

翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

## 28 Deep learning

### 28.1 Introduction

Many of the models we have looked at in this book have a simple two-layer architecture of the form  $z \rightarrow y$  for unsupervised latent variable models, or  $x \rightarrow y$  for supervised models. However, when we look at the brain, we seem many levels of processing. It is believed that each level is learning features or representations at increasing levels of abstraction. For example, the standard model of the visual cortex (Hubel and Wiesel 1962; Serre et al. 2005; Ranzato et al. 2007) suggests that (roughly speaking) the brain first extracts edges, then patches, then surfaces, then objects, etc. (See e.g., (Palmer 1999; Kandel et al. 2000) for more information about how the brain might perform vision.)

This observation has inspired a recent trend in machine learning known as deep learning (see e.g., (Bengio 2009), deeplearning.net, and the references therein), which attempts to replicate this kind of architecture in a computer. (Note the idea can be applied to non-vision problems as well, such as speech and language.)

In this chapter, we give a brief overview of this new field. However, we caution the reader that the topic of deep learning is currently evolving very quickly, so the material in this chapter may soon be outdated.

已思考 9s

## 逐段精确翻译（不遗漏）

### 28 深度学习

#### 28.1 引言

在本书中我们考察过的许多模型，都具有一种简单的两层架构：对于无监督的潜变量模型，形式为  $z \rightarrow y$ ；对于监督学习模型，形式为  $x \rightarrow y$ 。然而，当我们观察大脑时，我们会看到许多层次的处理。人们认为，每一层都在学习处于越来越高抽象水平的特征或表征。举例来说，视觉皮层的标准模型 (Hubel and Wiesel 1962; Serre 等 2005; Ranzato 等 2007) 表明（大致而言）大脑首先提取**边缘**，然后是**局部图块 (patches)**，接着是**表面 (surfaces)**，再然后是**物体 (objects)**，等等。（关于大脑可能如何实现视觉，参见例如 Palmer 1999; Kandel 等 2000。）

这一观察启发了机器学习中一个近期的潮流，称为**深度学习**（参见例如 Bengio 2009、网站 [deeplearning.net](#) 以及其中的参考文献），其试图在计算机中复制这种架构。（注意，这一思想也可以应用于非视觉问题，例如语音和语言。）

在本章中，我们对这一新领域做一个简要概述。然而，我们提醒读者：深度学习这一主题目前发展非常迅速，因此本章的材料可能很快就会过时。

**备注：**原文中 “when we look at the brain, we seem many levels of processing” 明显是排版/语法小误，语义应为 “we see many levels of processing”（我们会看到许多层次的处理）。

---

## 通俗而详尽的解释

### 1) “两层架构”是什么意思？

- **监督学习** ( $x \rightarrow y$ )：给定输入  $x$ （例如一张猫的图片），要预测输出  $y$ （例如标签“猫”）。箭头表示“把  $x$  直接映射到  $y$ ”。
- **无监督的潜变量模型** ( $z \rightarrow y$ )：这里  $z$  是**潜变量**，代表数据的隐藏因素（例如“光照强弱、姿态、风格”等），模型从这些隐藏因素生成观测到的  $y$ （例如一张图片或一个信号）。
- 之所以称为“两层”，是因为**只有输入层和输出层（或潜变量层和可观测层）**，中间几乎不做分层处理。

把它想成：

- 监督学习像“把原料 ( $x$ ) 直接做成成品 ( $y$ )”。
- 无监督潜变量像“先假设一些看不见的配方/参数 ( $z$ ) 会产生成品 ( $y$ )”，我们去学习这些配方。

### 2) 大脑为啥不是两层，而是“很多层”？

神经科学的发现表明：**视觉系统是分级的**。信息会经过一层又一层的处理，每一层提取更抽象的规律：

1. **边缘**：检测图像里哪里有明暗突变（像轮廓线）。
2. **局部图块 (patches)**：把多个边缘组合成简单纹理或小形状（如“角”“条纹片段”）。
3. **表面 (surfaces)**：再往上组合成更连续的区域与形状（比如一个物体的侧面）。
4. **物体 (objects)**：最终识别出完整的对象（猫、杯子、车）。

这就像**搭积木**：先有线段（边缘），再拼成小块（patch），再拼成大面（表面），最后拼出完整的东西（物体）。

### 3) “表征/特征、抽象程度”如何理解？

- **特征/表征 (feature/representation)** : 是模型“如何描述数据”的方式。
- **抽象程度**: 越高层的表征越“概括”，越不在意具体的像素细节，越关注“这像不像某个东西”。
  - 低层: 像素点、边缘 (非常具体)
  - 高层: 物体类别 (非常抽象)

类比: 识人时, 低层关注“这块像是眉毛的一条线”, 高层关注“这是张笑脸, 可能是小王”。

## 4) 深度学习在做什么?

- **核心思想**: 用**多层** (深层) 模型, 让机器**自动学会**从低层特征到高层表征的逐级提取与组合, **模仿大脑的层级处理**。
- **为什么有效**: 很多任务很难把“好特征”手工写死; 深度学习让模型自己在数据里学到这些特征, 并在高层形成对任务有用的抽象概念。
- **不只用于视觉**: 在**语音**里, 底层可能是声波特征, 中层是音素/发音单元, 高层是词与句子; 在**自然语言**里, 底层是词向量, 中层是短语/句法, 高层是语义与任务目标 (如情感、意图)。

## 5) 为何提示“可能很快过时”?

深度学习是**快速迭代**的领域: 模型结构 (如各类神经网络)、训练技巧、数据与评测方式都会不断更新。因此任何概述性章节都**有时效性**——几年后就可能不完全适用。

---

## 图式化小抄 (帮助理解)

- 传统 (浅层) 范式:
    - 监督:  $x \rightarrow y$  (直接从输入到输出)
    - 无监督潜变量:  $z \rightarrow y$  (潜在因素生成观测)
  - 大脑/深度学习的层级范式:
    - 视觉示例: 像素 → 边缘 → 图块 → 表面 → 物体 → 任务输出
    - 语音示例: 声谱 → 音素 → 音节/词 → 句子 → 任务输出
    - 语言示例: 字符/词 → 短语 → 句法/语义 → 篇章 → 任务输出
- 

## 小问题大白话

Q1: 为什么多层比两层更强?

因为现实世界的结构往往是**分层的**。多层网络可以逐级复用下层的结果来表达复杂结构, **用更少的参数表达更复杂的函数**, 也更容易泛化 (在足够数据和正则化下)。

## Q2：边缘、图块、表面这些词听起来抽象，能再打个比方吗？

把图片当拼图：

- 边缘 = 拼图块的边线；
- 图块 = 几块拼图拼成的小片段；
- 表面 = 更大的一整块区域；
- 物体 = 整幅拼好的图案。

## Q3：深度学习是不是只适合图像？

不是。它是一种**通用的分层表征学习方法**，图像、语音、文本、时间序列、推荐、强化学习等都可用。

---

## 要点总结

- 传统模型多是**两层映射**（监督： $x \rightarrow y$ ；无监督潜变量： $z \rightarrow y$ ）。
- **大脑处理是多层级的**：从低层到高层学习越来越抽象的特征（边缘→图块→表面→物体）。
- **深度学习旨在在计算机中复现这种层级结构**，自动学习表征；不仅用于视觉，也适用于**语音与语言**等领域。
- 该领域**发展极快**，综述性的材料**可能很快过时**。

翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

### 28.2 Deep generative models

Deep models often have millions of parameters. Acquiring enough labeled data to train such models is difficult, despite crowd sourcing sites such as Mechanical Turk. In simple settings, such as hand-written character recognition, it is possible to generate lots of labeled data by making modified copies of a small manually labeled training set (see e.g., Figure 16.13), but it seems unlikely that this approach will scale to complex scenes.

To overcome the problem of needing labeled training data, we will focus on unsupervised learning. The most natural way to perform this is to use generative models. In

this section, we discuss three different kinds of deep generative models: directed, undirected, and mixed.

### 28.2.1 Deep directed networks

Perhaps the most natural way to build a deep generative model is to construct a deep directed graphical model, as shown in Figure 28.1(a). The bottom level contains the observed pixels (or whatever the data is), and the remaining layers are hidden. We have assumed just 3 layers for notational simplicity. The number and size of layers is usually chosen by hand, although one can also use non-parametric Bayesian methods (Adams et al. 2010) or boosting (Chen et al. 2010) to infer the model structure. We shall call models of this form deep directed networks or DDNs. If all the nodes are binary, and all CPDs are logistic functions, this is called a sigmoid belief net (Neal 1992). In this case, the model defines the following joint distribution:

$$p(h_1, h_2, h_3, v|\theta) = i \text{Ber}(v_i|\text{sigm}(h_1^T w_0 i)) j \text{Ber}(h_1 j|\text{sigm}(h_1^T w_1 j)) \quad (28.1)$$

$$k \text{Ber}(h_2 k|\text{sigm}(h_2^T w_2 k)) l \text{Ber}(h_3 l|\text{sigm}(h_3^T w_3 l)) \quad (28.2)$$

Unfortunately, inference in directed models such as these is intractable because the posterior on the hidden nodes is correlated due to explaining away. One can use fast mean field approximations (Jaakkola and Jordan 1996a; Saul and Jordan 2000), but these may not be very accurate, since they approximate the correlated posterior with a factorial posterior. One can also use MCMC inference (Neal 1992; Adams et al. 2010), but this can be quite slow because the variables are highly correlated. Slow inference also results in slow learning.

