^2 - 1). \quad (19.53)$$

我们并没有直接假设  $p(x)$  取这种形式，而是通过最小化泛函从理论上得到了这个  $p(x)$  的表达式。为了解决这个最小化问题，我们需要选择  $\lambda$  的值来确保所有的约束都能够满足。我们有很大的自由去选择  $\lambda$ 。因为只要满足约束，拉格朗日关于  $\lambda$  这个变量的梯度就为 0。为了满足所有的约束，我们可以令  $\lambda_1 = 1 - \log \sigma \sqrt{2\pi}$ ,  $\lambda_2 = 0$ ,  $\lambda_3 = -\frac{1}{2\sigma^2}$ ，从而得到

$$p(x) = \mathcal{N}(x; \mu, \sigma^2). \quad (19.54)$$

这也是当我们不知道真实的分布时总是使用正态分布的一个原因。因为正态分布拥有最大的熵，我们通过这个假定来保证了最小可能量的结构。

当寻找熵的拉格朗日泛函的临界点并且给定一个固定的方差时，我们只能找到一个对应最大熵的临界点。那最小化熵的概率密度函数是什么样的呢？为什么我们无法发现对应着极小点的第二个临界点呢？原因是没有一个特定的函数能够达到最小的熵值。当函数把越多的概率密度加到  $x = \mu + \sigma$  和  $x = \mu - \sigma$  两个点上，越少的概率密度到其他点上时，它们的熵值会减少，而方差却不变。然而任何把所有的权重都放在这两点的函数的积分都不为 1，不是一个有效的概率分布。所以不存在一个最小熵的概率密度函数，就像不存在一个最小的正实数一样。然而，我们发现存在一个收敛的概率分布的序列，收敛到权重都在两个点上。这种情况能够退化为混合 Dirac 分布。因为 Dirac 分布并不是一个单独的概率密度函数，所以 Dirac 分布或

者混合 Dirac 分布并不能对应函数空间的一个点。所以对我们来说，当寻找一个泛函导数为 0 的函数空间的点时，这些分布是不可见的。这就是这种方法的局限之处。诸如 Dirac 分布这样的分布可以通过其他方法被找到，比如可以先猜测一个解，然后证明它是满足条件的。

### 19.4.3 连续型潜变量

当我们的图模型包含连续型潜变量时，我们仍然可以通过最大化  $\mathcal{L}$  进行变分推断和变分学习。然而，我们需要使用变分法来实现关于  $q(\mathbf{h} \mid \mathbf{v})$  最大化  $\mathcal{L}$ 。

在大多数情况下，研究者并不需要解决任何变分法的问题。取而代之的是，均值场固定点迭代更新有一个通用的方程。如果我们做了均值场近似：

$$q(\mathbf{h} \mid \mathbf{v}) = \prod_i q(h_i \mid \mathbf{v}), \quad (19.55)$$

并且对任何的  $j \neq i$  固定  $q(h_j \mid \mathbf{v})$ ，那么只需要满足分布  $p$  中任何联合分布变量的概率值不为 0，我们就可以通过归一化下面这个未归一的分布

$$\tilde{q}(h_i \mid \mathbf{v}) = \exp(\mathbb{E}_{\mathbf{h}_{-i} \sim q(\mathbf{h}_{-i} \mid \mathbf{v})} \log \tilde{p}(\mathbf{v}, \mathbf{h})) \quad (19.56)$$

来得到最优的  $q(h_i \mid \mathbf{v})$ 。在这个方程中计算期望就能得到正确的  $q(h_i \mid \mathbf{v})$  的表达式。我们只有在希望提出一种新形式的变分学习算法时才需要使用变分法来直接推导  $q$  的函数形式。式 (19.56) 给出了适用于任何概率模型的均值场近似。

式 (19.56) 是一个不动点方程，对每一个  $i$  它都被迭代地反复使用直到收敛。然而，它还包含着更多的信息。它还包含了最优解取到的泛函形式，无论我们是否能够通过不动点方程来解出它。这意味着我们可以利用方程中的泛函形式，把其中一些值当成参数，然后通过任何我们想用的优化算法来解决这个问题。

我们拿一个简单的概率模型作为例子，其中潜变量满足  $\mathbf{h} \in \mathbb{R}^2$ ，可见变量只有一个  $v$ 。假设  $p(\mathbf{h}) = \mathcal{N}(\mathbf{h}; \mathbf{0}, \mathbf{I})$  以及  $p(v \mid \mathbf{h}) = \mathcal{N}(v; \mathbf{w}^\top \mathbf{h}; 1)$ ，我们可以积掉  $\mathbf{h}$  来简化这个模型，结果是关于  $v$  的高斯分布。这个模型本身并不有趣。只是为了说明变分法如何应用在概率建模之中，我们才构造了这个模型。

忽略归一化常数时，真实的后验分布如下：

$$p(\mathbf{h} \mid \mathbf{v}) \quad (19.57)$$

$$\propto p(\mathbf{h}, \mathbf{v}) \quad (19.58)$$

$$= p(h_1)p(h_2)p(\mathbf{v} \mid \mathbf{h}) \quad (19.59)$$

$$\propto \exp\left(-\frac{1}{2}[h_1^2 + h_2^2 + (v - h_1 w_1 - h_2 w_2)^2]\right) \quad (19.60)$$

$$= \exp\left(-\frac{1}{2}[h_1^2 + h_2^2 + v^2 + h_1^2 w_1^2 + h_2^2 w_2^2 - 2vh_1 w_1 - 2vh_2 w_2 + 2h_1 w_1 h_2 w_2]\right). \quad (19.61)$$

在上式中，我们发现由于带有  $h_1, h_2$  乘积项的存在，真实的后验并不能关于  $h_1, h_2$  分解。

应用式(19.56)，我们可以得到

$$\tilde{q}(h_1 \mid \mathbf{v}) \quad (19.62)$$

$$= \exp(\mathbb{E}_{h_2 \sim q(h_2 \mid \mathbf{v})} \log \tilde{p}(\mathbf{v}, \mathbf{h})) \quad (19.63)$$

$$= \exp\left(-\frac{1}{2}\mathbb{E}_{h_2 \sim q(h_2 \mid \mathbf{v})}[h_1^2 + h_2^2 + v^2 + h_1^2 w_1^2 + h_2^2 w_2^2 \quad (19.64)\right.$$

$$\left.- 2vh_1 w_1 - 2vh_2 w_2 + 2h_1 w_1 h_2 w_2]\right). \quad (19.65)$$

从这里，我们可以发现其中我们只需要从  $q(h_2 \mid \mathbf{v})$  中获得两个有效值： $\mathbb{E}_{h_2 \sim q(h_2 \mid \mathbf{v})}[h_2]$  和  $\mathbb{E}_{h_2 \sim q(h_2 \mid \mathbf{v})}[h_2^2]$ 。把这两项记作  $\langle h_2 \rangle$  和  $\langle h_2^2 \rangle$ ，我们可以得到：

$$\tilde{q}(h_1 \mid \mathbf{v}) = \exp\left(-\frac{1}{2}[h_1^2 + \langle h_2^2 \rangle + v^2 + h_1^2 w_1^2 + \langle h_2^2 \rangle w_2^2 \quad (19.66)\right.$$

$$\left.- 2vh_1 w_1 - 2v\langle h_2 \rangle w_2 + 2h_1 w_1 \langle h_2 \rangle w_2]\right). \quad (19.67)$$

从这里，我们可以发现  $\tilde{q}$  的泛函形式满足高斯分布。因此，我们可以得到  $q(\mathbf{h} \mid \mathbf{v}) = \mathcal{N}(\mathbf{h}; \boldsymbol{\mu}, \boldsymbol{\beta}^{-1})$ ，其中  $\boldsymbol{\mu}$  和对角的  $\boldsymbol{\beta}$  是变分参数，我们可以使用任何方法来优化它。有必要再强调一下，我们并没有假设  $q$  是一个高斯分布，这个高斯的形式是使用变分法来关于分布  $q$  最大化  $\mathcal{L}$  而推导出来的。在不同的模型上应用相同的方法可能会得到不同泛函形式的分布  $q$ 。

当然，上述模型只是为了说明情况的一个简单例子。深度学习中关于变分学习中连续型变量的实际应用可以参考 Goodfellow et al. (2013f)。

### 19.4.4 学习和推断之间的相互作用

在学习算法中使用近似推断会影响学习的过程，反过来学习的过程也会影响推断算法的准确性。

具体来说，训练算法倾向于朝使得近似推断算法中的近似假设变得更加真实的方向来适应模型。当训练参数时，变分学习增加

$$\mathbb{E}_{\mathbf{h} \sim q} \log p(\mathbf{v}, \mathbf{h}). \quad (19.68)$$

对于一个特定的  $\mathbf{v}$ ，对于  $q(\mathbf{h} | \mathbf{v})$  中概率很大的  $\mathbf{h}$  它增加了  $p(\mathbf{h} | \mathbf{v})$ ；对于  $q(\mathbf{h} | \mathbf{v})$  中概率很小的  $\mathbf{h}$  它减小了  $p(\mathbf{h} | \mathbf{v})$ 。

这种行为使得我们做的近似假设变得合理。如果我们用单峰值近似后验来训练模型，那么所得具有真实后验的模型会比我们使用精确推断训练模型获得的模型更接近单峰值。

因此，估计变分近似对模型的破坏程度是很困难的。存在几种估计  $\log p(\mathbf{v})$  的方式。通常我们在训练模型之后估计  $\log p(\mathbf{v}; \boldsymbol{\theta})$ ，然后发现它和  $\mathcal{L}(\mathbf{v}, \boldsymbol{\theta}, q)$  的差距是很小的。从这里我们可以得出结论，对于特定的从学习过程中获得的  $\boldsymbol{\theta}$  来说，变分近似是很准确的。然而我们无法直接得到变分近似普遍很准确或者变分近似几乎不会对学习过程产生任何负面影响这样的结论。为了准确衡量变分近似带来的危害，我们需要知道  $\boldsymbol{\theta}^* = \max_{\boldsymbol{\theta}} \log p(\mathbf{v}; \boldsymbol{\theta})$ 。 $\mathcal{L}(\mathbf{v}, \boldsymbol{\theta}, q) \approx \log p(\mathbf{v}; \boldsymbol{\theta})$  和  $\log p(\mathbf{v}; \boldsymbol{\theta}) \ll \log p(\mathbf{v}; \boldsymbol{\theta}^*)$  同时成立是有可能的。如果存在  $\max_q \mathcal{L}(\mathbf{v}, \boldsymbol{\theta}^*, q) \ll \log p(\mathbf{v}; \boldsymbol{\theta}^*)$ ，即在  $\boldsymbol{\theta}^*$  点处后验分布太过复杂使得  $q$  分布族无法准确描述，那么学习过程永远无法到达  $\boldsymbol{\theta}^*$ 。这样的一类问题是很难发现的，因为只有在我们有一个能够找到  $\boldsymbol{\theta}^*$  的较好的学习算法时，才能确定地进行上述的比较。

## 19.5 学成近似推断

我们已经看到了推断可以被视作一个增加函数  $\mathcal{L}$  值的优化过程。显式地通过迭代方法（比如不动点方程或者基于梯度的优化算法）来进行优化的过程通常是代价很高且耗时巨大的。通过学习一个近似推断，许多推断算法避免了这种代价。具体地说，我们可以将优化过程视作将一个输入  $\mathbf{v}$  投影到一个近似分布  $q^* = \arg \max_q \mathcal{L}(\mathbf{v}, q)$  的一个  $f$  的函数。一旦我们将多步的迭代优化过程看作是一个函数，我们可以用一个近似函数为  $\hat{f}(\mathbf{v}; \boldsymbol{\theta})$  的神经网络来近似它。

### 19.5.1 醒眠算法

训练一个可以用  $v$  来推断  $h$  的模型的一个主要难点在于我们没有一个监督训练集来训练模型。给定一个  $v$ , 我们无法获知一个合适的  $h$ 。从  $v$  到  $h$  的映射依赖于模型族的选择, 并且在学习过程中随着  $\theta$  的改变而变化。醒眠 (wake sleep) 算法 (Hinton *et al.*, 1995b; Frey *et al.*, 1996) 通过从模型分布中抽取  $v$  和  $h$  的样本来解决这个问题。例如, 在有向模型中, 这可以通过执行从  $h$  开始并在  $v$  结束的原始采样来高效地完成。然后这个推断网络可以被训练来执行反向的映射: 预测哪一个  $h$  产生了当前的  $v$ 。这种方法的主要缺点是我们将只能在那些在当前模型上有较高概率的  $v$  值上训练推断网络。在学习早期, 模型分布与数据分布偏差较大, 因此推断网络将不具有在类似数据的样本上学习的机会。

在第 18.2 节中, 我们看到睡眠做梦在人类和动物中作用的一个可能解释是, 做梦可以提供蒙特卡罗训练算法用于近似无向模型中对数配分函数负梯度的负相样本。生物做梦的另一个可能解释是它提供来自  $p(h, v)$  的样本, 这可以用于训练推断网络在给定  $v$  的情况下预测  $h$ 。在某些意义上, 这种解释比配分函数的解释更令人满意。如果蒙特卡罗算法仅使用梯度的正相运行几个步骤, 然后仅对梯度的负相运行几个步骤, 那么结果通常不会很好。人类和动物通常连续清醒几个小时, 然后连续睡着几个小时。这个时间表如何支持无向模型的蒙特卡罗训练尚不清楚。然而, 基于最大化  $\mathcal{L}$  的学习算法可以通过长时间调整改进  $q$  和长期调整  $\theta$  来实现。如果生物做梦的作用是训练网络来预测  $q$ , 那么这解释了动物如何能够保持清醒几个小时 (它们清醒的时间越长,  $\mathcal{L}$  和  $\log p(v)$  之间的差距越大, 但是  $\mathcal{L}$  仍然是下限) 并且睡眠几个小时 (生成模型本身在睡眠期间不被修改), 而不损害它们的内部模型。当然, 这些想法纯粹是猜测性的, 没有任何确定的证据表明做梦实现了这些目标之一。做梦也可以通过从动物的过渡模型 (用来训练动物策略) 采样合成经验来服务于强化学习而不是概率建模。也许睡眠可以服务于一些机器学习社区尚未发现的其他目的。

### 19.5.2 学成推断的其他形式

这种学成近似推断策略已经被应用到了其他模型中。Salakhutdinov and Larochelle (2010) 证明了在学成推断网络中的单遍传递相比于在深度玻尔兹曼机中的迭代均值场不动点方程能够得到更快的推断。其训练过程是基于运行推断网络的, 然后运行一步均值场来改进其估计, 并训练推断网络来输出这个更精细的估计以代替其原始估计。

我们已经在第 14.8 节中看到，预测性的稀疏分解模型训练一个浅层编码器网络，从而预测输入的稀疏编码。这可以被看作是自编码器和稀疏编码之间的混合。为模型设计概率语义是可能的，其中编码器可以被视为执行学成近似 MAP 推断。由于其浅层的编码器，PSD 不能实现我们在均值场推断中看到的单元之间的那种竞争。然而，该问题可以通过训练深度编码器实现学成近似推断来补救，如 ISTA 技术 (Gregor and LeCun, 2010b)。

近来学成近似推断已经成为了变分自编码器形式的生成模型中的主要方法之一 (Kingma, 2013; Rezende *et al.*, 2014)。在这种优美的方法中，不需要为推断网络构造显式的目标。反之，推断网络仅仅被用来定义  $\mathcal{L}$ ，然后调整推断网络的参数来增大  $\mathcal{L}$ 。我们将在第 20.10.3 节中详细介绍这种模型。

我们可以使用近似推断来训练和使用很多不同的模型。其中许多模型将在下一章中描述。

# 第二十章 深度生成模型

在本章中，我们介绍几种具体的生成模型，这些模型可以使用第十六章至第十九章中出现的技术构建和训练。所有这些模型在某种程度上都代表了多个变量的概率分布。有些模型允许显式地计算概率分布函数。其他模型则不允许直接评估概率分布函数，但支持隐式获取分布知识的操作，如从分布中采样。这些模型中的一部分使用第十六章中的图模型语言，从图和因子的角度描述为结构化概率模型。其他的不能简单地从因子角度描述，但仍然代表概率分布。

## 20.1 玻尔兹曼机

玻尔兹曼机最初作为一种广义的“联结主义”引入，用来学习二值向量上的任意概率分布 (Fahlman *et al.*, 1983; Ackley *et al.*, 1985; Hinton *et al.*, 1984b; Hinton and Sejnowski, 1986)。玻尔兹曼机的变体（包含其他类型的变量）早已超过了原始玻尔兹曼机的流行程度。在本节中，我们简要介绍二值玻尔兹曼机并讨论训练模型和进行推断时出现的问题。

我们在  $d$  维二值随机向量  $\mathbf{x} \in \{0, 1\}^d$  上定义玻尔兹曼机。玻尔兹曼机是一种基于能量的模型（第 16.2.4 节），意味着我们可以使用能量函数定义联合概率分布：

$$P(\mathbf{x}) = \frac{\exp(-E(\mathbf{x}))}{Z}, \quad (20.1)$$

其中  $E(\mathbf{x})$  是能量函数， $Z$  是确保  $\sum_x P(\mathbf{x}) = 1$  的配分函数。玻尔兹曼机的能量函数如下给出：

$$E(\mathbf{x}) = -\mathbf{x}^\top \mathbf{U}\mathbf{x} - \mathbf{b}^\top \mathbf{x}, \quad (20.2)$$

其中  $\mathbf{U}$  是模型参数的“权重”矩阵， $\mathbf{b}$  是偏置向量。

在一般设定下，给定一组训练样本，每个样本都是  $n$  维的。式 (20.1) 描述了观察到的变量的联合概率分布。虽然这种情况显然可行，但它限制了观察到的变量和权重矩阵描述的变量之间相互作用的类型。具体来说，这意味着一个单元的概率由其他单元值的线性模型（逻辑回归）给出。

当不是所有变量都能被观察到时，玻尔兹曼机变得更强大。在这种情况下，潜变量类似于多层感知机中的隐藏单元，并模拟可见单元之间的高阶交互。正如添加隐藏单元将逻辑回归转换为 MLP，导致 MLP 成为函数的万能近似器，具有隐藏单元的玻尔兹曼机不再局限于建模变量之间的线性关系。相反，玻尔兹曼机变成了离散变量上概率质量函数的万能近似器 (Le Roux and Bengio, 2008)。

正式地，我们将单元  $\mathbf{x}$  分解为两个子集：可见单元  $\mathbf{v}$  和潜在（或隐藏）单元  $\mathbf{h}$ 。能量函数变为

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{v}^\top \mathbf{R}\mathbf{v} - \mathbf{v}^\top \mathbf{W}\mathbf{h} - \mathbf{h}^\top \mathbf{S}\mathbf{h} - \mathbf{b}^\top \mathbf{v} - \mathbf{c}^\top \mathbf{h}. \quad (20.3)$$

**玻尔兹曼机的学习** 玻尔兹曼机的学习算法通常基于最大似然。所有玻尔兹曼机都具有难以处理的配分函数，因此最大似然梯度必须使用第十八章中的技术来近似。

玻尔兹曼机有一个有趣的性质，当基于最大似然的学习规则训练时，连接两个单元的特定权重的更新仅取决于这两个单元在不同分布下收集的统计信息： $P_{\text{model}}(\mathbf{v})$  和  $\hat{P}_{\text{data}}(\mathbf{v})P_{\text{model}}(\mathbf{h} \mid \mathbf{v})$ 。网络的其余部分参与塑造这些统计信息，但权重可以在完全不知道网络其余部分或这些统计信息如何产生的情况下更新。这意味着学习规则是“局部”的，这使得玻尔兹曼机的学习似乎在某种程度上是生物学合理的。我们可以设想每个神经元都是玻尔兹曼机中随机变量的情况，那么连接两个随机变量的轴突和树突只能通过观察与它们物理上实际接触细胞的激发模式来学习。特别地，正相期间，经常同时激活的两个单元之间的连接会被加强。这是 Hebbian 学习规则 (Hebb, 1949) 的一个例子，经常总结为好记的短语——“fire together, wire together”。Hebbian 学习规则是生物系统学习中最古老的假设性解释之一，直至今天仍然有重大意义 (Giudice *et al.*, 2009)。

不仅仅使用局部统计信息的其他学习算法似乎需要假设更多的学习机制。例如，对于大脑在多层感知机中实现的反向传播，似乎需要维持一个辅助通信的网络，并借此向后传输梯度信息。已经有学者 (Hinton, 2007a; Bengio, 2015) 提出生物学上可行（和近似）的反向传播实现方案，但仍然有待验证，Bengio (2015) 还将梯度的反向传播关联到类似于玻尔兹曼机（但具有连续潜变量）的能量模型中的推断。

从生物学的角度看，玻尔兹曼机学习中的负相阶段有点难以解释。正如第 18.2 节所主张的，人类在睡眠时做梦可能是一种形式的负相采样。尽管这个想法更多的只是猜测。

## 20.2 受限玻尔兹曼机

受限玻尔兹曼机以簧风琴（harmonium）之名 (Smolensky, 1986) 面世之后，成为了深度概率模型中最常见的组件之一。我们之前在第 16.7.1 节简要介绍了 RBM。在这里我们回顾以前的内容并探讨更多的细节。RBM 是包含一层可观察变量和单层潜变量的无向概率图模型。RBM 可以堆叠起来（一个在另一个的顶部）形成更深的模型。图 20.1 展示了一些例子。特别地，图 20.1 a 显示 RBM 本身的图结构。它是一个二分图，观察层或潜层中的任何单元之间不允许存在连接。

我们从二值版本的受限玻尔兹曼机开始，但如我们之后所见，这还可以扩展为其他类型的可见和隐藏单元。

更正式地说，令观察层由一组  $n_v$  个二值随机变量组成，我们统称为向量  $\mathbf{v}$ 。我们将  $n_h$  个二值随机变量的潜在或隐藏层记为  $\mathbf{h}$ 。

就像普通的玻尔兹曼机，受限玻尔兹曼机也是基于能量的模型，其联合概率分布由能量函数指定：

$$P(\mathbf{v} = \mathbf{v}, \mathbf{h} = \mathbf{h}) = \frac{1}{Z} \exp(-E(\mathbf{v}, \mathbf{h})). \quad (20.4)$$

RBM 的能量函数由下给出

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{b}^\top \mathbf{v} - \mathbf{c}^\top \mathbf{h} - \mathbf{v}^\top \mathbf{W} \mathbf{h}, \quad (20.5)$$

其中  $Z$  是被称为配分函数的归一化常数：

$$Z = \sum_{\mathbf{v}} \sum_{\mathbf{h}} \exp\{-E(\mathbf{v}, \mathbf{h})\} \quad (20.6)$$

从配分函数  $Z$  的定义显而易见，计算  $Z$  的朴素方法（对所有状态进行穷举求和）计算上可能是难以处理的，除非有巧妙设计的算法可以利用概率分布中的规则来更快地计算  $Z$ 。在受限玻尔兹曼机的情况下，Long and Servedio (2010) 正式证明配分函数  $Z$  是难解的。难解的配分函数  $Z$  意味着归一化联合概率分布  $P(\mathbf{v})$  也难以评估。

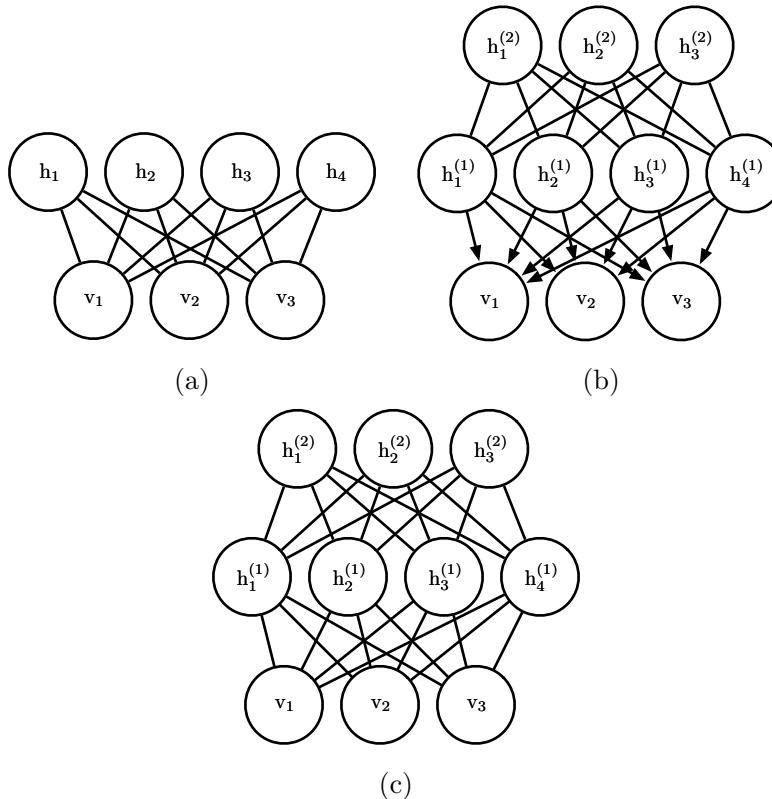


图 20.1: 可以用受限玻尔兹曼机构建的模型示例。(a)受限玻尔兹曼机本身是基于二分图的无向图模型, 在图的一部分具有可见单元, 另一部分具有隐藏单元。可见单元之间没有连接, 隐藏单元之间也没有任何连接。通常每个可见单元连接到每个隐藏单元, 但也可以构造稀疏连接的 RBM, 如卷积 RBM。(b)深度信念网络是涉及有向和无向连接的混合图模型。与 RBM 一样, 它也没有层内连接。然而, DBN 具有多个隐藏层, 因此隐藏单元之间的连接在分开的层中。深度信念网络所需的所有局部条件概率分布都直接复制 RBM 的局部条件概率分布。或者, 我们也可以用完全无向图表示深度信念网络, 但是它需要层内连接来捕获父节点间的依赖关系。(c)深度玻尔兹曼机是具有几层潜变量的无向图模型。与 RBM 和 DBN 一样, DBM 也缺少层内连接。DBM 与 RBM 的联系不如 DBN 紧密。当从 RBM 堆栈初始化 DBM 时, 有必要对 RBM 的参数稍作修改。某些种类的 DBM 可以直接训练, 而不用先训练一组 RBM。

### 20.2.1 条件分布

虽然  $P(\mathbf{v})$  难解，但 RBM 的二分图结构具有非常特殊的性质，其条件分布  $P(\mathbf{h} \mid \mathbf{v})$  和  $P(\mathbf{v} \mid \mathbf{h})$  是因子的，并且计算和采样是相对简单的。

从联合分布中导出条件分布是直观的：

$$P(\mathbf{h} \mid \mathbf{v}) = \frac{P(\mathbf{h}, \mathbf{v})}{P(\mathbf{v})} \quad (20.7)$$

$$= \frac{1}{P(\mathbf{v})} \frac{1}{Z} \exp \left\{ \mathbf{b}^\top \mathbf{v} + \mathbf{c}^\top \mathbf{h} + \mathbf{v}^\top \mathbf{W} \mathbf{h} \right\} \quad (20.8)$$

$$= \frac{1}{Z'} \exp \left\{ \mathbf{c}^\top \mathbf{h} + \mathbf{v}^\top \mathbf{W} \mathbf{h} \right\} \quad (20.9)$$

$$= \frac{1}{Z'} \exp \left\{ \sum_{j=1}^{n_h} \mathbf{c}_j^\top \mathbf{h}_j + \sum_{n_h}^{j=1} \mathbf{v}^\top \mathbf{W}_{:,j} \mathbf{h}_j \right\} \quad (20.10)$$

$$= \frac{1}{Z'} \prod_{j=1}^{n_h} \exp \left\{ \mathbf{c}_j^\top \mathbf{h}_j + \mathbf{v}^\top \mathbf{W}_{:,j} \mathbf{h}_j \right\}. \quad (20.11)$$

由于我们相对可见单元  $\mathbf{v}$  计算条件概率，相对于分布  $P(\mathbf{h} \mid \mathbf{v})$  我们可以将它们视为常数。条件分布  $P(\mathbf{h} \mid \mathbf{v})$  因子相乘的本质，我们可以将向量  $\mathbf{h}$  上的联合概率写成单独元素  $h_j$  上（未归一化）分布的乘积。现在原问题变成了对单个二值  $h_j$  上的分布进行归一化的简单问题。

$$P(h_j = 1 \mid \mathbf{v}) = \frac{\tilde{P}(h_j = 1 \mid \mathbf{v})}{\tilde{P}(h_j = 0 \mid \mathbf{v}) + \tilde{P}(h_j = 1 \mid \mathbf{v})} \quad (20.12)$$

$$= \frac{\exp\{c_j + \mathbf{v}^\top \mathbf{W}_{:,j}\}}{\exp\{0\} + \exp\{c_j + \mathbf{v}^\top \mathbf{W}_{:,j}\}} \quad (20.13)$$

$$= \sigma(c_j + \mathbf{v}^\top \mathbf{W}_{:,j}). \quad (20.14)$$

现在我们可以将关于隐藏层的完全条件分布表达为因子形式：

$$P(\mathbf{h} \mid \mathbf{v}) = \prod_{j=1}^{n_h} \sigma((2\mathbf{h} - 1) \odot (\mathbf{c} + \mathbf{W}^\top \mathbf{v}))_j. \quad (20.15)$$

类似的推导将显示我们感兴趣的另一条件分布， $P(\mathbf{v} \mid \mathbf{h})$  也是因子形式的分布：

$$P(\mathbf{v} \mid \mathbf{h}) = \prod_{i=1}^{n_v} \sigma((2\mathbf{v} - 1) \odot (\mathbf{b} + \mathbf{W} \mathbf{h}))_i. \quad (20.16)$$

## 20.2.2 训练受限玻尔兹曼机

因为 RBM 允许高效计算  $\tilde{P}(\mathbf{v})$  的估计和微分，并且还允许高效地（以块吉布斯采样的形式）进行 MCMC 采样，所以我们很容易使用第十八章中训练具有难以计

算配分函数的模型的技术来训练 RBM。这包括 CD、SML (PCD)、比率匹配等。与深度学习中使用的其他无向模型相比，RBM 可以相对直接地训练，因为我们可以以闭解形式计算  $P(\mathbf{h} | \mathbf{v})$ 。其他一些深度模型，如深度玻尔兹曼机，同时具备难处理的配分函数和难以推断的难题。

## 20.3 深度信念网络

深度信念网络 (deep belief network, DBN) 是第一批成功应用深度架构训练的非卷积模型之一 (Hinton *et al.*, 2006a; Hinton, 2007b)。2006 年深度信念网络的引入开始了当前深度学习的复兴。在引入深度信念网络之前，深度模型被认为太难以优化。具有凸目标函数的核机器引领了研究前沿。深度信念网络在 MNIST 数据集上表现超过内核化支持向量机，以此证明深度架构是能够成功的 (Hinton *et al.*, 2006a)。尽管现在与其他无监督或生成学习算法相比，深度信念网络大多已经失去了青睐并很少使用，但它们在深度学习历史中的重要作用仍应该得到承认。

深度信念网络是具有若干潜变量层的生成模型。潜变量通常是二值的，而可见单元可以是二值或实数。尽管构造连接比较稀疏的 DBN 是可能的，但在一般的模型中，每层的每个单元连接到每个相邻层中的每个单元（没有层内连接）。顶部两层之间的连接是无向的。而所有其他层之间的连接是有向的，箭头指向最接近数据的层。见图 20.1 b 的例子。

具有  $l$  个隐藏层的 DBN 包含  $l$  个权重矩阵： $\mathbf{W}^{(1)}, \dots, \mathbf{W}^{(l)}$ 。同时也包含  $l+1$  个偏置向量： $\mathbf{b}^{(0)}, \dots, \mathbf{b}^{(l)}$ ，其中  $\mathbf{b}^{(0)}$  是可见层的偏置。DBN 表示的概率分布由下式给出：

$$P(\mathbf{h}^{(l)}, \mathbf{h}^{(l-1)}) \propto \exp(\mathbf{b}^{(l)\top} \mathbf{h}^{(l)} + \mathbf{b}^{(l-1)\top} \mathbf{h}^{(l-1)} + \mathbf{h}^{(l-1)\top} \mathbf{W}^{(l)} \mathbf{h}^{(l)}), \quad (20.17)$$

$$P(h_i^{(k)} = 1 | \mathbf{h}^{(k+1)}) = \sigma(b_i^{(k)} + \mathbf{W}_{:,i}^{(k+1)\top} \mathbf{h}^{(k+1)}) \quad \forall i, \forall k \in 1, \dots, l-2, \quad (20.18)$$

$$P(v_i = 1 | \mathbf{h}^{(1)}) = \sigma(b_i^{(0)} + \mathbf{W}_{:,i}^{(1)\top} \mathbf{h}^{(1)}) \quad \forall i. \quad (20.19)$$

在实值可见单元的情况下，替换

$$\mathbf{v} \sim \mathcal{N}(\mathbf{v}; \mathbf{b}^{(0)} + \mathbf{W}^{(1)\top} \mathbf{h}^{(1)}, \boldsymbol{\beta}^{-1}) \quad (20.20)$$

为便于处理， $\boldsymbol{\beta}$  为对角形式。至少在理论上，推广到其他指数族的可见单元是直观的。只有一个隐藏层的 DBN 只是一个 RBM。

为了从 DBN 中生成样本，我们先在顶部的两个隐藏层上运行几个 Gibbs 采样步骤。这个阶段主要从 RBM（由顶部两个隐藏层定义）中采一个样本。然后，我们可以对模型的其余部分使用单次原始采样，以从可见单元绘制样本。

深度信念网络引发许多与有向模型和无向模型同时相关的问题。

由于每个有向层内的相消解释效应，并且由于无向连接的两个隐藏层之间的相互作用，深度信念网络中的推断是难解的。评估或最大化对数似然的标准证据下界也是难以处理的，因为证据下界基于大小等于网络宽度的团的期望。

评估或最大化对数似然，不仅需要面对边缘化潜变量时难以处理的推断问题，而且还需要处理顶部两层无向模型内难处理的配分函数问题。

为训练深度信念网络，我们可以先使用对比散度或随机最大似然方法训练 RBM 以最大化  $\mathbb{E}_{\mathbf{v} \sim p_{\text{data}}} \log p(\mathbf{v})$ 。RBM 的参数定义了 DBN 第一层的参数。然后，第二个 RBM 训练为近似最大化

$$\mathbb{E}_{\mathbf{v} \sim p_{\text{data}}} \mathbb{E}_{\mathbf{h}^{(1)} \sim p^{(1)}(\mathbf{h}^{(1)} | \mathbf{v})} \log p^{(2)}(\mathbf{h}^{(1)}), \quad (20.21)$$

其中  $p^{(1)}$  是第一个 RBM 表示的概率分布， $p^{(2)}$  是第二个 RBM 表示的概率分布。换句话说，第二个 RBM 被训练为模拟由第一个 RBM 的隐藏单元采样定义的分布，而第一个 RBM 由数据驱动。这个过程能无限重复，从而向 DBN 添加任意多层，其中每个新的 RBM 对前一个 RBM 的样本建模。每个 RBM 定义 DBN 的另一层。这个过程可以被视为提高数据在 DBN 下似然概率的变分下界 (Hinton *et al.*, 2006a)。

在大多数应用中，对 DBN 进行贪心逐层训练后，不再花功夫对其进行联合训练。然而，使用醒眠算法对其进行生成精调是可能的。

训练好的 DBN 可以直接用作生成模型，但是 DBN 的大多数兴趣来自于它们改进分类模型的能力。我们可以从 DBN 获取权重，并使用它们定义 MLP：

$$\mathbf{h}^{(1)} = \sigma(b^{(1)} + \mathbf{v}^\top \mathbf{W}^{(1)}), \quad (20.22)$$

$$\mathbf{h}^{(l)} = \sigma(b_i^{(l)} + \mathbf{h}^{(l-1)\top} \mathbf{W}^{(l)}) \quad \forall l \in 2, \dots, m. \quad (20.23)$$

利用 DBN 的生成训练后获得的权重和偏置初始化该 MLP 之后，我们可以训练该 MLP 来执行分类任务。这种 MLP 的额外训练是判别性精调的示例。

与第十九章中从基本原理导出的许多推断方程相比，这种特定选择的 MLP 有些随意。这个 MLP 是一个启发式选择，似乎在实践中效果不错，并在文献中一贯使用。许多近似推断技术是由它们在一些约束下，并在对数似然上找到最大紧变分下

界的能力所驱动的。我们可以使用 DBN 中 MLP 定义的隐藏单元的期望，构造对数似然的变分下界，但这对于隐藏单元上的任何概率分布都是如此，并没有理由相信该 MLP 提供了一个特别的紧界。特别地，MLP 忽略了 DBN 图模型中许多重要的相互作用。MLP 将信息从可见单元向上传播到最深的隐藏单元，但不向下或侧向传播任何信息。DBN 图模型解释了同一层内所有隐藏单元之间的相互作用以及层之间的自顶向下的相互作用。

虽然 DBN 的对数似然是难处理的，但它可以使用 AIS 近似 (Salakhutdinov and Murray, 2008)。通过近似，可以评估其作为生成模型的质量。

术语“深度信念网络”通常不正确地用于指代任意种类的深度神经网络，甚至没有潜变量意义的网络。这个术语应特指最深层中具有无向连接，而在所有其他连续层之间存在向下有向连接的模型。

这个术语也可能导致一些混乱，因为术语“信念网络”有时指纯粹的有向模型，而深度信念网络包含一个无向层。深度信念网络也与动态贝叶斯网络 (dynamic Bayesian networks) (Dean and Kanazawa, 1989) 共享首字母缩写 DBN，动态贝叶斯网络表示马尔可夫链的贝叶斯网络。

## 20.4 深度玻尔兹曼机

**深度玻尔兹曼机** (Deep Boltzmann Machine, DBM) (Salakhutdinov and Hinton, 2009a) 是另一种深度生成模型。与深度信念网络 (DBN) 不同的是，它是一个完全无向的模型。与 RBM 不同的是，DBM 有几层潜变量 (RBM 只有一层)。但是像 RBM 一样，每一层内的每个变量是相互独立的，并条件于相邻层中的变量。见图 20.2 中的图结构。深度玻尔兹曼机已经被应用于各种任务，包括文档建模 (Srivastava *et al.*, 2013)。

与 RBM 和 DBN 一样，DBM 通常仅包含二值单元 (正如我们为简化模型的演示而假设的)，但很容易就能扩展到实值可见单元。

DBM 是基于能量的模型，这意味着模型变量的联合概率分布由能量函数  $E$  参数化。在一个深度玻尔兹曼机包含一个可见层  $\mathbf{v}$  和三个隐藏层  $\mathbf{h}^{(1)}, \mathbf{h}^{(2)}$  和  $\mathbf{h}^{(3)}$  的情况下，联合概率由下式给出：

$$P(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}) = \frac{1}{Z(\boldsymbol{\theta})} \exp(-E(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}; \boldsymbol{\theta})). \quad (20.24)$$

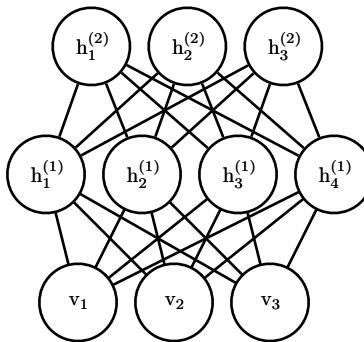


图 20.2: 具有一个可见层 (底部) 和两个隐藏层的深度玻尔兹曼机的图模型。仅在相邻层的单元之间存在连接。没有层内连接。

为简化表示, 下式省略了偏置参数。DBM 能量函数定义如下:

$$E(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \mathbf{h}^{(3)}; \boldsymbol{\theta}) = -\mathbf{v}^\top \mathbf{W}^{(1)} \mathbf{h}^{(1)} - \mathbf{h}^{(1)\top} \mathbf{W}^{(2)} \mathbf{h}^{(2)} - \mathbf{h}^{(2)\top} \mathbf{W}^{(3)} \mathbf{h}^{(3)}. \quad (20.25)$$

与 RBM 的能量函数 (式 (20.5)) 相比, DBM 能量函数以权重矩阵 ( $\mathbf{W}^{(2)}$  和  $\mathbf{W}^{(3)}$ ) 的形式表示隐藏单元 (潜变量) 之间的连接。正如我们将看到的, 这些连接对模型行为以及我们如何在模型中进行推断都有重要的影响。

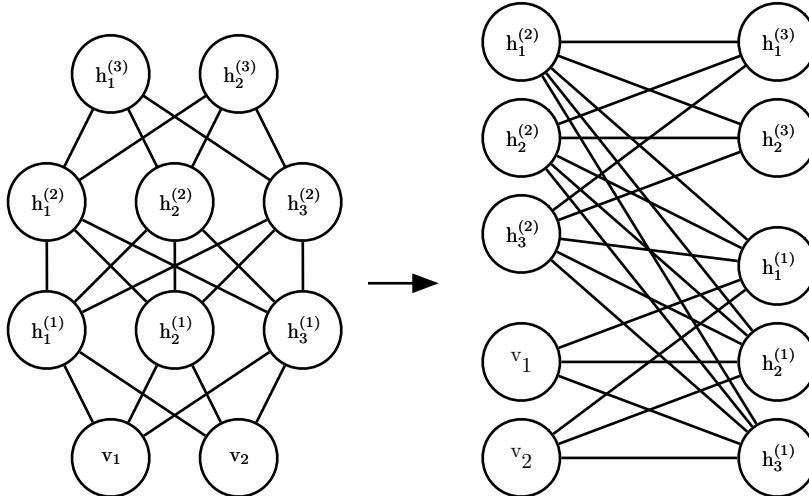


图 20.3: 深度玻尔兹曼机, 重新排列后显示为二分图结构。

与全连接的玻尔兹曼机 (每个单元连接到其他每个单元) 相比, DBM 提供了类

似于 RBM 的一些优点。

具体来说，如图 20.3 所示，DBM 的层可以组织成一个二分图，其中奇数层在一侧，偶数层在另一侧。容易发现，当我们条件于偶数层中的变量时，奇数层中的变量变得条件独立。当然，当我们条件于奇数层中的变量时，偶数层中的变量也会变得条件独立。

DBM 的二分图结构意味着我们可以应用之前用于 RBM 条件分布的相同式子来确定 DBM 中的条件分布。在给定相邻层值的情况下，层内的单元彼此条件独立，因此二值变量的分布可以由 Bernoulli 参数（描述每个单元的激活概率）完全描述。在具有两个隐藏层的示例中，激活概率由下式给出：

$$P(v_i = 1 \mid \mathbf{h}^{(1)}) = \sigma(\mathbf{W}_{i,:}^{(1)} \mathbf{h}^{(1)}), \quad (20.26)$$

$$P(h_i^{(1)} = 1 \mid \mathbf{v}, \mathbf{h}^{(2)}) = \sigma(\mathbf{v}^\top \mathbf{W}_{:,i}^{(1)} + \mathbf{W}_{i,:}^{(2)} \mathbf{h}^{(2)}), \quad (20.27)$$

和

$$P(h_k^{(2)} = 1 \mid \mathbf{h}^{(1)}) = \sigma(\mathbf{h}^{(1)\top} \mathbf{W}_{:,k}^{(2)}). \quad (20.28)$$

二分图结构使 Gibbs 采样能在深度玻尔兹曼机中高效采样。Gibbs 采样的方法是一次只更新一个变量。RBM 允许所有可见单元以一个块的方式更新，而所有隐藏单元在另一个块上更新。我们可以简单地假设具有  $l$  层的 DBM 需要  $l+1$  次更新，每次迭代更新由某层单元组成的块。然而，我们可以仅在两次迭代中更新所有单元。Gibbs 采样可以将更新分成两个块，一块包括所有偶数层（包括可见层），另一个包括所有奇数层。由于 DBM 二分连接模式，给定偶数层，关于奇数层的分布是因子的，因此可以作为块同时且独立地采样。类似地，给定奇数层，可以同时且独立地将偶数层作为块进行采样。高效采样对使用随机最大似然算法的训练尤其重要。

### 20.4.1 有趣的性质

深度玻尔兹曼机具有许多有趣的性质。

DBM 在 DBN 之后开发。与 DBN 相比，DBM 的后验分布  $P(\mathbf{h} \mid \mathbf{v})$  更简单。有点违反直觉的是，这种后验分布的简单性允许更加丰富的后验近似。在 DBN 的情况下，我们使用启发式的近似推断过程进行分类，其中我们可以通过 MLP（使用 sigmoid 激活函数并且权重与原始 DBN 相同）中的向上传播猜测隐藏单元合理的均匀场期望值。任何分布  $Q(\mathbf{h})$  可用于获得对数似然的变分下界。因此这种启发

式的过程让我们能够获得这样的下界。但是，该界没有以任何方式显式优化，所以该界可能是远远不紧的。特别地， $Q$  的启发式估计忽略了相同层内隐藏单元之间的相互作用以及更深层中隐藏单元对更接近输入的隐藏单元自顶向下的反馈影响。因为 DBN 中基于启发式 MLP 的推断过程不能考虑这些相互作用，所以得到的  $Q$  想必远不是最优的。DBM 中，在给定其他层的情况下，层内的所有隐藏单元都是条件独立的。这种层内相互作用的缺失使得通过不动点方程优化变分下界并找到真正最佳的均匀场期望（在一些数值容差内）变得可能的。

使用适当的均匀场允许 DBM 的近似推断过程捕获自顶向下反馈相互作用的影响。这从神经科学的角度来看是有趣的，因为根据已知，人脑使用许多自上而下的反馈连接。由于这个性质，DBM 已被用作真实神经科学现象的计算模型 (Series *et al.*, 2010; Reichert *et al.*, 2011)。

DBM 一个不理想的特性是从中采样是相对困难的。DBN 只需要在其顶部的一对层中使用 MCMC 采样。其他层仅在采样过程末尾涉及，并且只需在一个高效的原始采样过程。要从 DBM 生成样本，必须在所有层中使用 MCMC，并且模型的每一层都参与每个马尔可夫链转移。