### 28.2.2 Deep Boltzmann machines

A natural alternative to a directed model is to construct a deep undirected model. For example, we can stack a series of RBMs on top of each other, as shown in Figure 28.1(b). This is known as a deep Boltzmann machine or DBM (Salakhutdinov and Hinton 2009). If we have 3 hidden layers, the model is defined as follows:

$$p(h_1, h_2, h_3, v|\theta) = \frac{1}{Z(\theta)} \exp \left( \sum_{ij} v_i h_1 j W_{1ij} + \sum_{jk} h_1 j h_2 k W_{2jk} + \sum_{kl} h_2 k h_3 l W_{3kl} \right) \quad (28.3)$$

where we are ignoring constant offset or bias terms.

The main advantage over the directed model is that one can perform efficient block (layerwise) Gibbs sampling, or block mean field, since all the nodes in each layer are

conditionally independent of each other given the layers above and below (Salakhutdinov and Larochelle 2010).

The main disadvantage is that training undirected models is more difficult, because of the partition function.

However, below we will see a greedy layer-wise strategy for learning deep undirected models.

### 28.2.3 Deep belief networks

An interesting compromise is to use a model that is partially directed and partially undirected. In particular, suppose we construct a layered model which has directed arrows, except at the top, where there is an undirected bipartite graph, as shown in Figure 28.1(c). This model is known as a deep belief network (Hinton et al. 2006) or DBN.2 If we have 3 hidden layers, the model is defined as follows:

$$p(h_1, h_2, h_3, v|\theta) = i \text{Ber}(v_i|\text{sigm}(h_1^T w_1 i)) j \text{Ber}(h_2^T w_2 j) \quad (28.4)$$

$$1 Z(\theta) \exp k_l h_2^T w_3 l W_3 k_l \quad (28.5)$$

Essentially the top two layers act as an associative memory, and the remaining layers then generate the output.

The advantage of this peculiar architecture is that we can infer the hidden states in a fast, bottom-up fashion. To see why, suppose we only have two hidden layers, and that  $W_2 = W_1^T$ , so the second level weights are tied to the first level weights (see Figure 28.2(a)). This defines a model of the form  $p(h_1, h_2, v|W_1)$ . One can show that the distribution  $p(h_1, v|W_1) = h_2 p(h_1, h_2, v|W_1)$  has the form  $p(h_1, v|W_1) = 1 Z(W_1) \exp(v^T W_1 h_1)$ , which is equivalent to an RBM. Since the DBN is equivalent to the RBM as far as  $p(h_1, v|W_1)$  is concerned, we can infer the posterior  $p(h_1|v, W_1)$  in the DBN exactly as in the RBM. This posterior is exact, even though it is fully factorized.

Now the only way to get a factored posterior is if the prior  $p(h_1|W_1)$  is a complementary prior. This is a prior which, when multiplied by the likelihood  $p(v|h_1)$ , results in a perfectly factored posterior. Thus we see that the top level RBM in a DBN acts as a complementary prior for the bottom level directed sigmoidal likelihood function.

If we have multiple hidden levels, and/or if the weights are not tied, the correspondence between the DBN and the RBM does not hold exactly any more, but we can still use the factored inference rule as a form of approximate

bottom-up inference. Below we show that this is a valid variational lower bound. This bound also suggests a layer-wise training strategy, that we will explain in more detail later. Note, however, that top-down inference in a DBN is not tractable, so DBNs are usually only used in a feedforward manner.

#### 28.2.4 Greedy layer-wise learning of DBNs

The equivalence between DBNs and RBMs suggests the following strategy for learning a DBN.

- Fit an RBM to learn  $W_1$  using methods described in Section 27.7.2.
- Unroll the RBM into a DBN with 2 hidden layers, as in Figure 28.2(a). Now “freeze” the directed weights  $W_1$  and let  $W_2$  be “untied” so it is no longer forced to be equal to  $WT_1$ . We will now learn a better prior for  $p(h_1|W_2)$  by fitting a second RBM. The input data to this new RBM is the activation of the hidden units  $E[h_1|v, W_1]$  which can be computed using a factorial approximation.
- Continue to add more hidden layers until some stopping criterion is satisfied, e.g., you run out of time or memory, or you start to overfit the validation set. Construct the DBN from these RBMs, as illustrated in Figure 28.2(c). One can show (Hinton et al. 2006) that this procedure always increases a lower bound the observed data likelihood. Of course this procedure might result in overfitting, but that is a different matter.

In practice, we want to be able to use any number of hidden units in each level. This means we will not be able to initialize the weights so that  $W = WT - 1$ . This voids the theoretical guarantee. Nevertheless the method works well in practice, as we will see. The method can also be extended to train DBMs in a greedy way (Salakhutdinov and Larochelle 2010).

After using the greedy layer-wise training strategy, it is standard to “fine tune” the weights, using a technique called backfitting. This works as follows. Perform an upwards sampling pass to the top. Then perform brief Gibbs sampling in the top level RBM, and perform a CD update of the RBM parameters. Finally, perform a downwards ancestral sampling pass (which is an approximate sample from the posterior), and update the logistic CPD parameters using a small gradient step. This

is called the up-down procedure (Hinton et al. 2006).  
Unfortunately this procedure is very slow.

已思考 18s

## 逐段精确翻译（不遗漏）

### 28.2 深层生成模型 (Deep generative models)

深层模型通常拥有数以百万计的参数。即使有像 Mechanical Turk 这样的众包网站，要获得足够的**有标签**数据来训练这类模型仍然很困难。在一些简单场景中，比如手写字符识别，通过对一小份手工标注的训练集做各种**变形拷贝**来生成大量带标签的数据（例如见图 16.13），但这种方法很难扩展到**复杂场景**。

为克服对带标签训练数据的需求，我们将把注意力放在**无监督学习**上。最自然的做法就是使用**生成模型**。在本节中，我们讨论三种不同的深层生成模型：**有向**、**无向**以及**混合**。

#### 28.2.1 深层有向网络 (Deep directed networks)

也许构建深层生成模型最自然的方式，就是构造一个**深的有向图模型**，如图 28.1(a) 所示。底层包含观测到的像素（或任何数据），其余各层都是隐藏层。为了记号简洁，我们假设只有 3 层。层数和每层的大小通常**手工选择**，尽管也可以使用**非参数贝叶斯方法** (Adams 等, 2010) 或**提升法 (boosting)** (Chen 等, 2010) 来**推断模型结构**。

我们把这种形式的模型称为**深层有向网络 (DDN)**。如果所有结点都是二值的，且所有条件概率分布 (CPD) 都是 Logistic 函数，那么这叫做**Sigmoid 信念网络** (Neal, 1992)。在这种情况下，模型定义如下联合分布：

$$p(h_1, h_2, h_3, v \mid \theta) = \prod_i \text{Ber}(v_i \mid \text{sigm}(h_1^\top w_{0i})) \prod_j \text{Ber}(h_{1j} \mid \text{sigm}(h_2^\top w_{1j})) \quad (28.1)$$

$$\times \prod_k \text{Ber}(h_{2k} \mid \text{sigm}(h_3^\top w_{2k})) \prod_\ell \text{Ber}(h_{3\ell} \mid w_{3\ell}) \quad (28.2)$$

不幸的是，在这类有向模型中进行推断是不可 tractable (难以精确计算) 的，因为**隐藏结点的后验是相关的**，这源于所谓的\*\*“相互解释/遮蔽 (explaining away) ”现象。可以使用快速的平均场近似\*\* (Jaakkola 和 Jordan, 1996a; Saul 和 Jordan, 2000)，但它们可能不够准确，因为它们用**独立 (分解)** 的后验去近似一个**相关的后验**。也可以用 MCMC 推断 (Neal, 1992; Adams 等, 2010)，但由于变量**高度相关**，这会相当慢。推断慢也会导致**学习变慢**。

注：以上式子与原文一致；若你在别处见到下标或索引略有差异，多半是排版细节，不影响主旨。

## 28.2.2 深层玻尔兹曼机 (Deep Boltzmann machines)

一个自然的替代方案是构建**深的无向模型**。例如，我们可以把一系列 RBM (受限玻尔兹曼机) **逐层堆叠起来**，如图 28.1(b) 所示。这被称为**深层玻尔兹曼机 (DBM)** (Salakhutdinov 和 Hinton, 2009)。如果我们有 3 个隐藏层，模型定义为：

$$p(h_1, h_2, h_3, v \mid \theta) = \frac{1}{Z(\theta)} \exp\left(\sum_{ij} v_i h_{1j} W_{ij}^{(1)} + \sum_{jk} h_{1j} h_{2k} W_{jk}^{(2)} + \sum_{k\ell} h_{2k} h_{3\ell} W_{k\ell}^{(3)}\right) \quad (28.3)$$

其中我们忽略了常数偏置项 (bias)。

相较于有向模型，**主要优势在于可以进行高效的分块 (按层) Gibbs 采样，或分块平均场**，因为在给定其**上下相邻层**时，每一层内的所有结点**条件独立** (Salakhutdinov 和 Larochelle, 2010)。**主要劣势是训练无向模型更困难**，因为有配分函数  $Z(\theta)$  的存在。不过，下面我们将看到一种**逐层贪婪**的策略来学习深的无向模型。