## 20.4.2 DBM 均匀场推断

给定相邻层，一个 DBM 层上的条件分布是因子的。在有两个隐藏层的 DBM 的示例中，这些分布是  $P(\mathbf{v} | \mathbf{h}^{(1)})$ ,  $P(\mathbf{h}^{(1)} | \mathbf{v}, \mathbf{h}^{(2)})$  和  $P(\mathbf{h}^{(2)} | \mathbf{h}^{(1)})$ 。因为层之间的相互作用，所有隐藏层上的分布通常不是因子的。在有两个隐藏层的示例中，由于  $\mathbf{h}^{(1)}$  和  $\mathbf{h}^{(2)}$  之间的交互权重  $\mathbf{W}^{(2)}$  使得这些变量相互依赖， $P(\mathbf{h}^{(1)} | \mathbf{v}, \mathbf{h}^{(2)})$  不是因子的。

与 DBN 的情况一样，我们还是要找出近似 DBM 后验分布的方法。然而，与 DBN 不同，DBM 在其隐藏单元上的后验分布（复杂的）很容易用变分近似来近似（如第 19.4 节所讨论），具体是一个均匀场近似。均匀场近似是变分推断的简单形式，其中我们将近似分布限制为完全因子的分布。在 DBM 的情况下，均匀场方程捕获层之间的双向相互作用。在本节中，我们推导出由 Salakhutdinov and Hinton (2009a) 最初引入的迭代近似推断过程。

在推断的变分近似中，我们通过一些相当简单的分布族近似特定目标分布——在这里指给定可见单元时隐藏单元的后验分布。在均匀场近似的情况下，近似族是隐藏单元条件独立的分布集合。

我们现在为具有两个隐藏层的示例推导均匀场方法。令  $Q(\mathbf{h}^{(1)}, \mathbf{h}^{(2)} | \mathbf{v})$  为  $P(\mathbf{h}^{(1)}, \mathbf{h}^{(2)} | \mathbf{v})$  的近似。均匀场假设意味着

$$Q(\mathbf{h}^{(1)}, \mathbf{h}^{(2)} | \mathbf{v}) = \prod_j Q(h_j^{(1)} | \mathbf{v}) \prod_k Q(h_k^{(2)} | \mathbf{v}). \quad (20.29)$$

均匀场近似试图找到这个分布族中最适合真实后验  $P(\mathbf{h}^{(1)}, \mathbf{h}^{(2)} | \mathbf{v})$  的成员。重要的是，每次我们使用  $\mathbf{v}$  的新值时，必须再次运行推断过程以找到不同的分布  $Q$ 。

我们可以设想很多方法来衡量  $Q(\mathbf{h} | \mathbf{v})$  与  $P(\mathbf{h} | \mathbf{v})$  的拟合程度。均匀场方法是最小化

$$\text{KL}(Q || P) = \sum_h Q(\mathbf{h}^{(1)}, \mathbf{h}^{(2)} | \mathbf{v}) \log \left( \frac{Q(\mathbf{h}^{(1)}, \mathbf{h}^{(2)} | \mathbf{v})}{P(\mathbf{h}^{(1)}, \mathbf{h}^{(2)} | \mathbf{v})} \right). \quad (20.30)$$

一般来说，除了要保证独立性假设，我们不必提供参数形式的近似分布。变分近似过程通常能够恢复近似分布的函数形式。然而，在二值隐藏单元（我们在那里推导的情况）的均匀场假设的情况下，不会由于预先固定模型的参数而损失一般性。

我们将  $Q$  作为 Bernoulli 分布的乘积进行参数化，即我们将  $\mathbf{h}^{(1)}$  每个元素的概率与一个参数相关联。具体来说，对于每个  $j$ ,  $\hat{h}_j^{(1)} = Q(h_j^{(1)} = 1 | \mathbf{v})$ , 其中  $\hat{h}_j^{(1)} \in [0, 1]$ 。另外，对于每个  $k$ ,  $\hat{h}_k^{(2)} = Q(h_k^{(2)} = 1 | \mathbf{v})$ , 其中  $\hat{h}_k^{(2)} \in [0, 1]$ 。因此，我们有以下近似后验：

$$Q(\mathbf{h}^{(1)}, \mathbf{h}^{(2)} | \mathbf{v}) = \prod_j Q(h_j^{(1)} | \mathbf{v}) \prod_k Q(h_k^{(2)} | \mathbf{v}) \quad (20.31)$$

$$= \prod_j (\hat{h}_j^{(1)})^{h_j^{(1)}} (1 - \hat{h}_j^{(1)})^{(1-h_j^{(1)})} \times \prod_k (\hat{h}_k^{(2)})^{h_k^{(2)}} (1 - \hat{h}_k^{(2)})^{(1-h_k^{(2)})}. \quad (20.32)$$

当然，对于具有更多层的 DBM，近似后验的参数化可以通过明显的方式扩展，即利用图的二分结构，遵循 Gibbs 采样相同的调度，同时更新所有偶数层，然后同时更新所有奇数层。

现在我们已经指定了近似分布  $Q$  的函数族，但仍然需要指定用于选择该函数族中最适合  $P$  的成员的过程。最直接的方法是使用式 (19.56) 指定的均匀场方程。这些方程是通过求解变分下界导数为零的位置而导出。他们以抽象的方式描述如何优化任意模型的变分下界（只需对  $Q$  求期望）。

应用这些一般的方程，我们得到以下更新规则（再次忽略偏置项）：

$$h_j^{(1)} = \sigma \left( \sum_i v_i \mathbf{W}_{i,j}^{(1)} + \sum_{k'} \mathbf{W}_{j,k'}^{(2)} \hat{h}_{k'}^{(2)} \right), \forall j, \quad (20.33)$$

$$\hat{h}_k^{(2)} = \sigma \left( \sum_{j'} \mathbf{W}_{j',k}^{(2)} \hat{h}_{j'}^{(1)} \right), \forall k. \quad (20.34)$$

在该方程组的不动点处，我们具有变分下界  $\mathcal{L}(Q)$  的局部最大值。因此，这些不动点更新方程定义了迭代算法，其中我们交替更新  $h_j^{(1)}$ （使用式 (20.33)）和  $h_k^{(2)}$ （使用式 (20.34)）。对于诸如 MNIST 的小问题，少至 10 次迭代就足以找到用于学习的近似正相梯度，而 50 次通常足以获得要用于高精度分类的单个特定样本的高质量表示。将近似变分推断扩展到更深的 DBM 是直观的。

### 20.4.3 DBM 的参数学习

DBM 中的学习必须面对难解配分函数的挑战（使用第十八章中的技术），以及难解后验分布的挑战（使用第十九章中的技术）。

如第 20.4.2 节中所描述的，变分推断允许构建近似难处理的  $P(\mathbf{h} \mid \mathbf{v})$  的分布  $Q(\mathbf{h} \mid \mathbf{v})$ 。然后通过最大化  $\mathcal{L}(\mathbf{v}, Q, \boldsymbol{\theta})$ （难处理的对数似然的变分下界  $\log P(\mathbf{v}; \boldsymbol{\theta})$ ）学习。

对于具有两个隐藏层的深度玻尔兹曼机， $\mathcal{L}$  由下式给出

$$\mathcal{L}(Q, \boldsymbol{\theta}) = \sum_i \sum_{j'} v_i W_{i,j'}^{(1)} \hat{h}_{j'}^{(1)} + \sum_{j'} \sum_{k'} \hat{h}_{j'}^{(1)} W_{j',k'}^{(2)} \hat{h}_{k'}^{(2)} - \log Z(\boldsymbol{\theta}) + \mathcal{H}(Q). \quad (20.35)$$

该表达式仍然包含对数配分函数  $\log Z(\boldsymbol{\theta})$ 。由于深度玻尔兹曼机包含受限玻尔兹曼机作为组件，用于计算受限玻尔兹曼机的配分函数和采样的困难同样适用于深度玻尔兹曼机。这意味着评估玻尔兹曼机的概率质量函数需要近似方法，如退火重要采样。同样，训练模型需要近似对数配分函数的梯度。见第十八章对这些方法的一般性描述。DBM 通常使用随机最大似然训练。第十八章中描述的许多其他技术都不适用。诸如伪似然的技术需要评估非归一化概率的能力，而不是仅仅获得它们的变分下界。对于深度玻尔兹曼机，对比散度是缓慢的，因为它们不能在给定可见单元时对隐藏单元进行高效采样——反而，每当需要新的负相样本时，对比散度将需要磨合一条马尔可夫链。

非变分版本的随机最大似然算法已经在第 18.2 节讨论过。算法 20.1 给出了应用

于 DBM 的变分随机最大似然算法。回想一下，我们描述的是 DBM 的简化变体（缺少偏置参数）；很容易推广到包含偏置参数的情况。

#### 20.4.4 逐层预训练

不幸的是，随机初始化后使用随机最大似然训练（如上所述）的 DBM 通常导致失败。在一些情况下，模型不能学习如何充分地表示分布。在其他情况下，DBM 可以很好地表示分布，但是没有比仅使用 RBM 获得更高的似然。除第一层之外，所有层都具有非常小权重的 DBM 与 RBM 表示大致相同的分布。

如第 20.4.5 节所述，目前已经开发了允许联合训练的各种技术。然而，克服 DBM 的联合训练问题最初和最流行的方法是贪心逐层预训练。在该方法中，DBM 的每一层被单独视为 RBM，进行训练。第一层被训练为对输入数据进行建模。每个后续 RBM 被训练为对来自前一 RBM 后验分布的样本进行建模。在以这种方式训练了所有 RBM 之后，它们可以被组合成 DBM。然后可以用 PCD 训练 DBM。通常，PCD 训练将仅使模型的参数、由数据上的对数似然衡量的性能、或区分输入的能力发生微小的变化。见图 20.4 展示的训练过程。

这种贪心逐层训练过程不仅仅是坐标上升。因为我们在每个步骤优化参数的一个子集，它与坐标上升具有一些传递相似性。这两种方法是不同的，因为贪心逐层训练过程中，我们在每个步骤都使用了不同的目标函数。

DBM 的贪心逐层预训练与 DBN 的贪心逐层预训练不同。每个单独的 RBM 的参数可以直接复制到相应的 DBN。在 DBM 的情况下，RBM 的参数在包含到 DBM 中之前必须修改。RBM 栈的中间层仅使用自底向上的输入进行训练，但在栈组合形成 DBM 后，该层将同时具有自底向上和自顶向下的输入。为了解释这种效应，Salakhutdinov and Hinton (2009a) 提倡在将其插入 DBM 之前，将所有 RBM（顶部和底部 RBM 除外）的权重除 2。另外，必须使用每个可见单元的两个“副本”来训练底部 RBM，并且两个副本之间的权重约束为相等。这意味着在向上传播时，权重能有效地加倍。类似地，顶部 RBM 应当使用最顶层的两个副本训练。

为了使用深度玻尔兹曼机获得最好结果，我们需要修改标准的 SML 算法，即在联合 PCD 训练步骤的负相期间使用少量的均匀场 (Salakhutdinov and Hinton, 2009a)。具体来说，应当相对于其中所有单元彼此独立的均匀场分布来计算能量梯度的期望。这个均匀场分布的参数应该通过运行一次均匀场不动点方程获得。Goodfellow *et al.* (2013d) 比较了在负相中使用和不使用部分均匀场的中心化 DBM 的性能。

---

**算法 20.1** 用于训练具有两个隐藏层的DBM的变分随机最大似然算法
 

---

设步长  $\epsilon$  为一个小正数

设定吉布斯步数  $k$ , 大到足以让  $p(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}; \boldsymbol{\theta} + \epsilon\Delta_{\boldsymbol{\theta}})$  的马尔可夫链能磨合 (从来自  $p(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}; \boldsymbol{\theta})$  的样本开始)。

初始化三个矩阵,  $\tilde{\mathbf{V}}$ ,  $\tilde{\mathbf{H}}^{(1)}$  和  $\tilde{\mathbf{H}}^{(2)}$  每个都将  $m$  行设为随机值 (例如, 来自 Bernoulli 分布, 边缘分布大致与模型匹配)。

**while** 没有收敛 (学习循环) **do**

从训练数据采包含  $m$  个样本的小批量, 并将它们排列为设计矩阵  $\mathbf{V}$  的行。

初始化矩阵  $\hat{\mathbf{H}}^{(1)}$  和  $\hat{\mathbf{H}}^{(2)}$ , 使其大致符合模型的边缘分布。

**while** 没有收敛 (均匀场推断循环) **do**

$$\hat{\mathbf{H}}^{(1)} \leftarrow \text{sigmoid} \left( \mathbf{V}\mathbf{W}^{(1)} + \hat{\mathbf{H}}^{(2)}\mathbf{W}^{(2)\top} \right).$$

$$\hat{\mathbf{H}}^{(2)} \leftarrow \text{sigmoid} \left( \hat{\mathbf{H}}^{(1)}\mathbf{W}^{(2)} \right).$$

**end while**

$$\Delta_{\mathbf{W}^{(1)}} \leftarrow \frac{1}{m} \mathbf{V}^\top \hat{\mathbf{H}}^{(1)}$$

$$\Delta_{\mathbf{W}^{(2)}} \leftarrow \frac{1}{m} \hat{\mathbf{H}}^{(1)\top} \hat{\mathbf{H}}^{(2)}$$

**for**  $l = 1$  to  $k$  (Gibbs 采样) **do**

Gibbs block 1:

$$\forall i, j, \tilde{V}_{i,j} \text{ 采自 } P(\tilde{V}_{i,j} = 1) = \text{sigmoid} \left( \mathbf{W}_{j,:}^{(1)} \left( \hat{\mathbf{H}}_{i,:}^{(1)} \right)^\top \right).$$

$$\forall i, j, \tilde{H}_{i,j}^{(2)} \text{ 采自 } P(\tilde{H}_{i,j}^{(2)} = 1) = \text{sigmoid} \left( \tilde{\mathbf{H}}_{i,:}^{(1)} \mathbf{W}_{:,j}^{(2)} \right).$$

Gibbs block 2:

$$\forall i, j, \tilde{H}_{i,j}^{(1)} \text{ 采自 } P(\tilde{H}_{i,j}^{(1)} = 1) = \text{sigmoid} \left( \tilde{\mathbf{V}}_{i,:} \mathbf{W}_{:,j}^{(1)} + \tilde{\mathbf{H}}_{i,:}^{(2)} \mathbf{W}_{j,:}^{(2)\top} \right).$$

**end for**

$$\Delta_{\mathbf{W}^{(1)}} \leftarrow \Delta_{\mathbf{W}^{(1)}} - \frac{1}{m} \mathbf{V}^\top \tilde{\mathbf{H}}^{(1)}$$

$$\Delta_{\mathbf{W}^{(2)}} \leftarrow \Delta_{\mathbf{W}^{(2)}} - \frac{1}{m} \tilde{\mathbf{H}}^{(1)\top} \tilde{\mathbf{H}}^{(2)}$$

$\mathbf{W}^{(1)} \leftarrow \mathbf{W}^{(1)} + \epsilon \Delta_{\mathbf{W}^{(1)}}$  (这是大概的描述, 实践中使用的算法更高效, 如具有衰减学习率的动量)

$$\mathbf{W}^{(2)} \leftarrow \mathbf{W}^{(2)} + \epsilon \Delta_{\mathbf{W}^{(2)}}$$

**end while**

---

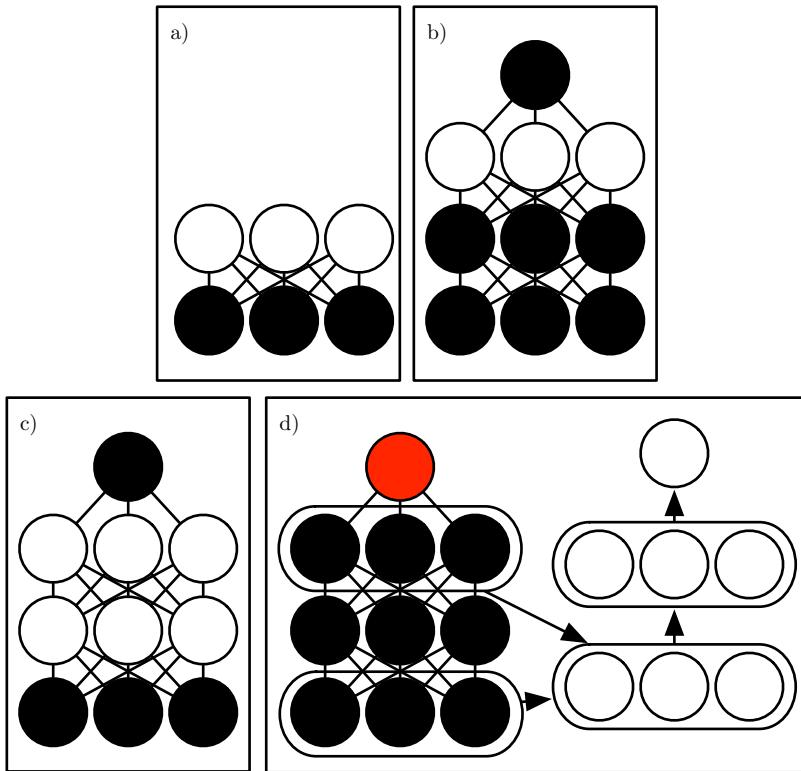


图 20.4: 用于分类 MNIST 数据集的深度玻尔兹曼机训练过程 (Salakhutdinov and Hinton, 2009a; Srivastava *et al.*, 2014)。(a) 使用CD近似最大化  $\log P(\mathbf{v})$  来训练RBM。(b) 训练第二个RBM, 使用CD- $k$  近似最大化  $\log P(\mathbf{h}^{(1)}, \mathbf{y})$  来建模  $\mathbf{h}^{(1)}$  和目标类  $\mathbf{y}$ , 其中  $\mathbf{h}^{(1)}$  采自第一个RBM条件于数据的后验。在学习期间将  $k$  从 1 增加到 20。(c) 将两个RBM组合为DBM。使用  $k = 5$  的随机最大似然训练, 近似最大化  $\log P(\mathbf{v}, \mathbf{y})$ 。(d) 将  $\mathbf{y}$  从模型中删除。定义新的一组特征  $\mathbf{h}^{(1)}$  和  $\mathbf{h}^{(2)}$ , 可在缺少  $\mathbf{y}$  的模型中运行均匀场推断后获得。使用这些特征作为MLP的输入, 其结构与均匀场的额外轮相同, 并且具有用于估计  $\mathbf{y}$  的额外输出层。初始化MLP的权重与DBM的权重相同。使用随机梯度下降和Dropout训练MLP近似最大化  $\log P(\mathbf{y} | \mathbf{v})$ 。图来自 Goodfellow *et al.* (2013d)。

## 20.4.5 联合训练深度玻尔兹曼机

经典 DBM 需要贪心无监督预训练, 并且为了更好的分类, 需要在它们提取的隐藏特征之上, 使用独立的基于 MLP 的分类器。这种方法有一些不理想的性质。因为我们不能在训练第一个 RBM 时评估完整 DBM 的属性, 所以在训练期间难以跟踪性能。因此, 直到相当晚的训练过程, 我们都很难知道我们的超参数表

现如何。DBM 的软件实现需要很多不同的模块，如用于单个 RBM 的 CD 训练、完整 DBM 的 PCD 训练以及基于反向传播的 MLP 训练。最后，玻尔兹曼机顶部的 MLP 失去了玻尔兹曼机概率模型的许多优点，例如当某些输入值丢失时仍能够进行推断的优点。

主要有两种方法可以处理深度玻尔兹曼机的联合训练问题。第一个是**中心化深度玻尔兹曼机**(centered deep Boltzmann machine) (Montavon and Muller, 2012)，通过重参数化模型使其在开始学习过程时代价函数的 Hessian 具有更好的条件数。这个模型不用经过贪心逐层预训练阶段就能训练。这个模型在测试集上获得出色的对数似然，并能产生高质量的样本。不幸的是，作为分类器，它仍然不能与适当正则化的 MLP 竞争。联合训练深度玻尔兹曼机的第二种方式是使用**多预测深度玻尔兹曼机** (multi-prediction deep Boltzmann machine, MP-DBM) (Goodfellow *et al.*, 2013d)。该模型的训练准则允许反向传播算法，以避免使用 MCMC 估计梯度的问题。不幸的是，新的准则不会导致良好的似然性或样本，但是相比 MCMC 方法，它确实会导致更好的分类性能和良好的推断缺失输入的能力。

如果我们回到玻尔兹曼机的一般观点，即包括一组权重矩阵  $\mathbf{U}$  和偏置  $\mathbf{b}$  的单元  $\mathbf{x}$ ，玻尔兹曼机中心化技巧是最容易描述的。回顾式(20.2)，能量函数由下式给出

$$E(\mathbf{x}) = -\mathbf{x}^\top \mathbf{U}\mathbf{x} - \mathbf{b}^\top \mathbf{x}. \quad (20.36)$$

在权重矩阵  $\mathbf{U}$  中使用不同的稀疏模式，我们可以实现不同架构的玻尔兹曼机，如 RBM 或具有不同层数的 DBM。将  $\mathbf{x}$  分割成可见和隐藏单元并将  $\mathbf{U}$  中不相互作用的单元的归零可以实现这些架构。中心化玻尔兹曼机引入了一个向量  $\boldsymbol{\mu}$ ，并从所有状态中减去：

$$E'(\mathbf{x}; \mathbf{U}, \mathbf{b}) = -(\mathbf{x} - \boldsymbol{\mu})^\top \mathbf{U}(\mathbf{x} - \boldsymbol{\mu}) - (\mathbf{x} - \boldsymbol{\mu})^\top \mathbf{b}. \quad (20.37)$$

通常  $\boldsymbol{\mu}$  在开始训练时固定为一个超参数。当模型初始化时，通常选择为  $\mathbf{x} - \boldsymbol{\mu} \approx 0$ 。这种重参数化不改变模型可表示的概率分布的集合，但它确实改变了应用于似然的随机梯度下降的动态。具体来说，在许多情况下，这种重参数化导致更好条件数的 Hessian 矩阵。Melchior *et al.* (2013) 通过实验证实了 Hessian 矩阵条件数的改善，并观察到中心化技巧等价于另一个玻尔兹曼机学习技术——增强梯度 (enhanced gradient) (Cho *et al.*, 2011)。即使在困难的情况下，例如训练多层的深度玻尔兹曼机，Hessian 矩阵条件数的改善也能使学习成功。

联合训练深度玻尔兹曼机的另一种方法是多预测深度玻尔兹曼机 (MP-DBM)，

它将均匀场方程视为定义一系列用于近似求解每个可能推断问题的循环网络 (Goodfellow *et al.*, 2013d)。模型被训练为使每个循环网络获得对相应推断问题的准确答案，而不是训练模型来最大化似然。训练过程如图 20.5 所示。它包括随机采一个训练样本、随机采样推断网络的输入子集，然后训练推断网络来预测剩余单元的值。

这种用于近似推断，通过计算图进行反向传播的一般原理已经应用于其他模型 (Stoyanov *et al.*, 2011; Brakel *et al.*, 2013)。在这些模型和 MP-DBM 中，最终损失不是似然的下界。相反，最终损失通常基于近似推断网络对缺失值施加的近似条件分布。这意味着这些模型的训练有些启发式。如果我们检查由 MP-DBM 学习出来的玻尔兹曼机表示  $p(\mathbf{v})$ ，在 Gibbs 采样产生较差样本的意义下，它倾向于有些缺陷。

通过推断图的反向传播有两个主要优点。首先，它以模型真正使用的方式训练模型——使用近似推断。这意味着在 MP-DBM 中，进行如填充缺失的输入或执行分类（尽管存在缺失的输入）的近似推断比在原始 DBM 中更准确。原始 DBM 不会自己做出准确的分类器；使用原始 DBM 的最佳分类结果是基于 DBM 提取的特征训练独立的分类器，而不是通过使用 DBM 中的推断来计算关于类标签的分布。MP-DBM 中的均匀场推断作为分类器，不需要进行特殊修改就获得良好的表现。通过近似推断反向传播的另一个优点是反向传播计算损失的精确梯度。对于优化而言，比 SML 训练中具有偏差和方差的近似梯度更好。这可能解释了为什么 MP-DBM 可以联合训练，而 DBM 需要贪心逐层预训练。近似推断图反向传播的缺点是它不提供一种优化对数似然的方法，而提供广义伪似然的启发式近似。

MP-DBM 启发了对 NADE 框架的扩展 NADE- $k$  (Raiko *et al.*, 2014)，我们将在第 20.10.10 节中描述。

MP-DBM 与 Dropout 有一定联系。Dropout 在许多不同的计算图之间共享相同的参数，每个图之间的差异是包括还是排除每个单元。MP-DBM 还在许多计算图之间共享参数。在 MP-DBM 的情况下，图之间的差异是每个输入单元是否被观察到。当没有观察到单元时，MP-DBM 不会像 Dropout 那样将其完全删除。相反，MP-DBM 将其视为要推断的潜变量。我们可以想象将 Dropout 应用到 MP-DBM，即额外去除一些单元而不是将它们变为潜变量。

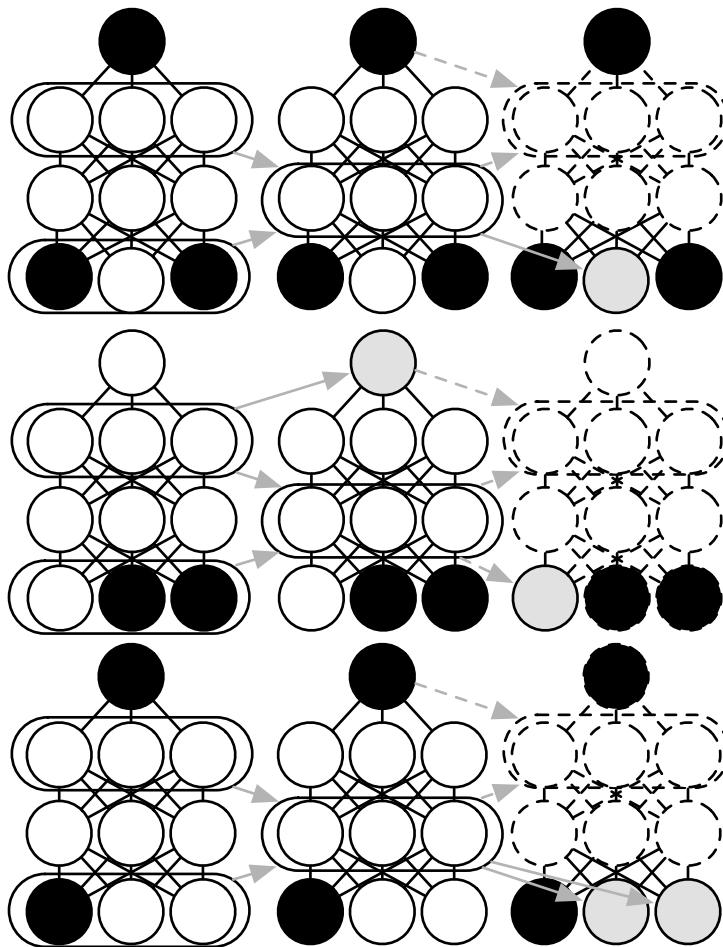


图 20.5: 深度玻尔兹曼机多预测训练过程的示意图。每一行指示相同训练步骤内小批量中的不同样本。每列表示均匀场推断过程中的时间步。对于每个样本，我们对数据变量的子集进行采样，作为推断过程的输入。这些变量以黑色阴影表示条件。然后我们运行均匀场推断过程，箭头指示过程中的哪些变量会影响其他变量。在实际应用中，我们将均匀场展开为几个步骤。在此示意图中，我们只展开为两个步骤。虚线箭头表示获得更多步骤需要如何展开该过程。未用作推断过程输入的数据变量成为目标，以灰色阴影表示。我们可以将每个样本的推断过程视为循环网络。为了使其在给定输入后能产生正确的目标，我们使用梯度下降和反向传播训练这些循环网络。这可以训练 MP-DBM 均匀场过程产生准确的估计。图改编自 Goodfellow *et al.* (2013d)。

## 20.5 实值数据上的玻尔兹曼机

虽然玻尔兹曼机最初是为二值数据而开发的，但是许多应用，例如图像和音频建模似乎需要表示实值上概率分布的能力。在一些情况下，我们可以将区间  $[0, 1]$  中的实值数据视为表示二值变量的期望。例如，Hinton (2000) 将训练集中灰度图像的像素值视为定义  $[0, 1]$  间的概率值。每个像素定义二值变量为 1 的概率，并且二值像素的采样都彼此独立。这是评估灰度图像数据集上二值模型的常见过程。然而，这种方法理论上并不特别令人满意，并且以这种方式独立采样的二值图像具有噪声表象。在本节中，我们介绍概率密度定义在实值数据上的玻尔兹曼机。

### 20.5.1 Gaussian-Bernoulli RBM

受限玻尔兹曼机可以用于许多指数族的条件分布 (Welling *et al.*, 2005)。其中，最常见的是具有二值隐藏单元和实值可见单元的 RBM，其中可见单元上的条件分布是高斯分布（均值为隐藏单元的函数）。

有很多方法可以参数化 Gaussian-Bernoulli RBM。首先，我们可以选择协方差矩阵或精度矩阵来参数化高斯分布。这里，我们介绍选择精度矩阵的情况。我们可以通过简单的修改获得协方差的形式。我们希望条件分布为

$$p(\mathbf{v} \mid \mathbf{h}) = \mathcal{N}(\mathbf{v}; \mathbf{Wh}, \boldsymbol{\beta}^{-1}). \quad (20.38)$$

通过扩展未归一化的对数条件分布可以找到需要添加到能量函数中的项：

$$\log \mathcal{N}(\mathbf{v}; \mathbf{Wh}, \boldsymbol{\beta}^{-1}) = -\frac{1}{2}(\mathbf{v} - \mathbf{Wh})^\top \boldsymbol{\beta} (\mathbf{v} - \mathbf{Wh}) + f(\boldsymbol{\beta}). \quad (20.39)$$

此处  $f$  封装所有的参数，但不包括模型中的随机变量。因为  $f$  的唯一作用是归一化分布，并且我们选择的任何可作为配分函数的能量函数都能起到这个作用，所以我们忽略  $f$ 。

如果我们在能量函数中包含式 (20.39) 中涉及  $\mathbf{v}$  的所有项（其符号被翻转），并且不添加任何其他涉及  $\mathbf{v}$  的项，那么我们的能量函数就能表示想要的条件分布  $p(\mathbf{v} \mid \mathbf{h})$ 。

其他条件分布比较自由，如  $p(\mathbf{h} \mid \mathbf{v})$ 。注意式 (20.39) 包含一项

$$\frac{1}{2} \mathbf{h}^\top \mathbf{W}^\top \boldsymbol{\beta} \mathbf{Wh}. \quad (20.40)$$

因为该项包含  $h_i h_j$  项，它不能被全部包括在内。这些对应于隐藏单元之间的边。如果我们包括这些项，我们将得到一个线性因子模型，而不是受限玻尔兹曼机。当设计我们的玻尔兹曼机时，我们简单地省略这些  $h_i h_j$  交叉项。省略这些项不改变条件分布  $p(\mathbf{v} | \mathbf{h})$ ，因此式 (20.39) 仍满足。然而，我们仍然可以选择是否包括仅涉及单个  $h_i$  的项。如果我们假设精度矩阵是对角的，就能发现对于每个隐藏单元  $h_i$ ，我们有一项

$$\frac{1}{2} h_i \sum_j \beta_j W_{j,i}^2. \quad (20.41)$$

在上面，我们使用了  $h_i^2 = h_i$  的事实（因为  $h_i \in \{0, 1\}$ ）。如果我们在能量函数中包含此项（符号被翻转），则当该单元的权重较大且以高精度连接到可见单元时，偏置  $h_i$  将自然被关闭。是否包括该偏置项不影响模型可以表示的分布族（假设我们包括隐藏单元的偏置参数），但是它确实会影响模型的学习动态。包括该项可以帮助隐藏单元（即使权重在幅度上快速增加时）保持合理激活。

因此，在 Gaussian-Bernoulli RBM 上定义能量函数的一种方式：

$$E(\mathbf{v}, \mathbf{h}) = \frac{1}{2} \mathbf{v}^\top (\boldsymbol{\beta} \odot \mathbf{v}) - (\mathbf{v} \odot \boldsymbol{\beta})^\top \mathbf{W} \mathbf{h} - \mathbf{b}^\top \mathbf{h}, \quad (20.42)$$

但我们还可以添加额外的项或者通过方差而不是精度参数化能量。

在这个推导中，我们没有在可见单元上添加偏置项，但添加这样的偏置是容易的。Gaussian-Bernoulli RBM 参数化一个最终变化的来源是如何处理精度矩阵的选择。它可以被固定为常数（可能基于数据的边缘精度估计）或学习出来。它也可以是标量乘以单位矩阵，或者是一个对角矩阵。在此情况下，由于一些操作需要对矩阵求逆，我们通常不允许非对角的精度矩阵，因为高斯分布的一些操作需要对矩阵求逆，一个对角矩阵可以非常容易地被求逆。在接下来的章节中，我们将看到其他形式的玻尔兹曼机，它们允许对协方差结构建模，并使用各种技术避免对精度矩阵求逆。

## 20.5.2 条件协方差的无向模型

虽然高斯 RBM 已成为实值数据的标准能量模型，Ranzato *et al.* (2010a) 认为高斯 RBM 感应偏置不能很好地适合某些类型的实值数据中存在的统计变化，特别是自然图像。问题在于自然图像中的许多信息内容嵌入于像素之间的协方差而不是原始像素值中。换句话说，图像中的大多数有用信息在于像素之间的关系，而不是其绝对值。由于高斯 RBM 仅对给定隐藏单元的输入条件均值建模，所以它不能捕

获条件协方差信息。为了回应这些评论，已经有学者提出了替代模型，设法更好地考虑实值数据的协方差。这些模型包括 **均值和协方差 RBM** ( mean and covariance RBM, mcRBM )<sup>1</sup>、**学生 t 分布均值乘积** ( mean product of Student t-distribution, mPoT ) 模型和 **尖峰和平板 RBM** ( spike and slab RBM, ssRBM )。

**均值和协方差 RBM** mcRBM 使用隐藏单元独立地编码所有可观察单元的条件均值和协方差。mcRBM 的隐藏层分为两组单元：均值单元和协方差单元。建模条件均值的那组单元是简单的高斯 RBM。另一半是 **协方差 RBM** ( covariance RBM, cRBM ) (Ranzato *et al.*, 2010a)，对条件协方差的结构进行建模（如下所述）。

具体来说，在二值均值的单元  $\mathbf{h}^{(m)}$  和二值协方差单元  $\mathbf{h}^{(c)}$  的情况下，mcRBM 模型被定义为两个能量函数的组合：

$$E_{\text{mc}}(\mathbf{x}, \mathbf{h}^{(m)}, \mathbf{h}^{(c)}) = E_{\text{m}}(\mathbf{x}, \mathbf{h}^{(m)}) + E_{\text{c}}(\mathbf{x}, \mathbf{h}^{(c)}), \quad (20.43)$$

其中  $E_{\text{m}}$  为标准的 Gaussian-Bernoulli RBM 能量函数<sup>2</sup>，

$$E_{\text{m}}(\mathbf{x}, \mathbf{h}^{(m)}) = \frac{1}{2} \mathbf{x}^\top \mathbf{x} - \sum_j \mathbf{x}^\top \mathbf{W}_{:,j} h_j^{(m)} - \sum_j b_j^{(m)} h_j^{(m)}, \quad (20.44)$$

$E_{\text{c}}$  是 cRBM 建模条件协方差信息的能量函数：

$$E_{\text{c}}(\mathbf{x}, \mathbf{h}^{(c)}) = \frac{1}{2} \sum_j h_j^{(c)} (\mathbf{x}^\top \mathbf{r}^{(j)})^2 - \sum_j b_j^{(c)} h_j^{(c)}. \quad (20.45)$$

参数  $\mathbf{r}^{(j)}$  与  $h_j^{(c)}$  关联的协方差权重向量对应， $\mathbf{b}^{(c)}$  是一个协方差偏置向量。组合后的能量函数定义联合分布，

$$p_{\text{mc}}(\mathbf{x}, \mathbf{h}^{(m)}, \mathbf{h}^{(c)}) = \frac{1}{Z} \exp \left\{ -E_{\text{mc}}(\mathbf{x}, \mathbf{h}^{(m)}, \mathbf{h}^{(c)}) \right\}, \quad (20.46)$$

以及给定  $\mathbf{h}^{(m)}$  和  $\mathbf{h}^{(c)}$  后，关于观察数据相应的条件分布（为一个多元高斯分布）：

$$p_{\text{mc}}(\mathbf{x} | \mathbf{h}^{(m)}, \mathbf{h}^{(c)}) = \mathcal{N} \left( \mathbf{x}; \mathbf{C}_{\mathbf{x}|\mathbf{h}}^{\text{mc}} \left( \sum_j \mathbf{W}_{:,j} h_j^{(m)} \right), \mathbf{C}_{\mathbf{x}|\mathbf{h}}^{\text{mc}} \right). \quad (20.47)$$

注意协方差矩阵  $\mathbf{C}_{\mathbf{x}|\mathbf{h}}^{\text{mc}} = \left( \sum_j h_j^{(c)} \mathbf{r}^{(j)} \mathbf{r}^{(j)T} + \mathbf{I} \right)^{-1}$  是非对角的，且  $\mathbf{W}$  是与建模条件均值的高斯 RBM 相关联的权重矩阵。由于非对角的条件协方差结构，难以通过对

<sup>1</sup>术语“mcRBM”根据字母 M-C-R-B-M 发音；“mc”不是“McDonald’s”中的“Mc”的发音。

<sup>2</sup>这个版本的 Gaussian-Bernoulli RBM 能量函数假定图像数据的每个像素具有零均值。考虑非零像素均值时，可以简单地将像素偏移添加到模型中。

比散度或持续性对比散度来训练 mcRBM。CD 和 PCD 需要从  $\mathbf{x}, \mathbf{h}^{(m)}, \mathbf{h}^{(c)}$  的联合分布中采样，这在标准RBM 中可以通过 Gibbs 采样在条件分布上采样实现。但是，在 mcRBM 中，从  $p_{\text{mc}}(\mathbf{x} | \mathbf{h}^{(m)}, \mathbf{h}^{(c)})$  中抽样需要在学习的每个迭代计算  $(\mathbf{C}^{\text{mc}})^{-1}$ 。这对于更大的观察数据可能是不切实际的计算负担。Ranzato and Hinton (2010) 通过使用 mcRBM 自由能上的哈密尔顿（混合）蒙特卡罗 (Neal, 1993) 直接从边缘  $p(\mathbf{x})$  采样，避免了直接从条件  $p_{\text{mc}}(\mathbf{x} | \mathbf{h}^{(m)}, \mathbf{h}^{(c)})$  抽样。

**学生 t 分布均值乘积** 学生  $t$  分布均值乘积 ( mPoT ) 模型 (Ranzato *et al.*, 2010b) 以类似 mcRBM 扩展 cRBM 的方式扩展 PoT 模型 (Welling *et al.*, 2003a)。通过添加类似高斯 RBM 中隐藏单元的非零高斯均值来实现。与 mcRBM 一样，观察值上的 PoT 条件分布是多元高斯（具有非对角的协方差）分布；然而，不同于 mcRBM，隐藏变量的互补条件分布是由条件独立的 Gamma 分布给出。Gamma 分布  $\mathcal{G}(k, \theta)$  是关于正实数且均值为  $k\theta$  的概率分布。我们只需简单地了解 Gamma 分布就足以理解 mPoT 模型的基本思想。

mPoT 的能量函数为：

$$E_{\text{mPoT}}(\mathbf{x}, \mathbf{h}^{(m)}, \mathbf{h}^{(c)}) \quad (20.48)$$

$$= E_m(\mathbf{x}, \mathbf{h}^{(m)}) + \sum_j \left( h_j^{(c)} \left( 1 + \frac{1}{2} (\mathbf{r}^{(j)T} \mathbf{x})^2 \right) + (1 - \gamma_j) \log h_j^{(c)} \right), \quad (20.49)$$

其中  $\mathbf{r}^{(j)}$  是与单元  $h_j^{(c)}$  相关联的协方差权重向量， $E_m(\mathbf{x}, \mathbf{h}^{(m)})$  如式 (20.44) 所定义。

正如 mcRBM 一样，mPoT 模型能量函数指定一个多元高斯分布，其中关于  $\mathbf{x}$  的条件分布具有非对角的协方差。mPoT 模型中的学习（也像 mcRBM）由于无法从非对角高斯条件分布  $p_{\text{mPoT}}(\mathbf{x} | \mathbf{h}^{(m)}, \mathbf{h}^{(c)})$  采样而变得复杂。因此 Ranzato *et al.* (2010b) 也倡导通过哈密尔顿（混合）蒙特卡罗 (Neal, 1993) 直接采样  $p(\mathbf{x})$ 。

**尖峰和平板 RBM** 尖峰和平板 RBM (spike and slab RBM, ssRBM) (Courville *et al.*, 2011b) 提供对实值数据的协方差结构建模的另一种方法。与 mcRBM 相比，ssRBM 具有既不需要矩阵求逆也不需要哈密尔顿蒙特卡罗方法的优点。就像 mcRBM 和 mPoT 模型，ssRBM 的二值隐藏单元通过使用辅助实值变量来编码跨像素的条件协方差。

尖峰和平板 RBM 有两类隐藏单元：二值 尖峰 (spike) 单元  $\mathbf{h}$  和实值 平板 (slab) 单元  $\mathbf{s}$ 。条件于隐藏单元的可见单元均值由  $(\mathbf{h} \odot \mathbf{s}) \mathbf{W}^\top$  给出。换句话说，每一列  $\mathbf{W}_{:, i}$

定义当  $h_i = 1$  时可出现在输入中的分量。相应的尖峰变量  $h_i$  确定该分量是否存在。如果存在的话，相应的平板变量  $s_i$  确定该分量的强度。当尖峰变量激活时，相应的平板变量将沿着  $\mathbf{W}_{:,i}$  定义的轴的输入增加方差。这允许我们对输入的协方差建模。幸运的是，使用 Gibbs 采样的对比散度和持续性对比散度仍然适用。此处无需对任何矩阵求逆。

形式上，ssRBM 模型通过其能量函数定义：

$$E_{ss}(\mathbf{x}, \mathbf{s}, \mathbf{h}) = -\sum_i \mathbf{x}^\top \mathbf{W}_{:,i} s_i h_i + \frac{1}{2} \mathbf{x}^\top \left( \mathbf{\Lambda} + \sum_i \mathbf{\Phi}_i h_i \right) \mathbf{x} \quad (20.50)$$

$$+ \frac{1}{2} \sum_i \alpha_i s_i^2 - \sum_i \alpha_i \mu_i s_i h_i - \sum_i b_i h_i + \sum_i \alpha_i \mu_i^2 h_i, \quad (20.51)$$

其中  $b_i$  是尖峰  $h_i$  的偏置， $\mathbf{\Lambda}$  是观测值  $\mathbf{x}$  上的对角精度矩阵。参数  $\alpha_i > 0$  是实值平板变量  $s_i$  的标量精度参数。参数  $\mathbf{\Phi}_i$  是定义  $\mathbf{x}$  上的  $\mathbf{h}$  调制二次惩罚的非负对角矩阵。每个  $\mu_i$  是平板变量  $s_i$  的均值参数。

利用能量函数定义的联合分布，能相对容易地导出 ssRBM 条件分布。例如，通过边缘化平板变量  $\mathbf{s}$ ，给定二值尖峰变量  $\mathbf{h}$ ，关于观察量的条件分布由下式给出

$$p_{ss}(\mathbf{x} | \mathbf{h}) = \frac{1}{P(\mathbf{h})} \frac{1}{Z} \int \exp\{-E(\mathbf{x}, \mathbf{s}, \mathbf{h})\} d\mathbf{s} \quad (20.52)$$

$$= \mathcal{N}\left(\mathbf{x}; \mathbf{C}_{x|h}^{ss} \sum_i \mathbf{W}_{:,i} \mu_i h_i, \mathbf{C}_{x|h}^{ss}\right) \quad (20.53)$$

其中  $\mathbf{C}_{x|h}^{ss} = (\mathbf{\Lambda} + \sum_i \mathbf{\Phi}_i h_i - \sum_i \alpha_i^{-1} h_i \mathbf{W}_{:,i} \mathbf{W}_{:,i}^\top)^{-1}$ 。最后的等式只有在协方差矩阵  $\mathbf{C}_{x|h}^{ss}$  正定时成立。

由尖峰变量选通意味着  $\mathbf{h} \odot \mathbf{s}$  上的真实边缘分布是稀疏的。这不同于稀疏编码，其中来自模型的样本在编码中“几乎从不”（在测度理论意义上）包含零，并且需要MAP推断来强加稀疏性。

相比 mcRBM 和 mPoT 模型，ssRBM 以明显不同的方式参数化观察量的条件协方差。mcRBM 和 mPoT 都通过  $(\sum_j h_j^{(c)} \mathbf{r}^{(j)} \mathbf{r}^{(j)\top} + \mathbf{I})^{-1}$  建模观察量的协方差结构，使用  $h_j > 0$  的隐藏单元的激活来对方向  $\mathbf{r}^{(j)}$  的条件协方差施加约束。相反，ssRBM 使用隐藏尖峰激活  $h_i = 1$  来指定观察结果的条件协方差，以沿着由相应权重向量指定的方向捏合精度矩阵。ssRBM 条件协方差与一个不同模型给出的类似：概率主成分分析的乘积（PoPPCA）(Williams and Agakov, 2002)。在过完备的设定下，ssRBM 参数化的稀疏激活仅允许在稀疏激活  $h_i$  的所选方向上有显著方差（高

于由  $\Lambda^{-1}$  给出的近似方差)。在 mcRBM 或 mPoT 模型中, 过完备的表示意味着, 捕获观察空间中特定方向上的变化需要在该方向上的正交投影下去除潜在的所有约束。这表明这些模型不太适合于过完备设定。

尖峰和平板 RBM 的主要缺点是参数的一些设置会对应于非正定的协方差矩阵。这种协方差矩阵会在离均值更远的值上放置更大的未归一化概率, 导致所有可能结果上的积分发散。通常这个问题可以通过简单的启发式技巧来避免。理论上还没有任何令人满意的解决方法。使用约束优化来显式地避免概率未定义的区域(不过分保守是很难做到的), 并且这还会阻止模型到达参数空间的高性能区域。

定性地, ssRBM 的卷积变体能产生自然图像的优秀样本。图 16.1 中展示了一些样例。

ssRBM 允许几个扩展, 包括平板变量的高阶交互和平均池化 (Courville *et al.*, 2014) 使得模型能够在标注数据稀缺时为分类器学习到出色的特征。向能量函数添加一项能防止配分函数在稀疏编码模型下变得不确定, 如尖峰和平板稀疏编码 (Goodfellow *et al.*, 2013g), 也称为 S3C。

## 20.6 卷积玻尔兹曼机

如第九章所示, 超高维度输入(如图像)会对机器学习模型的计算、内存和统计要求造成很大的压力。通过使用小核的离散卷积来替换矩阵乘法是解决具有空间平移不变性或时间结构的输入问题的标准方式。Desjardins 和 Bengio (2008) 表明这种方法应用于 RBM 时效果很好。

深度卷积网络通常需要池化操作, 使得每个连续层的空间大小减小。前馈卷积网络通常使用池化函数, 例如池化元素的最大值。目前尚不清楚如何将其推广到基于能量的模型的设定中。我们可以在  $n$  个二值检测器单元  $\mathbf{d}$  上引入二值池化单元  $p$ , 强制  $p = \max_i d_i$ , 并且当违反约束时将能量函数设置为  $\infty$ 。因为它需要评估  $2^n$  个不同的能量设置来计算归一化常数, 这种方式不能很好地扩展。对于小的  $3 \times 3$  池化区域, 每个池化单元需要评估  $2^9 = 512$  个能量函数!