## 28.2.3 深度置信网络 (Deep belief networks)

一个有趣的折中是使用**部分有向、部分无向**的模型。具体地，构造一个**分层模型**：除了最顶层外都用**有向箭头**，在顶层处放一个**无向的二部图**，如图 28.1(c) 所示。该模型称为**深度置信网络 (DBN)** (Hinton 等, 2006)。<sup>2</sup> 如果我们有 3 个隐藏层，模型定义如下：

$$p(h_1, h_2, h_3, v \mid \theta) = \prod_i \text{Ber}(v_i \mid \text{sigm}(h_1^\top w_{1i})) \prod_j \text{Ber}(h_{1j} \mid \text{sigm}(h_2^\top w_{2j})) \quad (28.4)$$

$$\times \frac{1}{Z(\theta)} \exp\left(\sum_{k\ell} h_{2k} h_{3\ell} W_{k\ell}^{(3)}\right) \quad (28.5)$$

本质上，**最顶的两层充当一个联想记忆 (associative memory)**，其余层再**生成输出**。

这种独特结构的好处是：我们可以进行**快速、自底向上的**隐藏状态推断。为了说明原因，假设只有两层隐藏层，且  $W_2 = W_1^\top$ ，也就是第二层的权重与第一层**绑定 (tied)** (见图 28.2(a))。这定义了模型  $p(h_1, h_2, v \mid W_1)$ 。可以证明，边缘分布

$$p(h_1, v \mid W_1) = \sum_{h_2} p(h_1, h_2, v \mid W_1)$$

具有如下形式：

$$p(h_1, v \mid W_1) = \frac{1}{Z(W_1)} \exp(v^\top W_1 h_1),$$

它等价于一个 RBM。因为就  $p(h_1, v \mid W_1)$  而言，**DBN 等价于 RBM**，我们就可以像在 RBM 中那样推断  $p(h_1 \mid v, W_1)$ 。这个后验是精确的，尽管它是完全**因子化 (分解) 的**。而只有当先验  $p(h_1 \mid W_1)$  是一个“互补先验 (complementary prior)”时，才能得到这种

**分解后的后验。** 所谓互补先验，是指它与似然  $p(v | h_1)$  相乘后，能够让后验完全分解。因此我们看到，在 DBN 中，顶层 RBM 相当于底层有向的 sigmoid 似然的一个互补先验。如果有**多层隐藏**，且/或**不再权重绑定**，DBN 与 RBM 的这种对应关系就不再完全成立，但我们仍可把因子化的推断规则当作一种**近似的自底向上推断**。下面我们会展示这对应一个**有效的变分下界**。该下界也启发了一种逐层训练策略，稍后会详细解释。不过需要注意，**自顶向下的推断在 DBN 中是不可 tractable 的**，因此 DBN 通常只前馈使用。

---

#### 28.2.4 DBN 的逐层贪婪学习 (Greedy layer-wise learning of DBNs)

DBN 与 RBM 的等价性提示了如下学习 DBN 的策略：

- 拟合第 1 个 RBM 来学习  $W_1$  (使用第 27.7.2 节所述方法)。
- 将该 RBM 展开为一个具有 2 个隐藏层的 DBN，如图 28.2(a)。现在\*\*“冻结”有向权重  $W_1$ ，并“解绑”\*\*  $W_2$ ，使其不再被强制等于  $W_1^\top$ 。我们现在通过拟合第二个 RBM 来为  $p(h_1 | W_2)$  学到一个**更好的先验**。这个新 RBM 的**输入数据**是隐藏单元的激活  $E[h_1 | v, W_1]$ ，可用**因子化近似计算**。
- 继续添加更多隐藏层，直到满足某个**停止准则**，例如时间或内存耗尽，或在验证集上开始过拟合。由这些 RBM 构造出 DBN，如图 28.2(c) 所示。

可以证明 (Hinton 等, 2006)，该过程总是会提升一个关于观测数据似然的下界。当然，这个过程可能会导致过拟合，但那是另一回事。

在实践中，我们希望每一层都能使用**任意数量**的隐藏单元。这意味着我们无法把权重初始化为满足  $W = W_{-1}^\top$  的形式。这会使上述**理论保证失效**。尽管如此，方法在实践中**仍然有效**，如我们将看到的。这一方法也可以扩展到**贪婪地训练 DBM** (Salakhutdinov 和 Larochelle, 2010)。

在采用逐层贪婪训练之后，通常还会用一种称为 backfitting (回合式微调) 的技术对权重进行微调 (fine-tune)。做法如下：先进行一次**向上的采样**传递到顶部；然后在顶层 RBM 中进行**短暂的 Gibbs 采样**，并对 RBM 参数做一次 CD (对比散度) 更新；最后进行一次**向下的“祖先式采样”传递** (这是对后验的一个近似采样)，并用**小的梯度步长**更新 logistic CPD 的参数。这称为 up-down (上-下) 过程 (Hinton 等, 2006)。不幸的是，这个过程**非常慢**。

---

## 通俗而详尽的解释

### 为何偏向“生成模型”和“无监督”？

- 深度模型参数多，需要很多数据。收集带标签的数据昂贵、慢、难扩展。

- 生成模型通过“学会数据是如何产生的”来刻画数据的整体结构，从而能在无标签或少标签下学到有用的表征。

## 三类深层生成模型一图看懂

- 有向 (DDN)**：像从上到下的因果链。每个结点由其父节点（上一层）通过条件分布生成。
  - 优点**：表达清晰、结构直观。
  - 难点**：后验推断易出现explaining away（两个原因解释同一现象时，彼此产生负相关），导致后验相关性强、推断难/慢。
- 无向 (DBM)**：像一块能量函数刻画的网络（玻尔兹曼机），层与层之间是对称连接，整体是马尔可夫随机场。
  - 优点**：在“给定上下层”的条件下，同层结点条件独立，可按层并行采样/平均场（效率高）。
  - 难点**：训练时要处理配分函数  $Z$ （归一化常数），这很难算。
- 混合 (DBN)**：顶层是无向（一个 RBM 作为联想记忆），下面是有向（逐层生成）。
  - 好处**：可以做快速的自下而上推断（像前馈网络一样），训练时还能用 RBM 的技巧逐层\*\*“预训练”\*\*。

## 关键概念用大白话

- Bernoulli (Ber)**：二项分布的 0/1 版本；每个结点就像一个灯泡（亮=1，灭=0）。
- sigm( $\cdot$ )**：Sigmoid 函数，把实数“压”到 (0,1) 变成概率。
- 配分函数  $Z$** ：把“能量”变成“概率”需要的归一化常数；全局依赖所有变量，难算。
- explaining away (相互解释/遮蔽)**：A 和 B 都能导致 C；一旦看到 C 发生，A、B 之间就会产生竞争关系（相关），这让后验计算变复杂。
- 平均场 (mean field)**：强行假设后验可分解（每个结点独立），用它去逼近真实后验；快，但可能不准。
- Gibbs 采样 (分块/按层)**：DBM 中，固定上下层时，同层结点互不影响，整层并行更新，大幅提速。
- 互补先验 (complementary prior)**：一种“神奇配对”的先验，使得先验  $\times$  似然的后验正好完全分解，便于快速推断。DBN 顶层 RBM 就在充当对下一层有向似然的互补先验。
- 逐层贪婪训练**：先训一个 RBM 得到第一层，再把它“展开”进 DBN，冻结它的有向权重，再在上一层数据的期望激活上训练下一个 RBM……一层一层往上堆。

- **变分下界**: 尽管我们做了近似, 但能证明每加一层并以这种方式训练, **数据似然的下界会上升** (理论保证在“权重绑定”等条件下最干净)。
- **微调 (fine-tune) 的 up-down:**
  1. 向上前馈/采样到顶层;
  2. 在顶层 RBM 中做**短 Gibbs + CD 更新**;
  3. 再**向下**做近似后验采样, 并对**有向层**做小步梯度更新。

这能整体协调各层参数, 但**很耗时**。

## 逐层贪婪训练直觉

- 想像在搭楼: 地基 (第 1 层特征) 稳了, 再在其“输出”(隐藏激活的期望) 上建第 2 层……
- 每层都把下层的表示\*\*“再抽象一点”, RBM 好训练、能并行, 为更深的结构打好初始化\*\*。
- 有了不错的初始化, 再用 up-down 微调, 整体性能更佳。

## 为什么 DBN 可以“快推断”?

- 在“**权重绑定**  $W_2 = W_1^\top$ 、两层隐藏”的特殊设置下, DBN 的  $p(h_1, v | W_1)$  等价于 RBM, 因而  $p(h_1 | v, W_1)$  可以像 RBM 一样快速、而且还是**因子化的精确后验**。
- 现实里往往**不做绑定**、层数也更多, 于是这种等价只剩**近似**; 但我们仍然使用那套**自底向上、因子化**的推断作为**有效近似**, 并可由**变分下界**加以支撑。

## 训练难点汇总

- DDN: 后验相关强 → 平均场不准、MCMC 慢。
- DBM: 采样推断高效, 但**学习要面对配分函数** (难核算/近似)。
- DBN: 结构混合, 能用 RBM 逐层预训练, 推断快; 但**自顶向下**精确推断仍难, 通常当作**前馈用**; 微调 (up-down) **耗时**。

## 要点总结

- 深层模型参数多, 带**标签数据**难, 因此转向**无监督 + 生成模型**。
- 三类深层生成模型:
  - DDN (**有向**): 表达直观, 但因 explaining away 导致**后验相关**、推断难慢。
  - DBM (**无向**): 同层条件独立, **按层 Gibbs/平均场**高效; 但训练受**配分函数拖累**。
  - DBN (**混合**): 顶层 RBM + 下层有向; 可**自底向上快推断**、**逐层贪婪预训练**。

- DBN  $\leftrightarrow$  RBM：在“**两层隐藏 + 权重绑定**”下，DBN 的  $p(h_1, v)$  等价 RBM；顶层 RBM 充当**互补先验**，使后验**因子化且精确**。
- **逐层贪婪训练**：一层层训练 RBM，堆成 DBN/DBM；每步都**提升似然下界**（在一定条件下）；随后用 up-down 进行**微调**，但过程**较慢**。
- 实务上即便**无严格绑定/证明条件**，上述流程**仍常常好用**。

这一节主要在讲什么？我不太懂

没关系，我用“超级白话”的方式给你捋一遍这节在说啥——

## 一句话版

深度学习里，如果**带标签数据不够**，就用**无监督的“生成模型”来学数据的内部结构**。本节介绍了**三种“深层生成模型”**（有向、无向、混合）以及一种**逐层搭楼的训练办法**，帮你把很深的模型训练起来。

## 核心动机（为啥要这么做）

- 深层模型参数多，**要很多标注**才好学；但标注很贵、很难搞到。
- 那就换条路：不用标签，直接学“**数据是怎么被生成出来的**”（这类模型叫**生成模型**）。

## 三类模型的“懒人包”

把“数据生成”想成做菜 ↗

### 1. 有向的深网 (DDN)：

- 像**有步骤的食谱**：上层决定下层，一步步“做”出数据。
- **问题**：多个原因一起解释同一现象会**互相牵制** (explaining away)，导致**推断很难又很慢**。

### 2. 无向的深网 (DBM)（把好多 RBM 叠起来）：

- 像**一锅互相影响的炖菜**：不是单向因果，而是大家彼此作用。
- **好处**：给定上下层时，同一层里的变量**互不打架**，所以能**整层一起采样/推断**，效率高。
- **难点**：训练要算一个全局的**配分函数 Z**，**超难**。

### 3. 深度置信网络 (DBN)（混合型）：

- **顶层用无向**（一个 RBM，当“联想记忆”）；**下面几层用有向往下“生成”**。

- **好处：**可以像前馈网络那样做**快速自底向上推断**，还方便用 RBM 进行**逐层预训练**。
- **提醒：**严格的自顶向下精确推断不现实，通常当成**前馈模型**来用。

## 怎么训练这么深？——“逐层贪婪搭楼”

- 先训**第一层** RBM，学到第一层特征。
- 用第一层的**隐藏激活**当“数据”，再训**第二层** RBM.....
- 一层层往上叠，最后把这些 RBM**拼成** DBN/DBM。
- 这样做通常会**提高数据似然的下界**（理论上更靠谱一些）。
- 最后再来一次**微调**（fine-tune）：上去采样、顶层短暂 Gibbs + CD 更新、再往下更新有向层——叫up-down过程（但很慢）。

## 各自的优缺点（超简表）

- DDN：表达清晰；但因为“相互解释”，**推断难/慢**。
- DBM：推断**按层并行**很爽；但**训练被配分函数卡脖子**。
- DBN：折中方案；**快推断 + 逐层预训练**好用；但**顶到底的精确推断不行**，微调也慢。

## 你需要记住的就这几条

1. **目的：**没标签也能学到**好表征**，靠**生成模型**刻画“数据怎么来”。
2. **三种形态：**有向（DDN）、无向（DBM）、混合（DBN）。
3. **训练诀窍：**逐层用 RBM 预训练，再整体微调。
4. **权衡：**DDN推断难；DBM训练难；DBN中庸但实用。

如果你愿意，我还能用小图/流程卡片把“逐层训练”和“三类模型差异”画出来，帮你记忆更牢。

翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

### 28.3 Deep neural networks

Given that DBNs are often only used in a feed-forward, or bottom-up, mode, they are effectively acting like neural networks. In view of this, it is natural to dispense with the generative story and try to fit deep neural networks directly, as we discuss below. The resulting training

methods are often simpler to implement, and can be faster.

Note, however, that performance with deep neural nets is sometimes not as good as with probabilistic models (Bengio et al. 2007). One reason for this is that probabilistic models support top-down inference as well as bottom-up inference. (DBNs do not support efficient top-down inference, but DBMs do, and this has been shown to help (Salakhutdinov and Larochelle 2010).) Top-down inference is useful when there is a lot of ambiguity about the correct interpretation of the signal.

It is interesting to note that in the mammalian visual cortex, there are many more feedback connections than there are feedforward connections (see e.g., (Palmer 1999; Kandel et al. 2000)). The role of these feedback connections is not precisely understood, but they presumably provide contextual prior information (e.g., coming from the previous “frame” or retinal glance) which can be used to disambiguate the current bottom-up signals (Lee and Mumford 2003).

Of course, we can simulate the effect of top-down inference using a neural network. However the models we discuss below do not do this.

### 28.3.1 Deep multi-layer perceptrons

Many decision problems can be reduced to classification, e.g., predict which object (if any) is present in an image patch, or predict which phoneme is present in a given acoustic feature vector. We can solve such problems by creating a deep feedforward neural network or multilayer perceptron (MLP), as in Section 16.5, and then fitting the parameters using gradient descent (aka back-propagation).

Unfortunately, this method does not work very well. One problem is that the gradient becomes weaker the further we move away from the data; this is known as the “vanishing gradient” problem (Bengio and Frasconi 1995). A related problem is that there can be large plateaus in the error surface, which cause simple first-order gradient-based methods to get stuck (Glorot and Bengio 2010). Consequently early attempts to learn deep neural networks proved unsuccessful. Recently there has been some progress, due to the adoption of GPUs (Ciresan et al. 2010) and second-order optimization algorithms

(Martens 2010). Nevertheless, such models remain difficult to train.

Below we discuss a way to initialize the parameters using unsupervised learning; this is called generative pre-training. The advantage of performing unsupervised learning first is that the model is forced to model a high-dimensional response, namely the input feature vector, rather than just predicting a scalar response. This acts like a data-induced regularizer, and helps backpropagation find local minima with good generalization properties (Erhan et al. 2010; Glorot and Bengio 2010).

### 28.3.2 Deep auto-encoders

An auto-encoder is a kind of unsupervised neural network that is used for dimensionality reduction and feature discovery. More precisely, an auto-encoder is a feedforward neural network that is trained to predict the input itself. To prevent the system from learning the trivial identity mapping, the hidden layer in the middle is usually constrained to be a narrow bottleneck. The system can minimize the reconstruction error by ensuring the hidden units capture the most relevant aspects of the data.

Suppose the system has one hidden layer, so the model has the form  $v \rightarrow h \rightarrow v$ . Further, suppose all the functions are linear. In this case, one can show that the weights to the  $K$  hidden units will span the same subspace as the first  $K$  principal components of the data (Karhunen and Joutsensalo 1995; Japkowicz et al. 2000). In other words, linear auto-encoders are equivalent to PCA. However, by using nonlinear activation functions, one can discover nonlinear representations of the data.

More powerful representations can be learned by using deep auto-encoders. Unfortunately training such models using back-propagation does not work well, because the gradient signal becomes too small as it passes back through multiple layers, and the learning algorithm often gets stuck in poor local minima.

One solution to this problem is to greedily train a series of RBMs and to use these to initialize an auto-encoder, as illustrated in Figure 28.3. The whole system can then be fine-tuned using backprop in the usual fashion. This approach, first suggested in (Hinton and Salakhutdinov 2006), works much better than trying to fit the deep auto-encoder directly starting with random weights.

### 28.3.3 Stacked denoising auto-encoders

A standard way to train an auto-encoder is to ensure that the hidden layer is narrower than the visible layer. This prevents the model from learning the identity function. But there are other ways to prevent this trivial solution, which allow for the use of an over-complete representation. One approach is to impose sparsity constraints on the activation of the hidden units (Ranzato et al. 2006). Another approach is to add noise to the inputs; this is called a denoising autoencoder (Vincent et al. 2010). For example, we can corrupt some of the inputs, for example by setting them to zero, so the model has to learn to predict the missing entries. This can be shown to be equivalent to a certain approximate form of maximum likelihood training (known as score matching) applied to an RBM (Vincent 2011).

Of course, we can stack these models on top of each other to learn a deep stacked denoising auto-encoder, which can be discriminatively fine-tuned just like a feedforward neural network, if desired.