Lee *et al.* (2009) 针对这个问题, 开发了一个称为 **概率最大池化** (probabilistic max pooling) 的解决方案(不要与“随机池化”混淆, “随机池化”是用于隐含地构建卷积前馈网络集成的技术)。概率最大池化背后的策略是约束检测器单元, 使得一次最多只有一个可以处于活动状态。这意味着仅存在  $n + 1$  个总状态 ( $n$  个检测器

单元中某一个状态为开和一个对应于所有检测器单元关闭的附加状态)。当且仅当检测器单元中的一个开启时,池化单元打开。所有单元的状态关闭时,能量被分配为零。我们可以认为这是在用包含  $n + 1$  个状态的单个变量来描述模型,或者等价地具有  $n + 1$  个变量的模型,除了  $n + 1$  个联合分配的变量之外的能量赋为  $\infty$ 。

虽然高效的概率最大池化确实能强迫检测器单元互斥,这在某些情景下可能是有用的正则化约束而在其他情景下是对模型容量有害的限制。它也不支持重叠池化区域。从前馈卷积网络获得最佳性能通常需要重叠的池化区域,因此这种约束可能大大降低了卷积玻尔兹曼机的性能。

Lee *et al.* (2009) 证明概率最大池化可以用于构建卷积深度玻尔兹曼机<sup>3</sup>。该模型能够执行诸如填补输入缺失部分的操作。虽然这种模型在理论上具有吸引力,让它在实践中工作是具有挑战性的,作为分类器通常不如通过监督训练的传统卷积网络。

许多卷积模型对于许多不同空间大小的输入同样有效。对于玻尔兹曼机,由于各种原因很难改变输入尺寸。配分函数随着输入大小的改变而改变。此外,许多卷积网络按与输入大小成比例地缩放池化区域来实现尺寸不变性,但缩放玻尔兹曼机池化区域是不优雅的。传统的卷积神经网络可以使用固定数量的池化单元并且动态地增加它们池化区域的大小,以此获得可变大小输入的固定尺寸的表示。对于玻尔兹曼机,大型池化区域的计算成本比朴素方法高很多。Lee *et al.* (2009) 的方法使得每个检测器单元在相同的池化区域中互斥,解决了计算问题,但仍然不允许大小可变的池化区域。例如,假设我们在学习边缘检测器时,检测器单元上具有  $2 \times 2$  的概率最大池化。这强制约束在每个  $2 \times 2$  的区域中只能出现这些边中的一条。如果我们随后在每个方向上将输入图像的大小增加 50%,则期望边缘的数量会相应地增加。相反,如果我们在每个方向上将池化区域的大小增加 50% 到  $3 \times 3$ ,则互斥性约束现在指定这些边中的每一个在  $3 \times 3$  区域中仅可以出现一次。当我们以这种方式增长模型的输入图像时,模型会生成密度较小的边。当然,这些问题只有在模型必须使用可变数量的池化,以便产出固定大小的输出向量时才会出现。只要模型的输出是可以与输入图像成比例缩放的特征图,使用概率最大池化的模型仍然可以接受可变大小的输入图像。

图像边界处的像素也带来一些困难,由于玻尔兹曼机中的连接是对称的事实而加剧。如果我们不隐式地补零输入,则将会导致比可见单元更少的隐藏单元,并且图像边界处的可见单元将不能被良好地建模,因为它们位于较少隐藏单元的接受场

<sup>3</sup>该论文将模型描述为“深度信念网络”,但因为它可以被描述为纯无向模型(具有易处理逐层均匀场不动点更新),所以它最适合深度玻尔兹曼机的定义。

中。然而，如果我们隐式地补零输入，则边界处的隐藏单元将由较少的输入像素驱动，并且可能在需要时无法激活。

## 20.7 用于结构化或序列输出的玻尔兹曼机

在结构化输出场景中，我们希望训练可以从一些输入  $\mathbf{x}$  映射到一些输出  $\mathbf{y}$  的模型， $\mathbf{y}$  的不同条目彼此相关，并且必须遵守一些约束。例如，在语音合成任务中， $\mathbf{y}$  是波形，并且整个波形听起来必须像连贯的发音。

表示  $\mathbf{y}$  中的条目之间关系的自然方式是使用概率分布  $p(\mathbf{y} \mid \mathbf{x})$ 。扩展到建模条件分布的玻尔兹曼机可以支持这种概率模型。

使用玻尔兹曼机条件建模的相同工具不仅可以用于结构化输出任务，还可以用于序列建模。在后一种情况下，模型必须估计变量序列上的概率分布  $p(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(\tau)})$ ，而不仅仅是将输入  $\mathbf{x}$  映射到输出  $\mathbf{y}$ 。为完成这个任务，条件玻尔兹曼机可以表示  $p(\mathbf{x}^{(\tau)} \mid \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(\tau-1)})$  形式的因子。

视频游戏和电影工业中一个重要序列建模任务是建模用于渲染 3-D 人物骨架关节角度的序列。这些序列通常通过记录角色移动的运动捕获系统收集。人物运动的概率模型允许生成新的（之前没见过的）但真实的动画。为了解决这个序列建模任务，Taylor *et al.* (2007) 针对小的  $m$  引入了条件 RBM 建模  $p(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-1)}, \dots, \mathbf{x}^{(t-m)})$ 。该模型是  $p(\mathbf{x}^{(t)})$  上的 RBM，其偏置参数是  $\mathbf{x}$  前面  $m$  个值的线性函数。当我们条件于  $\mathbf{x}^{(t-1)}$  的不同值和更早的变量时，我们会得到一个关于  $\mathbf{x}$  的新 RBM。RBM 关于  $\mathbf{x}$  的权重不会改变，但是条件于不同的过去值，我们可以改变 RBM 中的不同隐藏单元处于活动状态的概率。通过激活和去激活隐藏单元的不同子集，我们可以对  $\mathbf{x}$  上诱导的概率分布进行大的改变。条件 RBM 的其他变体 (Mnih *et al.*, 2011) 和使用条件 RBM 进行序列建模的其他变体是可能的 (Taylor and Hinton, 2009; Sutskever *et al.*, 2009; Boulanger-Lewandowski *et al.*, 2012)。

另一个序列建模任务是对构成歌曲音符序列的分布进行建模。Boulanger-Lewandowski *et al.* (2012) 引入了 **RNN-RBM** 序列模型并应用于这个任务。RNN-RBM 由 RNN（产生用于每个时间步的 RBM 参数）组成，是帧序列  $\mathbf{x}^{(t)}$  的生成模型。与之前只有 RBM 的偏置参数会在一个时间步到下一个发生变化的方法不同，RNN-RBM 使用 RNN 来产生 RBM 的所有参数（包括权重）。为了训练模型，我们需要能够通过 RNN 反向传播损失函数的梯度。损失函数不直接应用于 RNN 输出。

相反，它应用于 RBM。这意味着我们必须使用对比散度或相关算法关于 RBM 参数进行近似的微分。然后才可以使用通常的通过时间反向传播算法通过 RNN 反向传播该近似梯度。

## 20.8 其他玻尔兹曼机

玻尔兹曼机的许多其他变种是可能的。

玻尔兹曼机可以用不同的训练准则扩展。我们专注于训练为大致最大化生成标准  $\log p(\mathbf{v})$  的玻尔兹曼机。相反，旨在最大化  $\log p(y \mid \mathbf{v})$  来训练判别的 RBM 也是有可能的 (Larochelle and Bengio, 2008a)。当使用生成性和判别性标准的线性组合时，该方法通常表现最好。不幸的是，至少使用现有的方法来看，RBM 似乎并不如 MLP 那样的监督学习器强大。

在实践中使用的大多数玻尔兹曼机在其能量函数中仅具有二阶相互作用，意味着它们的能量函数是许多项的和，并且每个单独项仅包括两个随机变量之间的乘积。这种项的一个例子是  $v_i W_{i,j} h_j$ 。我们还可以训练高阶玻尔兹曼机 (Sejnowski, 1987)，其中能量函数项涉及许多变量的乘积。隐藏单元和两个不同图像之间的三向交互可以建模从一个视频帧到下一个帧的空间变换 (Memisevic and Hinton, 2007, 2010)。通过one-hot类别变量的乘法可以根据存在哪个类来改变可见单元和隐藏单元之间的关系 (Nair and Hinton, 2009)。使用高阶交互的一个最近的示例是具有两组隐藏单元的玻尔兹曼机，一组同时与可见单元  $\mathbf{v}$  和类别标签  $y$  交互，另一组仅与输入值  $\mathbf{v}$  交互 (Luo *et al.*, 2011)。这可以被解释为鼓励一些隐藏单元学习使用与类相关的特征来建模输入，而且还学习额外的隐藏单元（不需要根据样本类别，学习逼真  $\mathbf{v}$  样本所需的繁琐细节）。高阶交互的另一个用途是选通一些特征。Sohn *et al.* (2013) 介绍了一个带有三阶交互的玻尔兹曼机，以及与每个可见单元相关的二进制掩码变量。当这些掩码变量设置为零时，它们消除可见单元对隐藏单元的影响。这允许将与分类问题不相关的可见单元从估计类别的推断路径中移除。

更一般地说，玻尔兹曼机框架是一个丰富的模型空间，允许比迄今为止已经探索的更多的模型结构。开发新形式的玻尔兹曼机相比于开发新的神经网络层需要更多细心和创造力，因为它通常很难找到一个能保持玻尔兹曼机所需的所有不同条件分布的可解性的能量函数。尽管这需要努力，该领域仍对创新开放。

## 20.9 通过随机操作的反向传播

传统的神经网络对一些输入变量  $\mathbf{x}$  施加确定性变换。当开发生成模型时，我们经常希望扩展神经网络以实现  $\mathbf{x}$  的随机变换。这样做的一个直接方法是使用额外输入  $\mathbf{z}$ （从一些简单的概率分布采样得到，如均匀或高斯分布）来增强神经网络。神经网络在内部仍可以继续执行确定性计算，但是函数  $f(\mathbf{x}, \mathbf{z})$  对于不能访问  $\mathbf{z}$  的观察者来说将是随机的。假设  $f$  是连续可微的，我们可以像往常一样使用反向传播计算训练所需的梯度。

作为示例，让我们考虑从均值  $\mu$  和方差  $\sigma^2$  的高斯分布中采样  $y$  的操作：

$$y \sim \mathcal{N}(\mu, \sigma^2). \quad (20.54)$$

因为  $y$  的单个样本不是由函数产生的，而是由一个采样过程产生，它的输出会随我们的每次查询变化，所以取  $y$  相对于其分布的参数  $\mu$  和  $\sigma^2$  的导数似乎是违反直觉的。然而，我们可以将采样过程重写，对基本随机变量  $\mathbf{z} \sim \mathcal{N}(0, 1)$  进行转换以从期望的分布获得样本：

$$y = \mu + \sigma z. \quad (20.55)$$

现在我们将其视为具有额外输入  $\mathbf{z}$  的确定性操作，可以通过采样操作来反向传播。至关重要的是，额外输入是一个随机变量，其分布不是任何我们想对其计算导数的变量的函数。如果我们可以用相同的  $\mathbf{z}$  值再次重复采样操作，结果会告诉我们  $\mu$  或  $\sigma$  的微小变化将会如何改变输出。

能够通过该采样操作反向传播允许我们将其并入更大的图中。我们可以在采样分布的输出之上构建图元素。例如，我们可以计算一些损失函数  $J(y)$  的导数。我们还可以构建这样的图元素，其输出是采样操作的输入或参数。例如，我们可以通过  $\mu = f(\mathbf{x}; \boldsymbol{\theta})$  和  $\sigma = g(\mathbf{x}; \boldsymbol{\theta})$  构建更大的图。在这个增强图中，我们可以通过这些函数的反向传播导出  $\nabla_{\boldsymbol{\theta}} J(y)$ 。

在该高斯采样示例中使用的原理能更广泛地应用。我们可以将任何形为  $p(\mathbf{y}; \boldsymbol{\theta})$  或  $p(\mathbf{y} | \mathbf{x}; \boldsymbol{\theta})$  的概率分布表示为  $p(\mathbf{y} | \boldsymbol{\omega})$ ，其中  $\boldsymbol{\omega}$  是同时包含参数  $\boldsymbol{\theta}$  和输入  $\mathbf{x}$  的变量（如果适用的话）。给定从分布  $p(\mathbf{y} | \boldsymbol{\omega})$  采样的值  $\mathbf{y}$ （其中  $\boldsymbol{\omega}$  可以是其他变量的函数），我们可以将

$$\mathbf{y} \sim p(\mathbf{y} | \boldsymbol{\omega}) \quad (20.56)$$

重写为

$$\mathbf{y} = f(\mathbf{z}; \boldsymbol{\omega}), \quad (20.57)$$

其中  $\mathbf{z}$  是随机性的来源。只要  $f$  是几乎处处连续可微的，我们就可以使用传统工具（例如应用于  $f$  的反向传播算法）计算  $\mathbf{y}$  相对于  $\boldsymbol{\omega}$  的导数。至关重要的是， $\boldsymbol{\omega}$  不能是  $\mathbf{z}$  的函数，且  $\mathbf{z}$  不能是  $\boldsymbol{\omega}$  的函数。这种技术通常被称为**重参数化技巧** (reparametrization trick)、**随机反向传播**(stochastic back-propagation) 或**扰动分析** (perturbation analysis)。

要求  $f$  是连续可微的，当然需要  $\mathbf{y}$  是连续的。如果我们希望通过产生离散值样本的采样过程进行反向传播，则可以使用强化学习算法（如 REINFORCE 算法 (Williams, 1992) 的变体）来估计  $\boldsymbol{\omega}$  上的梯度，这将在第 20.9.1 节中讨论。

在神经网络应用中，我们通常选择从一些简单的分布中采样  $\mathbf{z}$ ，如单位均匀分布或单位高斯分布，并通过网络的确定性部分重塑其输入来实现更复杂的分布。

通过随机操作扩展梯度或优化的想法可追溯到二十世纪中叶 (Price, 1958; Bonnet, 1964)，并且首先在强化学习 (Williams, 1992) 的情景下用于机器学习。最近，它已被应用于变分近似 (Opper and Archambeau, 2009) 和随机生成神经网络 (Bengio *et al.*, 2013b; Kingma, 2013; Kingma and Welling, 2014b,a; Rezende *et al.*, 2014; Goodfellow *et al.*, 2014c)。许多网络，如去噪自编码器或使用 Dropout 的正则化网络，也被自然地设计为将噪声作为输入，而不需要任何特殊的重参数化就能使噪声独立于模型。

### 20.9.1 通过离散随机操作的反向传播

当模型发射离散变量  $\mathbf{y}$  时，重参数化技巧不再适用。假设模型采用输入  $\mathbf{x}$  和参数  $\boldsymbol{\theta}$ ，两者都封装在向量  $\boldsymbol{\omega}$  中，并且将它们与随机噪声  $\mathbf{z}$  组合以产生  $\mathbf{y}$ :

$$\mathbf{y} = f(\mathbf{z}; \boldsymbol{\omega}). \quad (20.58)$$

因为  $\mathbf{y}$  是离散的， $f$  必须是一个阶跃函数。阶跃函数的导数在任何点都是没用的。在每个阶跃边界，导数是未定义的，但这是一个小问题。大问题是导数在阶跃边界之间的区域几乎处处为零。因此，任何代价函数  $J(\mathbf{y})$  的导数无法给出如何更新模型参数  $\boldsymbol{\theta}$  的任何信息。

REINFORCE 算法 (REward Increment = nonnegative Factor  $\times$  Offset Reinforcement  $\times$  Characteristic Eligibility) 提供了定义一系列简单而强大解决方案的框架 (Williams, 1992)。其核心思想是, 即使  $J(f(\mathbf{z}; \boldsymbol{\omega}))$  是具有无用导数的阶跃函数, 期望代价  $\mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} J(f(\mathbf{z}; \boldsymbol{\omega}))$  通常是服从梯度下降的光滑函数。虽然当  $\mathbf{y}$  是高维 (或者是许多离散随机决策组合的结果) 时, 该期望通常是难解的, 但我们可以使用蒙特卡罗平均进行无偏估计。梯度的随机估计可以与 SGD 或其他基于随机梯度的优化技术一起使用。

通过简单地微分期望成本, 我们可以推导出 REINFORCE 最简单的版本:

$$\mathbb{E}_{\mathbf{z}}[J(\mathbf{y})] = \sum_{\mathbf{y}} J(\mathbf{y})p(\mathbf{y}), \quad (20.59)$$

$$\frac{\partial \mathbb{E}[J(\mathbf{y})]}{\partial \boldsymbol{\omega}} = \sum_{\mathbf{y}} J(\mathbf{y}) \frac{\partial p(\mathbf{y})}{\partial \boldsymbol{\omega}} \quad (20.60)$$

$$= \sum_{\mathbf{y}} J(\mathbf{y})p(\mathbf{y}) \frac{\partial \log p(\mathbf{y})}{\partial \boldsymbol{\omega}} \quad (20.61)$$

$$\approx \frac{1}{m} \sum_{\mathbf{y}^{(i)} \sim p(\mathbf{y}), i=1}^m J(\mathbf{y}^{(i)}) \frac{\partial \log p(\mathbf{y}^{(i)})}{\partial \boldsymbol{\omega}}. \quad (20.62)$$

式 (20.60) 依赖于  $J$  不直接引用  $\boldsymbol{\omega}$  的假设。放松这个假设来扩展该方法是简单的。式 (20.61) 利用对数的导数规则,  $\frac{\partial \log p(\mathbf{y})}{\partial \boldsymbol{\omega}} = \frac{1}{p(\mathbf{y})} \frac{\partial p(\mathbf{y})}{\partial \boldsymbol{\omega}}$ 。式 (20.62) 给出了该梯度的无偏蒙特卡罗估计。

在本节中我们写的  $p(\mathbf{y})$ , 可以等价地写成  $p(\mathbf{y} | \mathbf{x})$ 。这是因为  $p(\mathbf{y})$  由  $\boldsymbol{\omega}$  参数化, 并且如果  $\mathbf{x}$  存在,  $\boldsymbol{\omega}$  包含  $\boldsymbol{\theta}$  和  $\mathbf{x}$  两者。

简单 REINFORCE 估计的一个问题是其具有非常高的方差, 需要采  $\mathbf{y}$  的许多样本才能获得对梯度的良好估计, 或者等价地, 如果仅绘制一个样本, SGD 将收敛得非常缓慢并将需要较小的学习率。通过使用 **方差减小** (variance reduction) 方法 (Wilson, 1984; L'Ecuyer, 1994), 可以减少该估计的方差。想法是修改估计量, 使其预期值保持不变, 但方差减小。在 REINFORCE 的情况下提出的方差减小方法, 涉及计算用于偏移  $J(\mathbf{y})$  的基线 (baseline)。注意, 不依赖于  $\mathbf{y}$  的任何偏移  $b(\mathbf{w})$

都不会改变估计梯度的期望，因为

$$E_{p(\mathbf{y})} \left[ \frac{\partial \log p(\mathbf{y})}{\partial \boldsymbol{\omega}} \right] = \sum_{\mathbf{y}} p(\mathbf{y}) \frac{\partial \log p(\mathbf{y})}{\partial \boldsymbol{\omega}} \quad (20.63)$$

$$= \sum_{\mathbf{y}} \frac{\partial p(\mathbf{y})}{\partial \boldsymbol{\omega}} \quad (20.64)$$

$$= \frac{\partial}{\partial \boldsymbol{\omega}} \sum_{\mathbf{y}} p(\mathbf{y}) = \frac{\partial}{\partial \boldsymbol{\omega}} 1 = 0, \quad (20.65)$$

这意味着

$$E_{p(\mathbf{y})} \left[ (J(\mathbf{y}) - b(\boldsymbol{\omega})) \frac{\partial \log p(\mathbf{y})}{\partial \boldsymbol{\omega}} \right] = E_{p(\mathbf{y})} \left[ J(\mathbf{y}) \frac{\partial \log p(\mathbf{y})}{\partial \boldsymbol{\omega}} \right] - b(\boldsymbol{\omega}) E_{p(\mathbf{y})} \left[ \frac{\partial \log p(\mathbf{y})}{\partial \boldsymbol{\omega}} \right] \quad (20.66)$$

$$= E_{p(\mathbf{y})} \left[ J(\mathbf{y}) \frac{\partial \log p(\mathbf{y})}{\partial \boldsymbol{\omega}} \right]. \quad (20.67)$$

此外，我们可以通过计算  $(J(\mathbf{y}) - b(\boldsymbol{\omega})) \frac{\partial \log p(\mathbf{y})}{\partial \boldsymbol{\omega}}$  关于  $p(\mathbf{y})$  的方差，并关于  $b(\boldsymbol{\omega})$  最小化获得最优  $b(\boldsymbol{\omega})$ 。我们发现这个最佳基线  $b^*(\boldsymbol{\omega})_i$  对于向量  $\boldsymbol{\omega}$  的每个元素  $\omega_i$  是不同的：

$$b^*(\boldsymbol{\omega})_i = \frac{E_{p(\mathbf{y})} \left[ J(\mathbf{y}) \frac{\partial \log p(\mathbf{y})^2}{\partial \omega_i} \right]}{E_{p(\mathbf{y})} \left[ \frac{\partial \log p(\mathbf{y})^2}{\partial \omega_i} \right]}. \quad (20.68)$$

相对于  $\omega_i$  的梯度估计则变为

$$(J(\mathbf{y}) - b(\boldsymbol{\omega})_i) \frac{\partial \log p(\mathbf{y})}{\partial \omega_i}, \quad (20.69)$$

其中  $b(\boldsymbol{\omega})_i$  估计上述  $b^*(\boldsymbol{\omega})_i$ 。获得估计  $b$  通常需要将额外输出添加到神经网络，并训练新输出对  $\boldsymbol{\omega}$  的每个元素估计  $E_{p(\mathbf{y})}[J(\mathbf{y}) \frac{\partial \log p(\mathbf{y})^2}{\partial \omega_i}]$  和  $E_{p(\mathbf{y})}[\frac{\partial \log p(\mathbf{y})^2}{\partial \omega_i}]$ 。这些额外的输出可以用均方误差目标训练，对于给定的  $\boldsymbol{\omega}$ ，从  $p(\mathbf{y})$  采样  $\mathbf{y}$  时，分别用  $J(\mathbf{y}) \frac{\partial \log p(\mathbf{y})^2}{\partial \omega_i}$  和  $\frac{\partial \log p(\mathbf{y})^2}{\partial \omega_i}$  作目标。然后可以将这些估计代入式 (20.68) 就能恢复估计  $b$ 。Mnih and Gregor (2014) 倾向于使用通过目标  $J(\mathbf{y})$  训练的单个共享输出（跨越  $\boldsymbol{\omega}$  的所有元素  $i$ ），并使用  $b(\boldsymbol{\omega}) \approx E_{p(\mathbf{y})}[J(\mathbf{y})]$  作为基线。

在强化学习背景下引入的方差减小方法 (Sutton *et al.*, 2000; Weaver and Tao, 2001), Dayan (1990) 推广了二值奖励的前期工作。可以参考 Bengio *et al.* (2013b)、

Mnih and Gregor (2014)、Ba *et al.* (2014)、Mnih *et al.* (2014) 或 Xu *et al.* (2015) 中在深度学习的背景下使用减少方差的 REINFORCE 算法的现代例子。除了使用与输入相关的基线  $b(\omega)$ , Mnih and Gregor (2014) 发现可以在训练期间调整 ( $J(\mathbf{y}) - b(\omega)$ ) 的尺度 (即除以训练期间的移动平均估计的标准差), 即作为一种适应性学习率, 可以抵消训练过程中该量大小发生的重要变化的影响。Mnih and Gregor (2014) 称之为启发式方差归一化 (variance normalization)。

基于 REINFORCE 的估计器可以被理解为将  $\mathbf{y}$  的选择与  $J(\mathbf{y})$  的对应值相关联来估计梯度。如果在当前参数化下不太可能出现  $\mathbf{y}$  的良好值, 则可能需要很长时间来偶然获得它, 并且获得所需信号的配置应当被加强。

## 20.10 有向生成网络

如第十六章所讨论的, 有向图模型构成了一类突出的图模型。虽然有向图模型在更大的机器学习社群中非常流行, 但在较小的深度学习社群中, 大约直到 2013 年它们都掩盖在无向模型 (如 RBM) 的光彩之下。

在本节中, 我们回顾一些传统上与深度学习社群相关的标准有向图模型。

我们已经描述过部分有向的模型——深度信念网络。我们还描述过可以被认为是浅度有向生成模型的稀疏编码模型。尽管在样本生成和密度估计方面表现不佳, 在深度学习的背景下它们通常被用作特征学习器。我们接下来描述多种深度完全有向的模型。

### 20.10.1 sigmoid 信念网络

sigmoid 信念网络 (Neal, 1990) 是一种具有特定条件概率分布的有向图模型的简单形式。一般来说, 我们可以将 sigmoid 信念网络视为具有二值向量的状态  $\mathbf{s}$ , 其中状态的每个元素都受其祖先影响:

$$p(s_i) = \sigma\left(\sum_{j < i} W_{j,i}s_j + b_i\right). \quad (20.70)$$

sigmoid 信念网络最常见的结构是被分为许多层的结构, 其中原始采样通过一系列多个隐藏层进行, 然后最终生成可见层。这种结构与深度信念网络非常相似, 但它们在采样过程开始时的单元彼此独立, 而不是从受限玻尔兹曼机采样。这种结构

由于各种原因而令人感兴趣。一个原因是该结构是可见单元上概率分布的通用近似，即在足够深的情况下，可以任意良好地近似二值变量的任何概率分布（即使各个层的宽度受限于可见层的维度）(Sutskever and Hinton, 2008)。

虽然生成可见单元的样本在 sigmoid 信念网络中是非常高效的，但是其他大多数操作不是很高效。给定可见单元，对隐藏单元的推断是难解的。因为变分下界涉及对包含整个层的团求期望，均匀场推断也是难以处理的。这个问题一直困难到足以限制有向离散网络的普及。

在 sigmoid 信念网络中执行推断的一种方法是构造专用于 sigmoid 信念网络的不同下界 (Saul *et al.*, 1996)。这种方法只适用于非常小的网络。另一种方法是使用学成推断机制，如第 19.5 节中描述的。Helmholtz 机 (Dayan *et al.*, 1995; Dayan and Hinton, 1996) 结合了一个 sigmoid 信念网络与一个预测隐藏单元上均匀场分布参数的推断网络。sigmoid 信念网络的现代方法 (Gregor *et al.*, 2014; Mnih and Gregor, 2014) 仍然使用这种推断网络的方法。因为潜变量的离散本质，这些技术仍然是困难的。人们不能简单地通过推断网络的输出反向传播，而必须使用相对不可靠的机制即通过离散采样过程进行反向传播（如第 20.9.1 节所述）。最近基于重要采样、重加权的睡眠(Bornschein and Bengio, 2015) 或双向 Helmholtz 机 (Bornstein *et al.*, 2015) 的方法使得我们可以快速训练 sigmoid 信念网络，并在基准任务上达到最好的表现。

sigmoid 信念网络的一种特殊情况是没有潜变量的情况。在这种情况下学习是高效的，因为没有必要将潜变量边缘化到似然之外。一系列称为自回归网络的模型将这个完全可见的信念网络泛化到其他类型的变量（除二值变量）和其他结构（除对数线性关系）的条件分布。自回归网络将在第 20.10.7 节中描述。

### 20.10.2 可微生成器网络

许多生成模型基于使用可微生成器网络 (generator network) 的想法。这种模型使用可微函数  $g(\mathbf{z}; \theta^{(g)})$  将潜变量  $\mathbf{z}$  的样本变换为样本  $\mathbf{x}$  或样本  $\mathbf{x}$  上的分布，可微函数通常可以由神经网络表示。这类模型包括将生成器网络与推断网络配对的变分自编码器、将生成器网络与判别器网络配对的生成式对抗网络，以及孤立地训练生成器网络的技术。

生成器网络本质上仅是用于生成样本的参数化计算过程，其中的体系结构提供了从中采样的可能分布族以及选择这些族内分布的参数。

作为示例，从具有均值  $\mu$  和协方差  $\Sigma$  的正态分布绘制样本的标准过程是将来自零均值和单位协方差的正态分布的样本  $z$  馈送到非常简单的生成器网络中。这个生成器网络只包含一个仿射层：

$$\mathbf{x} = g(\mathbf{z}) = \boldsymbol{\mu} + \mathbf{L}\mathbf{z}, \quad (20.71)$$

其中  $\mathbf{L}$  由  $\Sigma$  的 Cholesky 分解给出。

伪随机数发生器也可以使用简单分布的非线性变换。例如，逆变换采样 (inverse transform sampling) (Devroye, 2013) 从  $U(0, 1)$  中采一个标量  $z$ ，并且对标量  $x$  应用非线性变换。在这种情况下， $g(z)$  由累积分布函数  $F(x) = \int_{-\infty}^x p(v)dv$  的反函数给出。如果我们能够指定  $p(x)$ ，在  $x$  上积分，并取所得函数的反函数，我们不用通过机器学习就能从  $p(x)$  进行采样。

为了从更复杂的分布（难以直接指定、难以积分或难以求所得积分的反函数）中生成样本，我们使用前馈网络来表示非线性函数  $g$  的参数族，并使用训练数据来推断参数以选择所期望的函数。

我们可以认为  $g$  提供了变量的非线性变化，将  $\mathbf{z}$  上的分布转换成  $\mathbf{x}$  上想要的分布。

回顾式 (3.47)，对于可求反函数的、可微的、连续的  $g$ ，

$$p_z(\mathbf{z}) = p_x(g(\mathbf{z})) \left| \det\left(\frac{\partial g}{\partial \mathbf{z}}\right) \right|. \quad (20.72)$$

这隐含地对  $\mathbf{x}$  施加概率分布：

$$p_x(\mathbf{x}) = \frac{p_z(g^{-1}(\mathbf{x}))}{\left| \det\left(\frac{\partial g}{\partial \mathbf{z}}\right) \right|}. \quad (20.73)$$

当然，取决于  $g$  的选择，这个公式可能难以评估，因此我们经常需要使用间接学习  $g$  的方法，而不是直接尝试最大化  $\log p(\mathbf{x})$ 。

在某些情况下，我们使用  $g$  来定义  $\mathbf{x}$  上的条件分布，而不是使用  $g$  直接提供  $\mathbf{x}$  的样本。例如，我们可以使用一个生成器网络，其最后一层由 sigmoid 输出组成，可以提供 Bernoulli 分布的平均参数：

$$p(\mathbf{x}_i = 1 | \mathbf{z}) = g(\mathbf{z})_i. \quad (20.74)$$

在这种情况下，我们使用  $g$  来定义  $p(\mathbf{x} | \mathbf{z})$  时，我们通过边缘化  $\mathbf{z}$  来对  $\mathbf{x}$  施加分布：

$$p(\mathbf{x}) = \mathbb{E}_{\mathbf{z}} p(\mathbf{x} | \mathbf{z}). \quad (20.75)$$

两种方法都定义了一个分布  $p_g(\mathbf{x})$ ，并允许我们使用第 20.9 节中的重参数化技巧来训练  $p_g$  的各种评估准则。

表示生成器网络的两种不同方法（发出条件分布的参数相对直接发射样品）具有互补的优缺点。当生成器网络在  $\mathbf{x}$  上定义条件分布时，它不但能生成连续数据，也能生成离散数据。当生成器网络直接提供采样时，它只能产生连续的数据（我们可以在前向传播中引入离散化，但这样做意味着模型不再能够使用反向传播进行训练）。直接采样的优点是，我们不再被迫使用条件分布（可以容易地写出来并由人类设计者进行代数操作的形式）。

基于可微生成器网络的方法是由分类可微前馈网络中梯度下降的成功应用而推动的。在监督学习的背景下，基于梯度训练学习的深度前馈网络在给定足够的隐藏单元和足够的训练数据的情况下，在实践中似乎能保证成功。这个同样的方案能成功转移到生成式建模上吗？

生成式建模似乎比分类或回归更困难，因为学习过程需要优化难以处理的准则。在可微生成器网络的情况下，准则是难以处理的，因为数据不指定生成器网络的输入  $\mathbf{z}$  和输出  $\mathbf{x}$ 。在监督学习的情况下，输入  $\mathbf{x}$  和输出  $\mathbf{y}$  同时给出，并且优化过程只需学习如何产生指定的映射。在生成建模的情况下，学习过程需要确定如何以有用的方式排布  $\mathbf{z}$  空间，以及额外的如何从  $\mathbf{z}$  映射到  $\mathbf{x}$ 。

Dosovitskiy *et al.* (2015) 研究了一个简化问题，其中  $\mathbf{z}$  和  $\mathbf{x}$  之间的对应关系已经给出。具体来说，训练数据是计算机渲染的椅子图。潜变量  $\mathbf{z}$  是渲染引擎的参数，描述了椅子模型的选择、椅子的位置以及影响图像渲染的其他配置细节。使用这种合成的生成数据，卷积网络能够学习将图像内容的描述  $\mathbf{z}$  映射到渲染图像的近似  $\mathbf{x}$ 。这表明当现代可微生成器网络具有足够的模型容量时，足以成为良好的生成模型，并且现代优化算法具有拟合它们的能力。困难在于当每个  $\mathbf{x}$  的  $\mathbf{z}$  的值不是固定的且在每次训练前是未知时，如何训练生成器网络。

在接下来的章节中，我们讨论仅给出  $\mathbf{x}$  的训练样本，训练可微生成器网络的几种方法。

### 20.10.3 变分自编码器

变分自编码器 (variational auto-encoder, VAE) (Kingma, 2013; Rezende *et al.*, 2014) 是一个使用学好的近似推断的有向模型，可以纯粹地使用基于梯度的方法进行

训练。

为了从模型生成样本，VAE 首先从编码分布  $p_{\text{model}}(\mathbf{z})$  中采样  $\mathbf{z}$ 。然后使样本通过可微生成器网络  $g(\mathbf{z})$ 。最后，从分布  $p_{\text{model}}(\mathbf{x}; g(\mathbf{z})) = p_{\text{model}}(\mathbf{x} | \mathbf{z})$  中采样  $\mathbf{x}$ 。然而在训练期间，近似推断网络（或编码器） $q(\mathbf{z} | \mathbf{x})$  用于获得  $\mathbf{z}$ ，而  $p_{\text{model}}(\mathbf{x} | \mathbf{z})$  则被视为解码器网络。

变分自编码器背后的关键思想是，它们可以通过最大化与数据点  $\mathbf{x}$  相关联的变分下界  $\mathcal{L}(q)$  来训练：

$$\mathcal{L}(q) = \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z} | \mathbf{x})} \log p_{\text{model}}(\mathbf{z}, \mathbf{x}) + \mathcal{H}(q(\mathbf{z} | \mathbf{x})) \quad (20.76)$$

$$= \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z} | \mathbf{x})} \log p_{\text{model}}(\mathbf{x} | \mathbf{z}) - D_{\text{KL}}(q(\mathbf{z} | \mathbf{x}) || p_{\text{model}}(\mathbf{z})) \quad (20.77)$$

$$\leq \log p_{\text{model}}(\mathbf{x}). \quad (20.78)$$

在式 (20.76) 中，我们将第一项视为潜变量的近似后验下可见和隐藏变量的联合对数似然性（正如 EM 一样，不同的是我们使用近似而不是精确后验）。第二项则可视为近似后验的熵。当  $q$  被选择为高斯分布，其中噪声被添加到预测平均值时，最大化该熵项促使该噪声标准偏差的增加。更一般地，这个熵项鼓励变分后验将高概率质量置于可能已经产生  $\mathbf{x}$  的许多  $\mathbf{z}$  值上，而不是坍缩到单个估计最可能值的点。在式 (20.77) 中，我们将第一项视为在其他自编码器中出现的重构对数似然。第二项试图使近似后验分布  $q(\mathbf{z} | \mathbf{x})$  和模型先验  $p_{\text{model}}(\mathbf{z})$  彼此接近。

变分推断和学习的传统方法是通过优化算法推断  $q$ ，通常是迭代不动点方程（第 19.4 节）。这些方法是缓慢的，并且通常需要以闭解形式计算  $\mathbb{E}_{\mathbf{z} \sim q} \log p_{\text{model}}(\mathbf{z}, \mathbf{x})$ 。变分自编码器背后的主要思想是训练产生  $q$  参数的参数编码器（有时也称为推断网络或识别模型）。只要  $\mathbf{z}$  是连续变量，我们就可以通过从  $q(\mathbf{z} | \mathbf{x}) = q(\mathbf{z}; f(\mathbf{x}; \boldsymbol{\theta}))$  中采样  $\mathbf{z}$  的样本反向传播，以获得相对于  $\boldsymbol{\theta}$  的梯度。学习则仅包括相对于编码器和解码器的参数最大化  $\mathcal{L}$ 。 $\mathcal{L}$  中的所有期望都可以通过蒙特卡罗采样来近似。

变分自编码器方法是优雅的，理论上令人愉快的，并且易于实现。它也获得了出色的结果，是生成式建模中的最先进方法之一。它的主要缺点是从在图像上训练的变分自编码器中采样的样本往往有些模糊。这种现象的原因尚不清楚。一种可能性是模糊性是最大似然的固有效应，因为我们需要最小化  $D_{\text{KL}}(p_{\text{data}} || p_{\text{model}})$ 。如图 3.6 所示，这意味着模型将为训练集中出现的点分配高的概率，但也可能为其他点分配高的概率。还有其他原因可以导致模糊图像。模型选择将概率质量置于模糊图像而不是空间的其他部分的部分原因是实际使用的变分自编码器通常在  $p_{\text{model}}(\mathbf{x}; g(\mathbf{z}))$  使用高斯分布。最大化这种分布似然性的下界与训练具有均方误差的传统自编码器类似，

这意味着它倾向于忽略由少量像素表示的特征或其中亮度变化微小的像素。如 Theis *et al.* (2015) 和 Huszar (2015) 指出的，该问题不是 VAE 特有的，而是与优化对数似然或  $D_{\text{KL}}(p_{\text{data}} \parallel p_{\text{model}})$  的生成模型共享的。现代 VAE 模型另一个麻烦的问题是，它们倾向于仅使用  $z$  维度中的小子集，就像编码器不能够将具有足够局部方向的输入空间变换到边缘分布与分解前匹配的空间。

VAE 框架可以直接扩展到大范围的模型架构。相比玻尔兹曼机，这是关键的优势，因为玻尔兹曼机需要非常仔细地设计模型来保持易解性。VAE 可以与广泛的可微算子族一起良好工作。一个特别复杂的 VAE 是深度循环注意写者 (DRAW) 模型 (Gregor *et al.*, 2015)。DRAW 使用一个循环编码器和循环解码器并结合注意力机制。DRAW 模型的生成过程包括顺序访问不同的小图像块并绘制这些点处的像素值。我们还可以通过在 VAE 框架内使用循环编码器和解码器来定义变分 RNN (Chung *et al.*, 2015b) 来扩展 VAE 以生成序列。从传统 RNN 生成样本仅在输出空间涉及非确定性操作。而变分 RNN 还具有由 VAE 潜变量捕获的潜在更抽象层的随机变化性。

VAE 框架已不仅仅扩展到传统的变分下界，还有 **重要加权自编码器**(importance-weighted autoencoder)(Burda *et al.*, 2015) 的目标：

$$\mathcal{L}_k(\mathbf{x}, q) = \mathbb{E}_{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(k)} \sim q(\mathbf{z}|\mathbf{x})} \left[ \log \frac{1}{k} \sum_{i=1}^k \frac{p_{\text{model}}(\mathbf{x}, \mathbf{z}^{(i)})}{q(\mathbf{z}^{(i)} | \mathbf{x})} \right]. \quad (20.79)$$

这个新的目标在  $k = 1$  时等同于传统的下界  $\mathcal{L}$ 。然而，它也可以被解释为基于提议分布  $q(\mathbf{z} | \mathbf{x})$  中  $z$  的重要采样而形成的真实  $\log p_{\text{model}}(\mathbf{x})$  估计。重要加权自编码器目标也是  $\log p_{\text{model}}(\mathbf{x})$  的下界，并且随着  $k$  增加而变得更紧。

变分自编码器与 MP-DBM 和其他涉及通过近似推断图的反向传播方法有一些有趣的联系 (Goodfellow *et al.*, 2013d; Stoyanov *et al.*, 2011; Brakel *et al.*, 2013)。这些以前的方法需要诸如均匀场不动点方程的推断过程来提供计算图。变分自编码器被定义为任意计算图，这使得它能适用于更广泛的概率模型族，因为它不需要将模型的选择限制到具有易处理的均匀场不动点方程的那些模型。变分自编码器还具有增加模型对数似然边界的优点，而 MP-DBM 和相关模型的准则更具启发性，并且除了使近似推断的结果准确外很少有概率的解释。变分自编码器的一个缺点是它仅针对一个问题学习推断网络，即给定  $\mathbf{x}$  推断  $\mathbf{z}$ 。较老的方法能够在给定任何其他变量子集的情况下对任何变量子集执行近似推断，因为均匀场不动点方程指定如何在所有这些不同问题的计算图之间共享参数。

变分自编码器的一个非常好的特性是，同时训练参数编码器与生成器网络的组合迫使模型学习一个编码器可以捕获的可预测的坐标系。这使得它成为一个优秀的流形学习算法。图 20.6 展示了由变分自编码器学到的低维流形的例子。图中所示的情况之一，算法发现了存在于面部图像中两个独立的变化因素：旋转角和情绪表达。



图 20.6: 由变分自编码器学习的高维流形在 2 维坐标系中的示例 (Kingma and Welling, 2014a)。我们可以在纸上直接绘制两个可视化的维度，因此可以使用 2 维潜在编码训练模型来了解模型的工作原理（即使我们认为数据流形的固有维度要高得多）。图中所示的图像不是来自训练集的样本，而是仅仅通过改变 2 维“编码” $z$ ，由模型  $p(x|z)$  实际生成的图像  $x$ （每个图像对应于“编码” $z$  位于 2 维均匀网格的不同选择）。（左）Frey 人脸流形的 2 维映射。其中一个维度（水平）已发现大致对应于面部的旋转，而另一个（垂直）对应于情绪表达。（右）MNIST 流形的 2 维映射。

#### 20.10.4 生成式对抗网络

**生成式对抗网络** (generative adversarial network, GAN) (Goodfellow *et al.*, 2014c) 是基于可微生成器网络的另一种生成式建模方法。

生成式对抗网络基于博奕论场景，其中生成器网络必须与对手竞争。生成器网络直接产生样本  $x = g(z; \theta^{(g)})$ 。其对手，**判别器网络** (discriminator network)，试图区分从训练数据抽取的样本和从生成器抽取的样本。判别器发出由  $d(x; \theta^{(d)})$  给出的概率值，指示  $x$  是真实训练样本而不是从模型抽取的伪造样本的概率。

形式化表示生成式对抗网络中学习的最简单方式是零和游戏，其中函数  $v(\theta^{(g)}, \theta^{(d)})$  确定判别器的收益。生成器接收  $-v(\theta^{(g)}, \theta^{(d)})$  作为它自己的收益。在学习期间，每个玩家尝试最大化自己的收益，因此收敛在

$$g^* = \arg \min_g \max_d v(g, d). \quad (20.80)$$

$v$  的默认选择是

$$v(\theta^{(g)}, \theta^{(d)}) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \log d(\mathbf{x}) + \mathbb{E}_{\mathbf{x} \sim p_{\text{model}}} \log(1 - d(\mathbf{x})). \quad (20.81)$$

这驱使判别器试图学习将样品正确地分类为真的或伪造的。同时，生成器试图欺骗分类器以让其相信样本是真实的。在收敛时，生成器的样本与实际数据不可区分，并且判别器处处都输出  $\frac{1}{2}$ 。然后就可以丢弃判别器。