已思考 19s

下面先逐段精确翻译（不遗漏），再做通俗详尽解释，最后给出要点总结。

## 逐段精确翻译

### 28.3 深度神经网络 (Deep neural networks)

鉴于 DBN 通常只以前馈 (feed-forward) 或自底向上 (bottom-up) 的方式使用，它们实际上表现得就像神经网络。基于此，自然就可以不再强调生成式故事 (generative story)，而是直接拟合深度神经网络，如下所述。由此得到的训练方法通常实现更简单，而且可能更快。

不过要注意，深度神经网络的性能有时不如概率模型 (Bengio 等, 2007)。其中一个原因是概率模型既支持自顶向下 (top-down) 推断，也支持自底向上推断。（DBN 不支持高效的自顶向下推断，但 DBM 支持，而且已被证明有帮助 (Salakhutdinov 和 Larochelle, 2010)。）当对信号的正确解释存在大量歧义时，自顶向下推断是有用的。

有趣的是，在哺乳动物视觉皮层中，反馈连接的数量远多于前馈连接（例如参见 Palmer 1999；Kandel 等 2000）。这些反馈连接的作用尚未被精确理解，但它们很可能提供上下文先验信息（例如来自先前的“帧”或一次视网膜扫视），这些信息可用于消解当前自底向上信号中的歧义 (Lee 和 Mumford, 2003)。

当然，我们可以用神经网络模拟自顶向下推断的效果。然而下面要讨论的模型并不这么做。

---

### 28.3.1 深层多层感知机 (Deep multi-layer perceptrons)

许多决策问题都可以归结为分类，例如：预测一个图像小块中存在哪个物体（如果有的话），或预测一个给定的声学特征向量中存在哪个音素。我们可以通过创建一个深度前馈神经网络或多层感知机 (MLP)（见第 16.5 节），然后用梯度下降（亦称反向传播）来拟合参数，从而解决此类问题。

不幸的是，这种方法效果并不理想。其中一个问题 是：离数据越远（层数越深），梯度越弱，这被称为“梯度消失”问题 (Bengio 和 Frasconi, 1995)。相关的问题是：误差曲面上可能存在很大的平台期 (plateaus)，这会使简单的一阶基于梯度的方法卡住 (Glorot 和 Bengio, 2010)。

因此，早期尝试学习深度神经网络是不成功的。最近，由于 GPU 的采用 (Ciresan 等, 2010) 以及二阶优化算法 (Martens, 2010)，出现了一些进展。尽管如此，这类模型依然难以训练。

下面我们将讨论一种用无监督学习来初始化参数的方法；这被称为生成式预训练 (generative pre-training)。先进行无监督学习的优势在于：模型被迫去刻画一个高维响应（即输入特征向量本身），而不是只预测一个标量响应。这在数据层面起到了正则化的作用，并帮助反向传播找到具有良好泛化性质的局部极小值 (Erhan 等, 2010; Glorot 和 Bengio, 2010)。

---

### 28.3.2 深度自编码器 (Deep auto-encoders)

自编码器是一种用于降维与特征发现的无监督神经网络。更准确地说，自编码器是一个被训练来预测其输入本身的前馈神经网络。为了防止系统学习到平凡的恒等映射，中间的隐藏层通常会被限制为一个狭窄的瓶颈。系统可以通过让隐藏单元捕获数据中最相关的方面来最小化重建误差。

假设系统只有一层隐藏层，因此模型形式为  $v \rightarrow h \rightarrow v$ 。进一步地，假设所有函数都是线性的。在这种情况下，可以证明：连接到  $K$  个隐藏单元的权重会张成与数据的前  $K$  个主成分相同的子空间 (Karhunen 和 Joutsensalo, 1995; Japkowicz 等, 2000)。换言之，线性自编码器等价于 PCA。然而，通过使用非线性激活函数，我们可以发现数据的非线性表征。

通过使用深度自编码器可以学习到更强的表征。不幸的是，使用反向传播来训练此类模型效果并不好，因为当梯度信号反向穿过多层时会变得过小，而且学习算法常常会陷入较差的局部极小值。

解决这一问题的一个办法是：贪婪地训练一系列 RBM (受限玻尔兹曼机)，并用它们来初

**始化一个自编码器**, 如图 28.3 所示。随后可以按常规方式用**反向传播**进行整体微调。这种方法最早由 (Hinton 和 Salakhutdinov, 2006) 提出, 且远胜于从随机权重直接拟合深度自编码器。

---

### 28.3.3 堆叠去噪自编码器 (Stacked denoising auto-encoders)

训练自编码器的一个标准方式是: 确保**隐藏层比可见层更窄**。这可以防止模型学到恒等函数。但也有**其他方法**能避免这一平凡解, 并允许使用**过完备表示** (over-complete representation)。一种方法是对隐藏单元的激活施加**稀疏性约束** (Ranzato 等, 2006)。另一种方法是在输入上**加入噪声**; 这被称为**去噪自编码器** (Vincent 等, 2010)。例如, 我们可以**破坏部分输入** (比如把它们置为 0), 从而迫使模型**学习去预测缺失的条目**。可以证明, 这等价于对 RBM 施加的一种**最大似然训练的近似形式** (称为 score matching, 得分匹配) (Vincent, 2011)。

当然, 我们可以**把这些模型堆叠起来**, 以学习一个**深度的堆叠去噪自编码器**; 如果需要, 它可以像前馈神经网络那样进行**判别式微调**。

---

## 通俗而详尽的解释

### 1) 为什么从“生成式”转向“直接训练神经网络”?

- **直觉**: 既然很多 DBN 最后都是**像前馈网一样用** (只自下而上), 那不如**直接训练前馈深网**, 实现简单、速度还可能更快。
- **代价**: 纯神经网络**缺少自顶向下推断的能力**; 而**概率模型** (尤其 DBM) 能“从高层往回看”, 在**歧义大时更能利用上下文**把意思弄清楚。

### 2) 大脑启示: 反馈很多

- 视觉皮层里**反馈通路比前馈通路**还多, 可能提供**上下文先验** (前一帧、前一眼的信息), 帮助**消除歧义**。
- 我们可以用 NN 去模拟这种自顶向下, 但**本节后续介绍的模型没有用**这种机制。

### 3) 深层 MLP (多层感知机) 为什么难训?

- **梯度消失**: 越往前 (靠近输入) 层, **信号衰减**越厉害, 学不动。
- **平台期**: 误差面大片**平坦**, 一阶方法 (普通梯度下降) 走不动。
- **改进**: GPU 加速、二阶优化 (考虑曲率, 如 Hessian 的信息) 能有所帮助, 但整体仍**难**。
- **生成式预训练**: 先用**无监督**逼着模型**重建输入** (高维任务), 像给模型打“**数据正则**”, 让后面的**监督微调**更容易收敛到**泛化更好的地方**。

## 4) 自编码器到底干嘛？

- **目标：**把输入**压缩到隐藏层（瓶颈）**，再**重建回来**；如果重建得好，说明瓶颈里抓住了**关键信息**。
- **线性自编码器 = PCA：**只有一层隐藏且线性时，它学到的空间等价于**主成分**。
- **非线性自编码器：**用激活函数（ReLU、sigmoid 等），就能学到**非线性的低维表示**。
- **深度自编码器：**更强，但**直接反传容易梯度太小、陷入坏局部最小值**。
- **解法：**先逐层训练 RBM 做预训练，再整体**反传微调**——实践上比随机初始化好很多。

## 5) 去噪自编码器与“过完备”

- **过完备：**隐藏维度 $\geq$ 输入维度也可以，但要防止抄作业（恒等映射）\*\*。
  - **两条路：**
    - 加**稀疏性**（大多数隐藏单元多数时候为 0）
    - 加**噪声做去噪自编码**（把部分输入抹掉，让模型**补全**）
  - **有趣的联系：**去噪自编码器与对 RBM 的一种**近似最大似然**（score matching）等价。
  - **可堆叠：**一层层堆起来形成**深度去噪自编码器**，最后像普通前馈网一样**做分类微调**。
- 

## 要点总结（记住这些就够）

- **为什么不用生成式？**因为很多 DBN 实际只前馈用，直接训练**深度前馈网更简单/快**。
- **但概率模型强在哪？**它们能做**自顶向下推断**，在**歧义大**时更有效（DBM 可以，DBN 基本不行）。
- **深层 MLP 难点：**梯度消失 + 平台期；GPU 与**二阶优化**有所缓解，但**仍然不容易**。
- **生成式预训练：**先**无监督重建输入**（自编码/RBM），相当于“**数据正则**”，再**监督微调，更稳更泛化**。
- **自编码器：**
  - 线性版  $\approx$  PCA；
  - 非线性版学**非线性表示**；
  - 深度版需RBM **逐层预训练 + 反传微调**更靠谱。
- **去噪自编码器：**对输入**加噪、学补全**；允许**过完备**，与 RBM 的 score matching 有**等价**；**可堆叠并判别式微调**。

如果你想，我可以把“**预训练→微调**”的流程画成小图卡，或给你一个“**训练深度自编码器的实操清单**”，帮助你动手做一遍更有感觉。

这一节主要在讲什么？我不太懂

已思考 7s

没事~我用“超白话速读卡”把 28.3 讲清楚 👇

## 一句话版

这节说：既然很多 DBN 最后都是**当前馈网来用**，那不如**直接训练深度神经网络**；但深网训练不容易，于是用\*\*无监督预训练（自编码器/RBM、去噪等）\*\*来帮忙，让后续反传更稳、更准。

## 核心内容 (3 块)

### 1. 为什么直接上深度神经网络？

- DBN 常只做**自下而上推断**，用法跟普通神经网络差不多；直接训练深网**更简单/可能更快**。
- 但**概率模型**（特别 DBM）能做**自顶向下推断**，遇到**歧义大的信号更有优势**（用“上下文”消歧）。大脑里**反馈连接很多**，也在暗示这点。