设计 GAN 的主要动机是学习过程既不需要近似推断也不需要配分函数梯度的近似。当  $\max_d v(g, d)$  在  $\theta^{(g)}$  中是凸的（例如，在概率密度函数的空间中直接执行优化的情况）时，该过程保证收敛并且是渐近一致的。

不幸的是，在实践中由神经网络表示的  $g$  和  $d$  以及  $\max_d v(g, d)$  不凸时，GAN 中的学习可能是困难的。Goodfellow (2014) 认为不收敛可能会引起 GAN 的欠拟合问题。一般来说，同时对两个玩家的成本梯度下降不能保证达到平衡。例如，考虑价值函数  $v(a, b) = ab$ ，其中一个玩家控制  $a$  并产生成本  $ab$ ，而另一玩家控制  $b$  并接收成本  $-ab$ 。如果我们将每个玩家建模为无穷小的梯度步骤，每个玩家以另一个玩家为代价降低自己的成本，则  $a$  和  $b$  进入稳定的圆形轨迹，而不是到达原点处的平衡点。注意，极小极大化游戏的平衡不是  $v$  的局部最小值。相反，它们是同时最小化的两个玩家成本的点。这意味着它们是  $v$  的鞍点，相对于第一个玩家的参数是局部最小值，而相对于第二个玩家的参数是局部最大值。两个玩家可以永远轮流增加然后减少  $v$ ，而不是正好停在玩家没有能力降低其成本的鞍点。目前不知道这种不收敛的问题会在多大程度上影响 GAN。

Goodfellow (2014) 确定了另一种替代的形式化收益公式，其中博弈不再是零和，每当判别器最优时，具有与最大似然学习相同的预期梯度。因为最大似然训练收敛，这种 GAN 博弈的重述在给定足够的样本时也应该收敛。不幸的是，这种替代的形式化似乎并没有提高实践中的收敛，可能是由于判别器的次优性或围绕期望梯度的高方差。

在真实实验中，GAN 博弈的最佳表现形式既不是零和也不等价于最大似然，而是 Goodfellow et al. (2014c) 引入的带有启发式动机的不同形式化。在这种最佳性能

的形式中，生成器旨在增加判别器发生错误的对数概率，而不是旨在降低判别器进行正确预测的对数概率。这种重述仅仅是观察的结果，即使在判别器确信地拒绝所有生成器样本的情况下，它也能导致生成器代价函数的导数相对于判别器的对数保持很大。

稳定 GAN 学习仍然是一个开放的问题。幸运的是，当仔细选择模型架构和超参数时，GAN 学习效果很好。Radford *et al.* (2015) 设计了一个深度卷积 GAN (DCGAN)，在图像合成的任务上表现非常好，并表明其潜在的表示空间能捕获到变化的重要因素，如图 20.7 所示。图 20.7 展示了 DCGAN 生成器生成的图像示例。



图 20.7：在 LSUN 数据集上训练后，由 GAN 生成的图像。(左) 由 DCGAN 模型生成的卧室图像，经 Radford *et al.* (2015) 许可转载。(右) 由 LAPGAN 模型生成的教堂图像，经 Denton *et al.* (2015) 许可转载。

GAN 学习问题也可以通过将生成过程分成许多级别的细节来简化。我们可以训练有条件的 GAN (Mirza and Osindero, 2014)，并学习从分布  $p(\mathbf{x} | \mathbf{y})$  中采样，而不是简单地从边缘分布  $p(\mathbf{x})$  中采样。Denton *et al.* (2015) 表明一系列的条件 GAN 可以被训练为首先生成非常低分辨率的图像，然后增量地向图像添加细节。由于使用拉普拉斯金字塔来生成包含不同细节水平的图像，这种技术被称为 LAPGAN 模型。LAPGAN 生成器不仅能够欺骗判别器网络，而且能够欺骗人类观察者，实验主体将高达 40% 的网络输出识别为真实数据。请看图 20.7 中 LAPGAN 生成器生成的图像示例。

GAN 训练过程中一个不寻常的能力是它可以拟合向训练点分配零概率的概率分布。生成器网络学习跟踪其点在某种程度上类似于训练点的流形，而不是最大化特定点的对数概率。有点矛盾的是，这意味着模型可以将负无穷大的对数似然分配

给测试集，同时仍然表示人类观察者判断为能捕获生成任务本质的流形。这不是明显的优点或缺点，并且只要向生成器网络最后一层所有生成的值添加高斯噪声，就可以保证生成器网络向所有点分配非零概率。以这种方式添加高斯噪声的生成器网络从相同分布的采样，即使用生成器网络参数化条件高斯分布的均值所获得的分布。

Dropout 似乎在判别器网络中很重要。特别地，在计算生成器网络的梯度时，单元应当被随机地丢弃。使用权重除以二的确定性版本的判别器的梯度似乎不是那么有效。同样，从不使用 Dropout 似乎会产生不良的结果。

虽然 GAN 框架被设计为用于可微生成器网络，但是类似的原理可以用于训练其他类型的模型。例如，**自监督提升** (self-supervised boosting) 可以用于训练 RBM 生成器以欺骗逻辑回归判别器 (Welling *et al.*, 2002)。

### 20.10.5 生成矩匹配网络

**生成矩匹配网络** (generative moment matching network) (Li *et al.*, 2015; Dziugaite *et al.*, 2015) 是另一种基于可微生成器网络的生成模型。与 VAE 和 GAN 不同，它们不需要将生成器网络与任何其他网络配对，如不需要与用于 VAE 的推断网络配对，也不需要与 GAN 的判别器网络。

生成矩匹配网络使用称为 **矩匹配** (moment matching) 的技术训练。矩匹配背后的基本思想是以如下的方式训练生成器——令模型生成的样本的许多统计量尽可能与训练集中的样本相似。在此情景下，**矩** (moment) 是对随机变量不同幂的期望。例如，第一矩是均值，第二矩是平方值的均值，以此类推。多维情况下，随机向量的每个元素可以被升高到不同的幂，因此使得矩可以是任意数量的形式

$$\mathbb{E}_x \prod_i x_i^{n_i}, \quad (20.82)$$

其中  $\mathbf{n} = [n_1, n_2, \dots, n_d]^\top$  是一个非负整数的向量。

在第一次检查时，这种方法似乎在计算上是不可行的。例如，如果我们想匹配形式为  $x_i x_j$  的所有矩，那么我们需要最小化在  $\mathbf{x}$  的维度上是二次的多个值之间的差。此外，甚至匹配所有第一和第二矩将仅足以拟合多变量高斯分布，其仅捕获值之间的线性关系。我们使用神经网络的野心是捕获复杂的非线性关系，这将需要更多的矩。GAN 通过使用动态更新的判别器避免了穷举所有矩的问题，该判别器自动将其注意力集中在生成器网络最不匹配的统计量上。

相反，我们可以通过最小化一个被称为最大平均偏差（maximum mean discrepancy, MMD）(Schölkopf and Smola, 2002; Gretton *et al.*, 2012) 的代价函数来训练生成矩匹配网络。该代价函数通过向核函数定义的特征空间隐式映射，在无限维空间中测量第一矩的误差，使得对无限维向量的计算变得可行。当且仅当所比较的两个分布相等时，MMD 代价为零。

从可视化方面看，来自生成矩匹配网络的样本有点令人失望。幸运的是，它们可以通过将生成器网络与自编码器组合来改进。首先，训练自编码器以重构训练集。接下来，自编码器的编码器用于将整个训练集转换到编码空间。然后训练生成器网络以生成编码样本，这些编码样本可以经解码器映射到视觉上令人满意的样本。

与 GAN 不同，代价函数仅关于一批同时来自训练集和生成器网络的实例定义。我们不可能将训练更新作为一个训练样本或仅来自生成器网络的一个样本的函数。这是因为必须将矩计算为许多样本的经验平均值。当批量大小太小时，MMD 可能低估采样分布的真实变化量。有限的批量大小都不足以大到完全消除这个问题，但是更大的批量大小减少了低估的量。当批量大小太大时，训练过程就会慢得不可行，因为计算单个小梯度步长必须一下子处理许多样本。

与 GAN 一样，即使生成器网络为训练点分配零概率，仍可以使用 MMD 训练生成器网络。

### 20.10.6 卷积生成网络

当生成图像时，将卷积结构的引入生成器网络通常是有用的（见 Goodfellow *et al.* (2014c) 或 Dosovitskiy *et al.* (2015) 的例子）。为此，我们使用卷积算子的“转置”，如第 9.5 节所述。这种方法通常能产生更逼真的图像，并且比不使用参数共享的全连接层使用更少的参数。

用于识别任务的卷积网络具有从图像到网络顶部的某些概括层（通常是类标签）的信息流。当该图像通过网络向上流动时，随着图像的表示变得对于有害变换保持不变，信息也被丢弃。在生成器网络中，情况恰恰相反。要生成图像的表示通过网络传播时必须添加丰富的详细信息，最后产生图像的最终表示，这个最终表示当然是带有所有细节的精细图像本身（具有对象位置、姿势、纹理以及明暗）。在卷积识别网络中丢弃信息的主要机制是池化层。而生成器网络似乎需要添加信息。由于大多数池化函数不可逆，我们不能将池化层求逆后放入生成器网络。更简单的操作是仅仅增加表示的空间大小。似乎可接受的方法是使用 Dosovitskiy *et al.* (2015) 引入的

“去池化”。该层对应于某些简化条件下最大池化的逆操作。首先，最大池化操作的步幅被约束为等于池化区域的宽度。其次，每个池化区域内的最大输入被假定为左上角的输入。最后，假设每个池化区域内所有非最大的输入为零。这些是非常强和不现实的假设，但它们允许我们对最大池化算子求逆。逆去池化的操作分配一个零张量，然后将每个值从输入的空间坐标  $i$  复制到输出的空间坐标  $i \times k$ 。整数值  $k$  定义池化区域的大小。即使驱动去池化算子定义的假设是不现实的，后续层也能够学习补偿其不寻常的输出，所以由整体模型生成的样本在视觉上令人满意。

### 20.10.7 自回归网络

自回归网络是没有潜在随机变量的有向概率模型。这些模型中的条件概率分布由神经网络表示（有时是极简单的神经网络，例如逻辑回归）。这些模型的图结构是完全图。它们可以通过概率的链式法则分解观察变量上的联合概率，从而获得形如  $P(x_d | x_{d-1}, \dots, x_1)$  条件概率的乘积。这样的模型被称为 **完全可见的贝叶斯网络**（fully-visible Bayes networks, FVBN），并成功地以许多形式使用，首先是对每个条件分布逻辑回归 (Frey, 1998)，然后是带有隐藏单元的神经网络 (Bengio and Bengio, 2000b; Larochelle and Murray, 2011)。在某些形式的自回归网络中，例如在第 20.10.10 节中描述的 NADE (Larochelle and Murray, 2011)，我们可以引入参数共享的一种形式，它能带来统计优点（较少的唯一参数）和计算优势（较少计算量）。这是深度学习中反复出现的主题——特征重用的另一个实例。

### 20.10.8 线性自回归网络

自回归网络的最简单形式是没有隐藏单元、没有参数或特征共享的形式。每个  $P(x_i | x_{i-1}, \dots, x_1)$  被参数化为线性模型（对于实值数据的线性回归，对于二值数据的逻辑回归，对于离散数据的softmax回归）。这个模型由 Frey (1998) 引入，当有  $d$  个变量要建模时，该模型有  $\mathcal{O}(d^2)$  个参数。如图 20.8 所示。

如果变量是连续的，线性自回归网络只是表示多元高斯分布的另一种方式，只能捕获观察变量之间线性的成对相互作用。

线性自回归网络本质上是线性分类方法在生成式建模上的推广。因此，它们具有与线性分类器相同的优缺点。像线性分类器一样，它们可以用凸损失函数训练，并且有时允许闭解形式（如在高斯情况下）。像线性分类器一样，模型本身不提供增加

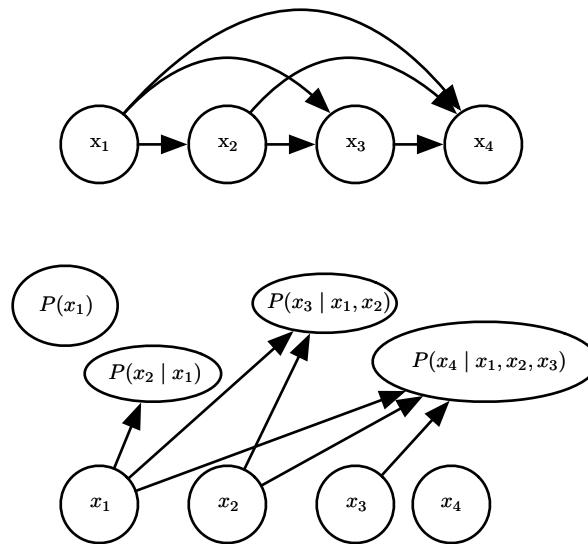


图 20.8: 完全可见的信念网络从前  $i - 1$  个变量预测第  $i$  个变量。(上) FVBN 的有向图模型。(下) 对数 FVBN 相应的计算图, 其中每个预测由线性预测器作出。

其容量的方法, 因此必须使用其他技术 (如输入的基扩展或核技巧) 来提高容量。

### 20.10.9 神经自回归网络

神经自回归网络 (Bengio and Bengio, 2000a,b) 具有与逻辑自回归网络相同的从左到右的图模型 (图 20.8), 但在该图模型结构内采用不同的条件分布参数。新的参数化更强大, 它可以根据需要随意增加容量, 并允许近似任意联合分布。新的参数化还可以引入深度学习中常见的参数共享和特征共享原理来改进泛化能力。设计这些模型的动机是避免传统表格图模型引起的维数灾难, 并与图 20.8 共享相同的结构。在表格离散概率模型中, 每个条件分布由概率表表示, 其中所涉及的变量的每个可能配置都具有一个条目和一个参数。通过使用神经网络, 可以获得两个优点:

1. 通过具有  $(i - 1) \times k$  个输入和  $k$  个输出的神经网络 (如果变量是离散的并有  $k$  个值, 使用one-hot编码) 参数化每个  $P(x_i | x_{i-1}, \dots, x_1)$ , 让我们不需要指数量级参数 (和样本) 的情况下就能估计条件概率, 然而仍然能够捕获随机变量之间的高阶依赖性。
2. 不需要对预测每个  $x_i$  使用不同的神经网络, 如图 20.9 所示的从左到右连接, 允

许将所有神经网络合并成一个。等价地，它意味着为预测  $x_i$  所计算的隐藏层特征可以重新用于预测  $x_{i+k}$  ( $k > 0$ )。因此隐藏单元被组织成第  $i$  组中的所有单元仅依赖于输入值  $x_1, \dots, x_i$  的特定的组。用于计算这些隐藏单元的参数被联合优化以改进对序列中所有变量的预测。这是重用原理的一个实例，这是从循环和卷积网络架构到多任务和迁移学习的场景中反复出现的深度学习原理。

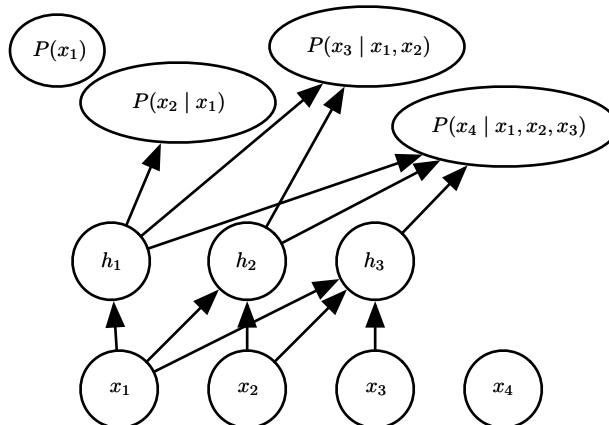


图 20.9: 神经自回归网络从前  $i - 1$  个变量预测第  $i$  个变量  $x_i$ ，但经参数化后，作为  $x_1, \dots, x_i$  函数的特征（表示为  $h_i$  的隐藏单元的组）可以在预测所有后续变量  $x_{i+1}, x_{i+2}, \dots, x_d$  时重用。

如在第 6.2.2.1 节中讨论的，使神经网络的输出预测  $x_i$  条件分布的参数，每个  $P(x_i | x_{i-1}, \dots, x_1)$  就可以表示一个条件分布。虽然原始神经自回归网络最初是在纯粹离散多变量数据（带有 sigmoid 输出的 Bernoulli 变量或 softmax 输出的 Multinoulli 变量）的背景下评估，但我们可以自然地将这样的模型扩展到连续变量或同时涉及离散和连续变量的联合分布。

## 20.10.10 NADE

**神经自回归密度估计器** (neural auto-regressive density estimator, NADE) 是最近非常成功的神经自回归网络的一种形式 (Larochelle and Murray, 2011)。与 Bengio and Bengio (2000b) 的原始神经自回归网络中的连接相同，但 NADE 引入了附加的参数共享方案，如图 20.10 所示。不同组  $j$  的隐藏单元的参数是共享的。

从第  $i$  个输入  $x_i$  到第  $j$  组隐藏单元的第  $k$  个元素  $h_k^{(j)}$  ( $j \geq i$ ) 的权重  $W'_{j,k,i}$  是

组内共享的：

$$W'_{j,k,i} = W_{k,i}. \quad (20.83)$$

其余  $j < i$  的权重为零。

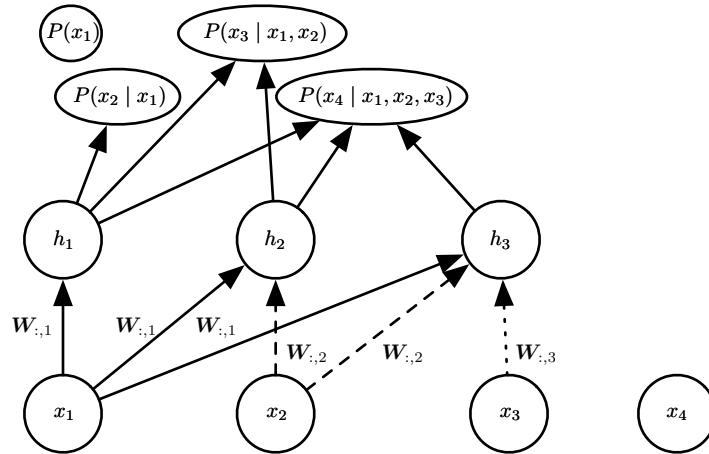


图 20.10：神经自回归密度估计器（NADE）的示意图。隐藏单元被组织在组  $\mathbf{h}^{(j)}$  中，使得只有输入  $x_1, \dots, x_i$  参与计算  $\mathbf{h}^{(i)}$  和预测  $P(x_j | x_{j-1}, \dots, x_1)$ （对于  $j > i$ ）。NADE 使用特定的权重共享模式区别于早期的神经自回归网络： $W'_{j,k,i} = W_{k,i}$  被共享于所有从  $x_i$  到任何  $j \geq i$  组中第  $k$  个单元的权重（在图中使用相同的线型表示复制权重的每个实例）。注意向量  $(W_{1,i}, W_{2,i}, \dots, W_{n,i})$  记为  $\mathbf{W}_{:,i}$ 。

Larochelle and Murray (2011) 选择了这种共享方案，使得 NADE 模型中的正向传播与在均匀场推断中执行的计算大致相似，以填充 RBM 中缺失的输入。这个均匀场推断对应于运行具有共享权重的循环网络，并且该推断的第一步与 NADE 中的相同。使用 NADE 的唯一区别是，连接隐藏单元到输出的输出权重独立于连接输入单元和隐藏单元的权重进行参数化。在 RBM 中，隐藏到输出的权重是输入到隐藏权重的转置。NADE 架构可以扩展为不仅仅模拟均匀场循环推断的一个时间步，而是  $k$  步。这种方法称为 NADE- $k$  (Raiko *et al.*, 2014)。

如前所述，自回归网络可以被扩展成处理连续数据。用于参数化连续密度的特别强大和通用的方法是混合权重为  $\alpha_i$ （组  $i$  的系数或先验概率），每组条件均值为  $\mu_i$  和每组条件方差为  $\sigma_i^2$  的高斯混合体。一个称为 RNADE 的模型 (Uria *et al.*, 2013) 使用这种参数化将 NADE 扩展到实值。与其他混合密度网络一样，该分布的参数是网络的输出，由 softmax 单元产生混合的权量概率以及参数化的方差，因此可使它

们为正的。由于条件均值  $\mu_i$  和条件方差  $\sigma_i^2$  之间的相互作用，随机梯度下降在数值上可能会表现不好。为了减少这种困难，Uria *et al.* (2013) 在后向传播阶段使用伪梯度代替平均值上的梯度。

另一个非常有趣的神经自回归架构的扩展摆脱了为观察到的变量选择任意顺序的需要 (Murray and Larochelle, 2014)。在自回归网络中，该想法是训练网络以能够通过随机采样顺序来处理任何顺序，并将信息提供给指定哪些输入被观察的隐藏单元（在条件条的右侧），以及哪些是被预测并因此被认为是缺失的（在条件条的左侧）。这是不错的性质，因为它允许人们非常高效地使用训练好的自回归网络来执行任何推断问题（即从给定任何变量的子集，从任何子集上的概率分布预测或采样）。最后，由于变量的许多顺序是可能的（对于  $n$  个变量是  $n!$ ），并且变量的每个顺序  $o$  产生不同的  $p(\mathbf{x} | o)$ ，我们可以组成许多  $o$  值模型的集成：

$$p_{\text{ensemble}}(\mathbf{x}) = \frac{1}{k} \sum_{i=1}^k p(\mathbf{x} | o^{(i)}). \quad (20.84)$$

这个集成模型通常能更好地泛化，并且为测试集分配比单个排序定义的单个模型更高的概率。

在同一篇文章中，作者提出了深度版本的架构，但不幸的是，这立即使计算成本像原始神经自回归网络一样高 (Bengio and Bengio, 2000b)。第一层和输出层仍然可以在  $\mathcal{O}(nh)$  的乘法-加法操作中计算，如在常规 NADE 中，其中  $h$  是隐藏单元的数量（图 20.10 和图 20.9 中的组  $h_i$  的大小），而它在 Bengio and Bengio (2000b) 中是  $\mathcal{O}(n^2h)$ 。然而，对于其他隐藏层的计算量是  $\mathcal{O}(n^2h^2)$ （假设在每个层存在  $n$  组  $h$  个隐藏单元，且在  $l$  层的每个“先前”组参与预测  $l+1$  层处的“下一个”组）。如在 Murray and Larochelle (2014) 中，使  $l+1$  层上的第  $i$  个组仅取决于第  $i$  个组， $l$  层处的计算量将减少到  $\mathcal{O}(nh^2)$ ，但仍然比常规 NADE 差  $h$  倍。

## 20.11 从自编码器采样

在第十四章中，我们看到许多种学习数据分布的自编码器。得分匹配、去噪自编码器和收缩自编码器之间有着密切的联系。这些联系表明某些类型的自编码器以某些方式学习数据分布。我们还没有讨论如何从这样的模型中采样。

某些类型的自编码器，例如变分自编码器，明确地表示概率分布并且允许直接的原始采样。而大多数其他类型的自编码器则需要 MCMC 采样。

收缩自编码器被设计为恢复数据流形切面的估计。这意味着使用注入噪声的重复编码和解码将引起沿着流形表面的随机游走 (Rifai *et al.*, 2012; Mesnil *et al.*, 2012)。这种流形扩散技术是马尔可夫链的一种。

更一般的马尔可夫链还可以从任何去噪自编码器中采样。

### 20.11.1 与任意去噪自编码器相关的马尔可夫链

上述讨论留下了一个开放问题——注入什么噪声和从哪获得马尔可夫链（可以根据自编码器估计的分布生成样本）。Bengio *et al.* (2013c) 展示了如何构建这种用于广义去噪自编码器(generalized denoising autoencoder) 的马尔可夫链。广义去噪自编码器由去噪分布指定，给定损坏输入后，对干净输入的估计进行采样。

根据估计分布生成的马尔可夫链的每个步骤由以下子步骤组成，如图 20.11 所示：

1. 从先前状态  $\mathbf{x}$  开始，注入损坏噪声，从  $C(\tilde{\mathbf{x}} | \mathbf{x})$  中采样  $\tilde{\mathbf{x}}$ 。
2. 将  $\tilde{\mathbf{x}}$  编码为  $\mathbf{h} = f(\tilde{\mathbf{x}})$ 。
3. 解码  $\mathbf{h}$  以获得  $p(\mathbf{x} | \omega = g(\mathbf{h})) = p(\mathbf{x} | \tilde{\mathbf{x}})$  的参数  $\omega = g(\mathbf{h})$ 。
4. 从  $p(\mathbf{x} | \omega = g(\mathbf{h})) = p(\mathbf{x} | \tilde{\mathbf{x}})$  采样下一状态  $\mathbf{x}$ 。

Bengio *et al.* (2014) 表明，如果自编码器  $p(\mathbf{x} | \tilde{\mathbf{x}})$  形成对应真实条件分布的一致估计量，则上述马尔可夫链的平稳分布形成数据生成分布  $\mathbf{x}$  的一致估计量（虽然是隐式的）。

### 20.11.2 夹合与条件采样

与玻尔兹曼机类似，去噪自编码器及其推广（例如下面描述的 GSN）可用于从条件分布  $p(\mathbf{x}_f | \mathbf{x}_o)$  中采样，只需夹合观察单元  $\mathbf{x}_f$  并在给定  $\mathbf{x}_f$  和采好的潜变量（如果有的话）下仅重采样自由单元  $\mathbf{x}_o$ 。例如，MP-DBM 可以被解释为去噪自编码器的一种形式，并且能够采样丢失的输入。GSN 随后将 MP-DBM 中的一些想法推广以执行相同的操作 (Bengio *et al.*, 2014)。Alain *et al.* (2015) 从 Bengio *et al.* (2014) 的命题 1 中发现了一个缺失条件，即转移算子（由从链的一个状态到下一个

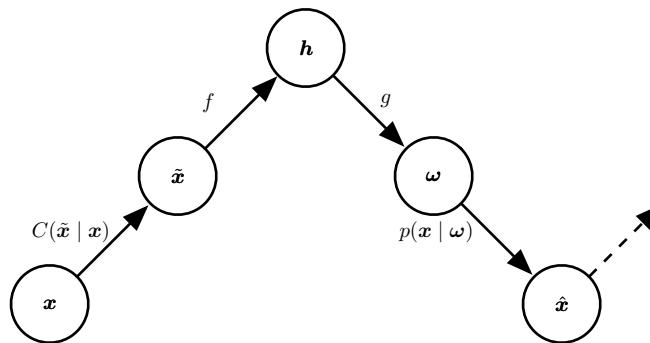


图 20.11：马尔可夫链的每个步骤与训练好的去噪自编码器相关联，根据由去噪对数似然准则隐式训练的概率模型生成样本。每个步骤包括：(a) 通过损坏过程  $C$  向状态  $x$  注入噪声产生  $\tilde{x}$ ，(b) 用函数  $f$  对其编码，产生  $h = f(\tilde{x})$ ，(c) 用函数  $g$  解码结果，产生用于重构分布的参数  $\omega$ ，(d) 给定  $\omega$ ，从重构分布  $p(x | \omega) = g(f(\tilde{x}))$  采样新状态。在典型的平方重构误差情况下， $g(h) = \hat{x}$ ，并估计  $E[x | \tilde{x}]$ ，损坏包括添加高斯噪声，并且从  $p(x | \omega)$  的采样包括第二次向重构  $\hat{x}$  添加高斯噪声。后者的噪声水平应对应于重构的均方误差，而注入的噪声是控制混合速度以及估计器平滑经验分布程度的超参数 (Vincent, 2011)。在这所示的例子中，只有  $C$  和  $p$  条件是随机步骤 ( $f$  和  $g$  是确定性计算)，我们也可以在自编码器内部注入噪声，如生成随机网络 (Bengio et al., 2014)。

状态的随机映射定义) 应该满足 **细致平衡** (detailed balance) 的属性，表明无论转移算子正向或反向运行，马尔可夫链都将保持平衡。

在图 20.12 中展示了夹合一半像素 (图像的右部分) 并在另一半上运行马尔可夫链的实验。

### 20.11.3 回退训练过程

回退训练过程由 Bengio et al. (2013c) 等人提出，作为一种加速去噪自编码器生成训练收敛的方法。不像执行一步编码-解码重建，该过程由交替的多个随机编码-解码步骤组成 (如在生成马尔可夫链中)，以训练样本初始化 (正如在第 18.2 节中描述的对比散度算法)，并惩罚最后的概率重建 (或沿途的所有重建)。

训练  $k$  个步骤与训练一个步骤是等价的 (在实现相同稳态分布的意义上)，但是实际上可以更有效地去除来自数据的伪模式。



图 20.12: 在每步仅重采样左半部分，夹合图像的右半部分并运行马尔可夫链的示意图。这些样本来自重构 MNIST 数字的 GSN (每个时间步使用回退过程)。

## 20.12 生成随机网络

生成随机网络 (generative stochastic network, GSN) (Bengio *et al.*, 2014) 是去噪自编码器的推广，除可见变量 (通常表示为  $\mathbf{x}$ ) 之外，在生成马尔可夫链中还包括潜变量  $\mathbf{h}$ 。

GSN 由两个条件概率分布参数化，指定马尔可夫链的一步：

1.  $p(\mathbf{x}^{(k)} \mid \mathbf{h}^{(k)})$  指示在给定当前潜在状态下如何产生下一个可见变量。这种“重建分布”也可以在去噪自编码器、RBM、DBN 和 DBM 中找到。
2.  $p(\mathbf{h}^{(k)} \mid \mathbf{h}^{(k-1)}, \mathbf{x}^{(k-1)})$  指示在给定先前的潜在状态和可见变量下如何更新潜在状态变量。

去噪自编码器和 GSN 不同于经典的概率模型 (有向或无向)，它们自己参数化生成过程而不是通过可见和潜变量的联合分布的数学形式。相反，后者如果存在则隐式地定义为生成马尔可夫链的稳态分布。存在稳态分布的条件是温和的，并且需要与标准 MCMC 方法相同的条件 (见第 17.3 节)。这些条件是保证链混合的必要条

件，但它们可能被某些过渡分布的选择（例如，如果它们是确定性的）所违反。

我们可以想象 GSN 不同的训练准则。由 Bengio *et al.* (2014) 提出和评估的只对可见单元上对数概率的重建，如应用于去噪自编码器。通过将  $\mathbf{x}^{(0)} = \mathbf{x}$  夹合到观察到的样本并且在一些后续时间步处使生成  $\mathbf{x}$  的概率最大化，即最大化  $\log p(\mathbf{x}^{(k)} = \mathbf{x} | \mathbf{h}^{(k)})$ ，其中给定  $\mathbf{x}^{(0)} = \mathbf{x}$  后， $\mathbf{h}^{(k)}$  从链中采样。为了估计相对于模型其他部分的  $\log p(\mathbf{x}^{(k)} = \mathbf{x} | \mathbf{h}^{(k)})$  的梯度，Bengio *et al.* (2014) 使用了在第 20.9 节中介绍的重参数化技巧。

回退训练过程（在第 20.11.3 节中描述）可以用来改善训练 GSN 的收敛性 (Bengio *et al.*, 2014)。

### 20.12.1 判别性 GSN

GSN 的原始公式 (Bengio *et al.*, 2014) 用于无监督学习和对观察数据  $\mathbf{x}$  的  $p(\mathbf{x})$  的隐式建模，但是我们可以修改框架来优化  $p(\mathbf{y} | \mathbf{x})$ 。

例如，Zhou and Troyanskaya (2014) 以如下方式推广 GSN，只反向传播输出变量上的重建对数概率，并保持输入变量固定。他们将这种方式成功应用于建模序列（蛋白质二级结构），并在马尔可夫链的转换算子中引入（一维）卷积结构。重要的是要记住，对于马尔可夫链的每一步，我们需要为每个层生成新序列，并且该序列用于在下一时间步计算其他层的值（例如下面一个和上面一个）的输入。

因此，马尔可夫链确实不只是输出变量（与更高层的隐藏层相关联），并且输入序列仅用于条件化该链，其中反向传播使得它能够学习输入序列如何条件化由马尔可夫链隐含表示的输出分布。因此这是在结构化输出中使用 GSN 的一个例子。

Zöhrer and Pernkopf (2014) 引入了一个混合模型，通过简单地添加（使用不同的权重）监督和非监督成本即  $\mathbf{y}$  和  $\mathbf{x}$  的重建对数概率，组合了监督目标（如上面的工作）和无监督目标（如原始的 GSN）。Larochelle and Bengio (2008a) 以前在 RBM 中就提出了这样的混合标准。他们展示了在这种方案下分类性能的提升。

## 20.13 其他生成方案

目前为止我们已经描述的方法，使用 MCMC 采样、原始采样或两者的一些混合来生成样本。虽然这些是生成式建模中最流行的方法，但它们绝不是唯一的方法。

Sohl-Dickstein *et al.* (2015) 开发了一种基于非平衡热力学学习生成模型的扩散反演 (diffusion inversion) 训练方案。该方法基于我们希望从中采样的概率分布具有结构的想法。这种结构会被递增地使概率分布具有更多熵的扩散过程逐渐破坏。为了形成生成模型，我们可以反过来运行该过程，通过训练模型逐渐将结构恢复到非结构化分布。通过迭代地应用使分布更接近目标分布的过程，我们可以逐渐接近该目标分布。在涉及许多迭代以产生样本的意义上，这种方法类似于 MCMC 方法。然而，模型被定义为由链的最后一步产生的概率分布。在这个意义上，没有由迭代过程诱导的近似。Sohl-Dickstein *et al.* (2015) 介绍的方法也非常接近于去噪自编码器的生成解释（第 20.11.1 节）。与去噪自编码器一样，扩散反演训练一个尝试概率地撤消添加的噪声效果的转移算子。不同之处在于，扩散反演只需要消除扩散过程的一个步骤，而不是一直返回到一个干净的数据点。这解决了去噪自编码器的普通重建对数似然目标中存在的以下两难问题：小噪声的情况下学习者只能看到数据点附近的配置，而在大噪声的情况下，去噪自编码器被要求做几乎不可能的工作（因为去噪分布是高度复杂和多峰值的）。利用扩散反演目标，学习者可以更精确地学习数据点周围的密度形状，以及去除可能在远离数据点处出现的假性模式。

样本生成的另一种方法是近似贝叶斯计算 (approximate Bayesian computation, ABC) 框架 (Rubin *et al.*, 1984)。在这种方法中，样本被拒绝或修改以使样本选定函数的矩匹配期望分布的那些矩。虽然这个想法与矩匹配一样使用样本的矩，但它不同于矩匹配，因为它修改样本本身，而不是训练模型来自动发出具有正确矩的样本。Bachman and Precup (2015) 展示了如何在深度学习的背景下使用 ABC 中的想法，即使用 ABC 来塑造 GSN 的 MCMC 轨迹。

我们期待更多其他等待发现的生成式建模方法。

## 20.14 评估生成模型

研究生成模型的研究者通常需要将一个生成模型与另一个生成模型比较，通常是为了证明新发明的生成模型比之前存在的模型更能捕获一些分布。

这可能是一个困难且微妙的任务。通常，我们不能实际评估模型下数据的对数概率，但仅可以评估一个近似。在这些情况下，重要的是思考和沟通清楚正在测量什么。例如，假设我们可以评估模型 A 对数似然的随机估计和模型 B 对数似然的确定性下界。如果模型 A 得分高于模型 B，哪个更好？如果我们关心确定哪个模型

具有分布更好的内部表示，我们实际上不能说哪个更好，除非我们有一些方法来确定模型 B 的边界有多松。然而，如果我们关心在实践中该模型能用得多好，例如执行异常检测，则基于特定于感兴趣的 actual task 的准则，可以公平地说模型是更好的，例如基于排名测试样例和排名标准，如精度和召回率。

评估生成模型的另一个微妙之处是，评估指标往往是自身困难的研究问题。可能很难确定模型是否被公平比较。例如，假设我们使用 AIS 来估计  $\log Z$  以便为我们刚刚发明的新模型计算  $\log \tilde{p}(\mathbf{x}) - \log Z$ 。AIS 计算经济的实现可能无法找到模型分布的几种模式并低估  $Z$ ，这将导致我们高估  $\log p(\mathbf{x})$ 。因此可能难以判断高似然估计是否是良好模型或不好的 AIS 实现导致的结果。

机器学习的其他领域通常允许在数据预处理中有一些变化。例如，当比较对象识别算法的准确性时，通常可接受的是对每种算法略微不同地预处理输入图像（基于每种算法具有何种输入要求）。而因为预处理的变化，会导致生成式建模的不同，甚至非常小和微妙的变化也是完全不可接受的。对输入数据的任何更改都会改变要捕获的分布，并从根本上改变任务。例如，将输入乘以 0.1 将人为地将概率增加 10 倍。

预处理的问题通常在基于 MNIST 数据集上的生成模型产生，MNIST 数据集是非常受欢迎的生成式建模基准之一。MNIST 由灰度图像组成。一些模型将 MNIST 图像视为实向量空间中的点，而其他模型将其视为二值。还有一些将灰度值视为二值样本的概率。我们必须将实值模型仅与其他实值模型比较，二值模型仅与其他二值模型进行比较。否则，测量的似然性不在相同的空间。对于二值模型，对数似然可以最多为零，而对于实值模型，它可以是任意高的，因为它是关于密度的测度。在二值模型中，比较使用完全相同的二值化模型是重要的。例如，我们可以将 0.5 设为阈值后，将灰度像素二值化为 0 或 1，或者通过由灰度像素强度给出样本为 1 的概率来采一个随机样本。如果我们使用随机二值化，我们可能将整个数据集二值化一次，或者我们可能为每个训练步骤采不同的随机样例，然后采多个样本进行评估。这三个方案中的每一个都会产生极不相同的似然数，并且当比较不同的模型时，两个模型使用相同的二值化方案来训练和评估是重要的。事实上，应用单个随机二值化步骤的研究者共享包含随机二值化结果的文件，使得基于二值化步骤的不同输出的结果没有差别。

因为从数据分布生成真实样本是生成模型的目标之一，所以实践者通常通过视觉检查样本来评估生成模型。在最好的情况下，这不是由研究人员本身，而是由不知道样品来源的实验受试者完成 (Denton *et al.*, 2015)。不幸的是，非常差的概率

模型可能会产生非常好的样本。验证模型是否仅复制一些训练示例的常见做法如图 16.1 所示。该想法是根据在  $\mathbf{x}$  空间中的欧几里得距离，为一些生成的样本显示它们在训练集中的最近邻。此测试旨在检测模型过拟合训练集并仅再现训练实例的情况。甚至可能同时欠拟合和过拟合，但仍然能产生单独看起来好的样本。想象一下，生成模型用狗和猫的图像训练时，但只是简单地学习来重现狗的训练图像。这样的模型明显过拟合，因为它不能产生不在训练集中的图像，但是它也欠拟合，因为它不给猫的训练图像分配概率。然而，人类观察者将判断狗的每个个体图像都是高质量的。在这个简单的例子中，对于能够检查许多样本的人类观察者来说，确定猫的不存在是容易的。在更实际的设定中，在具有数万个模式的数据上训练后的生成模型可以忽略少数模式，并且人类观察者不能容易地检查或记住足够的图像以检测丢失的变化。

由于样本的视觉质量不是可靠的标准，所以当计算可行时，我们通常还评估模型分配给测试数据的对数似然。不幸的是，在某些情况下，似然性似乎不可能测量我们真正关心的模型的任何属性。例如，MNIST 的实值模型可以将任意低的方差分配给从不改变的背景像素，获得任意高的似然。即使这不是一个非常有用的事情，检测这些常量特征的模型和算法可以获得无限的奖励。实现接近负无穷代价的可能性存在于任何实值的最大似然问题中，但是对于 MNIST 的生成模型问题尤为严重，因为许多输出值是不需要预测的。这强烈地表明需要开发评估生成模型的其他方法。

Theis *et al.* (2015) 回顾了评估生成模型所涉及的许多问题，包括上述的许多想法。他们强调了生成模型有许多不同的用途，并且指标的选择必须与模型的预期用途相匹配。例如，一些生成模型更好地为大多数真实的点分配高概率，而其他生成模型擅长于不将高概率分配给不真实的点。这些差异可能源于生成模型是设计为最小化  $D_{\text{KL}}(p_{\text{data}} \parallel p_{\text{model}})$  还是  $D_{\text{KL}}(p_{\text{model}} \parallel p_{\text{data}})$ ，如图 3.6 所示。不幸的是，即使我们将每个指标的使用限制在最适合的任务上，目前使用的所有指标仍存在严重的缺陷。因此，生成式建模中最重要的研究课题之一不仅仅是如何提升生成模型，事实上还包括了设计新的技术来衡量我们的进步。

## 20.15 结论

为了让模型理解表示在给定训练数据中的大千世界，训练具有隐藏单元的生成模型是一种有力方法。通过学习模型  $p_{\text{model}}(\mathbf{x})$  和表示  $p_{\text{model}}(\mathbf{h} \mid \mathbf{x})$ ，生成模型可以解答  $\mathbf{x}$  输入变量之间关系的许多推断问题，并且可以在层次的不同层对  $\mathbf{h}$  求期望来

提供表示  $x$  的许多不同方式。生成模型承诺为 AI 系统提供它们需要理解的、所有不同直观概念的框架，让它们有能力在面对不确定性的情况下推理这些概念。我们希望我们的读者能够找到增强这些方法的新途径，并继续探究学习和智能背后原理的旅程。

# 参考文献

- (-1). *JMLR*. 618, 649
- (-1a). Icml'08. In *ICML'08*. ACM. 649, 674
- (-1b). Icml'11. In *ICML'11*. 628, 634
- (-1c). Icml'13. In *ICML'13*. 635, 660
- (-1). International conference on learning representations 2014. In *ICLR'2014*. 661, 675
- (-1). Nips'13. In *NIPS26*. NIPS Foundation. 629, 635

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org. 24, 183, 380

Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science*, 9, 147–169. 486, 559

Alain, G. and Bengio, Y. (2013). What regularized auto-encoders learn from the data generating distribution. In *ICLR'2013, arXiv:1211.4246*. 433, 439, 445

Alain, G., Bengio, Y., Yao, L., Éric Thibodeau-Laufer, Yosinski, J., and Vincent, P. (2015). GSNs: Generative stochastic networks. *arXiv:1503.05571*. 436, 607

Anderson, E. (1935). The Irises of the Gaspé Peninsula. *Bulletin of the American Iris Society*, 59, 2–5. 18

- Ba, J., Mnih, V., and Kavukcuoglu, K. (2014). Multiple object recognition with visual attention. *arXiv:1412.7755*. 591
- Bachman, P. and Precup, D. (2015). Variational generative stochastic networks with collaborative shaping. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1964–1972. 611
- Bacon, P.-L., Bengio, E., Pineau, J., and Precup, D. (2015). Conditional computation in neural networks using a decision-theoretic approach. In *2nd Multidisciplinary Conference on Reinforcement Learning and Decision Making (RLDM 2015)*. 383
- Bagnell, J. A. and Bradley, D. M. (2009). Differentiable sparse coding. In *NIPS'2009*, pages 113–120. 425
- Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. In *ICLR'2015, arXiv:1409.0473*. 23, 89, 339, 356, 358, 395, 404, 405
- Bahl, L. R., Brown, P., de Souza, P. V., and Mercer, R. L. (1987). Speech recognition with continuous-parameter hidden Markov models. *Computer, Speech and Language*, **2**, 219–234. 390
- Baldi, P. and Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, **2**, 53–58. 245
- Baldi, P., Brunak, S., Frasconi, P., Soda, G., and Pollastri, G. (1999). Exploiting the past and the future in protein secondary structure prediction. *Bioinformatics*, **15**(11), 937–946. 337
- Baldi, P., Sadowski, P., and Whiteson, D. (2014). Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, **5**. 24
- Ballard, D. H., Hinton, G. E., and Sejnowski, T. J. (1983). Parallel vision computation. *Nature*. 385
- Barlow, H. B. (1989). Unsupervised learning. *Neural Computation*, **1**, 295–311. 128
- Barron, A. E. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Trans. on Information Theory*, **39**, 930–945. 172
- Bartholomew, D. J. (1987). *Latent variable models and factor analysis*. Oxford University Press. 418
- Basilevsky, A. (1994). *Statistical Factor Analysis and Related Methods: Theory and Applications*. Wiley. 418

- Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I., Bergeron, A., Bouchard, N., Warde-Farley, D., and Bengio, Y. (2012a). Theano: new features and speed improvements. Submitted to the Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, <http://www.iro.umontreal.ca/lisa/publications2/index.php/publications/show/551>. 23, 73, 380
- Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I. J., Bergeron, A., Bouchard, N., and Bengio, Y. (2012b). Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop. 182, 191
- Basu, S. and Christensen, J. (2013). Teaching classification boundaries to humans. In *AAAI'2013*. 280
- Baxter, J. (1995). Learning internal representations. In *Proceedings of the 8th International Conference on Computational Learning Theory (COLT'95)*, pages 311–320, Santa Cruz, California. ACM Press. 211
- Bayer, J. and Osendorfer, C. (2014). Learning stochastic recurrent networks. *ArXiv e-prints*. 228
- Becker, S. and Hinton, G. (1992). A self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, **355**, 161–163. 462
- Behnke, S. (2001). Learning iterative image reconstruction in the neural abstraction pyramid. *Int. J. Computational Intelligence and Applications*, **1**(4), 427–438. 440
- Beiu, V., Quintana, J. M., and Avedillo, M. J. (2003). VLSI implementations of threshold logic-a comprehensive survey. *Neural Networks, IEEE Transactions on*, **14**(5), 1217–1243. 384
- Belkin, M. and Niyogi, P. (2002). Laplacian eigenmaps and spectral techniques for embedding and clustering. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14 (NIPS'01)*, Cambridge, MA. MIT Press. 210
- Belkin, M. and Niyogi, P. (2003a). Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, **15**(6), 1373–1396. 443
- Belkin, M. and Niyogi, P. (2003b). Using manifold structure for partially labeled classification. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15 (NIPS'02)*, Cambridge, MA. MIT Press. 141
- Bengio, E., Bacon, P.-L., Pineau, J., and Precup, D. (2015a). Conditional computation in neural networks for faster models. arXiv:1511.06297. 383

- Bengio, S. and Bengio, Y. (2000a). Taking on the curse of dimensionality in joint distributions using neural networks. *IEEE Transactions on Neural Networks, special issue on Data Mining and Knowledge Discovery*, **11**(3), 550–557. 603
- Bengio, S., Vinyals, O., Jaitly, N., and Shazeer, N. (2015b). Scheduled sampling for sequence prediction with recurrent neural networks. Technical report, arXiv:1506.03099. 327
- Bengio, Y. (1991). *Artificial Neural Networks and their Application to Sequence Recognition*. Ph.D. thesis, McGill University, (Computer Science), Montreal, Canada. 347
- Bengio, Y. (2000). Gradient-based optimization of hyperparameters. *Neural Computation*, **12**(8), 1889–1900. 370
- Bengio, Y. (2002). New distributed probabilistic language models. Technical Report 1215, Dept. IRO, Université de Montréal. 397
- Bengio, Y. (2009). *Learning deep architectures for AI*. Now Publishers. 174, 531
- Bengio, Y. (2013). Deep learning of representations: looking forward. In *Statistical Language and Speech Processing*, volume 7978 of *Lecture Notes in Computer Science*, pages 1–37. Springer, also in arXiv at <http://arxiv.org/abs/1305.0445>. 382
- Bengio, Y. (2015). Early inference in energy-based models approximates back-propagation. Technical Report arXiv:1510.02777, Universite de Montreal. 560
- Bengio, Y. and Bengio, S. (2000b). Modeling high-dimensional discrete data with multi-layer neural networks. In *NIPS 12*, pages 400–406. MIT Press. 602, 603, 604, 606
- Bengio, Y. and Delalleau, O. (2009). Justifying and generalizing contrastive divergence. *Neural Computation*, **21**(6), 1601–1621. 438, 520
- Bengio, Y. and Grandvalet, Y. (2004). No unbiased estimator of the variance of k-fold cross-validation. In *JML ( 1)*, pages 1089–1105. 107
- Bengio, Y. and LeCun, Y. (2007a). Scaling learning algorithms towards AI. In *Large Scale Kernel Machines*. 17
- Bengio, Y. and LeCun, Y. (2007b). Scaling learning algorithms towards AI. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large Scale Kernel Machines*. MIT Press. 17
- Bengio, Y. and Monperrus, M. (2005). Non-local manifold tangent learning. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17 (NIPS'04)*, pages 129–136. MIT Press. 138, 444

- Bengio, Y. and Sénecal, J.-S. (2003). Quick training of probabilistic neural nets by importance sampling. In *Proceedings of AISTATS 2003*. 400
- Bengio, Y. and Sénecal, J.-S. (2008). Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Trans. Neural Networks*, **19**(4), 713–722. 400
- Bengio, Y., De Mori, R., Flammia, G., and Kompe, R. (1991). Phonetically motivated acoustic parameters for continuous speech recognition using artificial neural networks. In *Proceedings of EuroSpeech '91*. 21, 390
- Bengio, Y., De Mori, R., Flammia, G., and Kompe, R. (1992). Neural network-Gaussian mixture hybrid for speech recognition or density estimation. In *NIPS 4*, pages 175–182. Morgan Kaufmann. 390
- Bengio, Y., Frasconi, P., and Simard, P. (1993). The problem of learning long-term dependencies in recurrent networks. In *IEEE International Conference on Neural Networks*, pages 1183–1195, San Francisco. IEEE Press. (invited paper). 344
- Bengio, Y., Simard, P., and Frasconi, P. (1994a). Learning long-term dependencies with gradient descent is difficult. *IEEE Tr. Neural Nets*. 16
- Bengio, Y., Simard, P., and Frasconi, P. (1994b). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, **5**(2), 157–166. 343, 344, 345
- Bengio, Y., Simard, P., and Frasconi, P. (1994c). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, **5**(2), 157–166. 351
- Bengio, Y., Latendresse, S., and Dugas, C. (1999). Gradient-based learning of hyper-parameters. In *Learning Conference*. 370
- Bengio, Y., Ducharme, R., and Vincent, P. (2001a). A neural probabilistic language model. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13 (NIPS'00)*, pages 933–938. MIT Press. 16
- Bengio, Y., Ducharme, R., and Vincent, P. (2001b). A neural probabilistic language model. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *NIPS'2000*, pages 932–938. MIT Press. 380, 394, 396, 402, 406, 410
- Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *JMLR*, **3**, 1137–1155. 396, 402
- Bengio, Y., Delalleau, O., and Le Roux, N. (2006a). The curse of highly variable functions for local kernel machines. In *NIPS'2005*. 137

- Bengio, Y., Larochelle, H., and Vincent, P. (2006b). Non-local manifold Parzen windows. In *NIPS'2005*. MIT Press. 138, 444
- Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007a). Greedy layer-wise training of deep networks. In *NIPS'2006*. 13, 276
- Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007b). Greedy layer-wise training of deep networks. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19 (NIPS'06)*, pages 153–160. MIT Press. 173
- Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007c). Greedy layer-wise training of deep networks. In *Adv. Neural Inf. Proc. Sys. 19*, pages 153–160. 275
- Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007d). Greedy layer-wise training of deep networks. In *NIPS 19*, pages 153–160. MIT Press. 276, 451, 452
- Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *ICML'09*. ACM. 279
- Bengio, Y., Mesnil, G., Dauphin, Y., and Rifai, S. (2013a). Better mixing via deep representations. In *ICML'2013*. 514
- Bengio, Y., Léonard, N., and Courville, A. (2013b). Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv:1308.3432. 382, 383, 588, 590
- Bengio, Y., Yao, L., Alain, G., and Vincent, P. (2013c). Generalized denoising auto-encoders as generative models. In *NIPS'2013*. 433, 607, 608
- Bengio, Y., Courville, A., and Vincent, P. (2013d). Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **35**(8), 1798–1828. 473
- Bengio, Y., Thibodeau-Laufer, E., Alain, G., and Yosinski, J. (2014). Deep generative stochastic networks trainable by backprop. In *ICML'2014*. 607, 608, 609, 610
- Bennett, C. (1976). Efficient estimation of free energy differences from Monte Carlo data. *Journal of Computational Physics*, **22**(2), 245–268. 536
- Bennett, J. and Lanning, S. (2007). The Netflix prize. 408
- Berger, A. L., Della Pietra, V. J., and Della Pietra, S. A. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, **22**, 39–71. 403
- Berglund, M. and Raiko, T. (2013). Stochastic gradient estimate variance in contrastive divergence and persistent contrastive divergence. *CoRR*, **abs/1312.6002**. 523

- Bergstra, J. (2011). *Incorporating Complex Cells into Neural Networks for Pattern Classification.* Ph.D. thesis, Université de Montréal. 219
- Bergstra, J. and Bengio, Y. (2009). Slow, decorrelated features for pretraining complex cell-like networks. In *NIPS 22*, pages 99–107. MIT Press. 421
- Bergstra, J. and Bengio, Y. (2011). Random search for hyper-parameter optimization. *The Learning Workshop*, Fort Lauderdale, Florida. 369
- Bergstra, J. and Bengio, Y. (2012). Random search for hyper-parameter optimization. *J. Machine Learning Res.*, **13**, 281–305. 369, 370
- Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010a). Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*. Oral Presentation. 23, 73
- Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010b). Theano: a CPU and GPU math expression compiler. In *Proc. SciPy*. 182, 191
- Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010c). Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*. 380
- Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *NIPS'2011*. 371
- Berkes, P. and Wiskott, L. (2005). Slow feature analysis yields a rich repertoire of complex cell properties. *Journal of Vision*, **5**(6), 579–602. 423
- Bertsekas, D. P. and Tsitsiklis, J. (1996). *Neuro-Dynamic Programming*. Athena Scientific. 93
- Besag, J. (1975). Statistical analysis of non-lattice data. *The Statistician*, **24**(3), 179–195. 525
- Bishop, C. M. (1994). Mixture density networks. 163
- Bishop, C. M. (1995a). Regularization and complexity control in feed-forward networks. In *Proceedings International Conference on Artificial Neural Networks ICANN'95*, volume 1, page 141–148. 208, 215
- Bishop, C. M. (1995b). Training with noise is equivalent to Tikhonov regularization. *Neural Computation*, **7**(1), 108–116. 208

- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer. 87, 126
- Blum, A. L. and Rivest, R. L. (1992). Training a 3-node neural network is NP-complete. 250
- Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. (1989). Learnability and the Vapnik–Chervonenkis dimension. *Journal of the ACM*, **36**(4), 929–865. 100
- Bonnet, G. (1964). Transformations des signaux aléatoires à travers les systèmes non linéaires sans mémoire. *Annales des Télécommunications*, **19**(9–10), 203–220. 588
- Bordes, A., Weston, J., Collobert, R., and Bengio, Y. (2011). Learning structured embeddings of knowledge bases. In *AAAI 2011*. 411, 412
- Bordes, A., Glorot, X., Weston, J., and Bengio, Y. (2012). Joint learning of words and meaning representations for open-text semantic parsing. *AISTATS’2012*. 343, 411, 412
- Bordes, A., Glorot, X., Weston, J., and Bengio, Y. (2013a). A semantic matching energy function for learning with multi-relational data. *Machine Learning: Special Issue on Learning Semantics*. 411
- Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. (2013b). Translating embeddings for modeling multi-relational data. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2787–2795. Curran Associates, Inc. 411
- Bornschein, J. and Bengio, Y. (2015). Reweighted wake-sleep. In *ICLR’2015, arXiv:1406.2751*. 592
- Bornschein, J., Shabanian, S., Fischer, A., and Bengio, Y. (2015). Training bidirectional Helmholtz machines. Technical report, arXiv:1506.03877. 592
- Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *COLT ’92: Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, New York, NY, USA. ACM. 16, 123
- Bottou, L. (1998). Online algorithms and stochastic approximations. In D. Saad, editor, *Online Learning in Neural Networks*. Cambridge University Press, Cambridge, UK. 253
- Bottou, L. (2011). From machine learning to machine reasoning. Technical report, arXiv:1102.1808. 341, 342
- Bottou, L. (2015). Multilayer neural networks. Deep Learning Summer School. 374

- Bottou, L. and Bousquet, O. (2008a). The tradeoffs of large scale learning. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20 (NIPS'07)*, volume 20. MIT Press, Cambridge, MA. 241
- Bottou, L. and Bousquet, O. (2008b). The tradeoffs of large scale learning. In *NIPS'2008*. 252
- Boulanger-Lewandowski, N., Bengio, Y., and Vincent, P. (2012). Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *ICML'12*. 585
- Boureau, Y., Ponce, J., and LeCun, Y. (2010). A theoretical analysis of feature pooling in vision algorithms. In *Proc. International Conference on Machine learning (ICML'10)*. 292
- Boureau, Y., Le Roux, N., Bach, F., Ponce, J., and LeCun, Y. (2011). Ask the locals: multi-way local pooling for image recognition. In *Proc. International Conference on Computer Vision (ICCV'11)*. IEEE. 293
- Bourlard, H. and Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, **59**, 291–294. 429
- Bourlard, H. and Wellekens, C. (1989). Speech pattern discrimination and multi-layered perceptrons. *Computer Speech and Language*, **3**, 1–19. 390
- Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press, New York, NY, USA. 82
- Brady, M. L., Raghavan, R., and Slawny, J. (1989). Back-propagation fails to separate where perceptrons succeed. *IEEE Transactions on Circuits and Systems*, **36**(5), 665–674. 243
- Brakel, P., Stroobandt, D., and Schrauwen, B. (2013). Training energy-based models for time-series imputation. *Journal of Machine Learning Research*, **14**, 2771–2797. 576, 596
- Brand, M. (2003a). Charting a manifold. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15 (NIPS'02)*, pages 961–968. MIT Press. 141
- Brand, M. (2003b). Charting a manifold. In *NIPS'2002*, pages 961–968. MIT Press. 443
- Breiman, L. (1994). Bagging predictors. *Machine Learning*, **24**(2), 123–140. 220
- Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth International Group, Belmont, CA. 125
- Bridle, J. S. (1990). Alphanets: a recurrent ‘neural’ network architecture with a hidden Markov model interpretation. *Speech Communication*, **9**(1), 83–92. 160

- Briggman, K., Denk, W., Seung, S., Helmstaedter, M. N., and Turaga, S. C. (2009). Maximin affinity learning of image segmentation. In *NIPS'2009*, pages 1865–1873. 306
- Brown, P. F., Cocke, J., Pietra, S. A. D., Pietra, V. J. D., Jelinek, F., Lafferty, J. D., Mercer, R. L., and Roossin, P. S. (1990). A statistical approach to machine translation. *Computational linguistics*, **16**(2), 79–85. 18
- Brown, P. F., Pietra, V. J. D., DeSouza, P. V., Lai, J. C., and Mercer, R. L. (1992). Class-based  $n$ -gram models of natural language. *Computational Linguistics*, **18**, 467–479. 394
- Bryson, A. and Ho, Y. (1969). *Applied optimal control: optimization, estimation, and control*. Blaisdell Pub. Co. 194
- Bryson, Jr., A. E. and Denham, W. F. (1961). A steepest-ascent method for solving optimum programming problems. Technical Report BR-1303, Raytheon Company, Missle and Space Division. 194
- Buciluă, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541. ACM. 381
- Burda, Y., Grosse, R., and Salakhutdinov, R. (2015). Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*. 596
- Cai, M., Shi, Y., and Liu, J. (2013). Deep maxout neural networks for speech recognition. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 291–296. IEEE. 167
- Carreira-Perpiñan, M. A. and Hinton, G. E. (2005). On contrastive divergence learning. In *AISTATS'2005*, pages 33–40. 520
- Caruana, R. (1993). Multitask connectionist learning. In *Proceedings of the 1993 Connectionist Models Summer School*, pages 372–379. 210
- Cauchy, A. (1847). Méthode générale pour la résolution de systèmes d'équations simultanées. In *Compte rendu des séances de l'académie des sciences*, pages 536–538. 74, 194
- Cayton, L. (2005). Algorithms for manifold learning. Technical Report CS2008-0923, UCSD. 141
- Chandola, V., Banerjee, A., and Kumar, V. (2009). Anomaly detection: A survey. *ACM computing surveys (CSUR)*, **41**(3), 15. 90

- Chapelle, O., Weston, J., and Schölkopf, B. (2003). Cluster kernels for semi-supervised learning. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15 (NIPS'02)*, pages 585–592, Cambridge, MA. MIT Press. 210
- Chapelle, O., Schölkopf, B., and Zien, A., editors (2006). *Semi-Supervised Learning*. MIT Press, Cambridge, MA. 210, 462
- Chellapilla, K., Puri, S., and Simard, P. (2006). High Performance Convolutional Neural Networks for Document Processing. In Guy Lorette, editor, *Tenth International Workshop on Frontiers in Handwriting Recognition*, La Baule (France). Université de Rennes 1, Suvisoft. <http://www.suvisoft.com>. 20, 21, 379
- Chen, B., Ting, J.-A., Marlin, B. M., and de Freitas, N. (2010). Deep learning of invariant spatio-temporal features from video. NIPS\*2010 Deep Learning and Unsupervised Feature Learning Workshop. 307
- Chen, S. F. and Goodman, J. T. (1999). An empirical study of smoothing techniques for language modeling. *Computer, Speech and Language*, **13**(4), 359–393. 393, 394, 402
- Chen, T., Du, Z., Sun, N., Wang, J., Wu, C., Chen, Y., and Temam, O. (2014a). DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning. In *Proceedings of the 19th international conference on Architectural support for programming languages and operating systems*, pages 269–284. ACM. 384
- Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., and Zhang, Z. (2015). MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*. 23
- Chen, Y., Luo, T., Liu, S., Zhang, S., He, L., Wang, J., Li, L., Chen, T., Xu, Z., Sun, N., et al. (2014b). DaDianNao: A machine-learning supercomputer. In *Microarchitecture (MICRO), 2014 47th Annual IEEE/ACM International Symposium on*, pages 609–622. IEEE. 384
- Chilimbi, T., Suzue, Y., Apacible, J., and Kalyanaraman, K. (2014). Project Adam: Building an efficient and scalable deep learning training system. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI'14)*. 381
- Cho, K., Raiko, T., and Ilin, A. (2010a). Parallel tempering is efficient for learning restricted Boltzmann machines. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN 2010)*, Barcelona, Spain. 514
- Cho, K., Raiko, T., and Ilin, A. (2010b). Parallel tempering is efficient for learning restricted Boltzmann machines. In *IJCNN'2010*. 524

- Cho, K., Raiko, T., and Ilin, A. (2011). Enhanced gradient and adaptive learning rate for training restricted Boltzmann machines. In *ICML'2011*, pages 105–112. 575
- Cho, K., Van Merriënboer, B., Gülcöhre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014a). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734. Association for Computational Linguistics. 338
- Cho, K., van Merriënboer, B., Gulcehre, C., Bougares, F., Schwenk, H., and Bengio, Y. (2014b). Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP 2014)*. 403
- Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014c). On the properties of neural machine translation: Encoder-decoder approaches. *ArXiv e-prints*, **abs/1409.1259**. 351
- Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y. (2014). The loss surface of multilayer networks. 244, 245
- Chorowski, J., Bahdanau, D., Cho, K., and Bengio, Y. (2014). End-to-end continuous speech recognition using attention-based recurrent NN: First results. arXiv:1412.1602. 392
- Christianson, B. (1992). Automatic Hessians by reverse accumulation. *IMA Journal of Numerical Analysis*, **12**(2), 135–150. 193
- Chrupala, G., Kadar, A., and Alishahi, A. (2015). Learning language through pictures. arXiv 1506.03694. 351
- Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. NIPS'2014 Deep Learning workshop, arXiv 1412.3555. 351, 392
- Chung, J., Gülcöhre, C., Cho, K., and Bengio, Y. (2015a). Gated feedback recurrent neural networks. In *ICML'15*. 351
- Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A., and Bengio, Y. (2015b). A recurrent latent variable model for sequential data. In *NIPS'2015*. 596
- Ciresan, D., Meier, U., Masci, J., and Schmidhuber, J. (2012). Multi-column deep neural network for traffic sign classification. *Neural Networks*, **32**, 333–338. 22, 174

- Ciresan, D. C., Meier, U., Gambardella, L. M., and Schmidhuber, J. (2010). Deep big simple neural nets for handwritten digit recognition. *Neural Computation*, **22**, 1–14. 20, 21, 379
- Coates, A. and Ng, A. Y. (2011). The importance of encoding versus training with sparse coding and vector quantization. In *ICML'2011*. 21, 220, 425
- Coates, A., Lee, H., and Ng, A. Y. (2011). An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2011)*. 310, 387
- Coates, A., Huval, B., Wang, T., Wu, D., Catanzaro, B., and Andrew, N. (2013). Deep learning with COTS HPC systems. In S. Dasgupta and D. McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28 (3), pages 1337–1345. JMLR Workshop and Conference Proceedings. 20, 21, 310, 381
- Cohen, N., Sharir, O., and Shashua, A. (2015). On the expressive power of deep learning: A tensor analysis. arXiv:1509.05009. 472
- Collobert, R. (2004). *Large Scale Machine Learning*. Ph.D. thesis, Université de Paris VI, LIP6. 170
- Collobert, R. (2011). Deep learning for efficient discriminative parsing. In *AISTATS'2011*. 89, 406
- Collobert, R. and Weston, J. (2008a). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML'2008*. 401, 406
- Collobert, R. and Weston, J. (2008b). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML'2008*. 455
- Collobert, R., Bengio, S., and Bengio, Y. (2001). A parallel mixture of SVMs for very large scale problems. Technical Report 12, IDIAP. 383
- Collobert, R., Bengio, S., and Bengio, Y. (2002). Parallel mixture of SVMs for very large scale problem. *Neural Computation*. 383
- Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011a). Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, **12**, 2493–2537. 279, 406, 455, 456
- Collobert, R., Kavukcuoglu, K., and Farabet, C. (2011b). Torch7: A Matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*. 23, 182, 380

- Comon, P. (1994). Independent component analysis - a new concept? *Signal Processing*, **36**, 287–314. 419
- Cortes, C. and Vapnik, V. (1995). Support vector networks. *Machine Learning*, **20**, 273–297. 16, 123
- Couprie, C., Farabet, C., Najman, L., and LeCun, Y. (2013). Indoor semantic segmentation using depth information. In *International Conference on Learning Representations (ICLR2013)*. 22, 174
- Courbariaux, M., Bengio, Y., and David, J.-P. (2015). Low precision arithmetic for deep learning. In *Arxiv:1412.7024, ICLR'2015 Workshop*. 384
- Courville, A., Bergstra, J., and Bengio, Y. (2011a). Unsupervised models of images by spike-and-slab RBMs. In *ICML'2011*. 477
- Courville, A., Bergstra, J., and Bengio, Y. (2011b). Unsupervised models of images by spike-and-slab RBMs. In *ICM ( 1b)*. 581
- Courville, A., Desjardins, G., Bergstra, J., and Bengio, Y. (2014). The spike-and-slab RBM and extensions to discrete and sparse data distributions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **36**(9), 1874–1887. 583
- Cover, T. M. and Thomas, J. A. (2006). *Elements of Information Theory, 2nd Edition*. Wiley-Interscience. 66
- Cox, D. and Pinto, N. (2011). Beyond simple features: A large-scale feature search approach to unconstrained face recognition. In *Automatic Face & Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on*, pages 8–15. IEEE. 310
- Cramér, H. (1946). *Mathematical methods of statistics*. Princeton University Press. 118, 252
- Crick, F. H. C. and Mitchison, G. (1983). The function of dream sleep. *Nature*, **304**, 111–114. 518
- Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, **2**, 303–314. 171
- Dahl, G. E., Ranzato, M., Mohamed, A., and Hinton, G. E. (2010). Phone recognition with the mean-covariance restricted Boltzmann machine. In *Advances in Neural Information Processing Systems (NIPS)*. 22
- Dahl, G. E., Yu, D., Deng, L., and Acero, A. (2012). Context-dependent pre-trained deep neural networks for large vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, **20**(1), 33–42. 391

- Dahl, G. E., Sainath, T. N., and Hinton, G. E. (2013). Improving deep neural networks for LVCSR using rectified linear units and dropout. In *ICASSP'2013*. 391
- Dahl, G. E., Jaitly, N., and Salakhutdinov, R. (2014). Multi-task neural networks for QSAR predictions. arXiv:1406.1231. 24
- Dauphin, Y. and Bengio, Y. (2013). Stochastic ratio matching of RBMs for sparse high-dimensional inputs. In *NIP* ( 1). 528
- Dauphin, Y., Glorot, X., and Bengio, Y. (2011). Large-scale learning of embeddings with reconstruction sampling. In *ICML'2011*. 401
- Dauphin, Y., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., and Bengio, Y. (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *NIPS'2014*. 244, 245
- Davis, A., Rubinstein, M., Wadhwa, N., Mysore, G., Durand, F., and Freeman, W. T. (2014). The visual microphone: Passive recovery of sound from video. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, **33**(4), 79:1–79:10. 385
- Dayan, P. (1990). Reinforcement comparison. In *Connectionist Models: Proceedings of the 1990 Connectionist Summer School*, San Mateo, CA. 590
- Dayan, P. and Hinton, G. E. (1996). Varieties of Helmholtz machine. *Neural Networks*, **9**(8), 1385–1403. 592
- Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. (1995). The Helmholtz machine. *Neural computation*, **7**(5), 889–904. 592
- Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Le, Q., Mao, M., Ranzato, M., Senior, A., Tucker, P., Yang, K., and Ng, A. Y. (2012). Large scale distributed deep networks. In *NIPS'2012*. 23, 381
- Dean, T. and Kanazawa, K. (1989). A model for reasoning about persistence and causation. *Computational Intelligence*, **5**(3), 142–150. 566
- Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, **41**(6), 391–407. 406, 410
- Delalleau, O. and Bengio, Y. (2011). Shallow vs. deep sum-product networks. In *NIPS*. 17, 472
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*. 18

- Deng, J., Berg, A. C., Li, K., and Fei-Fei, L. (2010a). What does classifying more than 10,000 image categories tell us? In *Proceedings of the 11th European Conference on Computer Vision: Part V*, ECCV'10, pages 71–84, Berlin, Heidelberg. Springer-Verlag. 18
- Deng, L. and Yu, D. (2014). Deep learning – methods and applications. *Foundations and Trends in Signal Processing*. 391
- Deng, L., Seltzer, M., Yu, D., Acero, A., Mohamed, A., and Hinton, G. (2010b). Binary coding of speech spectrograms using a deep auto-encoder. In *Interspeech 2010*, Makuhari, Chiba, Japan. 22
- Denil, M., Bazzani, L., Larochelle, H., and de Freitas, N. (2012). Learning where to attend with deep architectures for image tracking. *Neural Computation*, **24**(8), 2151–2184. 313
- Denton, E., Chintala, S., Szlam, A., and Fergus, R. (2015). Deep generative image models using a Laplacian pyramid of adversarial networks. *NIPS*. 599, 612
- Desjardins, G. and Bengio, Y. (2008). Empirical evaluation of convolutional RBMs for vision. Technical Report 1327, Département d’Informatique et de Recherche Opérationnelle, Université de Montréal. 583
- Desjardins, G., Courville, A. C., Bengio, Y., Vincent, P., and Delalleau, O. (2010). Tempered Markov chain Monte Carlo for training of restricted Boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, pages 145–152. 514, 524
- Desjardins, G., Courville, A., and Bengio, Y. (2011). On tracking the partition function. In *NIPS'2011*. 537
- Devlin, J., Zbib, R., Huang, Z., Lamar, T., Schwartz, R., and Makhoul, J. (2014). Fast and robust neural network joint models for statistical machine translation. In *Proc. ACL'2014*. 403
- Devroye, L. (2013). *Non-Uniform Random Variate Generation*. SpringerLink : Bücher. Springer New York. 593
- DiCarlo, J. J. (2013). Mechanisms underlying visual object recognition: Humans vs. neurons vs. machines. *NIPS Tutorial*. 24, 312
- Dinh, L., Krueger, D., and Bengio, Y. (2014). NICE: Non-linear independent components estimation. arXiv:1410.8516. 421
- Donahue, J., Hendricks, L. A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T. (2014). Long-term recurrent convolutional networks for visual recognition and description. arXiv:1411.4389. 90

- Donoho, D. L. and Grimes, C. (2003). Hessian eigenmaps: new locally linear embedding techniques for high-dimensional data. Technical Report 2003-08, Dept. Statistics, Stanford University. 141, 443
- Dosovitskiy, A., Springenberg, J. T., and Brox, T. (2015). Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1538–1546. 594, 601
- Doya, K. (1993). Bifurcations of recurrent neural networks in gradient descent learning. *IEEE Transactions on Neural Networks*, **1**, 75–80. 343, 345
- Dreyfus, S. E. (1962). The numerical solution of variational problems. *Journal of Mathematical Analysis and Applications*, **5**(1), 30–45. 194
- Dreyfus, S. E. (1973). The computational solution of optimal control problems with time lag. *IEEE Transactions on Automatic Control*, **18**(4), 383–385. 194
- Drucker, H. and LeCun, Y. (1992). Improving generalisation performance using double back-propagation. *IEEE Transactions on Neural Networks*, **3**(6), 991–997. 233
- Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*. 261
- Dudik, M., Langford, J., and Li, L. (2011). Doubly robust policy evaluation and learning. In *Proceedings of the 28th International Conference on Machine learning*, ICML '11. 410
- Dugas, C., Bengio, Y., Bélisle, F., and Nadeau, C. (2001). Incorporating second-order functional knowledge for better option pricing. In T. Leen, T. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13 (NIPS'00)*, pages 472–478. MIT Press. 61, 170
- Dziugaite, G. K., Roy, D. M., and Ghahramani, Z. (2015). Training generative neural networks via maximum mean discrepancy optimization. *arXiv preprint arXiv:1505.03906*. 600
- El Hihi, S. and Bengio, Y. (1996). Hierarchical recurrent neural networks for long-term dependencies. In *NIPS 8*. MIT Press. 340, 348
- Elkahky, A. M., Song, Y., and He, X. (2015). A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th International Conference on World Wide Web*, pages 278–288. 408
- Elman, J. L. (1993). Learning and development in neural networks: The importance of starting small. *Cognition*, **48**, 781–799. 279

- Erhan, D., Manzagol, P.-A., Bengio, Y., Bengio, S., and Vincent, P. (2009). The difficulty of training deep architectures and the effect of unsupervised pre-training. In *AISTATS'2009*, pages 153–160. 174
- Erhan, D., Bengio, Y., Courville, A., Manzagol, P., Vincent, P., and Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *J. Machine Learning Res.* 452, 454, 455, 456
- Fahlman, S. E., Hinton, G. E., and Sejnowski, T. J. (1983). Massively parallel architectures for AI: NETL, thistle, and Boltzmann machines. In *Proceedings of the National Conference on Artificial Intelligence AAAI-83*. 486, 559
- Fang, H., Gupta, S., Iandola, F., Srivastava, R., Deng, L., Dollár, P., Gao, J., He, X., Mitchell, M., Platt, J. C., Zitnick, C. L., and Zweig, G. (2015). From captions to visual concepts and back. arXiv:1411.4952. 90
- Farabet, C., LeCun, Y., Kavukcuoglu, K., Culurciello, E., Martini, B., Akselrod, P., and Talay, S. (2011). Large-scale FPGA-based convolutional networks. In R. Bekkerman, M. Bilenko, and J. Langford, editors, *Scaling up Machine Learning: Parallel and Distributed Approaches*. Cambridge University Press. 447
- Farabet, C., Couprie, C., Najman, L., and LeCun, Y. (2013). Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **35**(8), 1915–1929. 22, 174, 306
- Fei-Fei, L., Fergus, R., and Perona, P. (2006). One-shot learning of object categories. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **28**(4), 594–611. 459
- Finn, C., Tan, X. Y., Duan, Y., Darrell, T., Levine, S., and Abbeel, P. (2015). Learning visual feature spaces for robotic manipulation with deep spatial autoencoders. *arXiv preprint arXiv:1509.06113*. 23
- Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, **7**, 179–188. 18, 92
- Földiák, P. (1989). Adaptive network for optimal linear feature extraction. In *International Joint Conference on Neural Networks (IJCNN)*, volume 1, pages 401–405, Washington 1989. IEEE, New York. 421
- Franzius, M., Sprekeler, H., and Wiskott, L. (2007). Slowness and sparseness lead to place, head-direction, and spatial-view cells. 423

- Franzius, M., Wilbert, N., and Wiskott, L. (2008). Invariant object recognition with slow feature analysis. In *Proceedings of the 18th international conference on Artificial Neural Networks, Part I*, ICANN '08, pages 961–970, Berlin, Heidelberg. Springer-Verlag. 423
- Frasconi, P., Gori, M., and Sperduti, A. (1997). On the efficient classification of data structures by neural networks. In *Proc. Int. Joint Conf. on Artificial Intelligence*. 341, 342
- Frasconi, P., Gori, M., and Sperduti, A. (1998). A general framework for adaptive processing of data structures. *IEEE Transactions on Neural Networks*, **9**(5), 768–786. 341, 342
- Freund, Y. and Schapire, R. E. (1996a). Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of Thirteenth International Conference*, pages 148–156, USA. ACM. 222
- Freund, Y. and Schapire, R. E. (1996b). Game theory, on-line prediction and boosting. In *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, pages 325–332. 222
- Frey, B. J. (1998). *Graphical models for machine learning and digital communication*. MIT Press. 602
- Frey, B. J., Hinton, G. E., and Dayan, P. (1996). Does the wake-sleep algorithm learn good density estimators? In D. Touretzky, M. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems 8 (NIPS'95)*, pages 661–670. MIT Press, Cambridge, MA. 557
- Frobenius, G. (1908). Über matrizen aus positiven elementen, s. *B. Preuss. Akad. Wiss. Berlin, Germany*. 508
- Fukushima, K. (1975). Cognitron: A self-organizing multilayered neural network. *Biological Cybernetics*, **20**, 121–136. 14, 195, 451
- Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, **36**, 193–202. 14, 20, 21, 195, 313
- Gal, Y. and Ghahramani, Z. (2015). Bayesian convolutional neural networks with Bernoulli approximate variational inference. *arXiv preprint arXiv:1506.02158*. 227
- Gallinari, P., LeCun, Y., Thiria, S., and Fogelman-Soulie, F. (1987). Memoires associatives distribuees. In *Proceedings of COGNITIVA 87*, Paris, La Villette. 440

- Garcia-Duran, A., Bordes, A., Usunier, N., and Grandvalet, Y. (2015). Combining two and three-way embeddings models for link prediction in knowledge bases. *arXiv preprint arXiv:1506.00999*. 412
- Garofolo, J. S., Lamel, L. F., Fisher, W. M., Fiscus, J. G., and Pallett, D. S. (1993). Darpa timit acoustic-phonetic continuous speech corpus cd-rom. nist speech disc 1-1.1. *NASA STI/Recon Technical Report N*, **93**, 27403. 390
- Garson, J. (1900). The metric system of identification of criminals, as used in Great Britain and Ireland. *The Journal of the Anthropological Institute of Great Britain and Ireland*, (2), 177–227. 18
- Gers, F. A., Schmidhuber, J., and Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural computation*, **12**(10), 2451–2471. 349, 352
- Ghahramani, Z. and Hinton, G. E. (1996). The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1, Dpt. of Comp. Sci., Univ. of Toronto. 417
- Gillick, D., Brunk, C., Vinyals, O., and Subramanya, A. (2015). Multilingual language processing from bytes. *arXiv preprint arXiv:1512.00103*. 406
- Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2015). Region-based convolutional networks for accurate object detection and segmentation. 363
- Giudice, M. D., Manera, V., and Keysers, C. (2009). Programmed to learn? The ontogeny of mirror neurons. *Dev. Sci.*, **12**(2), 350--363. 560
- Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *AISTATS'2010*. 258
- Glorot, X., Bordes, A., and Bengio, Y. (2011a). Deep sparse rectifier neural networks. In *AISTATS'2011*. 15, 150, 170, 195
- Glorot, X., Bordes, A., and Bengio, Y. (2011b). Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML'2011*. 433
- Glorot, X., Bordes, A., and Bengio, Y. (2011c). Domain adaptation for large-scale sentiment classification: A deep learning approach. In ICM ( 1b), pages 97–110. 457
- Goldberger, J., Roweis, S., Hinton, G. E., and Salakhutdinov, R. (2005). Neighbourhood components analysis. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17 (NIPS'04)*. MIT Press. 101

- Gong, S., McKenna, S., and Psarrou, A. (2000). *Dynamic Vision: From Images to Face Recognition*. Imperial College Press. 142, 443
- Goodfellow, I., Le, Q., Saxe, A., and Ng, A. (2009). Measuring invariances in deep networks. In Y. Bengio, D. Schuurmans, C. Williams, J. Lafferty, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22 (NIPS'09)*, pages 646–654. 219
- Goodfellow, I., Koenig, N., Muja, M., Pantofaru, C., Sorokin, A., and Takayama, L. (2010). Help me help you: Interfaces for personal robots. In *Proc. of Human Robot Interaction (HRI)*, Osaka, Japan. ACM Press, ACM Press. 88
- Goodfellow, I., Mirza, M., Xiao, D., Courville, A., and Bengio, Y. (2014a). An empirical investigation of catastrophic forgetting in gradient-based neural networks. In *ICLR'14*. 168
- Goodfellow, I. J. (2010). Technical report: Multidimensional, downsampled convolution for autoencoders. Technical report, Université de Montréal. 302
- Goodfellow, I. J. (2014). On distinguishability criteria for estimating generative models. In *International Conference on Learning Representations, Workshops Track*. 531, 598
- Goodfellow, I. J., Courville, A., and Bengio, Y. (2011). Spike-and-slab sparse coding for unsupervised feature discovery. In *NIPS Workshop on Challenges in Learning Hierarchical Models*. 454, 458
- Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013a). Maxout networks. In *ICML'2013*. 167
- Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013b). Maxout networks. In ICM ( 1c), pages 1319–1327. 227, 292, 312
- Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013c). Maxout networks. Technical Report arXiv:1302.4389, Université de Montréal. 387
- Goodfellow, I. J., Mirza, M., Courville, A., and Bengio, Y. (2013d). Multi-prediction deep Boltzmann machines. In NIP ( 1). 89, 526, 572, 574, 575, 576, 577, 596
- Goodfellow, I. J., Warde-Farley, D., Lamblin, P., Dumoulin, V., Mirza, M., Pascanu, R., Bergstra, J., Bastien, F., and Bengio, Y. (2013e). Pylearn2: a machine learning research library. *arXiv preprint arXiv:1308.4214*. 23, 380
- Goodfellow, I. J., Courville, A., and Bengio, Y. (2013f). Scaling up spike-and-slab models for unsupervised feature learning. *IEEE T. PAMI*, pages 1902–1914. 425, 426, 555

- Goodfellow, I. J., Courville, A., and Bengio, Y. (2013g). Scaling up spike-and-slab models for unsupervised feature learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **35**(8), 1902–1914. 583
- Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014b). Explaining and harnessing adversarial examples. *CoRR*, **abs/1412.6572**. 230, 231, 233, 473, 474
- Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014c). Generative adversarial networks. In *NIPS'2014*. 464, 588, 597, 598, 601
- Goodfellow, I. J., Bulatov, Y., Ibarz, J., Arnoud, S., and Shet, V. (2014d). Multi-digit number recognition from Street View imagery using deep convolutional neural networks. In *International Conference on Learning Representations*. 22, 89, 174, 175, 334, 359, 382
- Goodfellow, I. J., Vinyals, O., and Saxe, A. M. (2015). Qualitatively characterizing neural network optimization problems. In *International Conference on Learning Representations*. 244, 245, 246, 248
- Goodman, J. (2001). Classes for fast maximum entropy training. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Utah. 397
- Gori, M. and Tesi, A. (1992). On the problem of local minima in backpropagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-14**(1), 76–86. 243
- Gosset, W. S. (1908). The probable error of a mean. *Biometrika*, **6**(1), 1–25. Originally published under the pseudonym “Student”. 18
- Gouws, S., Bengio, Y., and Corrado, G. (2014). BilBOWA: Fast bilingual distributed representations without word alignments. Technical report, arXiv:1410.2455. 406, 459
- Graf, H. P. and Jackel, L. D. (1989). Analog electronic neural network circuits. *Circuits and Devices Magazine, IEEE*, **5**(4), 44–49. 384
- Graves, A. (2011). Practical variational inference for neural networks. In *NIPS'2011*. 208
- Graves, A. (2012). *Supervised Sequence Labelling with Recurrent Neural Networks*. Studies in Computational Intelligence. Springer. 320, 336, 351, 392
- Graves, A. (2013). Generating sequences with recurrent neural networks. Technical report, arXiv:1308.0850. 164, 349, 351, 354, 358
- Graves, A. and Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. In *ICML'2014*. 349

- Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, **18**(5), 602–610. 337
- Graves, A. and Schmidhuber, J. (2009). Offline handwriting recognition with multidimensional recurrent neural networks. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *NIPS'2008*, pages 545–552. 337
- Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *ICML'2006*, pages 369–376, Pittsburgh, USA. 392
- Graves, A., Liwicki, M., Bunke, H., Schmidhuber, J., and Fernández, S. (2008). Unconstrained on-line handwriting recognition with recurrent neural networks. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *NIPS'2007*, pages 577–584. 337
- Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., and Schmidhuber, J. (2009). A novel connectionist system for unconstrained handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **31**(5), 855–868. 349
- Graves, A., Mohamed, A., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *ICASSP'2013*, pages 6645–6649. 337, 340, 349, 392
- Graves, A., Wayne, G., and Danihelka, I. (2014). Neural Turing machines. arXiv:1410.5401. 23, 356
- Grefenstette, E., Hermann, K. M., Suleyman, M., and Blunsom, P. (2015). Learning to transduce with unbounded memory. In *NIPS'2015*. 356
- Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. (2015). LSTM: a search space odyssey. *arXiv preprint arXiv:1503.04069*. 352
- Gregor, K. and LeCun, Y. (2010a). Emergence of complex-like cells in a temporal product network with local receptive fields. Technical report, arXiv:1006.0448. 300
- Gregor, K. and LeCun, Y. (2010b). Learning fast approximations of sparse coding. In L. Bottou and M. Littman, editors, *Proceedings of the Twenty-seventh International Conference on Machine Learning (ICML-10)*. ACM. 558
- Gregor, K., Danihelka, I., Mnih, A., Blundell, C., and Wierstra, D. (2014). Deep autoregressive networks. In *International Conference on Machine Learning (ICML'2014)*. 592
- Gregor, K., Danihelka, I., Graves, A., and Wierstra, D. (2015). DRAW: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*. 596

- Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *The Journal of Machine Learning Research*, **13**(1), 723–773. 601
- Guillaume Desjardins, Karen Simonyan, R. P. K. K. (2015). Natural neural networks. Technical report, arXiv:1507.00210. 273
- Gulcehre, C. and Bengio, Y. (2013). Knowledge matters: Importance of prior information for optimization. Technical Report arXiv:1301.4083, Universite de Montreal. 22
- Guo, H. and Gelfand, S. B. (1992). Classification trees with neural network feature extraction. *Neural Networks, IEEE Transactions on*, **3**(6), 923–933. 383
- Gupta, S., Agrawal, A., Gopalakrishnan, K., and Narayanan, P. (2015). Deep learning with limited numerical precision. *CoRR*, **abs/1502.02551**. 384
- Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of The Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS'10)*. 529
- Hadsell, R., Sermanet, P., Ben, J., Erkan, A., Han, J., Muller, U., and LeCun, Y. (2007). Online learning for offroad robots: Spatial label propagation to learn long-range traversability. In *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA. 386
- Hajnal, A., Maass, W., Pudlak, P., Szegedy, M., and Turan, G. (1993). Threshold circuits of bounded depth. *J. Comput. System. Sci.*, **46**, 129–154. 172
- Håstad, J. (1986). Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th annual ACM Symposium on Theory of Computing*, pages 6–20, Berkeley, California. ACM Press. 172
- Håstad, J. and Goldmann, M. (1991). On the power of small-depth threshold circuits. *Computational Complexity*, **1**, 113–129. 172
- Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The elements of statistical learning: data mining, inference and prediction*. Springer Series in Statistics. Springer Verlag. 126
- He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. *arXiv preprint arXiv:1502.01852*. 23, 167
- Hebb, D. O. (1949). *The Organization of Behavior*. Wiley, New York. 13, 15, 560
- Henaff, M., Jarrett, K., Kavukcuoglu, K., and LeCun, Y. (2011). Unsupervised learning of sparse features for scalable audio classification. In *ISMIR'11*. 447

- Henderson, J. (2003). Inducing history representations for broad coverage statistical parsing. In *HLT-NAACL*, pages 103–110. 406
- Henderson, J. (2004). Discriminative training of a neural network statistical parser. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, page 95. 406
- Henniges, M., Puertas, G., Bornschein, J., Eggert, J., and Lücke, J. (2010). Binary sparse coding. In *Latent Variable Analysis and Signal Separation*, pages 450–457. Springer. 546
- Herault, J. and Ans, B. (1984). Circuits neuronaux à synapses modifiables: Décodage de messages composites par apprentissage non supervisé. *Comptes Rendus de l' Académie des Sciences*, **299(III-13)**, 525--528. 419
- Hinton, G., Deng, L., Dahl, G. E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., and Kingsbury, B. (2012a). Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, **29**(6), 82–97. 22, 391
- Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*. 381
- Hinton, G. E. (1989). Connectionist learning procedures. *Artificial Intelligence*, **40**, 185–234. 421
- Hinton, G. E. (1990). Mapping part-whole hierarchies into connectionist networks. *Artificial Intelligence*, **46**(1), 47–75. 356
- Hinton, G. E. (1999). Products of experts. In *Proceedings of the Ninth International Conference on Artificial Neural Networks (ICANN)*, volume 1, pages 1–6, Edinburgh, Scotland. IEE. 486
- Hinton, G. E. (2000). Training products of experts by minimizing contrastive divergence. Technical Report GCNU TR 2000-004, Gatsby Unit, University College London. 519, 578
- Hinton, G. E. (2006). To recognize shapes, first learn to generate images. Technical Report UTML TR 2006-003, University of Toronto. 451
- Hinton, G. E. (2007a). How to do backpropagation in a brain. Invited talk at the NIPS'2007 Deep Learning Workshop. 560
- Hinton, G. E. (2007b). Learning multiple layers of representation. *Trends in cognitive sciences*, **11**(10), 428–434. 564
- Hinton, G. E. (2010). A practical guide to training restricted Boltzmann machines. Technical Report UTML TR 2010-003, Comp. Sc., University of Toronto. 519