### 2. 深层 MLP 难点 & 解决思路

- 难点：**梯度消失**、误差面有**大平台**，一阶优化容易卡住；后来有 GPU 和**二阶优化**稍微改善。
- 关键招：**生成式预训练**——先用无监督任务（重建输入）把网络“练热身”，相当于**数据诱导的正则化**，再用反传做监督微调，更容易找到**泛化好的解**。

### 3. 自编码器家族（预训练的主力）

- **自编码器 (AE)**：输入→瓶颈→重建输入；逼它通过“窄瓶颈”，学到**关键信息**。
  - 线性 AE  $\approx$  PCA；非线性 AE 学**非线性表示**。
  - **深度 AE** 直接反传不好训 → 用**逐层 RBM 预训练再微调**，效果显著更好。
- **去噪自编码器 (DAE)**：给输入**加噪/抹零**，让模型学**补全**，既能防“**抄作业**”（恒等映射），又允许**过完备表示**；可**堆叠**并像前馈网那样做**判别微调**。

## 打个比方

- **自顶向下推断**就像读句子时用**上下文**补全含义；没有它，碰到歧义更容易猜错。
- **自编码器**像“**先压缩，再还原**”的复印机：能还原得好，说明压缩里抓住了重点。
- **去噪自编码器**像做**填空题**：把输入抹掉一部分，强迫模型学会“**合理补全**”。

## 要点总结

- 这节主旨：从生成式 DBN 转到直接训练深网，因为用法相近且更简便。
- 但概率模型的顶向下推断在消歧上有优势；纯深网通常缺这点。
- 深网训练难在梯度消失/平台面；无监督预训练（AE/RBM）能充当正则 & 好初始化，再反传微调。
- 线性 AE= PCA；深度 AE需逐层预训练+微调才好训。
- 去噪 AE通过加噪补全避免恒等映射，能做过完备，还能堆叠+判别微调。

翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

## 28.4 Applications of deep networks

In this section, we mention a few applications of the models we have been discussing.

### 28.4.1 Handwritten digit classification using DBNs

Figure 28.4(a) shows a DBN (from (Hinton et al. 2006)) consisting of 3 hidden layers. The visible layer corresponds to binary images of handwritten digits from the MNIST data set. In addition, the top RBM is connected to a softmax layer with 10 units, representing the class label.

The first 2 hidden layers were trained in a greedy unsupervised fashion from 50,000 MNIST digits, using 30 epochs (passes over the data) and stochastic gradient descent, with the CD heuristic. This process took "a few hours per layer" (Hinton et al. 2006, p1540). Then the top layer was trained using as input the activations of the lower hidden layer, as well as the class labels. The corresponding generative model had a test error of about 2.5%. The network weights were then carefully fine-tuned on all 60,000 training images using the up-down procedure. This process took "about a week" (Hinton et al. 2006, p1540). The model can be used to classify by performing a deterministic bottom-up pass, and then computing the free energy for the top-level RBM for each possible class label. The final error on the test set was about 1.25%. The misclassified examples are shown in Figure 28.4(b).

This was the best error rate of any method on the

permutation-invariant version of MNIST at that time. (By permutation-invariant, we mean a method that does not exploit the fact that the input is an image. Generic methods work just as well on permuted versions of the input (see Figure 1.5), and can therefore be applied to other kinds of datasets.) The only other method that comes close is an SVM with a degree 9 polynomial kernel, which has achieved an error rate of 1.4% (Decoste and Schoelkopf 2002). By way of comparison, 1-nearest neighbor (using all 60,000 examples) achieves 3.1% (see `mnist1NNdemo`). This is not as good, although 1-NN is much simpler.

#### 28.4.2 Data visualization and feature discovery using deep auto-encoders

Deep autoencoders can learn informative features from raw data. Such features are often used as input to standard supervised learning methods.

To illustrate this, consider fitting a deep auto-encoder with a 2d hidden bottleneck to some text data. The results are shown in Figure 28.5. On the left we show the 2d embedding produced by LSA (Section 27.2.2), and on the right, the 2d embedding produced by the auto-encoder. It is clear that the low-dimensional representation created by the auto-encoder has captured a lot of the meaning of the documents, even though class labels were not used. Note that various other ways of learning low-dimensional continuous embeddings of words have been proposed. See e.g., (Turian et al. 2010) for details.

#### 28.4.3 Information retrieval using deep auto-encoders (semantic hashing)

In view of the success of RBMs for information retrieval discussed in Section 27.7.3.1, it is natural to wonder if deep models can do even better. In fact they can, as is shown in Figure 28.6.

More interestingly, we can use a binary low-dimensional representation in the middle layer of the deep auto-encoder, rather than a continuous representation as we used above. This enables very fast retrieval of related documents. For example, if we use a 20-bit code, we can precompute the binary representation for all the documents, and then create a hash-table mapping codewords to documents. This approach is known as semantic hashing, since the binary representation of

semantically similar documents will be close in Hamming distance.

For the 402,207 test documents in Reuters RCV1-v2, this results in about 0.4 documents per entry in the table. At test time, we compute the codeword for the query, and then simply retrieve the relevant documents in constant time by looking up the contents of the relevant address in memory. To find other other related documents, we can compute all the codewords within a Hamming distance of, say, 4. This results in retrieving about  $6196 \times 0.4 \approx 2500$  documents. The key point is that the total time is independent of the size of the corpus.

Of course, there are other techniques for fast document retrieval, such as inverted indices. These rely on the fact that individual words are quite informative, so we can simply intersect all the documents that contain each word. However, when performing image retrieval, it is clear that we do not want to work at the pixel level. Recently (Krizhevsky and Hinton 2010) showed that a deep autoencoder could learn a good semantic hashing function that outperformed previous techniques (Torralba et al. 2008; Weiss et al. 2008) on the 80 million tiny images dataset. It is hard to apply inverted indexing techniques to real-valued data (although one could imagine vector quantizing image patches).

#### 28.4.4 Learning audio features using 1d convolutional DBNs

To apply DBNs to time series of unbounded length, it is necessary to use some form of parameter tying. One way to do this is to use convolutional DBNs (Lee et al. 2009; Desjardins and Bengio 2008), which use convolutional RBMs as their basic unit. These models are a generative version of convolutional neural nets discussed in Section 16.5.1. The basic idea is illustrated in Figure 28.7. The hidden activation vector for each group is computed by convolving the input vector with that group's filter (weight vector or matrix). In other words, each node within a hidden group is a weighted combination of a subset of the inputs. We compute the activation of all the hidden nodes by "sliding" this weight vector over the input. This allows us to model translation invariance, since we use the same weights no matter where in the input vector the pattern occurs. Each group has its own filter,

corresponding to its own pattern detector.

More formally, for binary 1d signals, we can define the full conditionals in a convolutional RBM as follows (Lee et al. 2009):

$$p(h_k t=1|v) = \text{sigm}((w_k \otimes v)t + b_k) \quad (28.6)$$

$$p(v_k s=1|h) = \text{sigm}(k(w_k \otimes h)s + c_k) \quad (28.7)$$

where  $w_k$  is the weight vector for group  $k$ ,  $b_k$  and  $c_k$  are bias terms, and  $a \otimes b$  represents the convolution of vectors  $a$  and  $b$ .

It is common to add a max pooling layer as well as a convolutional layer, which computes a local maximum over the filtered response. This allows for a small amount of translation invariance. It also reduces the size of the higher levels, which speeds up computation considerably. Defining this for a neural network is simple, but defining this in a way which allows for information flow backwards as well as forwards is a bit more involved. The basic idea is similar to a noisy-OR CPD (Section 10.2.3), where we define a probabilistic relationship between the max node and the parts it is maxing over. See (Lee et al. 2009) for details. Note, however, that the top-down generative process will be difficult, since the max pooling operation throws away so much information.

(Lee et al. 2009) applies 1d convolutional DBNs of depth 2 to auditory data. When the input consists of speech signals, the method recovers a representation that is similar to phonemes. When applied to music classification and speaker identification, their method outperforms techniques using standard features such as MFCC. (All features were fed into the same discriminative classifier.) In (Seide et al. 2011), a deep neural net was used in place of a GMM inside a conventional HMM. The use of DNNs significantly improved performance on conversational speech recognition. In an interview, the tech lead of this project said "historically, there have been very few individual technologies in speech recognition that have led to improvements of this magnitude".<sup>7</sup>

#### 28.4.5 Learning image features using 2d convolutional DBNs

We can extend a convolutional DBN from 1d to 2d in a straightforward way (Lee et al. 2009), as illustrated in Figure 28.8. The results of a 3-layer system trained on four classes of visual objects (cars, motorbikes, faces and

airplanes) from the Caltech 101 dataset are shown in Figure 28.9. We only show the results for layers 2 and 3, because layer 1 learns Gabor-like filters that are very similar to those learned by sparse coding, shown in Figure 13.21(b). We see that layer 2 has learned some generic visual parts that are shared amongst object classes, and layer 3 seems to have learned filters that look like grandmother cells, that are specific to individual object classes, and in some cases, to individual objects.