- Hinton, G. E. (2012). Tutorial on deep learning. IPAM Graduate Summer School: Deep Learning, Feature Learning. 262
- Hinton, G. E. and Ghahramani, Z. (1997). Generative models for discovering sparse distributed representations. *Philosophical Transactions of the Royal Society of London*. 128
- Hinton, G. E. and McClelland, J. L. (1988). Learning representations by recirculation. In *NIPS'1987*, pages 358–366. 429
- Hinton, G. E. and Roweis, S. (2003). Stochastic neighbor embedding. In *NIPS'2002*. 443
- Hinton, G. E. and Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*, **313**(5786), 504–507. 435, 448, 451, 452, 454
- Hinton, G. E. and Sejnowski, T. J. (1986). Learning and relearning in Boltzmann machines. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, volume 1, chapter 7, pages 282–317. MIT Press, Cambridge. 486, 559
- Hinton, G. E. and Sejnowski, T. J. (1999). *Unsupervised learning: foundations of neural computation*. MIT press. 462
- Hinton, G. E. and Shallice, T. (1991). Lesioning an attractor network: investigations of acquired dyslexia. *Psychological review*, **98**(1), 74. 12
- Hinton, G. E. and Zemel, R. S. (1994). Autoencoders, minimum description length, and Helmholtz free energy. In *NIPS'1993*. 429
- Hinton, G. E., Sejnowski, T. J., and Ackley, D. H. (1984a). Boltzmann machines: Constraint satisfaction networks that learn. Technical Report TR-CMU-CS-84-119, Carnegie-Mellon University, Dept. of Computer Science. 486
- Hinton, G. E., Sejnowski, T. J., and Ackley, D. H. (1984b). Boltzmann machines: Constraint satisfaction networks that learn. Technical Report TR-CMU-CS-84-119, Carnegie-Mellon University, Dept. of Computer Science. 559
- Hinton, G. E., McClelland, J., and Rumelhart, D. (1986). Distributed representations. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1, pages 77–109. MIT Press, Cambridge. 16, 194, 449
- Hinton, G. E., Revow, M., and Dayan, P. (1995a). Recognizing handwritten digits using mixtures of linear models. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7 (NIPS'94)*, pages 1015–1022. MIT Press, Cambridge, MA. 417

- Hinton, G. E., Dayan, P., Frey, B. J., and Neal, R. M. (1995b). The wake-sleep algorithm for unsupervised neural networks. *Science*, **268**, 1558–1161. 431, 557
- Hinton, G. E., Dayan, P., and Revow, M. (1997). Modelling the manifolds of images of handwritten digits. *IEEE Transactions on Neural Networks*, **8**, 65–74. 426
- Hinton, G. E., Welling, M., Teh, Y. W., and Osindero, S. (2001). A new view of ICA. In *Proceedings of 3rd International Conference on Independent Component Analysis and Blind Signal Separation (ICA'01)*, pages 746–751, San Diego, CA. 419
- Hinton, G. E., Osindero, S., and Teh, Y. (2006a). A fast learning algorithm for deep belief nets. *Neural Computation*, **18**, 1527–1554. 13, 17, 21, 506, 564, 565
- Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006b). A fast learning algorithm for deep belief nets. *Neural Computation*, **18**, 1527–1554. 125, 451, 452
- Hinton, G. E., Deng, L., Yu, D., Dahl, G. E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., and Kingsbury, B. (2012b). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Process. Mag.*, **29**(6), 82–97. 89
- Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2012c). Improving neural networks by preventing co-adaptation of feature detectors. Technical report, arXiv:1207.0580. 205, 226
- Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2012d). Improving neural networks by preventing co-adaptation of feature detectors. Technical report, arXiv:1207.0580. 229
- Hinton, G. E., Vinyals, O., and Dean, J. (2014). Dark knowledge. Invited talk at the BayLearn Bay Area Machine Learning Symposium. 381
- Hochreiter, S. (1991a). Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, T.U. München. 343, 344
- Hochreiter, S. (1991b). Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München. 16
- Hochreiter, S. and Schmidhuber, J. (1995). Simplifying neural nets by discovering flat minima. In *Advances in Neural Information Processing Systems 7*, pages 529–536. MIT Press. 209
- Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, **9**(8), 1735–1780. 16, 349, 351

- Hochreiter, S., Bengio, Y., and Frasconi, P. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In J. Kolen and S. Kremer, editors, *Field Guide to Dynamical Recurrent Networks*. IEEE Press. 351
- Holi, J. L. and Hwang, J.-N. (1993). Finite precision error analysis of neural network hardware implementations. *Computers, IEEE Transactions on*, **42**(3), 281–290. 384
- Holt, J. L. and Baker, T. E. (1991). Back propagation simulations using limited precision calculations. In *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, volume 2, pages 121–126. IEEE. 384
- Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, **2**, 359–366. 171
- Hornik, K., Stinchcombe, M., and White, H. (1990). Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural networks*, **3**(5), 551–560. 171
- Hsu, F.-H. (2002). *Behind Deep Blue: Building the Computer That Defeated the World Chess Champion*. Princeton University Press, Princeton, NJ, USA. 2
- Huang, F. and Ogata, Y. (2002). Generalized pseudo-likelihood estimates for Markov random fields on lattice. *Annals of the Institute of Statistical Mathematics*, **54**(1), 1–18. 525
- Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., and Heck, L. (2013). Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2333–2338. ACM. 408
- Hubel, D. and Wiesel, T. (1968). Receptive fields and functional architecture of monkey striate cortex. *Journal of Physiology (London)*, **195**, 215–243. 311
- Hubel, D. H. and Wiesel, T. N. (1959). Receptive fields of single neurons in the cat's striate cortex. *Journal of Physiology*, **148**, 574–591. 311
- Hubel, D. H. and Wiesel, T. N. (1962). Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex. *Journal of Physiology (London)*, **160**, 106–154. 311
- Huszár, F. (2015). How (not) to train your generative model: schedule sampling, likelihood, adversary? *arXiv:1511.05101*. 596
- Hutter, F., Hoos, H., and Leyton-Brown, K. (2011). Sequential model-based optimization for general algorithm configuration. In *LION-5*. Extended version as UBC Tech report TR-2010-10. 371

- Hyvönen, H. (1996). Turing machines are recurrent neural networks. In *STeP'96*, pages 13–24. 325
- Hyvönen, A. (1999). Survey on independent component analysis. *Neural Computing Surveys*, **2**, 94–128. 419
- Hyvönen, A. (2005a). Estimation of non-normalized statistical models using score matching. *Journal of Machine Learning Research*, **6**, 695–709. 437
- Hyvönen, A. (2005b). Estimation of non-normalized statistical models using score matching. *J. Machine Learning Res.*, **6**. 526
- Hyvönen, A. (2007a). Connections between score matching, contrastive divergence, and pseudolikelihood for continuous-valued variables. *IEEE Transactions on Neural Networks*, **18**, 1529–1531. 527
- Hyvönen, A. (2007b). Some extensions of score matching. *Computational Statistics and Data Analysis*, **51**, 2499–2512. 527
- Hyvönen, A. and Hoyer, P. O. (1999). Emergence of topography and complex cell properties from natural images using extensions of ica. In *NIPS*, pages 827–833. 421
- Hyvönen, A. and Pajunen, P. (1999). Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks*, **12**(3), 429–439. 420
- Hyvönen, A., Karhunen, J., and Oja, E. (2001a). *Independent Component Analysis*. Wiley-Interscience. 419
- Hyvönen, A., Hoyer, P. O., and Inki, M. O. (2001b). Topographic independent component analysis. *Neural Computation*, **13**(7), 1527–1558. 421
- Hyvönen, A., Hurri, J., and Hoyer, P. O. (2009). *Natural Image Statistics: A probabilistic approach to early computational vision*. Springer-Verlag. 316
- Iba, Y. (2001). Extended ensemble Monte Carlo. *International Journal of Modern Physics*, **C12**, 623–656. 514
- Inayoshi, H. and Kurita, T. (2005). Improved generalization by adding both auto-association and hidden-layer noise to neural-network-based-classifiers. *IEEE Workshop on Machine Learning for Signal Processing*, pages 141–146. 440
- Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. 88, 271, 273

- Jacobs, R. A. (1988). Increased rates of convergence through learning rate adaptation. *Neural networks*, **1**(4), 295–307. 261
- Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, **3**, 79–87. 163, 383
- Jaeger, H. (2003). Adaptive nonlinear system identification with echo state networks. In *Advances in Neural Information Processing Systems 15*. 345
- Jaeger, H. (2007a). Discovering multiscale dynamical features with hierarchical echo state networks. Technical report, Jacobs University. 340
- Jaeger, H. (2007b). Echo state network. *Scholarpedia*, **2**(9), 2330. 345
- Jaeger, H. (2012). Long short-term memory in echo state networks: Details of a simulation study. Technical report, Technical report, Jacobs University Bremen. 346
- Jaeger, H. and Haas, H. (2004). Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, **304**(5667), 78–80. 21, 345
- Jaeger, H., Lukosevicius, M., Popovici, D., and Siewert, U. (2007). Optimization and applications of echo state networks with leaky-integrator neurons. *Neural Networks*, **20**(3), 335–352. 348
- Jain, V., Murray, J. F., Roth, F., Turaga, S., Zhigulin, V., Briggman, K. L., Helmstaedter, M. N., Denk, W., and Seung, H. S. (2007). Supervised learning of image restoration with convolutional networks. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE. 306
- Jaitly, N. and Hinton, G. (2011). Learning a better representation of speech soundwaves using restricted Boltzmann machines. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5884–5887. IEEE. 390
- Jaitly, N. and Hinton, G. E. (2013). Vocal tract length perturbation (VTLP) improves speech recognition. In *ICML'2013*. 207
- Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y. (2009a). What is the best multi-stage architecture for object recognition? In *Proc. International Conference on Computer Vision (ICCV'09)*, pages 2146–2153. IEEE. 15, 167
- Jarrett, K., Kavukcuoglu, K., Ranzato, M., and LeCun, Y. (2009b). What is the best multi-stage architecture for object recognition? In *ICCV'09*. 20, 21, 150, 195, 310, 447

- Jarzynski, C. (1997). Nonequilibrium equality for free energy differences. *Phys. Rev. Lett.*, **78**, 2690–2693. 533, 536
- Jaynes, E. T. (2003). *Probability Theory: The Logic of Science*. Cambridge University Press. 47
- Jean, S., Cho, K., Memisevic, R., and Bengio, Y. (2014). On using very large target vocabulary for neural machine translation. arXiv:1412.2007. 403
- Jelinek, F. and Mercer, R. L. (1980). Interpolated estimation of Markov source parameters from sparse data. In E. S. Gelsema and L. N. Kanal, editors, *Pattern Recognition in Practice*. North-Holland, Amsterdam. 393, 402
- Jia, Y. (2013). Caffe: An open source convolutional architecture for fast feature embedding. <http://caffe.berkeleyvision.org/>. 23, 182
- Jia, Y., Huang, C., and Darrell, T. (2012). Beyond spatial pyramids: Receptive field learning for pooled image features. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3370–3377. IEEE. 293
- Jim, K.-C., Giles, C. L., and Horne, B. G. (1996). An analysis of noise in recurrent neural networks: convergence and generalization. *IEEE Transactions on Neural Networks*, **7**(6), 1424–1438. 208
- Jordan, M. I. (1998). *Learning in Graphical Models*. Kluwer, Dordrecht, Netherlands. 16
- Joulin, A. and Mikolov, T. (2015). Inferring algorithmic patterns with stack-augmented recurrent nets. *arXiv preprint arXiv:1503.01007*. 356
- Jozefowicz, R., Zaremba, W., and Sutskever, I. (2015). An empirical evaluation of recurrent network architectures. In *ICML'2015*. 260, 351, 352
- Judd, J. S. (1989). *Neural Network Design and the Complexity of Learning*. MIT press. 250
- Jutten, C. and Herault, J. (1991). Blind separation of sources, part I: an adaptive algorithm based on neuromimetic architecture. *Signal Processing*, **24**, 1–10. 419
- Kahou, S. E., Pal, C., Bouthillier, X., Froumenty, P., Gülcühre, c., Memisevic, R., Vincent, P., Courville, A., Bengio, Y., Ferrari, R. C., Mirza, M., Jean, S., Carrier, P. L., Dauphin, Y., Boulanger-Lewandowski, N., Aggarwal, A., Zumer, J., Lamblin, P., Raymond, J.-P., Desjardins, G., Pascanu, R., Warde-Farley, D., Torabi, A., Sharma, A., Bengio, E., Côté, M., Konda, K. R., and Wu, Z. (2013). Combining modality specific deep neural networks for emotion recognition in video. In *Proceedings of the 15th ACM on International Conference on Multimodal Interaction*. 174

- Kalchbrenner, N. and Blunsom, P. (2013). Recurrent continuous translation models. In *EMNLP'2013*. 403
- Kalchbrenner, N., Danihelka, I., and Graves, A. (2015). Grid long short-term memory. *arXiv preprint arXiv:1507.01526*. 338
- Kamyshanska, H. and Memisevic, R. (2015). The potential energy of an autoencoder. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 439
- Karpathy, A. and Li, F.-F. (2015). Deep visual-semantic alignments for generating image descriptions. In *CVPR'2015*. arXiv:1412.2306. 90
- Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *CVPR*. 18
- Karush, W. (1939). *Minima of Functions of Several Variables with Inequalities as Side Constraints*. Master's thesis, Dept. of Mathematics, Univ. of Chicago. 85
- Katz, S. M. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **ASSP-35**(3), 400–401. 393, 402
- Kavukcuoglu, K., Ranzato, M., and LeCun, Y. (2008). Fast inference in sparse coding algorithms with applications to object recognition. Technical report, Computational and Biological Learning Lab, Courant Institute, NYU. Tech Report CBLL-TR-2008-12-01. 447
- Kavukcuoglu, K., Ranzato, M.-A., Fergus, R., and LeCun, Y. (2009). Learning invariant features through topographic filter maps. In *CVPR'2009*. 447
- Kavukcuoglu, K., Sermanet, P., Boureau, Y.-L., Gregor, K., Mathieu, M., and LeCun, Y. (2010). Learning convolutional feature hierarchies for visual recognition. In *NIPS'2010*. 310, 447
- Kelley, H. J. (1960). Gradient theory of optimal flight paths. *ARS Journal*, **30**(10), 947–954. 194
- Khan, F., Zhu, X., and Mutlu, B. (2011). How do humans teach: On curriculum learning and teaching dimension. In *Advances in Neural Information Processing Systems 24 (NIPS'11)*, pages 1449–1457. 280
- Kim, S. K., McAfee, L. C., McMahon, P. L., and Olukotun, K. (2009). A highly scalable restricted Boltzmann machine FPGA implementation. In *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, pages 367–372. IEEE. 384

- Kindermann, R. (1980). *Markov Random Fields and Their Applications (Contemporary Mathematics ; V. 1)*. American Mathematical Society. 482
- Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. 262
- Kingma, D. and LeCun, Y. (2010a). Regularized estimation of image statistics by score matching. In *NIPS'2010*. 438
- Kingma, D. and LeCun, Y. (2010b). Regularized estimation of image statistics by score matching. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1126–1134. 528
- Kingma, D., Rezende, D., Mohamed, S., and Welling, M. (2014). Semi-supervised learning with deep generative models. In *NIPS'2014*. 363
- Kingma, D. P. (2013). Fast gradient-based inference with continuous latent variable models in auxiliary form. Technical report, arxiv:1306.0733. 558, 588, 594
- Kingma, D. P. and Welling, M. (2014a). Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*. 588, 597
- Kingma, D. P. and Welling, M. (2014b). Efficient gradient-based inference through transformations between bayes nets and neural nets. Technical report, arxiv:1402.0480. 588
- Kirkpatrick, S., Jr., C. D. G., , and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, **220**, 671–680. 279
- Kiros, R., Salakhutdinov, R., and Zemel, R. (2014a). Multimodal neural language models. In *ICML'2014*. 90
- Kiros, R., Salakhutdinov, R., and Zemel, R. (2014b). Unifying visual-semantic embeddings with multimodal neural language models. *arXiv:1411.2539 [cs.LG]*. 90, 349
- Klementiev, A., Titov, I., and Bhattacharai, B. (2012). Inducing crosslingual distributed representations of words. In *Proceedings of COLING 2012*. 406, 459
- Knowles-Barley, S., Jones, T. R., Morgan, J., Lee, D., Kasthuri, N., Lichtman, J. W., and Pfister, H. (2014). Deep learning for the connectome. *GPU Technology Conference*. 24
- Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press. 496, 506, 551

- Konig, Y., Bourlard, H., and Morgan, N. (1996). REMAP: Recursive estimation and maximization of a posteriori probabilities – application to transition-based connectionist speech recognition. In D. Touretzky, M. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems 8 (NIPS'95)*. MIT Press, Cambridge, MA. 390
- Koren, Y. (2009). The BellKor solution to the Netflix grand prize. 222, 408
- Kotzias, D., Denil, M., de Freitas, N., and Smyth, P. (2015). From group to individual labels using deep features. In *ACM SIGKDD*. 93
- Koutnik, J., Greff, K., Gomez, F., and Schmidhuber, J. (2014). A clockwork RNN. In *ICML'2014*. 348
- Kočiský, T., Hermann, K. M., and Blunsom, P. (2014). Learning Bilingual Word Representations by Marginalizing Alignments. In *Proceedings of ACL*. 404
- Krause, O., Fischer, A., Glasmachers, T., and Igel, C. (2013). Approximation properties of DBNs with binary hidden units and real-valued visible units. In *ICML'2013*. 472
- Krizhevsky, A. (2010). Convolutional deep belief networks on CIFAR-10. Technical report, University of Toronto. Unpublished Manuscript: <http://www.cs.utoronto.ca/~kriz/conv-cifar10-aug2010.pdf>. 380
- Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto. 18, 477
- Krizhevsky, A. and Hinton, G. E. (2011). Using very deep autoencoders for content-based image retrieval. In *ESANN*. 448
- Krizhevsky, A., Sutskever, I., and Hinton, G. (2012a). ImageNet classification with deep convolutional neural networks. In *NIPS'2012*. 20, 21, 88, 174, 317
- Krizhevsky, A., Sutskever, I., and Hinton, G. (2012b). ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25 (NIPS'2012)*. 22, 386, 389
- Krueger, K. A. and Dayan, P. (2009). Flexible shaping: how learning in small steps helps. *Cognition*, **110**, 380–394. 279
- Kuhn, H. W. and Tucker, A. W. (1951). Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, pages 481–492, Berkeley, Calif. University of California Press. 85

- Kumar, A., Irsoy, O., Ondruska, P., Iyyer, M., Bradbury, J., Gulrajani, I., and Socher, R. (2015a). Ask me anything: Dynamic memory networks for natural language processing. Technical report, arXiv:1506.07285. 356
- Kumar, A., Irsoy, O., Su, J., Bradbury, J., English, R., Pierce, B., Ondruska, P., Iyyer, M., Gulrajani, I., and Socher, R. (2015b). Ask me anything: Dynamic memory networks for natural language processing. *arXiv:1506.07285*. 412
- Kumar, M. P., Packer, B., and Koller, D. (2010). Self-paced learning for latent variable models. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1189–1197. 279
- Lang, K. J. and Hinton, G. E. (1988). The development of the time-delay neural network architecture for speech recognition. Technical Report CMU-CS-88-152, Carnegie-Mellon University. 313, 319, 347
- Lang, K. J., Waibel, A. H., and Hinton, G. E. (1990). A time-delay neural network architecture for isolated word recognition. *Neural networks*, 3(1), 23–43. 319
- Langford, J. and Zhang, T. (2008). The epoch-greedy algorithm for contextual multi-armed bandits. In *NIPS'2008*, pages 1096--1103. 409
- Lappalainen, H., Giannakopoulos, X., Honkela, A., and Karhunen, J. (2000). Nonlinear independent component analysis using ensemble learning: Experiments and discussion. In *Proc. ICA*. Citeseer. 420
- Larochelle, H. and Bengio, Y. (2008a). Classification using discriminative restricted Boltzmann machines. In *ICML'2008*. 210, 586, 610
- Larochelle, H. and Bengio, Y. (2008b). Classification using discriminative restricted Boltzmann machines. In *ICML ( 1a)*, pages 536–543. 219, 453
- Larochelle, H. and Hinton, G. E. (2010). Learning to combine foveal glimpses with a third-order Boltzmann machine. In *Advances in Neural Information Processing Systems 23*, pages 1243–1251. 313
- Larochelle, H. and Murray, I. (2011). The Neural Autoregressive Distribution Estimator. In *AISTATS'2011*. 602, 604, 605
- Larochelle, H., Erhan, D., and Bengio, Y. (2008). Zero-data learning of new tasks. In *AAAI Conference on Artificial Intelligence*. 459
- Larochelle, H., Bengio, Y., Louradour, J., and Lamblin, P. (2009). Exploring strategies for training deep neural networks. In *JML ( 1)*, pages 1–40. 455

- Lasserre, J. A., Bishop, C. M., and Minka, T. P. (2006). Principled hybrids of generative and discriminative models. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR'06)*, pages 87–94, Washington, DC, USA. IEEE Computer Society. 210, 218
- Le, Q., Ngiam, J., Chen, Z., hao Chia, D. J., Koh, P. W., and Ng, A. (2010). Tiled convolutional neural networks. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23 (NIPS'10)*, pages 1279–1287. 300
- Le, Q., Ngiam, J., Coates, A., Lahiri, A., Prochnow, B., and Ng, A. (2011). On optimization methods for deep learning. In *Proc. ICML'2011*. ACM. 270
- Le, Q., Ranzato, M., Monga, R., Devin, M., Corrado, G., Chen, K., Dean, J., and Ng, A. (2012). Building high-level features using large scale unsupervised learning. In *ICML'2012*. 20, 21
- Le Roux, N. and Bengio, Y. (2008). Representational power of restricted Boltzmann machines and deep belief networks. *Neural Computation*, **20**(6), 1631–1649. 472, 560
- Le Roux, N. and Bengio, Y. (2010). Deep belief networks are compact universal approximators. *Neural Computation*, **22**(8), 2192–2207. 472
- LeCun, Y. (1985). Une procédure d'apprentissage pour Réseau à seuil assymétrique. In *Cognitiva 85: A la Frontière de l'Intelligence Artificielle, des Sciences de la Connaissance et des Neurosciences*, pages 599–604, Paris 1985. CESTA, Paris. 194
- LeCun, Y. (1986). Learning processes in an asymmetric threshold network. In E. Bienenstock, F. Fogelman-Soulie, and G. Weisbuch, editors, *Disordered Systems and Biological Organization*, pages 233–240. Springer-Verlag, Berlin, Les Houches 1985. 298
- LeCun, Y. (1987). *Modèles connexionnistes de l'apprentissage*. Ph.D. thesis, Université de Paris VI. 16, 429, 440
- LeCun, Y. (1989). Generalization and network design strategies. Technical Report CRG-TR-89-4, University of Toronto. 281, 298
- LeCun, Y., Jackel, L. D., Boser, B., Denker, J. S., Graf, H. P., Guyon, I., Henderson, D., Howard, R. E., and Hubbard, W. (1989). Handwritten digit recognition: Applications of neural network chips and automatic learning. *IEEE Communications Magazine*, **27**(11), 41–46. 314
- LeCun, Y., Bottou, L., Orr, G. B., and Müller, K.-R. (1998a). Efficient backprop. In *Neural Networks, Tricks of the Trade*, Lecture Notes in Computer Science LNCS 1524. Springer Verlag. 265

- LeCun, Y., Bottou, L., Orr, G. B., and Müller, K. (1998b). Efficient backprop. In *Neural Networks, Tricks of the Trade*. 365
- LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998c). Gradient based learning applied to document recognition. *Proc. IEEE*. 14, 16, 18, 21, 317, 390, 392
- LeCun, Y., Kavukcuoglu, K., and Farabet, C. (2010). Convolutional networks and applications in vision. In *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, pages 253–256. IEEE. 317
- L'Ecuyer, P. (1994). Efficiency improvement and variance reduction. In *Proceedings of the 1994 Winter Simulation Conference*, pages 122--132. 589
- Lee, C.-Y., Xie, S., Gallagher, P., Zhang, Z., and Tu, Z. (2014). Deeply-supervised nets. *arXiv preprint arXiv:1409.5185*. 278
- Lee, H., Battle, A., Raina, R., and Ng, A. (2007). Efficient sparse coding algorithms. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19 (NIPS'06)*, pages 801–808. MIT Press. 544
- Lee, H., Ekanadham, C., and Ng, A. (2008). Sparse deep belief net model for visual area V2. In *NIPS'07*. 219
- Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. (2009). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In L. Bottou and M. Littman, editors, *Proceedings of the Twenty-sixth International Conference on Machine Learning (ICML'09)*. ACM, Montreal, Canada. 310, 583, 584
- Lee, Y. J. and Grauman, K. (2011). Learning the easy things first: self-paced visual category discovery. In *CVPR'2011*. 279
- Leibniz, G. W. (1676). Memoir using the chain rule. (Cited in TMME 7:2&3 p 321-332, 2010). 194
- Lenat, D. B. and Guha, R. V. (1989). *Building large knowledge-based systems; representation and inference in the Cyc project*. Addison-Wesley Longman Publishing Co., Inc. 2
- Leshno, M., Lin, V. Y., Pinkus, A., and Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6, 861--867. 171, 172
- Levenberg, K. (1944). A method for the solution of certain non-linear problems in least squares. *Quarterly Journal of Applied Mathematics*, II(2), 164–168. 266

- L'Hôpital, G. F. A. (1696). *Analyse des infiniment petits, pour l'intelligence des lignes courbes.* Paris: L'Imprimerie Royale. 194
- Li, Y., Swersky, K., and Zemel, R. S. (2015). Generative moment matching networks. *CoRR*, **abs/1502.02761**. 600
- Lin, T., Horne, B. G., Tino, P., and Giles, C. L. (1996). Learning long-term dependencies is not as difficult with NARX recurrent neural networks. *IEEE Transactions on Neural Networks*, **7**(6), 1329–1338. 347
- Lin, Y., Liu, Z., Sun, M., Liu, Y., and Zhu, X. (2015). Learning entity and relation embeddings for knowledge graph completion. In *Proc. AAAI'15*. 412
- Linde, N. (1992). The machine that changed the world, episode 3. Documentary miniseries. 2
- Lindsey, C. and Lindblad, T. (1994). Review of hardware neural networks: a user's perspective. In *Proc. Third Workshop on Neural Networks: From Biology to High Energy Physics*, pages 195--202, Isola d'Elba, Italy. 384
- Linnainmaa, S. (1976). Taylor expansion of the accumulated rounding error. *BIT Numerical Mathematics*, **16**(2), 146–160. 194
- LISA (2008). Deep learning tutorials: Restricted Boltzmann machines. Technical report, LISA Lab, Université de Montréal. 501
- Long, P. M. and Servedio, R. A. (2010). Restricted Boltzmann machines are hard to approximately evaluate or simulate. In *Proceedings of the 27th International Conference on Machine Learning (ICML'10)*. 561
- Lotter, W., Kreiman, G., and Cox, D. (2015). Unsupervised learning of visual structure using predictive generative networks. *arXiv preprint arXiv:1511.06380*. 464, 465
- Lovelace, A. (1842). Notes upon L. F. Menabrea's "Sketch of the Analytical Engine invented by Charles Babbage". 1
- Lu, L., Zhang, X., Cho, K., and Renals, S. (2015). A study of the recurrent neural network encoder-decoder for large vocabulary speech recognition. In *Proc. Interspeech*. 392
- Lu, T., Pál, D., and Pál, M. (2010). Contextual multi-armed bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 485–492. 409
- Luenberger, D. G. (1984). *Linear and Nonlinear Programming*. Addison Wesley. 270
- Lukoševičius, M. and Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, **3**(3), 127–149. 345

- Luo, H., Shen, R., Niu, C., and Ullrich, C. (2011). Learning class-relevant features and class-irrelevant features via a hybrid third-order RBM. In *International Conference on Artificial Intelligence and Statistics*, pages 470–478. 586
- Luo, H., Carrier, P. L., Courville, A., and Bengio, Y. (2013). Texture modeling with convolutional spike-and-slab RBMs and deep extensions. In *AISTATS'2013*. 90
- Lyu, S. (2009). Interpretation and generalization of score matching. In *Proceedings of the Twenty-fifth Conference in Uncertainty in Artificial Intelligence (UAI'09)*. 527
- Ma, J., Sheridan, R. P., Liaw, A., Dahl, G. E., and Svetnik, V. (2015). Deep neural nets as a method for quantitative structure – activity relationships. *J. Chemical information and modeling*. 452
- Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *ICML Workshop on Deep Learning for Audio, Speech, and Language Processing*. 167
- Maass, W. (1992). Bounds for the computational power and learning complexity of analog neural nets (extended abstract). In *Proc. of the 25th ACM Symp. Theory of Computing*, pages 335–344. 172
- Maass, W., Schnitger, G., and Sontag, E. D. (1994). A comparison of the computational power of sigmoid and Boolean threshold circuits. *Theoretical Advances in Neural Computation and Learning*, pages 127–151. 172
- Maass, W., Natschlaeger, T., and Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, **14**(11), 2531–2560. 345
- MacKay, D. (2003). *Information Theory, Inference and Learning Algorithms*. Cambridge University Press. 66
- Maclaurin, D., Duvenaud, D., and Adams, R. P. (2015). Gradient-based hyperparameter optimization through reversible learning. *arXiv preprint arXiv:1502.03492*. 370
- Mao, J., Xu, W., Yang, Y., Wang, J., and Yuille, A. (2014). Deep captioning with multimodal recurrent neural networks (m-rnn). *arXiv:1412.6632 [cs.CV]*. 90
- Marcotte, P. and Savard, G. (1992). Novel approaches to the discrimination problem. *Zeitschrift für Operations Research (Theory)*, **36**, 517–545. 237
- Marlin, B. and de Freitas, N. (2011). Asymptotic efficiency of deterministic estimators for discrete energy-based models: Ratio matching and pseudolikelihood. In *UAI'2011*. 526, 528

- Marlin, B., Swersky, K., Chen, B., and de Freitas, N. (2010). Inductive principles for restricted Boltzmann machine learning. In *AISTATS'2010*, pages 509–516. 522, 527
- Marquardt, D. W. (1963). An algorithm for least-squares estimation of non-linear parameters. *Journal of the Society of Industrial and Applied Mathematics*, **11**(2), 431–441. 266
- Marr, D. and Poggio, T. (1976). Cooperative computation of stereo disparity. *Science*, **194**. 313
- Martens, J. (2010). Deep learning via Hessian-free optimization. In *ICML'2010*, pages 735–742. 259
- Martens, J. and Medabalimi, V. (2014). On the expressive efficiency of sum product networks. *arXiv:1411.7717*. 472
- Martens, J. and Sutskever, I. (2011). Learning recurrent neural networks with Hessian-free optimization. In *Proc. ICML'2011*. ACM. 352, 353
- Mase, S. (1995). Consistency of the maximum pseudo-likelihood estimator of continuous state space Gibbsian processes. *The Annals of Applied Probability*, **5**(3), pp. 603–612. 525
- McClelland, J., Rumelhart, D., and Hinton, G. (1995). The appeal of parallel distributed processing. In *Computation & intelligence*, pages 305–341. American Association for Artificial Intelligence. 15
- McCulloch, W. S. and Pitts, W. (1943). A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, **5**, 115–133. 13
- Mead, C. and Ismail, M. (2012). *Analog VLSI implementation of neural systems*, volume 80. Springer Science & Business Media. 384
- Melchior, J., Fischer, A., and Wiskott, L. (2013). How to center binary deep Boltzmann machines. *arXiv preprint arXiv:1311.1354*. 575
- Memisevic, R. and Hinton, G. E. (2007). Unsupervised learning of image transformations. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR'07)*. 586
- Memisevic, R. and Hinton, G. E. (2010). Learning to represent spatial transformations with factored higher-order Boltzmann machines. *Neural Computation*, **22**(6), 1473–1492. 586
- Mesnil, G., Dauphin, Y., Glorot, X., Rifai, S., Bengio, Y., Goodfellow, I., Lavoie, E., Muller, X., Desjardins, G., Warde-Farley, D., Vincent, P., Courville, A., and Bergstra, J. (2011). Unsupervised and transfer learning challenge: a deep learning approach. In *JMLR W&CP: Proc. Unsupervised and Transfer Learning*, volume 7. 174, 454, 458

- Mesnil, G., Rifai, S., Dauphin, Y., Bengio, Y., and Vincent, P. (2012). Surfing on the manifold. Learning Workshop, Snowbird. 607
- Miikkulainen, R. and Dyer, M. G. (1991). Natural language processing with modular PDP networks and distributed lexicon. *Cognitive Science*, **15**, 343–399. 406
- Mikolov, T. (2012). *Statistical Language Models based on Neural Networks*. Ph.D. thesis, Brno University of Technology. 353
- Mikolov, T., Deoras, A., Kombrink, S., Burget, L., and Cernocky, J. (2011a). Empirical evaluation and combination of advanced language modeling techniques. In *Proc. 12th annual conference of the international speech communication association (INTERSPEECH 2011)*. 402
- Mikolov, T., Deoras, A., Povey, D., Burget, L., and Cernocky, J. (2011b). Strategies for training large scale neural network language models. In *Proc. ASRU'2011*. 279, 402
- Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. In *International Conference on Learning Representations: Workshops Track*. 456
- Mikolov, T., Le, Q. V., and Sutskever, I. (2013b). Exploiting similarities among languages for machine translation. Technical report, arXiv:1309.4168. 459
- Minka, T. (2005). Divergence measures and message passing. *Microsoft Research Cambridge UK Tech Rep MSRTR2005173*, **72**(TR-2005-173). 533
- Minsky, M. L. and Papert, S. A. (1969). *Perceptrons*. MIT Press, Cambridge. 14
- Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*. 599
- Mishkin, D. and Matas, J. (2015). All you need is a good init. *arXiv preprint arXiv:1511.06422*. 259
- Misra, J. and Saha, I. (2010). Artificial neural networks in hardware: A survey of two decades of progress. *Neurocomputing*, **74**(1), 239–255. 384
- Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, New York. 87
- Miyato, T., Maeda, S., Koyama, M., Nakae, K., and Ishii, S. (2015). Distributional smoothing with virtual adversarial training. In *ICLR*. Preprint: arXiv:1507.00677. 231
- Mnih, A. and Gregor, K. (2014). Neural variational inference and learning in belief networks. In *ICML'2014*. 590, 591, 592

- Mnih, A. and Hinton, G. E. (2007). Three new graphical models for statistical language modelling. In Z. Ghahramani, editor, *Proceedings of the Twenty-fourth International Conference on Machine Learning (ICML'07)*, pages 641–648. ACM. 396
- Mnih, A. and Hinton, G. E. (2009). A scalable hierarchical distributed language model. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21 (NIPS'08)*, pages 1081–1088. 397
- Mnih, A. and Kavukcuoglu, K. (2013). Learning word embeddings efficiently with noise-contrastive estimation. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2265–2273. Curran Associates, Inc. 401, 530
- Mnih, A. and Teh, Y. W. (2012). A fast and simple algorithm for training neural probabilistic language models. In *ICML'2012*, pages 1751–1758. 401
- Mnih, V. and Hinton, G. (2010). Learning to detect roads in high-resolution aerial images. In *Proceedings of the 11th European Conference on Computer Vision (ECCV)*. 90
- Mnih, V., Larochelle, H., and Hinton, G. (2011). Conditional restricted Boltzmann machines for structure output prediction. In *Proc. Conf. on Uncertainty in Artificial Intelligence (UAI)*. 585
- Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., and Wierstra, D. (2013). Playing Atari with deep reinforcement learning. Technical report, arXiv:1312.5602. 93
- Mnih, V., Heess, N., Graves, A., and Kavukcuoglu, K. (2014). Recurrent models of visual attention. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *NIPS'2014*, pages 2204–2212. 591
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, **518**, 529–533. 23
- Mobahi, H. and Fisher, III, J. W. (2015). A theoretical analysis of optimization by Gaussian continuation. In *AAAI'2015*. 279
- Mobahi, H., Collobert, R., and Weston, J. (2009). Deep learning from temporal coherence in video. In L. Bottou and M. Littman, editors, *Proceedings of the 26th International Conference on Machine Learning*, pages 737–744, Montreal. Omnipress. 421
- Mohamed, A., Dahl, G., and Hinton, G. (2009). Deep belief networks for phone recognition. 391

- Mohamed, A., Sainath, T. N., Dahl, G., Ramabhadran, B., Hinton, G. E., and Picheny, M. A. (2011). Deep belief networks using discriminative features for phone recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*, pages 5060–5063. IEEE. 391
- Mohamed, A., Dahl, G., and Hinton, G. (2012a). Acoustic modeling using deep belief networks. *IEEE Trans. on Audio, Speech and Language Processing*, **20**(1), 14–22. 391
- Mohamed, A., Hinton, G., and Penn, G. (2012b). Understanding how deep belief networks perform acoustic modelling. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 4273–4276. IEEE. 391
- Moller, M. (1993). *Efficient Training of Feed-Forward Neural Networks*. Ph.D. thesis, Aarhus University, Aarhus, Denmark. 270
- Montavon, G. and Muller, K.-R. (2012). Deep Boltzmann machines and the centering trick. In G. Montavon, G. Orr, and K.-R. Müller, editors, *Neural Networks: Tricks of the Trade*, volume 7700 of *Lecture Notes in Computer Science*, pages 621–637. Preprint: <http://arxiv.org/abs/1203.3783>. 575
- Montúfar, G. (2014). Universal approximation depth and errors of narrow belief networks with discrete units. *Neural Computation*, **26**. 472
- Montúfar, G. and Ay, N. (2011). Refinements of universal approximation results for deep belief networks and restricted Boltzmann machines. *Neural Computation*, **23**(5), 1306–1319. 472
- Montufar, G. F., Pascanu, R., Cho, K., and Bengio, Y. (2014). On the number of linear regions of deep neural networks. In *NIPS’2014*. 17, 172, 173
- Mor-Yosef, S., Samueloff, A., Modan, B., Navot, D., and Schenker, J. G. (1990). Ranking the risk factors for cesarean: logistic regression analysis of a nationwide study. *Obstet Gynecol*, **75**(6), 944–7. 2
- Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In *AISTATS’2005*. 397, 399
- Mozer, M. C. (1992). The induction of multiscale temporal structure. In J. M. S. Hanson and R. Lippmann, editors, *Advances in Neural Information Processing Systems 4 (NIPS’91)*, pages 275–282, San Mateo, CA. Morgan Kaufmann. 348
- Murphy, K. P. (2012). *Machine Learning: a Probabilistic Perspective*. MIT Press, Cambridge, MA, USA. 56, 87, 126

- Murray, B. U. I. and Larochelle, H. (2014). A deep and tractable density estimator. In *ICML'2014*. 164, 606
- Nair, V. and Hinton, G. (2010a). Rectified linear units improve restricted Boltzmann machines. In *ICML'2010*. 150, 170
- Nair, V. and Hinton, G. E. (2009). 3d object recognition with deep belief nets. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1339–1347. Curran Associates, Inc. 586
- Nair, V. and Hinton, G. E. (2010b). Rectified linear units improve restricted Boltzmann machines. In L. Bottou and M. Littman, editors, *Proceedings of the Twenty-seventh International Conference on Machine Learning (ICML-10)*, pages 807–814. ACM. 14
- Narayanan, H. and Mitter, S. (2010). Sample complexity of testing the manifold hypothesis. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1786–1794. 141
- Naumann, U. (2008). Optimal Jacobian accumulation is NP-complete. *Mathematical Programming*, **112**(2), 427–441. 191
- Navigli, R. and Velardi, P. (2005). Structural semantic interconnections: a knowledge-based approach to word sense disambiguation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **27**(7), 1075--1086. 412
- Neal, R. and Hinton, G. (1999). A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*. MIT Press, Cambridge, MA. 541
- Neal, R. M. (1990). Learning stochastic feedforward networks. Technical report. 591
- Neal, R. M. (1993). Probabilistic inference using Markov chain Monte-Carlo methods. Technical Report CRG-TR-93-1, Dept. of Computer Science, University of Toronto. 581
- Neal, R. M. (1994). Sampling from multimodal distributions using tempered transitions. Technical Report 9421, Dept. of Statistics, University of Toronto. 514
- Neal, R. M. (1996). *Bayesian Learning for Neural Networks*. Lecture Notes in Statistics. Springer. 228
- Neal, R. M. (2001). Annealed importance sampling. *Statistics and Computing*, **11**(2), 125–139. 533, 535, 536

- Neal, R. M. (2005). Estimating ratios of normalizing constants using linked importance sampling. 536, 537
- Nesterov, Y. (1983). A method of solving a convex programming problem with convergence rate  $O(1/k^2)$ . *Soviet Mathematics Doklady*, **27**, 372–376. 256
- Nesterov, Y. (2004). *Introductory lectures on convex optimization : a basic course*. Applied optimization. Kluwer Academic Publ., Boston, Dordrecht, London. 256
- Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. Deep Learning and Unsupervised Feature Learning Workshop, NIPS. 18
- Ney, H. and Kneser, R. (1993). Improved clustering techniques for class-based statistical language modelling. In *European Conference on Speech Communication and Technology (Eurospeech)*, pages 973–976, Berlin. 394
- Ng, A. (2015). Advice for applying machine learning. <https://see.stanford.edu/materials/aimlcs229/ML-advice.pdf>. 359
- Niesler, T. R., Whittaker, E. W. D., and Woodland, P. C. (1998). Comparison of part-of-speech and automatically derived category-based language models for speech recognition. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 177–180. 394
- Ning, F., Delhomme, D., LeCun, Y., Piano, F., Bottou, L., and Barbano, P. E. (2005). Toward automatic phenotyping of developing embryos from videos. *Image Processing, IEEE Transactions on*, **14**(9), 1360–1371. 306
- Nocedal, J. and Wright, S. (2006). *Numerical Optimization*. Springer. 82, 85
- Norouzi, M. and Fleet, D. J. (2011). Minimal loss hashing for compact binary codes. In *ICML'2011*. 448
- Nowlan, S. J. (1990). Competing experts: An experimental investigation of associative mixture models. Technical Report CRG-TR-90-5, University of Toronto. 383
- Nowlan, S. J. and Hinton, G. E. (1992). Adaptive soft weight tying using Gaussian mixtures. In J. M. S. Hanson and R. Lippmann, editors, *Advances in Neural Information Processing Systems 4 (NIPS'91)*, pages 993–1000, San Mateo, CA. Morgan Kaufmann. 122
- Olshausen, B. and Field, D. J. (2005). How close are we to understanding V1? *Neural Computation*, **17**, 1665–1699. 14

- Olshausen, B. A. and Field, D. J. (1996). Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, **381**, 607–609. 128, 219, 316, 423
- Olshausen, B. A., Anderson, C. H., and Van Essen, D. C. (1993). A neurobiological model of visual attention and invariant pattern recognition based on dynamic routing of information. *J. Neurosci.*, **13**(11), 4700–4719. 383
- Opper, M. and Archambeau, C. (2009). The variational Gaussian approximation revisited. *Neural computation*, **21**(3), 786–792. 588
- Quab, M., Bottou, L., Laptev, I., and Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1717–1724. IEEE. 456
- Osindero, S. and Hinton, G. E. (2008). Modeling image patches with a directed hierarchy of Markov random fields. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20 (NIPS'07)*, pages 1121–1128, Cambridge, MA. MIT Press. 539
- Ovid and Martin, C. (2004). *Metamorphoses*. W.W. Norton. 1
- Paccanaro, A. and Hinton, G. E. (2000). Extracting distributed representations of concepts and relations from positive and negative propositions. In *International Joint Conference on Neural Networks (IJCNN)*, Como, Italy. IEEE, New York. 411, 412
- Paine, T. L., Khorrami, P., Han, W., and Huang, T. S. (2014). An analysis of unsupervised pre-training in light of recent advances. *arXiv preprint arXiv:1412.6597*. 454
- Palatucci, M., Pomerleau, D., Hinton, G. E., and Mitchell, T. M. (2009). Zero-shot learning with semantic output codes. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1410–1418. Curran Associates, Inc. 459
- Parker, D. B. (1985). Learning-logic. Technical Report TR-47, Center for Comp. Research in Economics and Management Sci., MIT. 194
- Pascanu, R., Mikolov, T., and Bengio, Y. (2013a). On the difficulty of training recurrent neural networks. In *ICML'2013*. 247, 343, 348, 353, 354, 355
- Pascanu, R., Mikolov, T., and Bengio, Y. (2013b). On the difficulty of training recurrent neural networks. In *ICM ( 1c)*. 345
- Pascanu, R., Gulcehre, C., Cho, K., and Bengio, Y. (2014a). How to construct deep recurrent neural networks. In *ICLR*. 17, 228, 340, 341, 349, 392

- Pascanu, R., Montufar, G., and Bengio, Y. (2014b). On the number of inference regions of deep feed forward networks with piece-wise linear activations. In *ICL* ( 1). 469
- Pati, Y., Rezaiifar, R., and Krishnaprasad, P. (1993). Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of the 27 th Annual Asilomar Conference on Signals, Systems, and Computers*, pages 40–44. 220
- Pearl, J. (1985). Bayesian networks: A model of self-activated memory for evidential reasoning. In *Proceedings of the 7th Conference of the Cognitive Science Society, University of California, Irvine*, pages 329–334. 480
- Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann. 48
- Perron, O. (1907). Zur theorie der matrices. *Mathematische Annalen*, **64**(2), 248–263. 508
- Petersen, K. B. and Pedersen, M. S. (2006). The matrix cookbook. Version 20051003. 27
- Peterson, G. B. (2004). A day of great illumination: B. F. Skinner’s discovery of shaping. *Journal of the Experimental Analysis of Behavior*, **82**(3), 317–328. 279
- Pham, D.-T., Garat, P., and Jutten, C. (1992). Separation of a mixture of independent sources through a maximum likelihood approach. In *EUSIPCO*, pages 771–774. 419
- Pham, P.-H., Jelaca, D., Farabet, C., Martini, B., LeCun, Y., and Culurciello, E. (2012). Neu-Flow: dataflow vision processing system-on-a-chip. In *Circuits and Systems (MWSCAS), 2012 IEEE 55th International Midwest Symposium on*, pages 1044–1047. IEEE. 384
- Pinheiro, P. H. O. and Collobert, R. (2014). Recurrent convolutional neural networks for scene labeling. In *ICML’2014*. 306
- Pinheiro, P. H. O. and Collobert, R. (2015). From image-level to pixel-level labeling with convolutional networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 306
- Pinto, N., Cox, D. D., and DiCarlo, J. J. (2008). Why is real-world visual object recognition hard? *PLoS Comput Biol*, **4**. 388
- Pinto, N., Stone, Z., Zickler, T., and Cox, D. (2011). Scaling up biologically-inspired computer vision: A case study in unconstrained face recognition on facebook. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, pages 35–42. IEEE. 310

- Pollack, J. B. (1990). Recursive distributed representations. *Artificial Intelligence*, **46**(1), 77–105. 341
- Polyak, B. and Juditsky, A. (1992). Acceleration of stochastic approximation by averaging. *SIAM J. Control and Optimization*, **30**(4), 838–855. 274
- Polyak, B. T. (1964). Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, **4**(5), 1–17. 253
- Poole, B., Sohl-Dickstein, J., and Ganguli, S. (2014). Analyzing noise in autoencoders and deep networks. *CoRR*, **abs/1406.1831**. 207
- Poon, H. and Domingos, P. (2011). Sum-product networks for deep learning. In *Learning Workshop*, Fort Lauderdale, FL. 472
- Presley, R. K. and Haggard, R. L. (1994). A fixed point implementation of the backpropagation learning algorithm. In *Southeastcon'94. Creative Technology Transfer-A Global Affair., Proceedings of the 1994 IEEE*, pages 136–138. IEEE. 384
- Price, R. (1958). A useful theorem for nonlinear devices having Gaussian inputs. *IEEE Transactions on Information Theory*, **4**(2), 69–72. 588
- Quiroga, R. Q., Reddy, L., Kreiman, G., Koch, C., and Fried, I. (2005). Invariant visual representation by single neurons in the human brain. *Nature*, **435**(7045), 1102–1107. 312
- Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*. 470, 471, 599
- Raiko, T., Yao, L., Cho, K., and Bengio, Y. (2014). Iterative neural autoregressive distribution estimator (NADE-k). Technical report, arXiv:1406.1485. 576, 605
- Raina, R., Madhavan, A., and Ng, A. Y. (2009a). Large-scale deep unsupervised learning using graphics processors. In L. Bottou and M. Littman, editors, *Proceedings of the Twenty-sixth International Conference on Machine Learning (ICML'09)*, pages 873–880, New York, NY, USA. ACM. 21
- Raina, R., Madhavan, A., and Ng, A. Y. (2009b). Large-scale deep unsupervised learning using graphics processors. In *ICML'2009*. 379
- Ramsey, F. P. (1926). Truth and probability. In R. B. Braithwaite, editor, *The Foundations of Mathematics and other Logical Essays*, chapter 7, pages 156–198. McMaster University Archive for the History of Economic Thought. 49

- Ranzato, M. and Hinton, G. H. (2010). Modeling pixel means and covariances using factorized third-order Boltzmann machines. In *CVPR'2010*, pages 2551–2558. 581
- Ranzato, M., Poultney, C., Chopra, S., and LeCun, Y. (2007a). Efficient learning of sparse representations with an energy-based model. In *NIPS'2006*. 13, 433, 451, 452
- Ranzato, M., Poultney, C., Chopra, S., and LeCun, Y. (2007b). Efficient learning of sparse representations with an energy-based model. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19 (NIPS'06)*, pages 1137–1144. MIT Press. 17
- Ranzato, M., Huang, F., Boureau, Y., and LeCun, Y. (2007c). Unsupervised learning of invariant feature hierarchies with applications to object recognition. In *CVPR'07*. 310
- Ranzato, M., Boureau, Y., and LeCun, Y. (2008). Sparse feature learning for deep belief networks. In *NIPS'2007*. 433
- Ranzato, M., Krizhevsky, A., and Hinton, G. E. (2010a). Factored 3-way restricted Boltzmann machines for modeling natural images. In *Proceedings of AISTATS 2010*. 579, 580
- Ranzato, M., Mnih, V., and Hinton, G. (2010b). Generating more realistic images using gated MRFs. In *NIPS'2010*. 581
- Rao, C. (1945). Information and the accuracy attainable in the estimation of statistical parameters. *Bulletin of the Calcutta Mathematical Society*, **37**, 81–89. 118, 252
- Rasmus, A., Valpola, H., Honkala, M., Berglund, M., and Raiko, T. (2015). Semi-supervised learning with ladder network. *arXiv preprint arXiv:1507.02672*. 363, 453
- Recht, B., Re, C., Wright, S., and Niu, F. (2011). Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *NIPS'2011*. 380
- Reichert, D. P., Seriès, P., and Storkey, A. J. (2011). Neuronal adaptation for sampling-based probabilistic inference in perceptual bistability. In *Advances in Neural Information Processing Systems*, pages 2357–2365. 569
- Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *ICML'2014*. Preprint: arXiv:1401.4082. 558, 588, 594
- Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. (2011a). Contractive auto-encoders: Explicit invariance during feature extraction. In *ICML'2011*. 445, 446, 447

- Rifai, S., Mesnil, G., Vincent, P., Muller, X., Bengio, Y., Dauphin, Y., and Glorot, X. (2011b). Higher order contractive auto-encoder. In *ECML PKDD*. 445, 446
- Rifai, S., Dauphin, Y., Vincent, P., Bengio, Y., and Muller, X. (2011c). The manifold tangent classifier. In *NIPS'2011*. 233, 446
- Rifai, S., Dauphin, Y., Vincent, P., Bengio, Y., and Muller, X. (2011d). The manifold tangent classifier. In *NIPS'2011*. Student paper award. 233
- Rifai, S., Bengio, Y., Dauphin, Y., and Vincent, P. (2012). A generative process for sampling contractive auto-encoders. In *ICML'2012*. 607
- Ringach, D. and Shapley, R. (2004). Reverse correlation in neurophysiology. *Cognitive Science*, **28**(2), 147–166. 314
- Roberts, S. and Everson, R. (2001). *Independent component analysis: principles and practice*. Cambridge University Press. 420
- Robinson, A. J. and Fallside, F. (1991). A recurrent error propagation network speech recognition system. *Computer Speech and Language*, **5**(3), 259–274. 21, 390
- Rockafellar, R. T. (1997). Convex analysis. princeton landmarks in mathematics. 82
- Romero, A., Ballas, N., Ebrahimi Kahou, S., Chassang, A., Gatta, C., and Bengio, Y. (2015). Fitnets: Hints for thin deep nets. In *ICLR'2015, arXiv:1412.6550*. 277
- Rosen, J. B. (1960). The gradient projection method for nonlinear programming. part i. linear constraints. *Journal of the Society for Industrial and Applied Mathematics*, **8**(1), pp. 181–217. 83
- Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, **65**, 386–408. 13, 21
- Rosenblatt, F. (1962). *Principles of Neurodynamics*. Spartan, New York. 21
- Rosenblatt, M. (1956). Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, **27**(3), 832–837. 13
- Roweis, S. and Saul, L. K. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, **290**(5500). 141, 443
- Roweis, S., Saul, L., and Hinton, G. (2002). Global coordination of local linear models. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14 (NIPS'01)*, Cambridge, MA. MIT Press. 417

- Rubin, D. B. *et al.* (1984). Bayesianly justifiable and relevant frequency calculations for the applied statistician. *The Annals of Statistics*, **12**(4), 1151–1172. 611
- Rumelhart, D., Hinton, G., and Williams, R. (1986a). Learning representations by back-propagating errors. *Nature*, **323**, 533–536. 13, 194, 406, 410
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986b). Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, volume 1, chapter 8, pages 318–362. MIT Press, Cambridge. 18, 21, 194
- Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986c). Learning representations by back-propagating errors. *Nature*, **323**, 533–536. 16, 175, 319
- Rumelhart, D. E., McClelland, J. L., and the PDP Research Group (1986d). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press, Cambridge. 15, 22
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2014a). ImageNet Large Scale Visual Recognition Challenge. 18
- Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., *et al.* (2014b). Imagenet large scale visual recognition challenge. *arXiv preprint arXiv:1409.0575*. 23
- Russel, S. J. and Norvig, P. (2003). *Artificial Intelligence: a Modern Approach*. Prentice Hall. 77
- Rust, N., Schwartz, O., Movshon, J. A., and Simoncelli, E. (2005). Spatiotemporal elements of macaque V1 receptive fields. *Neuron*, **46**(6), 945–956. 313
- Sainath, T., Mohamed, A., Kingsbury, B., and Ramabhadran, B. (2013). Deep convolutional neural networks for LVCSR. In *ICASSP 2013*. 391
- Salakhutdinov, R. (2010). Learning in Markov random fields using tempered transitions. In Y. Bengio, D. Schuurmans, C. Williams, J. Lafferty, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22 (NIPS'09)*. 514
- Salakhutdinov, R. and Hinton, G. (2009a). Deep Boltzmann machines. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 5, pages 448–455. 20, 21, 452, 566, 569, 572, 574
- Salakhutdinov, R. and Hinton, G. (2009b). Semantic hashing. In *International Journal of Approximate Reasoning*. 448

- Salakhutdinov, R. and Hinton, G. E. (2007a). Learning a nonlinear embedding by preserving class neighbourhood structure. In *Proceedings of AISTATS-2007*. 450
- Salakhutdinov, R. and Hinton, G. E. (2007b). Semantic hashing. In *SIGIR'2007*. 448
- Salakhutdinov, R. and Hinton, G. E. (2008). Using deep belief nets to learn covariance kernels for Gaussian processes. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20 (NIPS'07)*, pages 1249–1256, Cambridge, MA. MIT Press. 210
- Salakhutdinov, R. and Larochelle, H. (2010). Efficient learning of deep Boltzmann machines. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010), JMLR W&CP*, volume 9, pages 693–700. 557
- Salakhutdinov, R. and Mnih, A. (2008). Probabilistic matrix factorization. In *NIPS'2008*. 408
- Salakhutdinov, R. and Murray, I. (2008). On the quantitative analysis of deep belief networks. In W. W. Cohen, A. McCallum, and S. T. Roweis, editors, *Proceedings of the Twenty-fifth International Conference on Machine Learning (ICML'08)*, volume 25, pages 872–879. ACM. 536, 566
- Salakhutdinov, R., Mnih, A., and Hinton, G. (2007). Restricted Boltzmann machines for collaborative filtering. In *ICML*. 408
- Sanger, T. D. (1994). Neural network learning control of robot manipulators using gradually increasing task difficulty. *IEEE Transactions on Robotics and Automation*, **10**(3). 279
- Saul, L. K. and Jordan, M. I. (1996). Exploiting tractable substructures in intractable networks. In D. Touretzky, M. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems 8 (NIPS'95)*. MIT Press, Cambridge, MA. 544
- Saul, L. K., Jaakkola, T., and Jordan, M. I. (1996). Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research*, **4**, 61–76. 21, 592
- Savich, A. W., Moussa, M., and Areibi, S. (2007). The impact of arithmetic representation on implementing mlp-bp on fpgas: A study. *Neural Networks, IEEE Transactions on*, **18**(1), 240–252. 384
- Saxe, A. M., Koh, P. W., Chen, Z., Bhand, M., Suresh, B., and Ng, A. (2011). On random weights and unsupervised feature learning. In *Proc. ICML'2011*. ACM. 310
- Saxe, A. M., McClelland, J. L., and Ganguli, S. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *ICLR*. 244, 245, 258

- Schaul, T., Antonoglou, I., and Silver, D. (2014). Unit tests for stochastic optimization. In *International Conference on Learning Representations*. 263
- Schmidhuber, J. (1992). Learning complex, extended sequences using the principle of history compression. *Neural Computation*, **4**(2), 234–242. 340
- Schmidhuber, J. (1996). Sequential neural text compression. *IEEE Transactions on Neural Networks*, **7**(1), 142–146. 406
- Schmidhuber, J. (2012). Self-delimiting neural networks. *arXiv preprint arXiv:1210.0118*. 333
- Schölkopf, B. and Smola, A. J. (2002). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT press. 601
- Schölkopf, B., Burges, C. J. C., and Smola, A. J. (1998a). *Advances in kernel methods: support vector learning*. MIT Press, Cambridge, MA. 141
- Schölkopf, B., Smola, A., and Müller, K.-R. (1998b). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, **10**, 1299–1319. 443
- Schölkopf, B., Burges, C. J. C., and Smola, A. J. (1999). *Advances in Kernel Methods — Support Vector Learning*. MIT Press, Cambridge, MA. 16, 124
- Schölkopf, B., Janzing, D., Peters, J., Sgouritsa, E., Zhang, K., and Mooij, J. (2012). On causal and anticausal learning. In *ICML'2012*, pages 1255–1262. 465
- Schuster, M. (1999). On supervised learning from sequential data with applications for speech recognition. 164
- Schuster, M. and Paliwal, K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, **45**(11), 2673–2681. 336
- Schwenk, H. (2007). Continuous space language models. *Computer speech and language*, **21**, 492–518. 396
- Schwenk, H. (2010). Continuous space language models for statistical machine translation. *The Prague Bulletin of Mathematical Linguistics*, **93**, 137–146. 402
- Schwenk, H. (2014). Cleaned subset of WMT '14 dataset. 18
- Schwenk, H. and Bengio, Y. (1998). Training methods for adaptive boosting of neural networks. In M. Jordan, M. Kearns, and S. Solla, editors, *Advances in Neural Information Processing Systems 10 (NIPS'97)*, pages 647–653. MIT Press. 222

- Schwenk, H. and Gauvain, J.-L. (2002). Connectionist language modeling for large vocabulary continuous speech recognition. In *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 765–768, Orlando, Florida. 396
- Schwenk, H., Costa-jussà, M. R., and Fonollosa, J. A. R. (2006). Continuous space language models for the IWSLT 2006 task. In *International Workshop on Spoken Language Translation*, pages 166–173. 402
- Seide, F., Li, G., and Yu, D. (2011). Conversational speech transcription using context-dependent deep neural networks. In *Interspeech 2011*, pages 437–440. 22
- Sejnowski, T. (1987). Higher-order Boltzmann machines. In *AIP Conference Proceedings 151 on Neural Networks for Computing*, pages 398–403. American Institute of Physics Inc. 586
- Series, P., Reichert, D. P., and Storkey, A. J. (2010). Hallucinations in Charles Bonnet syndrome induced by homeostasis: a deep Boltzmann machine model. In *Advances in Neural Information Processing Systems*, pages 2020–2028. 569
- Sermanet, P., Chintala, S., and LeCun, Y. (2012). Convolutional neural networks applied to house numbers digit classification. In *International Conference on Pattern Recognition (ICPR 2012)*. 388
- Sermanet, P., Kavukcuoglu, K., Chintala, S., and LeCun, Y. (2013). Pedestrian detection with unsupervised multi-stage feature learning. In *Proc. International Conference on Computer Vision and Pattern Recognition (CVPR'13)*. IEEE. 22, 174
- Shilov, G. (1977). *Linear Algebra*. Dover Books on Mathematics Series. Dover Publications. 27
- Siegelmann, H. (1995). Computation beyond the Turing limit. *Science*, **268**(5210), 545–548. 324
- Siegelmann, H. and Sontag, E. (1991). Turing computability with neural nets. *Applied Mathematics Letters*, **4**(6), 77–80. 324
- Siegelmann, H. T. and Sontag, E. D. (1995). On the computational power of neural nets. *Journal of Computer and Systems Sciences*, **50**(1), 132–150. 324, 325, 345
- Sietsma, J. and Dow, R. (1991). Creating artificial neural networks that generalize. *Neural Networks*, **4**(1), 67–79. 207
- Simard, D., Steinkraus, P. Y., and Platt, J. C. (2003). Best practices for convolutional neural networks. In *ICDAR'2003*. 317

- Simard, P. and Graf, H. P. (1994). Backpropagation without multiplication. In *Advances in Neural Information Processing Systems*, pages 232–239. 384
- Simard, P., Victorri, B., LeCun, Y., and Denker, J. (1992). Tangent prop - A formalism for specifying selected invariances in an adaptive network. In *NIPS'1991*. 232, 233, 301
- Simard, P. Y., LeCun, Y., and Denker, J. (1993). Efficient pattern recognition using a new transformation distance. In *NIPS'92*. 232
- Simard, P. Y., LeCun, Y. A., Denker, J. S., and Victorri, B. (1998). Transformation invariance in pattern recognition — tangent distance and tangent propagation. *Lecture Notes in Computer Science*, **1524**. 232
- Simons, D. J. and Levin, D. T. (1998). Failure to detect changes to people during a real-world interaction. *Psychonomic Bulletin & Review*, **5**(4), 644–649. 463
- Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *ICLR*. 275
- Sjöberg, J. and Ljung, L. (1995). Overtraining, regularization and searching for a minimum, with application to neural networks. *International Journal of Control*, **62**(6), 1391–1407. 215
- Skinner, B. F. (1958). Reinforcement today. *American Psychologist*, **13**, 94–99. 279
- Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, volume 1, chapter 6, pages 194–281. MIT Press, Cambridge. 486, 499, 561
- Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. In *NIPS'2012*. 371
- Socher, R., Huang, E. H., Pennington, J., Ng, A. Y., and Manning, C. D. (2011a). Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS'2011*. 341, 342
- Socher, R., Manning, C., and Ng, A. Y. (2011b). Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the Twenty-Eighth International Conference on Machine Learning (ICML'2011)*. 341
- Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., and Manning, C. D. (2011c). Semi-supervised recursive autoencoders for predicting sentiment distributions. In *EMNLP'2011*. 341, 342

- Socher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. (2013a). Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP'2013*. 341, 342
- Socher, R., Ganjoo, M., Manning, C. D., and Ng, A. Y. (2013b). Zero-shot learning through cross-modal transfer. In *27th Annual Conference on Neural Information Processing Systems (NIPS 2013)*. 459
- Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. (2015). Deep unsupervised learning using nonequilibrium thermodynamics. 610, 611
- Sohn, K., Zhou, G., and Lee, H. (2013). Learning and selecting features jointly with point-wise gated Boltzmann machines. In *ICML'2013*. 586
- Solomonoff, R. J. (1989). A system for incremental learning based on algorithmic probability. 279
- Sontag, E. D. (1998). VC dimension of neural networks. *NATO ASI Series F Computer and Systems Sciences*, **168**, 69–96. 467, 470
- Sontag, E. D. and Sussman, H. J. (1989). Backpropagation can give rise to spurious local minima even for networks without hidden layers. *Complex Systems*, **3**, 91–106. 243
- Sparkes, B. (1996). *The Red and the Black: Studies in Greek Pottery*. Routledge. 1
- Spitkovsky, V. I., Alshawi, H., and Jurafsky, D. (2010). From baby steps to leapfrog: how “less is more” in unsupervised dependency parsing. In *HLT'10*. 279
- Squire, W. and Trapp, G. (1998). Using complex variables to estimate derivatives of real functions. *SIAM Rev.*, **40**(1), 110--112. 373
- Srebro, N. and Shraibman, A. (2005). Rank, trace-norm and max-norm. In *Proceedings of the 18th Annual Conference on Learning Theory*, pages 545–560. Springer-Verlag. 206
- Srivastava, N. (2013). *Improving Neural Networks With Dropout*. Master’s thesis, U. Toronto. 456
- Srivastava, N. and Salakhutdinov, R. (2012). Multimodal learning with deep Boltzmann machines. In *NIPS'2012*. 460
- Srivastava, N., Salakhutdinov, R. R., and Hinton, G. E. (2013). Modeling documents with deep Boltzmann machines. *arXiv preprint arXiv:1309.6865*. 566

- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, **15**, 1929–1958. 222, 227, 228, 229, 574
- Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Highway networks. *arXiv:1505.00387*. 278
- Steinkrau, D., Simard, P. Y., and Buck, I. (2005). Using GPUs for machine learning algorithms. *2013 12th International Conference on Document Analysis and Recognition*, **0**, 1115–1119. 379
- Stoyanov, V., Ropson, A., and Eisner, J. (2011). Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 15 of *JMLR Workshop and Conference Proceedings*, pages 725–733, Fort Lauderdale. Supplementary material (4 pages) also available. 576, 596
- Sukhbaatar, S., Szlam, A., Weston, J., and Fergus, R. (2015). Weakly supervised memory networks. *arXiv preprint arXiv:1503.08895*. 356
- Supancic, J. and Ramanan, D. (2013). Self-paced learning for long-term tracking. In *CVPR'2013*. 280
- Sussillo, D. (2014). Random walks: Training very deep nonlinear feed-forward networks with smart initialization. *CoRR*, **abs/1412.6558**. 248, 259, 260, 344
- Sutskever, I. (2012). *Training Recurrent Neural Networks*. Ph.D. thesis, Department of computer science, University of Toronto. 347, 353
- Sutskever, I. and Hinton, G. E. (2008). Deep narrow sigmoid belief networks are universal approximators. *Neural Computation*, **20**(11), 2629–2636. 592
- Sutskever, I. and Tieleman, T. (2010). On the Convergence Properties of Contrastive Divergence. In *AISTATS'2010*. 521
- Sutskever, I., Hinton, G., and Taylor, G. (2009). The recurrent temporal restricted Boltzmann machine. In *NIPS'2008*. 585
- Sutskever, I., Martens, J., and Hinton, G. E. (2011). Generating text with recurrent neural networks. In *ICML'2011*, pages 1017–1024. 406
- Sutskever, I., Martens, J., Dahl, G., and Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In *ICML*. 256, 347, 353

- Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *NIPS'2014, arXiv:1409.3215.* 23, 89, 338, 349, 351, 403
- Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction.* MIT Press. 93
- Sutton, R. S., Mcallester, D., Singh, S., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *NIPS'1999*, pages 1057--1063. MIT Press. 590
- Swersky, K., Ranzato, M., Buchman, D., Marlin, B., and de Freitas, N. (2011). On autoencoders and score matching for energy based models. In *ICML'2011.* ACM. 438
- Swersky, K., Snoek, J., and Adams, R. P. (2014). Freeze-thaw Bayesian optimization. *arXiv preprint arXiv:1406.3896.* 371
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014a). Going deeper with convolutions. Technical report, arXiv:1409.4842. 20, 21, 174, 222, 231, 278, 295
- Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. (2014b). Intriguing properties of neural networks. *ICLR, abs/1312.6199.* 230, 233
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2015). Rethinking the Inception Architecture for Computer Vision. *ArXiv e-prints.* 209, 275
- Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2014). DeepFace: Closing the gap to human-level performance in face verification. In *CVPR'2014.* 88
- Tandy, D. W. (1997). *Works and Days: A Translation and Commentary for the Social Sciences.* University of California Press. 1
- Tang, Y. and Eliasmith, C. (2010). Deep networks for robust visual recognition. In *Proceedings of the 27th International Conference on Machine Learning, June 21-24, 2010, Haifa, Israel.* 207
- Tang, Y., Salakhutdinov, R., and Hinton, G. (2012). Deep mixtures of factor analysers. *arXiv preprint arXiv:1206.4635.* 417
- Taylor, G. and Hinton, G. (2009). Factored conditional restricted Boltzmann machines for modeling motion style. In L. Bottou and M. Littman, editors, *Proceedings of the Twenty-sixth International Conference on Machine Learning (ICML'09)*, pages 1025–1032, Montreal, Quebec, Canada. ACM. 585

- Taylor, G., Hinton, G. E., and Roweis, S. (2007). Modeling human motion using binary latent variables. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19 (NIPS'06)*, pages 1345–1352. MIT Press, Cambridge, MA. 585
- Teh, Y., Welling, M., Osindero, S., and Hinton, G. E. (2003). Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, **4**, 1235–1260. 419
- Tenenbaum, J., de Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, **290**(5500), 2319–2323. 141, 443, 456
- Theis, L., van den Oord, A., and Bethge, M. (2015). A note on the evaluation of generative models. arXiv:1511.01844. 596, 613
- Thompson, J., Jain, A., LeCun, Y., and Bregler, C. (2014). Joint training of a convolutional network and a graphical model for human pose estimation. In *NIPS'2014*. 306
- Thrun, S. (1995). Learning to play the game of chess. In *NIPS'1994*. 232
- Tibshirani, R. J. (1995). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B*, **58**, 267–288. 204
- Tieleman, T. (2008). Training restricted Boltzmann machines using approximations to the likelihood gradient. In *ICML'2008*, pages 1064–1071. 521
- Tieleman, T. and Hinton, G. (2009). Using fast weights to improve persistent contrastive divergence. In *ICML'2009*. 524
- Tipping, M. E. and Bishop, C. M. (1999). Probabilistic principal components analysis. *Journal of the Royal Statistical Society B*, **61**(3), 611–622. 419
- Torralba, A., Fergus, R., and Weiss, Y. (2008). Small codes and large databases for recognition. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR'08)*, pages 1–8. 448
- Touretzky, D. S. and Minton, G. E. (1985). Symbols among the neurons: Details of a connectionist inference architecture. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'85, pages 238–243, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. 15
- Tu, K. and Honavar, V. (2011). On the utility of curricula in unsupervised learning of probabilistic grammars. In *IJCAI'2011*. 279
- Turaga, S. C., Murray, J. F., Jain, V., Roth, F., Helmstaedter, M., Briggman, K., Denk, W., and Seung, H. S. (2010). Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural Computation*, **22**, 511–538. 306

- Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning. In *Proc. ACL'2010*, pages 384–394. 455
- Töscher, A., Jahrer, M., and Bell, R. M. (2009). The BigChaos solution to the Netflix grand prize. 408
- Uria, B., Murray, I., and Larochelle, H. (2013). Rnade: The real-valued neural autoregressive density-estimator. In *NIPS'2013*. 605, 606
- van den Oord, A., Dieleman, S., and Schrauwen, B. (2013). Deep content-based music recommendation. In *NIPS'2013*. 408
- van der Maaten, L. and Hinton, G. E. (2008). Visualizing data using t-SNE. *J. Machine Learning Res.*, **9**. 406, 443
- Vanhoucke, V., Senior, A., and Mao, M. Z. (2011). Improving the speed of neural networks on CPUs. In *Proc. Deep Learning and Unsupervised Feature Learning NIPS Workshop*. 378, 384
- Vapnik, V. N. (1982). *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, Berlin. 100
- Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer, New York. 100
- Vapnik, V. N. and Chervonenkis, A. Y. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and Its Applications*, **16**, 264–280. 100
- Vincent, P. (2011). A connection between score matching and denoising autoencoders. *Neural Computation*, **23**(7). 438, 439, 440, 608
- Vincent, P. and Bengio, Y. (2003). Manifold Parzen windows. In *NIPS'2002*. MIT Press. 444
- Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008a). Extracting and composing robust features with denoising autoencoders. In *ICML ( 1a)*, pages 1096–1103. 207
- Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008b). Extracting and composing robust features with denoising autoencoders. In *ICML 2008*. 440
- Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *J. Machine Learning Res.*, **11**. 440
- Vincent, P., de Brébisson, A., and Bouthillier, X. (2015). Efficient exact gradient update for training deep networks with very large sparse targets. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1108–1116. Curran Associates, Inc. 396

- Vinyals, O., Kaiser, L., Koo, T., Petrov, S., Sutskever, I., and Hinton, G. (2014a). Grammar as a foreign language. *arXiv preprint arXiv:1412.7449*. 349
- Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2014b). Show and tell: a neural image caption generator. *arXiv* 1411.4555. 349
- Vinyals, O., Fortunato, M., and Jaitly, N. (2015a). Pointer networks. *arXiv preprint arXiv:1506.03134*. 356
- Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015b). Show and tell: a neural image caption generator. In *CVPR'2015*. *arXiv:1411.4555*. 90
- Viola, P. and Jones, M. (2001). Robust real-time object detection. In *International Journal of Computer Vision*. 382
- Visin, F., Kastner, K., Cho, K., Matteucci, M., Courville, A., and Bengio, Y. (2015). ReNet: A recurrent neural network based alternative to convolutional networks. *arXiv preprint arXiv:1505.00393*. 338
- Von Melchner, L., Pallas, S. L., and Sur, M. (2000). Visual behaviour mediated by retinal projections directed to the auditory pathway. *Nature*, **404**(6780), 871–876. 14
- Wager, S., Wang, S., and Liang, P. (2013). Dropout training as adaptive regularization. In *Advances in Neural Information Processing Systems 26*, pages 351–359. 228
- Waibel, A., Hanazawa, T., Hinton, G. E., Shikano, K., and Lang, K. (1989). Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **37**, 328–339. 319, 386, 390
- Wan, L., Zeiler, M., Zhang, S., LeCun, Y., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In *ICML'2013*. 229
- Wang, S. and Manning, C. (2013). Fast dropout training. In *ICML'2013*. 228
- Wang, Z., Zhang, J., Feng, J., and Chen, Z. (2014a). Knowledge graph and text jointly embedding. In *Proc. EMNLP'2014*. 411
- Wang, Z., Zhang, J., Feng, J., and Chen, Z. (2014b). Knowledge graph embedding by translating on hyperplanes. In *Proc. AAAI'2014*. 412
- Warde-Farley, D., Goodfellow, I. J., Courville, A., and Bengio, Y. (2014). An empirical analysis of dropout in piecewise linear networks. In *ICL (1)*. 225, 228, 229
- Wawrynek, J., Asanovic, K., Kingsbury, B., Johnson, D., Beck, J., and Morgan, N. (1996). Spert-II: A vector microprocessor system. *Computer*, **29**(3), 79–86. 384

- Weaver, L. and Tao, N. (2001). The optimal reward baseline for gradient-based reinforcement learning. In *Proc. UAI'2001*, pages 538–545. 590
- Weinberger, K. Q. and Saul, L. K. (2004a). Unsupervised learning of image manifolds by semidefinite programming. In *Proceedings of the Computer Vision and Pattern Recognition Conference (CVPR'04)*, volume 2, pages 988–995, Washington D.C. 141
- Weinberger, K. Q. and Saul, L. K. (2004b). Unsupervised learning of image manifolds by semidefinite programming. In *CVPR'2004*, pages 988–995. 443
- Weiss, Y., Torralba, A., and Fergus, R. (2008). Spectral hashing. In *NIPS*, pages 1753–1760. 448
- Welling, M., Zemel, R. S., and Hinton, G. E. (2002). Self supervised boosting. In *Advances in Neural Information Processing Systems*, pages 665–672. 600
- Welling, M., Hinton, G. E., and Osindero, S. (2003a). Learning sparse topographic representations with products of Student-t distributions. In *NIPS'2002*. 581
- Welling, M., Zemel, R., and Hinton, G. E. (2003b). Self-supervised boosting. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15 (NIPS'02)*, pages 665–672. MIT Press. 531
- Welling, M., Rosen-Zvi, M., and Hinton, G. E. (2005). Exponential family harmoniums with an application to information retrieval. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17 (NIPS'04)*, volume 17, Cambridge, MA. MIT Press. 578
- Werbos, P. J. (1981). Applications of advances in nonlinear sensitivity analysis. In *Proceedings of the 10th IFIP Conference, 31.8 - 4.9, NYC*, pages 762–770. 194
- Weston, J., Bengio, S., and Usunier, N. (2010). Large scale image annotation: learning to rank with joint word-image embeddings. *Machine Learning*, **81**(1), 21–35. 343
- Weston, J., Chopra, S., and Bordes, A. (2014). Memory networks. *arXiv preprint arXiv:1410.3916*. 356, 412
- Widrow, B. and Hoff, M. E. (1960). Adaptive switching circuits. In *1960 IRE WESCON Convention Record*, volume 4, pages 96–104. IRE, New York. 13, 18, 20, 21
- Wikipedia (2015). List of animals by number of neurons — Wikipedia, the free encyclopedia. [Online; accessed 4-March-2015]. 20, 21

- Williams, C. K. I. and Agakov, F. V. (2002). Products of Gaussians and Probabilistic Minor Component Analysis. *Neural Computation*, **14**(5), 1169–1182. 582
- Williams, C. K. I. and Rasmussen, C. E. (1996). Gaussian processes for regression. In D. Touretzky, M. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems 8 (NIPS'95)*, pages 514–520. MIT Press, Cambridge, MA. 124
- Williams, R. J. (1992). Simple statistical gradient-following algorithms connectionist reinforcement learning. *Machine Learning*, **8**, 229–256. 588, 589
- Williams, R. J. and Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, **1**, 270–280. 192
- Wilson, D. R. and Martinez, T. R. (2003). The general inefficiency of batch training for gradient descent learning. *Neural Networks*, **16**(10), 1429–1451. 239
- Wilson, J. R. (1984). Variance reduction techniques for digital simulation. *American Journal of Mathematical and Management Sciences*, **4**(3), 277--312. 589
- Wiskott, L. and Sejnowski, T. J. (2002). Slow feature analysis: Unsupervised learning of invariances. *Neural Computation*, **14**(4), 715–770. 421, 422
- Wolpert, D. and MacReady, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, **1**, 67–82. 250
- Wolpert, D. H. (1996). The lack of a priori distinction between learning algorithms. *Neural Computation*, **8**(7), 1341–1390. 102
- Wu, R., Yan, S., Shan, Y., Dang, Q., and Sun, G. (2015). Deep image: Scaling up image recognition. arXiv:1501.02876. 381
- Wu, Z. (1997). Global continuation for distance geometry problems. *SIAM Journal of Optimization*, **7**, 814–836. 279
- Xiong, H. Y., Barash, Y., and Frey, B. J. (2011). Bayesian prediction of tissue-regulated splicing using RNA sequence and cellular context. *Bioinformatics*, **27**(18), 2554–2562. 228
- Xu, K., Ba, J. L., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *ICML'2015*, arXiv:1502.03044. 90, 349, 591
- Yildiz, I. B., Jaeger, H., and Kiebel, S. J. (2012). Re-visiting the echo state property. *Neural networks*, **35**, 1–9. 346

- Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *NIPS 27*, pages 3320–3328. Curran Associates, Inc. 277, 456
- Younes, L. (1998). On the convergence of Markovian stochastic algorithms with rapidly decreasing ergodicity rates. In *Stochastics and Stochastics Models*, pages 177–228. 521
- Yu, D., Wang, S., and Deng, L. (2010). Sequential labeling using deep-structured conditional random fields. *IEEE Journal of Selected Topics in Signal Processing*. 276
- Zaremba, W. and Sutskever, I. (2014). Learning to execute. arXiv 1410.4615. 280
- Zaremba, W. and Sutskever, I. (2015). Reinforcement learning neural Turing machines. *arXiv:1505.00521*. 358
- Zaslavsky, T. (1975). *Facing Up to Arrangements: Face-Count Formulas for Partitions of Space by Hyperplanes*. Number no. 154 in Memoirs of the American Mathematical Society. American Mathematical Society. 469
- Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *ECCV'14*. 5
- Zeiler, M. D., Ranzato, M., Monga, R., Mao, M., Yang, K., Le, Q., Nguyen, P., Senior, A., Vanhoucke, V., Dean, J., and Hinton, G. E. (2013). On rectified linear units for speech processing. In *ICASSP 2013*. 391
- Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. (2015). Object detectors emerge in deep scene CNNs. ICLR'2015, arXiv:1412.6856. 470
- Zhou, J. and Troyanskaya, O. G. (2014). Deep supervised and convolutional generative stochastic network for protein secondary structure prediction. In *ICML'2014*. 610
- Zhou, Y. and Chellappa, R. (1988). Computation of optical flow using a neural network. In *Neural Networks, 1988., IEEE International Conference on*, pages 71–78. IEEE. 290
- Zöhrer, M. and Pernkopf, F. (2014). General stochastic networks for classification. In *NIPS'2014*. 610

# 术语

- 绝对值整流 absolute value rectification 167, 172, 173
- 准确率 accuracy 91, 360, 372–375
- 声学 acoustic 392
- 激活函数 activation function 147, 245, 257, 258, 260, 271–273, 277, 278
- AdaGrad** AdaGrad 261, 262
- 对抗 adversarial 464
- 对抗样本 adversarial example 230, 231
- 对抗训练 adversarial training 230, 231, 233, 474
- 几乎处处 almost everywhere 64
- 几乎必然 almost sure 114
- 几乎必然收敛 almost sure convergence 114
- 选择性剪接数据集 alternative splicing dataset 456
- 原始采样 Ancestral Sampling 494, 495, 507, 513, 557, 565, 569, 591, 606, 610
- 退火重要采样 annealed importance sampling 533–537, 566, 571, 612
- 专用集成电路 application-specific integrated circuit 384
- 近似贝叶斯计算 approximate Bayesian computation 611
- 近似推断 approximate inference 490, 497, 499, 539–542, 556–558
- 架构 architecture 170
- 人工智能 artificial intelligence 1–4, 6–10, 16, 17, 21, 47, 49, 136, 138, 141, 279, 362, 377, 385, 411, 416, 444, 462, 471, 472, 476, 614
- 人工神经网络 artificial neural network 12, 13, 20, 21, 377
- 渐近无偏 asymptotically unbiased 109

- 异步随机梯度下降 Asynchronous Stochastic Gradient Descent 380
- 异步 asynchronous 240
- 注意力机制 attention mechanism 313, 339, 358, 382, 383, 404, 405, 596
- 属性 attribute 411
- 自编码器 autoencoder xv, 4, 20, 21, 169, 233, 234, 245, 260, 293, 301–303, 322, 373, 420, 425, 426, 428–442, 445–448, 450, 452, 453, 464, 474, 514, 521, 528, 551, 558, 595, 596, 601, 606–608
- 自动微分 automatic differentiation 191
- 自动语音识别 Automatic Speech Recognition 390, 391
- 自回归网络 auto-regressive network 592, 602, 603, 605, 606
- 反向传播 back propagate 425
- 反向传播 back propagation 147, 175, 406, 429, 530, 560, 575–577, 585–588, 592, 594–596, 610
- 回退 back-off 478
- 反向传播 backprop 153, 176, 181, 182, 185, 187, 188, 384, 385
- 通过时间反向传播 back-propagation through time 326–328, 586
- 反向传播 backward propagation 257–259, 271, 326, 328, 329, 345–347, 354, 355, 358
- 词袋 bag of words 401
- Bagging bootstrap aggregating 220–222, 224, 225, 229
- bandit bandit 409
- 批量 batch viii, 237–239, 251–253, 256, 261, 273
- 批标准化 batch normalization 230, 271–273, 362, 455, 456
- 贝叶斯误差 Bayes error 102, 103, 360
- 贝叶斯规则 Bayes' rule 63, 64, 119, 463, 465, 535
- 贝叶斯推断 Bayesian inference 87, 121, 122, 450
- 贝叶斯网络 Bayesian network 480, 483, 496, 566
- 贝叶斯概率 Bayesian probability 49
- 贝叶斯统计 Bayesian statistics 118
- 基准 benchmark 106, 360
- 信念网络 belief network 21, 480, 592, 603
- Bernoulli 分布 Bernoulli distribution 56, 61, 157–159, 369, 435, 548, 570, 573, 593

**基准** baseline 362, 363, 375

**BFGS** BFGS 270

**偏置** bias in affine function 96, 199, 202, 243, 257, 260, 326, 334, 350–354, 371, 396, 408, 546, 547, 559, 564, 565, 567, 571, 572, 575, 579, 580, 582, 585

**偏差** bias in statistics 197, 198, 265, 400

**有偏** biased 240, 248

**有偏重要采样** biased importance sampling 400, 505

**偏差** bias 114

**二元语法** bigram 393, 400

**二元关系** binary relation 410

**二值稀疏编码** binary sparse coding 546–551

**比特** bit 66

**块坐标下降** block coordinate descent 274

**块吉布斯采样** block Gibbs Sampling 500, 510, 563

**玻尔兹曼分布** Boltzmann distribution 485

**玻尔兹曼机** Boltzmann Machine 248, 260, 293, 485, 486, 500, 513, 520, 559–561, 567, 571, 575, 576, 578, 579, 584–586, 596, 607

**Boosting** Boosting 222, 229

**桥式采样** bridge sampling 533, 536, 537

**广播** broadcasting 29

**磨合** Burning-in 509, 518–520, 522, 523, 571, 573

**变分法** calculus of variations 155, 156, 544, 545, 551, 554, 555

**容量** capacity 98, 99, 101, 104, 106, 114, 215, 222, 237, 359, 364–367, 381, 382, 394, 401, 402, 430, 431, 433, 434, 436, 440, 441, 470, 523, 528

**级联** cascade 382, 384

**灾难遗忘** catastrophic forgetting 168

**范畴分布** categorical distribution 56, 369

**因果因子** causal factor 466, 470, 472, 473

**因果模型** causal modeling 53

**中心差分** centered difference 373

- 中心极限定理 central limit theorem 58, 504
- 链式法则 chain rule 53, 76, 525
- 混沌 chaos 258
- 弦 chord 492, 493
- 弦图 chordal graph 493
- 梯度截断 clip gradient 164
- 截断梯度 clipping the gradient 353
- 团 clique 482–486, 491–494, 496, 497, 539, 543, 565
- 团势能 clique potential 482, 484, 485
- 闭式解 closed form solution 206, 420, 422
- 级联 coalesced 379, 383
- 编码 code 429–431, 433–435, 445, 447, 448
- 协同过滤 collaborative filtering 407, 408
- 列 column 28
- 列空间 column space 33
- 共因 common cause 489
- 完全图 complete graph 491
- 复杂细胞 complex cell 312
- 计算图 computational graph 176, 247, 320–322, 328–330, 341, 355, 498, 576, 596, 603
- 计算机视觉 Computer Vision 218, 363, 377, 384–386, 389, 421, 470
- 概念漂移 concept drift 457, 458
- 条件计算 conditional computation 382
- 条件概率 conditional probability 52, 53, 64, 69, 524
- 条件独立的 conditionally independent 53, 418, 481, 487, 488, 492
- 共轭 conjugate 268
- 共轭方向 conjugate directions 267
- 共轭梯度 conjugate gradient 267–270
- 联结主义 connectionism 12, 13, 15, 16, 19, 377, 559
- 一致性 consistency 114
- 约束优化 constrained optimization 82, 83, 85, 220, 485

- 特定环境下的独立 context-specific independences 488
- contextual bandit** contextual bandit 409, 410
- 延拓法 continuation method 278, 279
- 收缩 contractive 346, 445–447
- 收缩自编码器 contractive autoencoder 434, 438, 440, 442, 445, 446, 606, 607
- 对比散度 contrastive divergence 248, 438, 519–523, 527, 529, 564, 565, 571, 574, 575, 580–582, 586, 608
- 凸优化 Convex optimization 82, 241–243, 261, 274
- 卷积 convolution 281, 282, 450, 499
- 卷积玻尔兹曼机 Convolutional Boltzmann Machine 293
- 卷积玻尔兹曼机 convolutional Boltzmann machine 584
- 卷积网络 convolutional net 472
- 卷积网络 convolutional network 20, 21, 144, 175, 242, 246, 281, 282, 285, 287, 288, 290, 293–297, 299, 301, 302, 304, 306–314, 317–319, 337, 338, 360, 362, 363, 375, 379, 391, 395, 402, 403, 408, 456, 469, 470, 583, 584, 594, 601, 604
- 卷积神经网络 convolutional neural network 145, 218, 229, 281, 284, 285, 290, 295, 306
- 坐标上升 coordinate ascent 541, 543, 572
- 坐标下降 coordinate descent 274
- 共父 coparent 539, 547
- 相关系数 correlation 55
- 代价 cost 119, 134, 243–246, 248, 252, 257, 360, 361, 365, 370, 455, 506
- 代价函数 cost function 26, 74, 76, 78, 87, 104, 115, 116, 132–134, 152, 201, 203, 204, 208, 209, 214, 215, 231, 235–237, 242–249, 251, 252, 255, 269, 271, 272, 274, 275, 278, 279, 353, 360, 365, 375, 413, 421, 423, 433, 437, 438, 453, 465, 506, 524, 575, 588, 601
- 协方差 covariance 55, 60, 202, 220, 427
- 协方差矩阵 covariance matrix 55, 58, 60, 418, 427
- 协方差 RBM covariance RBM 580, 581
- 覆盖 coverage 361, 375
- 准则 criterion 74, 210, 251, 254, 256, 262–265, 267, 269, 322, 327, 345, 401, 435, 437–439, 446, 447, 575, 586, 594, 596, 608, 610
- 临界点 critical point 74–77, 79–82, 242–245, 249, 250, 266, 453, 551, 553

- 临界温度 critical temperatures 514
- 互相关函数 cross-correlation 283
- 交叉熵 cross-entropy 68, 116, 153–156, 189–191, 194, 330, 333, 396, 397
- 累积函数 cumulative function 504
- 课程学习 curriculum learning 279, 280, 327
- 维数灾难 curse of dimensionality 135, 136, 138, 394, 395, 468, 473, 603
- 曲率 curvature 78–81, 99, 201, 242, 253, 266
- 控制论 cybernetics 12, 13
- 衰减 damping 551
- 数据生成分布 data generating distribution 97, 236, 240, 241, 251
- 数据生成过程 data generating process 97, 449
- 数据并行 data parallelism 380
- 数据点 data point 92
- 数据集 dataset 87, 92–95, 97, 98, 101, 104, 106, 107, 113–115, 118, 119, 125, 128, 131, 133, 134, 141
- 数据集增强 dataset augmentation 386, 389
- 决策树 decision tree 125, 127, 382–384, 466
- 解码器 decoder 4, 338, 339, 402–404, 417, 420, 421, 423–427, 429–431, 434–436, 439–441, 447, 469, 595
- 分解 decompose 38
- 深度信念网络 deep belief network 17, 21, 310, 452, 472, 520, 536, 538, 562, 564–566, 568, 569, 572, 584, 591, 609
- 深度玻尔兹曼机 Deep Boltzmann Machine xiv, 20, 21, 452, 513, 520, 523, 526, 527, 538, 539, 551, 557, 562, 564, 566–577, 584, 609
- 深度回路 deep circuit 472
- 深度前馈网络 deep feedforward network 145, 147, 391, 417, 428
- 深度生成模型 deep generative model 452
- 深度学习 deep learning 1, 4–7, 10–15, 17, 18, 22–24, 26, 73, 74, 76, 79, 82, 87–89, 92, 93, 100, 105, 125, 128, 132, 133, 135–138, 141, 144, 197, 198, 210–212, 230, 235, 237, 239, 248, 251, 256, 261, 262, 266, 269, 270, 275, 345, 358, 362, 364, 371, 374, 377, 379, 381, 383–386, 390–392, 407, 408, 410, 412, 415, 416, 444, 448, 456, 458, 462, 466, 472, 474–476, 484, 496–499, 501, 506, 507, 510, 516, 518, 521, 526, 538, 539, 542, 543, 555