下面先逐段精确翻译（不遗漏），再做通俗详尽解释，最后给出要点总结。

---

## 逐段精确翻译

### 28.4 深度网络的应用 (Applications of deep networks)

在本节中，我们提及一些我们一直在讨论的模型的应用。

---

#### 28.4.1 使用 DBN 的手写数字分类

图 28.4(a) 展示了一个由 (Hinton 等, 2006) 提出、包含 3 个隐藏层的 DBN。可见层对应 MNIST 数据集中二值的手写数字图像。此外，顶层 RBM 还连接到一个 10 单元的 softmax 层，表示类别标签。

前 2 个隐藏层是在 5 万个 MNIST 数字上，以贪婪的无监督方式训练的，使用 30 个 epoch (在数据上往返 30 次) 和随机梯度下降，并采用 CD 启发式。这个过程“每层花费数小时” (Hinton 等, 2006, 第 1540 页)。随后，顶层使用下层隐藏层的激活以及类别标签作为输入进行训练。相应的生成模型在测试集上的错误率约为 2.5%。然后使用 up-down 过程在全部 6 万张训练图像上仔细微调网络权重。这个过程“大约花了一周” (Hinton 等, 2006, 第 1540 页)。该模型可以通过确定性的自底向上传播进行分类，然后为每个可能的类别标签计算顶层 RBM 的自由能。最终在测试集上的错误率约为 1.25%。误分类的样本如图 28.4(b) 所示。

这在当时是置换不变 (permutation-invariant) 版本的 MNIST 上的最佳错误率。(所谓置换不变，是指一种方法不利用输入是图像这一事实。通用方法在输入被打乱像素位置的版本上也同样有效 (见图 1.5)，因此可以应用到其它类型的数据集。) 唯一接近的另一种方法是使用 9 次多项式核的 SVM，其错误率达到 1.4% (Decoste 和 Schoelkopf, 2002)。作为比较，1-近邻 (使用全部 6 万个样本) 达到 3.1% (见 mnist1NNdemo)。这没有那么好，尽管 1-NN 要简单得多。

---

## 28.4.2 使用深度自编码器进行数据可视化与特征发现

深度自编码器可以从原始数据中学习到信息量丰富的特征。这些特征常被用作标准监督学习方法的输入。

为说明这一点，考虑对一些文本数据拟合一个瓶颈为二维隐藏层的深度自编码器。结果如图 28.5 所示。左图为 LSA（第 27.2.2 节）产生的 2D 嵌入，右图为自编码器产生的 2D 嵌入。很明显，即使没有使用类别标签，由自编码器创建的低维表示已经捕获了大量文档的语义。

请注意，还有各种其他学习词的低维连续嵌入的方法被提出。细节可参见例如 (Turian 等, 2010)。

---

## 28.4.3 使用深度自编码器的信息检索（语义哈希）

鉴于第 27.7.3.1 节讨论的 RBM 在信息检索中的成功，自然会想：深度模型能否做得更好？事实上可以，如图 28.6 所示。

更有趣的是，我们可以在深度自编码器的中间层使用二值的低维表示，而不是上面所用的连续表示。这使得相关文档的快速检索成为可能。例如，如果我们使用 20 比特码，可以为所有文档预先计算其二值表示，然后创建一个哈希表，将码字映射到文档。这种方法称为语义哈希 (semantic hashing)，因为语义相似的文档的二值表示在汉明距离上会很接近。

对于 Reuters RCV1-v2 中的 402,207 个测试文档，这在表中每个条目（哈希桶）大约对应 0.4 个文档。在测试时，我们计算查询的码字，然后只需在内存中查找相应地址的内容，就能以常数时间检索相关文档。为了找到更多相关文档，我们可以计算与该码字汉明距离不超过（比如）4 的所有码字。这样会检索到大约  $6196 \times 0.4 \approx 2500$  个文档。关键在于：总时间与语料库大小无关。

当然，还有其它快速文档检索技术，比如倒排索引。这些方法依赖于单个词具有相当的信息量，因此我们可以简单地对包含每个词的文档集合做求交。然而，当进行图像检索时，很明显我们不想在像素级工作。最近 (Krizhevsky 和 Hinton, 2010) 表明，一个深度自编码器可以学习到一种良好的语义哈希函数，在 8000 万张微小图像数据集上优于先前技术 (Torralba 等, 2008; Weiss 等, 2008)。而把倒排索引应用到实值数据上是困难的（尽管可以设想对图像块进行矢量量化）。

---

## 28.4.4 使用一维卷积 DBN 学习音频特征

要将 DBN 应用于无限长的时间序列，就必须使用某种形式的参数共享。一种方式是使用卷积 DBN (Lee 等, 2009; Desjardins 和 Bengio, 2008)，它们使用卷积 RBM 作为基本单元。这些模型是第 16.5.1 节讨论的卷积神经网络的生成式版本。基本思想如图 28.7 所示。对每个组 (group) 来说，其隐藏激活向量是通过将输入向量与该组的滤波器（权重

向量或矩阵) 做卷积得到的。换言之，隐藏组内的每个结点都是输入中某个子集的加权组合。通过在输入上\*\*“滑动”该权重向量，我们可以计算出所有隐藏结点的激活。这让我们能够建模平移不变性\*\*，因为无论模式出现在输入向量的何处，我们都使用相同的权重。每个组都有其自身的滤波器，对应其自身的模式检测器。

更形式化地，对于二值的一维信号，我们可以如下定义卷积 RBM 中的全条件分布 (Lee 等, 2009)：

$$p(h_t^{(k)} = 1 \mid v) = \text{sigm}((w^{(k)} \otimes v)_t + b_t) \quad (28.6)$$

$$p(v_s = 1 \mid h) = \text{sigm}(\sum_k (w^{(k)} \otimes h^{(k)})_s + c_s) \quad (28.7)$$

其中  $w^{(k)}$  是第  $k$  个组的权重向量， $b_t$  与  $c_s$  为偏置项， $a \otimes b$  表示向量  $a$  与  $b$  的卷积。通常会在卷积层之外再加入一个最大池化 (max pooling) 层，它对滤波响应取局部最大值。这允许获得少量的平移不变性。它也会减小高层的尺寸，从而显著加速计算。在神经网络里定义这一点很简单，但要以一种既允许信息前向流动又允许反向流动的方式来定义它就更复杂一些。基本思想与 noisy-OR 条件概率分布 (第 10.2.3 节) 类似：我们在最大单元与其所取最大值的组成部分之间定义一种概率关系。细节见 (Lee 等, 2009)。不过要注意，自顶向下的生成过程会很困难，因为最大池化操作丢弃了大量信息。

(Lee 等, 2009) 将深度为 2 的一维卷积 DBN 应用于声音数据。当输入是语音信号时，该方法恢复出一种类似音素的表示。应用于音乐分类和说话人识别时，他们的方法优于使用标准特征 (如 MFCC) 的技术。(所有特征都输入到同一个判别式分类器中。)

在 (Seide 等, 2011) 中，一个深度神经网络被用来替代传统 HMM 中的 GMM。使用 DNN 显著提升了对会话语音识别的性能。在一次采访中，该项目的技术负责人表示：“从历史上看，在语音识别领域，能带来这种幅度改进的单项技术是非常少的”。^7

#### 28.4.5 使用二维卷积 DBN 学习图像特征

我们可以将卷积 DBN 从 1D 直接扩展到 2D (Lee 等, 2009)，如图 28.8 所示。对 Caltech 101 数据集中四类视觉对象 (汽车、摩托车、脸、飞机) 训练的三层系统的结果如图 28.9 所示。我们只展示第 2 层和第 3 层的结果，因为第 1 层学习到的 Gabor 样滤波器与稀疏编码 (图 13.21(b)) 所学的非常相似。我们看到，第 2 层学到一些类通用的视觉部件 (在类别之间共享)，而第 3 层似乎学到了看起来像祖母细胞 (grandmother cells) 的滤波器，即对个别对象类别，甚至在某些情况下对个别对象都很特异。

## 通俗而详尽的解释

### 1) 手写体分类 (DBN)

- **流程：**先无监督逐层训练 (RBM + CD)，再用**类别标签**训练顶层，最后up-down整网微调。
- **为什么这样做：**
  - 无监督预训练让网络先学“**怎么表示图像**”；
  - 再用**标签对齐到类别**；
  - 微调则**整体协调**。
- **结果：**测试错误率1.25%（当时在“置換不变”设定下很强）。置換不变=算法**不依赖像素的空间位置结构**，因此适用于非图像数据。
- **对比：**SVM (9 次核) 1.4%、1-NN 3.1%。DBN 的优势在于**表示学习能力强**。

## 2) 自编码器做可视化/特征

- **自编码器**把高维数据压到**低维瓶颈**，再重建回来。
- 2D **瓶颈**可以当作**可视化坐标**：相似文档在图上会靠近。即使没用标签，也能捕捉**语义结构**。
- **替代/补充**：还有诸如 LSA、词向量等方法，目标相近：把语义**连续化、低维化**。

## 3) 语义哈希 (极快检索)

- **思路：**让中间层输出**二值码**（比如 20 位）。
- **离线：**给每篇文档算出码字，放进**哈希表**（桶）。
- **在线查询：**算出查询的码字，O(1) 地取出同桶文档；再找**汉明距离  $\leq r$** 的邻居码字，得到更多候选。
- **关键优势：****时间与库大小无关**；适用于**实值/图像**这种不适合倒排索引的场景（像素级倒排不现实）。
- **效果：**在 RCV1-v2 上，每桶约 0.4 篇文档；以半径 4 搜索约 6196 个码字，返回约  $0.4 \times 6196 \approx 2500$  篇。对**8000 万微图像**，深度自编码器学到的哈希优于早期方法。