- 深度模型** deep model 93, 235, 236, 241, 243, 245, 257, 263, 277, 452, 522, 526
- 深度网络** deep network 144, 211, 258, 272, 278, 471
- 信任度** degree of belief 49
- 去噪** denoising 90, 92, 433, 437, 438, 440, 445, 476, 528
- 去噪自编码器** denoising autoencoder xv, 207, 433, 434, 436–440, 442, 445, 454, 457, 588, 606–611
- 去噪得分匹配** denoising score matching 438, 528
- 依赖** dependency 474, 476, 488, 492, 496
- 深度** depth 145
- 导数** derivative 74, 76, 77, 81, 86
- 描述** description 70
- 设计矩阵** design matrix 93–95, 129
- 细致平衡** detailed balance 608
- 探测级** detector stage 290
- 确定性** deterministic 238
- 对角矩阵** diagonal matrix 36
- 微分熵** differential entropy 67, 552
- 微分方程** differential equation 255
- 降维** dimensionality reduction 406, 429, 448
- Dirac delta 函数** Dirac delta function 59
- Dirac 分布** dirac distribution 59, 60, 528, 542, 543, 553, 554
- 有向** directed 69
- 有向图模型** directed graphical model 331, 334, 418, 462, 480–482, 491, 494, 495, 591, 603
- 有向模型** Directed Model 481, 482, 485, 488, 490–492, 495, 507, 538, 557, 565, 566, 594
- 方向导数** directional derivative 76, 77
- 判别 RBM** discriminative RBM 453
- 判别器网络** discriminator network 597
- 分布式表示** distributed representation 16, 138, 228, 394–396, 404, 406–408, 410–412, 444, 449, 459, 466–471, 473, 498, 499
- 深度神经网络** DNN 247, 261, 262, 265, 271, 273, 381, 384, 391, 450–453, 471, 566
- 领域自适应** domain adaption 457

- 点积 dot product 30, 35, 123, 124
- 双反向传播 double backprop 233, 474
- 双重分块循环矩阵 doubly block circulant matrix 284, 307
- 降采样 downsampling 293, 298
- Dropout** Dropout 208, 222–230, 252, 257, 362, 364, 366, 367, 381, 383, 391, 455, 456, 574, 576, 588, 600
- Dropout Boosting** Dropout Boosting 229
- d-分离** d-separation 488, 490
- 动态规划 dynamic programming 188
- 动态结构 dynamic structure 382, 383
- 提前终止 early stopping 212–217, 237, 258, 362, 454, 455
- 回声状态网络 echo state network 21, 345–348
- 有效容量 effective capacity 100
- 特征分解 eigendecomposition 37–39
- 特征值 eigenvalue 37
- 特征向量 eigenvector 37
- 基本单位向量 elementary basis vectors 485
- 元素对应乘积 element-wise product 30
- 嵌入 embedding 442, 443
- 经验分布 empirical distribution 59, 60, 236, 238, 528
- 经验频率 empirical frequency 59
- 经验风险 empirical risk 236
- 经验风险最小化 empirical risk minimization 236, 237
- 编码器 encoder 4, 338, 339, 402–404, 421, 424–427, 429–432, 434–440, 442, 443, 445, 447, 451, 558, 595, 596
- 端到端的 end-to-end 359, 362, 363, 374, 392, 496
- 能量函数 energy function 485, 486, 499, 500, 511, 518, 559–561, 566, 567, 575, 578–583, 586
- 基于能量的模型 Energy-based model 485–487, 499, 506, 507, 510, 511, 513, 514, 559, 561, 566, 583
- 集成 ensemble 197, 220–223, 225–227, 229, 381, 402, 450

- 集成学习 ensemble learning 420
- 轮 epoch 242, 374
- 轮数 epochs 213
- 等式约束 equality constraint 83, 84
- 均衡分布 Equilibrium Distribution 508, 509
- 等变 equivariance 286
- 等变表示 equivariant representations 285
- 误差条 error bar 103
- 误差函数 error function 74
- 误差度量 error metric 359, 360
- 错误率 error rate 91, 360, 361, 366
- 估计量 estimator 108–115, 197, 456, 468, 520, 523
- 欧几里得范数 Euclidean norm 34
- 欧拉-拉格朗日方程 Euler-Lagrange Equation 552
- 证据下界 evidence lower bound 539, 540, 543, 544, 548, 565
- 样本 example 13, 23, 88, 90–95, 97, 99, 100, 102, 106, 107, 109, 110, 112–119, 123–125, 128, 129, 131–133, 135–138, 141, 210
- 额外误差 excess error 252, 256
- 期望 expectation 54, 56
- 期望最大化 expectation maximization 419, 541–544, 595
- E 步 expectation step 541
- 期望值 expected value 54
- 经验 experience, E 87, 88, 92, 94, 95
- 专家网络 expert network 383
- 相消解释 explaining away 538, 550, 565
- 相消解释作用 explaining away effect 489
- 解释因子 explanatory factor 463, 471, 473, 474
- 梯度爆炸 exploding gradient 248
- 利用 exploitation 409, 410
- 探索 exploration 409, 410

- 指数分布** exponential distribution 58
- 因子** factor 482–484, 486, 493, 494, 559, 585
- 因子分析** factor analysis 418, 420, 426
- 因子图** factor graph 493, 494
- 因子** factorial 417, 425, 426, 501, 544, 551, 562, 563, 568, 569
- 分解** factorization 69, 70
- 分解的** factorized 474
- 变差因素** factors of variation 4, 6, 173, 470, 472, 473
- 快速 Dropout** fast dropout 228
- 快速持续性对比散度** fast persistent contrastive divergence 524
- 可行** feasible 83, 84, 86
- 特征** feature 88, 92–96, 98, 99, 104, 123–125, 128–131
- 特征提取器** feature extractor 422, 425, 453, 469, 543
- 特征映射** feature map 282, 389
- 特征选择** feature selection 204
- 反馈** feedback 145
- 前向** feedforward 145
- 前馈分类器** feedforward classifier 464
- 前馈网络** feedforward network 145–150, 156, 169, 171, 172, 174, 193–196, 245, 247, 248, 259, 276, 319, 321, 330, 334, 337, 344, 347, 361, 362, 429, 432, 434, 435, 437, 449, 450, 464, 465, 472, 474, 593
- 前馈神经网络** feedforward neural network 145–148, 151, 153, 165, 171, 175, 246, 434
- 现场可编程门阵列** field programmable gated array 384
- 精调** fine-tune 451, 452, 455, 520
- 精调** fine-tuning 275, 276, 425, 565
- 有限差分** finite difference 373
- 第一层** first layer 145
- 不动点方程** fixed point equation 545, 549, 550, 554, 556, 557, 569, 572
- 定点运算** fixed-point arithmetic 378
- 翻转** flip 283

- 浮点运算 float-point arithmetic 378
- 遗忘门 forget gate 350–352
- 前向模式累加 forward mode accumulation 192
- 前向传播 forward propagation 175, 182, 183, 257–259, 285, 301, 302, 309, 325, 326, 338, 346, 349
- 傅立叶变换 Fourier transform 308, 309
- 中央凹 fovea 313
- 自由能 free energy 487
- 频率派概率 frequentist probability 49
- 频率派统计 frequentist statistics 118
- Frobenius** 范数 Frobenius norm 35, 41, 44, 45
- F** 分数 F-score 361
- 全 full 297
- 泛函 functional 155, 551–555
- 泛函导数 functional derivative 551–554
- Gabor** 函数 Gabor function 314–317
- Gamma** 分布 Gamma distribution 581
- 门控 gated 349–352, 355
- 门控循环网络 gated recurrent net 362
- 门控循环单元 gated recurrent unit 349, 351, 362
- 门控 **RNN** gated RNN 349, 351
- 选通器 gater 383
- 高斯分布 Gaussian distribution xxvi, 57, 58, 60, 68, 154, 156, 162, 163, 165, 295, 418, 426, 554, 555, 578, 580, 581, 587, 588, 595, 600, 602
- 高斯核 Gaussian kernel 124, 466
- 高斯混合模型 Gaussian Mixture Model 60, 61, 390, 391, 496
- 高斯混合体 Gaussian mixtures 466
- 高斯输出分布 Gaussian output distribution 155
- 高斯 **RBM** Gaussian RBM 579–581
- Gaussian-Bernoulli RBM** Gaussian-Bernoulli RBM xiv, 578–580

- 通用 GPU** general purpose GPU 379
- 泛化** generalization 97, 99, 136, 137, 146–149, 151, 172, 174, 194, 197, 198, 257, 277, 364, 381, 386, 389, 425, 457–459, 465, 468, 469, 472
- 泛化误差** generalization error 97, 100–102, 114, 236, 239–241, 250, 252, 257, 259, 261, 362, 364–367, 425
- 泛化** generalize 257, 457–459, 468–470, 473, 592, 603, 606
- 广义函数** generalized function 59
- 广义 Lagrange 函数** generalized Lagrange function 83, 84, 204
- 广义 Lagrangian** generalized Lagrangian 83, 85
- 广义伪似然** generalized pseudolikelihood 525, 526, 576
- 广义伪似然估计** generalized pseudolikelihood estimator 525
- 广义得分匹配** generalized score matching 527, 528
- 生成式对抗框架** generative adversarial framework 465
- 生成式对抗网络** generative adversarial network 464, 465, 513, 531, 592, 597–601
- 生成模型** generative model 385, 417, 419, 420, 422, 425, 426, 428, 431–433, 440, 453, 464, 465, 470, 471, 498, 513, 515, 531, 537, 557–559, 564–566, 585, 587, 591, 592, 594, 596, 600, 611–614
- 生成式建模** generative modeling 594, 595, 597, 602, 610–613
- 生成矩匹配网络** generative moment matching network 600, 601
- 生成随机网络** generative stochastic network xv, 431, 607–611
- 生成器网络** generator network 592–595, 597, 599–601
- 吉布斯分布** Gibbs distribution 484
- Gibbs 采样** Gibbs Sampling 495, 499–501, 510–513, 515, 522, 527, 565, 568, 570, 573, 576, 581, 582
- 吉布斯步数** Gibbs steps 519, 521, 523, 573
- 全局对比度归一化** Global contrast normalization 387–389
- 全局极小值** global minima 245, 246
- 全局最小点** global minimum 75, 76, 82, 85, 243, 244, 249, 279
- 梯度** gradient 76–78, 82, 83, 85, 86, 199–201, 203, 205, 214, 215, 323, 326–330, 343, 344, 346, 347, 349, 352–358, 438, 439
- 梯度上升** gradient ascent 548

- 梯度截断 gradient clipping 246, 248, 258, 354, 355
- 梯度下降 gradient descent 74, 75, 77–83, 85, 123, 132–134, 205, 206, 215, 237, 238, 242, 245–247, 249, 251–255, 258, 259, 266, 272–274, 354, 365, 371, 380, 381, 405, 421, 429, 437, 447, 453, 470, 511, 541, 545, 549, 577, 589, 594
- 图模型 graphical model 69, 331–334, 396, 475, 476, 479, 481, 487, 488, 491, 494–499, 501, 538, 543, 544, 550, 551, 554, 559, 561, 562, 566, 567, 591, 603
- 图形处理器 Graphics Processing Unit 239, 378–380, 383, 384
- 贪心 greedy 451, 452
- 贪心算法 greedy algorithm 275, 451
- 贪心逐层预训练 greedy layer-wise pretraining 310, 572, 575, 576
- 贪心逐层训练 greedy layer-wise training 572
- 贪心逐层无监督预训练 greedy layer-wise unsupervised pretraining 450–452
- 贪心监督预训练 greedy supervised pretraining 275, 276
- 贪心无监督预训练 greedy unsupervised pretraining 452, 574
- 网格搜索 grid search 368–370
- Hadamard 乘积** Hadamard product xxv, 30
- 汉明距离 Hamming distance 528
- 硬专家混合体 hard mixture of experts 383
- 硬双曲正切函数 hard tanh 170
- 簧风琴 harmonium 499, 561
- 哈里斯链 Harris Chain 509
- Helmholtz 机** Helmholtz machine 431, 592
- Hessian** Hessian xxv, 78–82, 200, 201, 203, 204, 215, 239, 242, 244, 246, 248, 253, 266–268, 270, 271, 279, 352, 453, 454, 575
- 异方差 heteroscedastic 162
- 隐藏层 hidden layer 5, 13, 146–148, 150, 165, 171–173, 184, 188, 190, 195, 224, 271, 272, 275–278, 301, 324, 429, 434, 440, 446, 448, 449, 471, 472, 527, 538, 561–571, 573, 580, 591, 604, 606, 610
- 隐马尔可夫模型 Hidden Markov Model 390–392

**隐藏单元** hidden unit vi, 5, 15, 16, 20, 21, 148, 154, 156, 165, 166, 168–172, 175, 190, 195, 206–208, 211, 215, 218, 220, 222–224, 226, 229, 230, 243, 257, 260, 273, 295, 300, 321, 323–327, 329, 332, 334, 335, 339, 345, 348–350, 352, 363, 365–368, 374, 382, 383, 387, 405, 421, 434, 437, 445, 446, 466, 469–472, 492, 496, 499, 500, 510, 517, 520, 522, 527, 539, 541, 543, 546, 547, 550, 560–562, 565–571, 575, 578–582, 584–586, 592, 594, 602, 604–606, 613

**隐藏变量** hidden variable 526, 538

**爬山** hill climbing 77

**超参数** hyperparameter 253, 254, 259, 261–264, 359, 363–371, 375, 455

**超参数优化** hyperparameter optimization 368

**假设空间** hypothesis space 98

**同分布的** identically distributed 97

**可辨认的** identifiable 243

**单位矩阵** identity matrix xxiii, 31

**独立同分布假设** i.i.d. assumption 97

**病态** ill conditioning 242

**不道德** immorality 491, 492

**重要采样** Importance Sampling 400, 401, 504–506, 532–536, 592, 596

**相互独立的** independent 53, 97

**独立成分分析** independent component analysis 418–422

**独立同分布** independent identically distributed 503, 531

**独立子空间分析** independent subspace analysis 421

**索引** index of matrix 27, 28

**指示函数** indicator function 58

**不等式约束** inequality constraint 83–85

**推断** inference xiv, 2, 208, 225, 227–229, 393, 394, 415, 431, 432, 497, 542, 559, 560, 565, 567–571, 573–577, 582, 586, 592–596, 598, 600, 605, 606, 613

**无限** infinite 456

**信息检索** information retrieval 448

**内积** inner product 123

**输入** input 282, 453

- 输入分布 input distribution 453, 454, 457
- 干预查询 intervention query 53
- 不变 invariant 291
- 求逆 invert 579
- Isomap** Isomap 456
- 各向同性 isotropic 58, 61
- Jacobian** Jacobian xxv, 77, 78, 176, 178, 180, 185–187, 233, 278, 329, 343, 345, 346, 373, 421, 445, 446
- Jacobian 矩阵** Jacobian matrix 65, 178, 192
- 联合概率分布 joint probability distribution 50, 52, 53, 69, 559–561, 566
- Karush–Kuhn–Tucker** Karush–Kuhn–Tucker 83–85, 204, 206
- 核函数 kernel function 123, 282
- 核机器 kernel machine 124, 125, 146, 210, 345, 466, 564
- 核方法 kernel method 124
- 核技巧 kernel trick 123, 124, 133, 146
- KL 散度** KL divergence 116, 219, 539, 545
- 知识库 knowledge base 2, 411, 412
- 知识图谱 knowledge graph 412
- Krylov 方法** Krylov method 193
- KL 散度** Kullback–Leibler (KL) divergence xxvi, 67, 68
- 标签 label 92, 94, 124, 136, 453, 459, 470, 472
- 标注 labeled 363, 364, 375, 450, 454, 456, 458, 459, 461, 462
- 拉格朗日乘子 Lagrange multiplier 552, 553
- 语言模型 language model 355, 392–394, 402, 403, 406, 410, 506
- Laplace 分布** Laplace distribution 58
- 大学习步骤 large learning step 544
- 潜在 latent 163, 418, 419, 426, 431, 451, 463, 496, 522, 560, 561, 597, 599, 602, 609
- 潜层 latent layer 561

- 潜变量** latent variable xiii, 60, 163, 243, 396, 417–419, 429, 431–433, 435, 452, 462, 466, 472, 486, 487, 496–499, 501, 512, 514, 517, 521, 527, 538, 539, 541, 542, 544, 545, 548, 554, 560–562, 564–567, 576, 592, 594–596, 607, 609
- 大数定理** Law of large number 503
- 逐层的** layer-wise 451
- L-BFGS** L-BFGS 270, 271
- 渗漏整流线性单元** Leaky ReLU 167, 362
- 渗漏单元** leaky unit 347–349
- 学成** learned 450, 454, 458, 459, 465, 467, 470, 473, 474, 557, 558, 592
- 学习近似推断** learned approximate inference 447
- 学习器** learner 106, 138, 240, 457, 459, 463, 469, 472, 473
- 学习率** learning rate 77, 79, 133, 235, 239, 242, 251, 252, 254, 256, 261–264, 266, 268, 271, 362, 363, 365–368, 372, 523, 524, 573, 589, 591
- 勒贝格可积** Lebesgue-integrable 517
- 左特征向量** left eigenvector 37
- 左奇异向量** left singular vector 40
- 莱布尼兹法则** Leibniz's rule 517
- 似然** likelihood 49
- 线搜索** line search 77, 83, 269
- 线性自回归网络** linear auto-regressive network 602
- 线性分类器** linear classifier 237, 428, 449, 453, 458, 467, 470
- 线性组合** linear combination 33
- 线性相关** linear dependence 33
- 线性因子模型** linear factor model 417, 418, 420, 421, 423, 425, 426, 428, 501, 543, 579
- 线性模型** linear model 14, 198, 203, 204, 206, 215, 228, 231, 560, 602
- 线性回归** linear regression 87, 94, 96–98, 100, 101, 104, 108, 117–119, 121–123, 133, 134, 198, 200–203, 205, 206, 219, 228, 260, 345, 428, 544, 602
- 线性阀值单元** linear threshold units 469, 470
- 线性无关** linearly independent 33
- 链接预测** link prediction 412
- 链接重要采样** linked importance sampling 537

**Lipschitz** Lipschitz 82

**Lipschitz 常数** Lipschitz constant 82

**Lipschitz 连续** Lipschitz continuous 82

**流体状态机** liquid state machine 345

**局部条件概率分布** local conditional probability distribution 480

**局部不变性先验** local constancy prior 136

**局部对比度归一化** local contrast normalization 388, 389

**局部下降** local descent 250

**局部核** local kernel 137, 466

**局部极大值** local maxima 127, 245

**局部极大点** local maximum 74, 75, 79, 80, 244, 549

**局部极小值** local minima 243–245, 249, 279, 453

**局部极小点** local minimum 74–76, 79, 80, 82, 213, 214, 237, 243, 244, 249, 255, 453

**对数尺度** logarithmic scale 368, 369

**逻辑回归** logistic regression 2, 3, 6, 123, 146, 153, 155, 177, 198, 206, 231, 310, 362, 367, 397, 530, 560, 600, 602

**logistic sigmoid** logistic sigmoid vi, 61, 62, 122, 157, 159, 168, 171

**分对数** logit 63, 158

**对数线性模型** log-linear model 486

**长短期记忆** long short-term memory ix, 16, 22, 260, 278, 349–353, 355, 356, 358, 362, 392

**长期依赖** long-term dependency 247, 341, 343–345, 347, 348, 351, 355

**环** loop 492, 493

**环状信念传播** loopy belief propagation 498, 499

**损失** loss 91, 116, 132, 528, 576

**损失函数** loss function 74, 107, 134, 219, 236, 237, 245, 248, 249, 253, 278, 325–327, 355, 365, 396, 401, 422, 430, 431, 433, 435, 447, 585, 587, 602

**机器学习** machine learning 2–4, 7, 10, 12–18, 20, 24, 26, 72, 86–95, 97–100, 102, 104, 105, 108, 112, 113, 118, 119, 123, 126, 132, 134, 135, 138, 139, 141, 197, 204, 206–208, 220, 222, 232, 234–237, 240, 241, 251, 252, 260, 279, 319, 353, 359–364, 371, 372, 374, 377, 378, 380–382, 401, 407, 408, 410, 411, 429, 440, 443, 449, 453, 458, 473, 474, 476, 486, 490, 496, 498, 502, 506, 518, 519, 542, 551, 552, 557

- 机器学习模型** machine learning model 452
- 机器翻译** machine translation 362, 459
- 主对角线** main diagonal 29
- 流形** manifold 139, 141, 142, 233, 426, 427, 438–446, 473, 474, 496, 511, 513, 597
- 流形假设** manifold hypothesis 140
- 流形学习** manifold learning 139, 434, 442–444, 597
- 边缘概率分布** marginal probability distribution 52
- 马尔可夫链** Markov Chain xv, 506–514, 518–524, 527, 534, 566, 569, 571, 573, 607–610
- 马尔可夫链蒙特卡罗** Markov Chain Monte Carlo 415, 504, 506, 507, 509, 511, 513, 518–520, 524, 528, 534, 563, 569, 575, 606, 609–611
- 马尔可夫网络** Markov network 482, 486, 496, 500
- 马尔可夫随机场** Markov random field 482, 486
- 掩码** mask 222–225, 228, 229
- 矩阵** matrix 28
- 矩阵逆** matrix inversion 31, 32
- 矩阵乘积** matrix product 29
- 最大范数** max norm 35
- 池** pool 291, 293, 294
- 最大池化** max pooling 290–293, 301, 469, 602
- 极大值** maxima 244, 245
- M 步** maximization step 541, 542
- 最大后验** Maximum A Posteriori v, 121, 122, 204, 392, 432, 542–544, 558, 582
- 最大似然** maximum likelihood 420, 424, 516, 545, 546
- 最大似然估计** maximum likelihood estimation 115–119, 121, 122, 134, 238, 393, 520, 525, 529, 543, 545
- 最大平均偏差** maximum mean discrepancy 601
- maxout** maxout 213, 243, 259, 278, 292, 317, 362
- maxout 单元** maxout unit 167, 168, 172, 317, 365
- 平均绝对误差** mean absolute error 156
- 均值和协方差 RBM** mean and covariance RBM 580–583
- 学生 t 分布均值乘积** mean product of Student t-distribution 580–583

- 均方误差** mean squared error 95, 96, 103, 104, 113, 116–118, 120, 129, 148, 154–156, 158, 194, 195, 345, 422, 430, 435, 437, 464, 465, 590, 595, 608
- 均值-协方差 RBM** mean-covariance restricted Boltzmann machine 486
- 均匀场** meanfield 21, 568–570, 572–574, 576, 577, 584, 592, 596, 605
- 均值场** mean-field 544–551, 554, 557, 558
- 测度论** measure theory 64
- 零测度** measure zero 64
- 记忆网络** memory network 356, 358, 412
- 信息传输** message passing 551
- 小批量** minibatch viii, 132, 183, 189, 190, 221–223, 237–241, 248, 251–254, 256, 259, 261–265, 270, 272, 320, 353, 354, 374, 380, 383, 429, 436, 453, 502, 509, 519, 521, 523, 541, 573, 577
- 小批量随机** minibatch stochastic 239
- 极小值** minima 245, 249
- 极小点** minimum 250, 251, 553
- 混合** Mixing 511–515, 521–524
- 混合时间** Mixing Time 509, 510
- 混合密度网络** mixture density network 163
- 混合分布** mixture distribution 59
- 专家混合体** mixture of experts 383, 466
- 模态** modality 460
- 峰值** mode xiii, 511–515, 520, 522–524, 551
- 模型** model 452
- 模型平均** model averaging 220–222
- 模型压缩** model compression 381
- 模型可辨识性** model identifiability 243
- 模型并行** model parallelism 380
- 矩** moment 600, 601, 611
- 矩匹配** moment matching 600, 611
- 动量** momentum 253–256, 261, 263, 264, 277, 362
- 蒙特卡罗** Monte Carlo 227, 400, 502–504, 506, 515, 518, 524, 532, 557, 581, 589, 595
- Moore-Penrose 伪逆** Moore-Penrose pseudoinverse xxv, 41, 99, 105

- 道德化 moralization 491, 492
- 道德图 moralized graph 491, 492
- 多层感知机 multilayer perceptron 5, 20, 21, 145, 188, 189, 194, 275, 276, 298, 340, 341, 403, 440, 471, 560, 565, 566, 568, 569, 574, 575, 586
- 多峰值 multimodal 533, 550, 611
- 多模态学习 multimodal learning 460
- 多项式分布 multinomial distribution 56
- Multinoulli 分布 multinoulli distribution 56, 59, 60, 73, 159, 163
- 多预测深度玻尔兹曼机 multi-prediction deep Boltzmann machine 575–577, 596, 607
- 多任务学习 multitask learning 210, 211, 457, 458
- 多维正态分布 multivariate normal distribution 58, 418, 512
- 朴素贝叶斯 naive Bayes 2
- 奈特 nats 66
- 自然语言处理 Natural Language Processing 246, 363, 377, 392, 395, 396, 406, 407, 410, 455
- 最近邻 nearest neighbor 137, 450, 466–468
- 最近邻图 nearest neighbor graph 443
- 最近邻回归 nearest neighbor regression 101, 125
- 负定 negative definite 38
- 负部函数 negative part function 63
- 负相 negative phase 517–520, 522–524, 526, 527, 557, 561, 571, 572
- 半负定 negative semidefinite 38
- Nesterov 动量 Nesterov momentum 256
- 网络 network 145
- 神经自回归密度估计器 neural auto-regressive density estimator xiv, 602, 604–606
- 神经自回归网络 neural auto-regressive network 603–606
- 神经语言模型 Neural Language Model 394, 396, 397, 399, 401, 402, 406, 411
- 神经机器翻译 Neural Machine Translation 395
- 神经网络 neural network 12–17, 19–23, 197–199, 205–207, 215, 218, 221, 222, 225, 229–232, 234, 235, 241–250, 257, 258, 261, 262, 266, 267, 269, 270, 273–275, 277–280, 319, 341, 349, 356, 358, 377–379, 384, 387, 390–392, 395, 396, 401, 402, 405, 406, 408, 411, 429, 444, 447, 452–455, 466, 470, 506, 556, 587

- 神经网络图灵机 neural Turing machine 356, 357
- 牛顿法 Newton's method 81, 82, 85, 242, 243, 245, 250, 266–268, 270, 274
- n*-gram n-gram 393, 394, 396, 397, 401–403, 467, 478
- 没有免费午餐定理 no free lunch theorem 102, 105, 472
- 噪声 noise 101, 140, 239, 248, 253, 279, 362, 363, 453, 528–531
- 噪声分布 noise distribution 529–531
- 噪声对比估计 noise-contrastive estimation 529–531
- 非凸 nonconvex 241, 243–246, 262, 266, 275, 279
- 非分布式 nondistributed 467–469
- 非分布式表示 nondistributed representation 466–468
- 非线性共轭梯度 nonlinear conjugate gradients 269, 270
- 非线性独立成分估计 nonlinear independent components estimation 420, 421
- 非参数 non-parametric 100, 394, 442–444
- 范数 norm 34
- 正态分布 normal distribution 57, 58, 61, 504, 553
- 正规方程 normal equation 96, 98, 99, 133, 148
- 归一化的 normalized 51
- 标准初始化 normalized initialization 258
- 数值 numeric value 182
- 数值优化 numerical optimization 235, 242, 246
- 对象识别 object recognition 246, 362, 364, 385, 389, 390, 423, 425, 459, 612
- 目标 objective 455
- 目标函数 objective function 74, 77, 84, 197–202, 204, 205, 213, 214, 217, 221, 236–238, 241, 246–248, 250, 252, 253, 265–267, 269, 274, 278, 279, 353, 359, 368, 374, 450, 470, 525, 527, 564, 572
- 奥卡姆剃刀 Occam's razor 100
- one-hot one-hot 125, 131, 161, 193, 394, 395, 454, 455, 459, 466, 468, 586, 603
- 一次学习 one-shot learning 459
- 在线 online 238
- 在线学习 online learning 240

**操作** operation 176

**最佳容量** optimal capacity 101, 103, 114

**原点** origin 33

**正交** orthogonal 36

**正交矩阵** orthogonal matrix 37

**标准正交** orthonormal 36, 39

**输出** output 453

**输出层** output layer 145

**过完备** overcomplete 431, 434, 582, 583

**过估计** overestimation 506

**过拟合** overfitting 98, 99, 105, 114, 197, 198, 215, 237, 241, 252, 258, 359, 363, 365, 366, 372, 375, 381, 450, 454, 455, 478, 613

**过拟合机制** overfitting regime 101

**上溢** overflow 72, 73, 535

**并行分布式处理** Parallel Distributed Processing 194

**并行回火** parallel tempering 514, 524, 537

**参数** parameter 94

**参数服务器** parameter server 381

**参数共享** parameter sharing 218, 225, 229, 285, 286, 288, 300, 313, 319, 320, 322, 323, 332, 333, 402, 601, 602, 604

**有参情况** parametric case 118

**参数化整流线性单元** parametric ReLU 167, 362

**偏导数** partial derivative 76, 77, 445, 551

**配分函数** Partition Function 415, 484, 486, 502, 506, 515, 516, 518, 519, 524, 525, 527–529, 531–537, 557, 559–561, 564, 565, 571, 578, 583, 584, 598

**性能度量** performance measures 87, 88, 91, 95, 361, 362

**性能度量** performance metrics 359, 360, 362, 370, 372, 374, 375

**置换不变性** permutation invariant 296

**持续性对比散度** persistent contrastive divergence 521, 523, 564, 572, 575, 581, 582

**音素** phoneme 390–392, 457

- 语音 phonetic 392
- 分段 piecewise 362
- 点估计 point estimator 108
- 策略 policy 409, 410
- 策略梯度 policy gradient 383
- 池化 pooling 207, 229, 281, 287, 290–295, 299, 306, 309, 310, 312, 313, 386, 421
- 池化函数 pooling function 290
- 病态条件 poor conditioning 74, 81, 239, 242, 246, 248, 250, 253, 454
- 正定 positive definite 38
- 正部函数 positive part function 63
- 正相 positive phase 517–520, 523, 524, 557, 560, 571
- 半正定 positive semidefinite 38
- 后验概率 posterior probability 60
- 幂方法 power method 248
- PR 曲线 PR curve 361
- 精度 precision 57, 361, 373, 612
- 精度矩阵 precision matrix 58
- 预测稀疏分解 predictive sparse decomposition 447
- 预训练 pretraining 275–278, 391, 425, 451–456, 498, 521, 527
- 初级视觉皮层 primary visual cortex 311
- 主成分分析 principal components analysis xi, 42–44, 128–130, 134, 210, 235, 302, 388, 418–420, 422, 424, 426, 427, 430, 441, 446, 448
- 先验概率 prior probability 60
- 先验概率分布 prior probability distribution 118, 295
- 概率 PCA probabilistic PCA 418–420, 426, 538, 539
- 概率密度函数 probability density function 51, 52, 57–59, 64, 503, 551–553, 598
- 概率分布 probability distribution 47, 50–56, 58–61, 66, 67, 69, 70, 360, 472, 516, 529, 531
- 概率质量函数 probability mass function 50, 51, 90, 560, 571
- 专家之积 product of expert 486
- 乘法法则 product rule 53

- 成比例 proportional 70
- 提议分布 proposal distribution 400, 532, 534–536
- 伪似然 pseudolikelihood 524–530, 571
- 象限对 quadrature pair 316
- 量子力学 quantum mechanics 48
- 径向基函数 radial basis function 124, 146, 170, 471
- 随机搜索 random search 369–371
- 随机变量 random variable 49–56, 58–60, 64, 65, 67, 69, 70, 472, 525, 530, 534
- 值域 range 33
- 比率匹配 ratio matching 527, 528, 564
- 召回率 recall 361, 382, 612
- 接受域 receptive field 287, 295
- 再循环 recirculation 429
- 推荐系统 recommender system 407–409
- 重构 reconstruction 429, 430, 436–439, 441, 442, 445–447, 608, 609
- 重构误差 reconstruction error 419, 422, 426, 427, 431, 433, 437, 438, 440, 445, 446, 448, 454, 514, 608
- 整流线性 rectified linear 151, 167, 230, 243, 273, 290
- 整流线性变换 rectified linear transformation 152
- 整流线性单元 rectified linear unit 14, 15, 150, 151, 165–168, 170–172, 177, 195, 233, 278, 362, 375, 391, 433, 455
- 整流网络 rectifier network 172, 173, 195
- 循环 recurrence 450
- 循环卷积网络 recurrent convolutional network 307
- 循环网络 recurrent network 145, 246–248, 307, 319–324, 326, 330, 333, 338, 341, 343–347, 349, 350, 353, 354, 357, 412, 417, 440, 474, 550, 576, 577
- 循环神经网络 recurrent neural network ix, 21, 22, 144, 145, 208, 228, 247, 306, 318–325, 328, 330–341, 343–345, 348, 349, 352, 355, 358, 392, 403, 550, 585, 586, 596
- 回归 regression 103
- 正则化 regularization 104, 105, 118, 122, 197–206, 208, 209, 212–220, 222, 227–236, 258, 355, 359, 362, 364–366, 387–389, 422, 431, 432, 434, 438, 440, 446, 453, 455, 472

- 正则化** regularize 239, 365, 421, 422, 455, 456, 514, 528, 575, 584, 588  
**正则化项** regularizer 104, 122, 126, 134, 362, 452, 454, 455, 467  
**强化学习** reinforcement learning 23, 93, 232, 383, 409, 410, 458, 557, 588, 590  
**关系** relation 410–412  
**关系型数据库** relational database 411  
**重参数化** reparametrization 575, 588  
**重参数化技巧** reparametrization trick 588, 594, 610  
**表示** representation 2–7, 16, 210, 219, 220, 297, 357, 367, 394, 395, 403, 404, 411, 430, 431, 433, 440–442, 448  
**表示学习** representation learning 4, 403, 417, 419, 448–450, 452, 457, 458, 461–463, 466, 472–474, 501, 514  
**表示容量** representational capacity 100, 104  
**储层计算** reservoir computing 345  
**受限玻尔兹曼机** Restricted Boltzmann Machine 228, 301, 391, 408, 437, 438, 448, 450, 472, 490, 499–501, 510, 514, 515, 517, 519–523, 533, 536–538, 561–568, 571, 572, 574, 575, 578, 579, 581, 583, 585, 586, 591, 600, 605, 609, 610  
**反向相关** reverse correlation 314  
**反向模式累加** reverse mode accumulation 191  
**岭回归** ridge regression 199  
**右特征向量** right eigenvector 37  
**右奇异向量** right singular vector 40  
**风险** risk 236  
**行** row 28  
**扫视** saccade 313  
**鞍点** saddle point 75, 76, 79, 80, 82, 244–246, 248, 249, 266, 267  
**无鞍牛顿法** saddle-free Newton method 245  
**相同** same 297, 298  
**样本均值** sample mean 110  
**样本方差** sample variance 110, 111  
**饱和** saturate 61  
**标量** scalar 27

- 得分 score 437–440, 526, 527  
得分匹配 score matching 437, 438, 445, 526–530, 606  
二阶导数 second derivative 77–80  
二阶导数测试 second derivative test 80  
第二层 second layer 145  
二阶方法 second-order method 245  
自对比估计 self-contrastive estimation 531  
自信息 self-information 66  
语义哈希 semantic hashing 448  
半受限玻尔兹曼机 semi-restricted Boltzmann Machine 539  
半监督 semi-supervised 363, 415  
半监督学习 semi-supervised learning 209, 210, 231, 450, 452, 454, 462, 463, 473  
可分离的 separable 309, 449, 453  
分离的 separate 473  
分离 separation 487, 488, 495  
情景 setting 458, 459, 469, 471  
浅度回路 shadow circuit 472  
香农熵 Shannon entropy xxvi, 66, 67  
香农 shannons 66  
塑造 shaping 279, 560, 611  
短列表 shortlist 396, 397  
**sigmoid** sigmoid 157–162, 168, 169, 195, 278, 362, 425, 511  
**sigmoid** 信念网络 sigmoid Belief Network 591, 592  
简单细胞 simple cell 311  
奇异的 singular 34  
奇异值 singular value 39, 40  
奇异值分解 singular value decomposition 39–41, 130, 408  
奇异向量 singular vector 39  
跳跃连接 skip connection 340, 341, 347, 348  
慢特征分析 slow feature analysis 421–423, 474

- 慢性原则** slowness principle 421–423
- 平滑** smoothing 394
- 平滑先验** smoothness prior 136
- softmax** softmax 449
- softmax 函数** softmax function 72, 73, 209, 226, 227, 325, 328, 372, 375, 383
- softmax 单元** softmax unit 375
- softplus** softplus 170
- softplus 函数** softplus function 61–63, 158, 170
- 生成子空间** span 33
- 稀疏** sparse 203, 204, 218–220, 227, 431–434, 440
- 稀疏激活** sparse activation 195
- 稀疏编码** sparse coding 274, 423–426, 432, 440, 447, 451, 490, 492, 496, 501, 527, 538, 543, 544, 551, 558, 582, 583, 591
- 稀疏连接** sparse connectivity 285–287
- 稀疏初始化** sparse initialization 259
- 稀疏交互** sparse interactions 285
- 稀疏权重** sparse weights 285
- 谱半径** spectral radius 345–347
- 语音识别** Speech Recognition 362, 377, 381, 390–392, 457
- sphering** sphering 388
- 尖峰和平板** spike and slab 317, 425, 426
- 尖峰和平板 RBM** spike and slab RBM 580–583
- 虚假模态** spurious modes 520, 522
- 方阵** square 34
- 标准差** standard deviation 54, 112, 238, 272, 273, 386–389
- 标准差** standard error 57, 111, 112, 238
- 标准正态分布** standard normal distribution 57
- 声明** statement 47, 48
- 平稳的** stationary 333
- 平稳分布** Stationary Distribution 508–510, 512

- 驻点 stationary point 74, 84
- 统计效率 statistic efficiency 118
- 统计学习理论 statistical learning theory 97
- 统计量 statistics 108
- 最陡下降 steepest descent 247
- 随机 stochastic 238, 239
- 随机课程 stochastic curriculum 280
- 随机梯度上升 Stochastic Gradient Ascent 541
- 随机梯度下降 stochastic gradient descent 14, 87, 132, 133, 205, 206, 216, 222, 228, 238–242, 246, 251–254, 256, 258, 270, 277, 344, 353, 354, 356, 362, 380, 437, 506, 518, 574, 575, 589, 606
- 随机矩阵 Stochastic Matrix 508
- 随机最大似然 stochastic maximum likelihood 521–524, 526, 528, 529, 564, 565, 568, 571–574, 576
- 流 stream 240
- 步幅 stride 287, 291, 293, 294, 297, 298, 301, 302, 306
- 结构学习 structure learning 496, 498
- 结构化概率模型 structured probabilistic model 47, 69, 70, 472, 475, 477, 479–482, 495, 498, 559
- 结构化变分推断 structured variational inference 544
- 亚原子 subatomic 48
- 子采样 subsample 502
- 求和法则 sum rule 52
- 和-积网络 sum-product network 472
- 监督 supervised 92, 210, 211, 218, 231, 236, 310, 311, 317, 379, 425, 440, 449–453, 455, 557, 584
- 监督学习 supervised learning xxvii, 87, 92–94, 101, 107, 116, 122, 123, 125, 126, 134, 140, 144, 210, 232, 236, 342, 362, 397, 407, 409, 410, 415, 432, 449, 450, 452, 453, 455–458, 462, 463, 472, 529, 594
- 监督学习算法 supervised learning algorithm 92
- 监督模型 supervised model 453
- 监督预训练 supervised pretraining 456
- 支持向量 support vector 124, 466

- 代理损失函数 surrogate loss function 237, 248  
符号 symbol 182  
符号表示 symbolic representation 182, 466, 468  
对称 symmetric 36
- 切面距离 tangent distance 232  
切平面 tangent plane 440, 443, 446  
正切传播 tangent prop 232–234  
目标 target 92–95, 101, 102, 105, 108, 116, 122, 128, 134, 135, 137, 138, 141  
泰勒 taylor 79, 81, 203, 215, 242  
导师驱动过程 teacher forcing 327, 328  
温度 temperature 514  
回火转移 tempered transition 514  
回火 tempering 514  
张量 tensor 28  
测试误差 test error 97, 98, 101, 103, 241, 363, 365, 366, 371, 372, 375, 452, 454, 455  
测试集 test set 91, 95, 97, 98, 106, 107, 112, 235, 237, 252, 277, 363, 364, 366, 372, 375, 454  
碰撞情况 the collider case 489  
绑定的权重 tied weights 285  
**Tikhonov 正则** Tikhonov regularization 199  
平铺卷积 tiled convolution 300, 301, 303, 305  
时延神经网络 time delay neural network 314, 319, 391  
时间步 time step 168, 247, 248, 265, 319–335, 339–341, 343, 346, 348–350, 352–354, 357, 392, 404, 405, 423, 577, 585, 605, 609, 610  
**Toeplitz 矩阵** Toeplitz matrix 284  
标记 token 392, 393, 411  
容差 tolerance 85, 549  
**地质 ICA** topographic ICA 421  
训练误差 training error 97, 98, 100–103, 236, 241, 364–366, 372, 375, 454  
训练集 training set 97, 98, 235–241, 243, 249, 251, 252, 254, 256, 260, 262–265, 267, 269, 274, 277, 280, 360, 362–364, 366, 367, 372, 373, 375, 462, 464, 468

- 转录 transcribe 89, 91, 94
- 转录系统 transcription system 359, 361, 372, 374, 375
- 迁移学习 transfer learning 454, 456–461, 604
- 转移 transition 322
- 转置 transpose 29
- 三角不等式 triangle inequality 34
- 三角形化 triangulate 493
- 三角形化图 triangulated graph 493
- 三元语法 trigram 393
- 无偏 unbiased 109, 240, 241, 251, 503–505, 528
- 无偏样本方差 unbiased sample variance 111
- 欠完备 undercomplete 430, 431
- 欠定的 underdetermined 552
- 欠估计 underestimation 506
- 欠拟合 underfitting 98, 99, 105, 114, 197–199, 241, 295, 359, 365, 366, 372, 373, 375, 598, 613
- 欠拟合机制 underfitting regime 101
- 下溢 underflow 72, 73
- 潜在 underlying 236, 237, 462–466, 470–474
- 潜在成因 underlying cause 461, 463, 473
- 无向 undirected 69
- 无向模型 undirected Model 482–488, 490–493, 495, 500, 502, 510, 515–518, 538, 557, 564, 565, 591
- 展开图 unfolded graph 322, 323, 326, 392
- 展开 unfolding 320–322, 340, 392
- 均匀分布 uniform distribution 51, 52, 55, 67, 165, 456
- 一元语法 unigram 393, 400
- 单峰值 unimodal 514, 556
- 单元 unit 146
- 单位范数 unit norm 36, 43
- 单位向量 unit vector 36

- 万能近似定理 universal approximation theorem 171, 172, 434
- 万能近似器 universal approximator 60, 471, 472, 560
- 万能函数近似器 universal function approximator 151
- 未标注 unlabeled 450, 454, 455, 459, 461, 463, 472
- 未归一化概率函数 unnormalized probability function 483, 484, 486, 493
- 非共享卷积 unshared convolution 299
- 无监督 unsupervised 20, 21, 92, 210, 218, 228, 363, 391, 415, 423, 425, 440, 447, 449–453, 455, 458, 459, 462, 463
- 无监督学习 unsupervised learning 87, 92–94, 107, 128, 134, 207, 210, 211, 234, 236, 363, 391, 415, 432, 443, 450–455, 457, 458, 462–464, 529, 610
- 无监督学习算法 unsupervised learning algorithm 92
- 无监督预训练 unsupervised pretraining 450, 452–457
- 有效 valid 284, 297, 298
- 验证集 validation set 106, 237, 242, 259, 368–370, 455
- 梯度消失与爆炸问题 vanishing and exploding gradient problem 247, 248, 259
- 梯度消失 vanishing gradient 248
- Vapnik-Chervonenkis 维度** Vapnik-Chervonenkis dimension 100, 467, 470
- 变量消去 variable elimination 547
- 方差 variance 54, 56, 57, 111, 197–199, 202, 206, 220
- 方差减小 variance reduction 589, 590
- 变分自编码器 variational auto-encoder 195, 431, 506, 558, 592, 594–597, 600, 606
- 变分导数 variational derivative 551
- 变分自由能 variational free energy 539
- 变分推断 variational inference 497, 499, 526
- 去噪 denoise 128, 386
- 向量 vector 27
- 虚拟对抗样本 virtual adversarial example 231
- 虚拟对抗训练 virtual adversarial training 452
- 可见层 visible layer 5
- V-结构** V-structure 489, 539

- 睡眠** wake sleep 557, 565, 592
- warp** warp 380, 383
- 支持向量机** support vector machine 123–125, 153, 310, 367, 522
- 无向图模型** undirected graphical model 516, 531
- 权重** weight 94
- 权重衰减** weight decay 104–106, 199–202, 205, 206, 209, 213, 215, 217, 218, 227, 228, 243, 258, 274, 364–367, 432, 454, 524, 544
- 权重比例推断规则** weight scaling inference rule 226–229
- 权重空间对称性** weight space symmetry 243
- 条件概率分布** conditional probability distribution 534
- 白化** whitening 388
- 宽度** width 146
- 赢者通吃** winner-take-all 161
- 正切传播** tangent propagation 474
- 流形正切分类器** manifold tangent classifier 474
- 词嵌入** word embedding 363, 395, 404, 406, 408, 454, 459
- 词义消歧** word-sense disambiguation 412
- 零数据学习** zero-data learning 459, 461
- 零次学习** zero-shot learning 459–461