## 4) 一维卷积 DBN (音频)

- **为什么卷积：**时间序列很长，需要**参数共享**，对**平移**（时移）**不敏感**。
- **卷积 RBM：**
  - 隐层激活： $h^{(k)} = \text{sigm}(w^{(k)} \otimes v + b)$
  - 可见重建： $v = \text{sigm}(\sum_k w^{(k)} \otimes h^{(k)} + c)$
  - $\otimes$  = 卷积；等价于把同一个滤波器**在时间上滑动**。
- **最大池化：**对局部响应取**最大值**，提升**小范围平移不变性**、降维、加速。但会**丢信息**，因此**自顶向下生成**更难。

- **经验：**对语音能学出类似音素的单位；在**音乐分类/说话人识别**上优于MFCC等手工特征（相同分类器下）。把DNN换掉HMM里的GMM也带来**显著提升**（行业里罕见的大幅改进）。

## 5) 二维卷积 DBN (图像)

- 2D 扩展很直接：滤波器在**空间上卷积**。
  - **层次表征：**
    - 第1层学到类似 Gabor 的**边缘/纹理**滤波器；
    - 第2层学到**通用部件**（轮廓片段、形状组合），可跨类别共享；
    - 第3层出现类似“**祖母细胞**”的高专一滤波器，对特定类别/对象**强响应**。
  - **含义：**网络逐层从**低级**（边缘）到**高级**（部件→对象）自动**分层抽象**。
- 

## 要点总结

- **DBN 手写体分类：**无监督预训练 + 顶层监督 + up-down 微调，测试错率1.25%（当时置换不变设定下 SOTA 之一）。
- **深度自编码器：**学到**有用特征与低维可视化**，无需标签也能呈现语义结构。
- **语义哈希：**把中间表示二值化（如 20 位），可**哈希检索与汉明邻域扩展**，查询与库大小无关，在**文本/图像**上都有效。
- **一维卷积 DBN (音频)：**卷积 + 池化实现**参数共享与时移不变性**，在语音/音乐任务上**优于传统特征**；用 DNN 替 GMM 显著提升 HMM 系统。
- **二维卷积 DBN (图像)：**层级特征从**Gabor 边缘**到**通用部件**再到**类别专属单元**，体现**逐层抽象**的表示学习。

如果你愿意，我可以把“语义哈希”的检索流程画成一张小示意图，或给你一段简洁伪代码，帮你把概念和步骤串起来。

这一节主要在讲什么？我不太懂

已思考 12s

没事～我用“超白话速读卡”把 28.4 应用篇 讲清楚 ↗

## 一句话版

这节就是在**展示深度网络怎么落地**：用在手写体识别、可视化与特征学习、超快检索（语义哈希）、音频（一维卷积）、图像（二维卷积）上，各给出思路和代表性结果。

# 五个小例子在说啥

## 1. DBN 做手写数字分类 (MNIST)

- **怎么训：**先对前两层做**无监督贪婪预训练** (RBM+CD)，再用类别标签训练顶层，最后用 up-down 整体**微调**。
- **怎么判别：**自底向上前馈后，在顶层 RBM 上对每个类别算**自由能**，选最小的。
- **效果：**测试错率约 1.25% (当时在“**置换不变**”MNIST 设定下的最佳之一)。对比：SVM (九次多项式核) 1.4%，1-NN 3.1%。
- **意义：**说明“**先学表示→再微调**”这一套在当年非常管用。

## 2. 深度自编码器做可视化/特征

- 训练一个**瓶颈=2D** 的深度自编码器，让它**重建输入**。
- 得到的**二维嵌入**能把语义相近的文本拉近 (即使**没用标签**)，常作为**下游监督学习**的输入。
- 和传统 LSA、词向量思路类似，但**非线性**更强。

## 3. 信息检索：语义哈希 (用深度自编码器)

- 把中间层设成**二值码** (如 20 位)，离线为所有文档算码字并建**哈希表**。
- **查询时：**算出查询码字，O(1) 直接取同桶文档；再在**汉明距离**  $\leq r$  的邻居码字里扩展 (例：距 4 的邻域约 6196 个码字  $\times$  0.4 文档/码字  $\approx$  2500 篇)。
- **关键点：****查询时间与库大小无关**；对图像等**实值数据**也好用 (倒排索引不适合像素级)。在“**8000 万微图像**”上，深度自编码器学的哈希优于早期方法。

## 4. 一维卷积 DBN 学音频特征

- 为了应对**长时间序列**，用**卷积 RBM** (参数共享) + **最大池化** (小范围时移不变性)。
- **形式** (理解即可)：
  - 隐层激活  $h^{(k)} = \text{sigm}(w^{(k)} \otimes v + b)$ ，可见层重建  $v = \text{sigm}(\sum_k w^{(k)} \otimes h^{(k)} + c)$ 。
  - **效果：**能学到**类似音素**的表示；在**音乐分类/说话人识别**上优于用 MFCC 的传统方案。同一时期把 DNN 换进 HMM (替掉 GMM) 也让**会话语音识别**显著提升。

## 5. 二维卷积 DBN 学图像特征

- 1D 卷积自然扩展到 2D。在 Caltech101 (车、摩托、脸、飞机) 上：
  - **第1层**学到类似 Gabor 的边缘/纹理滤波器；
  - **第2层**学到可跨类复用的**通用部件**；
  - **第3层**出现类似“**祖母细胞**”的**类别/对象专属滤波器**。
- **意义：**清晰展示了**从低级→中级→高级的分层抽象**。

# 比喻快速记

- 语义哈希像给每篇文档做“**20位书脊条码**”，同类书条码相近，查找时直接按条码抽屉拿书。
- 卷积+池化**像用同一把“**花纹模板**”在时间/空间上**来回盖章**，不管出现在哪里都能认出来。

## 要点总结

- 本节用**五个应用**说明：深度网络不仅能分类，还能**学表示、可视化、极快检索**，并在**音频/图像**等模态中通过**卷积+池化**获得**平移不变与参数共享**。
- DBN + 预训练 + 微调**在当年 MNIST（置换不变）上达到了**很低错率**，验证了**无监督表示学习**的价值。
- 深度自编码器**既能做**非线性降维**，又能产出**二值语义码**，让大规模检索实现**近常数时间**。
- 卷积 DBN (1D/2D)**学到的层级特征与人类直觉一致：**边缘→部件→对象/类别**；在**语音与图像任务**上有实际收益。

翻译并解释以下内容，翻译不能有遗漏，解释请通俗易懂且详尽，因为我的数学基础不太好，最后总结要点。

### 28.5 Discussion

So far, we have been discussing models inspired by low-level processing in the brain. These models have produced useful features for simple classification tasks. But can this pure bottom-up approach scale to more challenging problems, such as scene interpretation or natural language understanding?

To put the problem in perspective, consider the DBN for handwritten digit classification in Figure 28.4(a). This has about 1.6M free parameters ( $28 \times 28 \times 500 + 500 \times 500 + 510 \times 2000 = 1,662,000$ ).

Although this is a lot, it is tiny compared to the number of neurons in the brain. As Hinton says,

This is about as many parameters as 0.002 cubic millimetres of mouse cortex, and several hundred networks of this complexity could fit within a single voxel of a high-resolution fMRI scan. This suggests that much

bigger networks may be required to compete with human shape recognition abilities. — (Hinton et al. 2006, p1547). To scale up to more challenging problems, various groups are using GPUs (see e.g., (Raina et al. 2009)) and/or parallel computing. But perhaps a more efficient approach is to work at a higher level of abstraction, where inference is done in the space of objects or their parts, rather than in the space of bits and pixels. That is, we want to bridge the signal-to-symbol divide, where by "symbol" we mean something atomic, that can be combined with other symbols in a compositional way.

The question of how to convert low level signals into a more structured/ "semantic" representation is known as the symbol grounding problem (Harnard 1990).

Traditionally such symbols are associated with words in natural language, but it seems unlikely we can jump directly from low-level signals to high-level semantic concepts. Instead, what we need is an intermediate level of symbolic or atomic parts.

A very simple way to create such parts from real-valued signals, such as images, is to apply vector quantization. This generates a set of visual words. These can then be modelled using some of the techniques from Chapter 27 for modeling bags of words. Such models, however, are still quite "shallow".

It is possible to define, and learn, deep models which use discrete latent parts. Here we just mention a few recent approaches, to give a flavor of the possibilities.

(Salakhutdinov et al. 2011) combine RBMs with hierarchical latent Dirichlet allocation methods, trained in an unsupervised way. (Zhu et al. 2010) use latent and-or graphs, trained in a manner similar to a latent structural SVM. A similar approach, based on grammars, is described in (Girshick et al. 2011). What is interesting about these techniques is that they apply data-driven machine learning methods to rich structured/symbolic "AI-style" models. This seems like a promising future direction for machine learning.

已思考 10s

下面先逐段精确翻译（不遗漏），再做通俗详尽解释，最后给出要点总结。

## 逐段精确翻译

### 28.5 讨论 (Discussion)

到目前为止，我们一直在讨论受大脑低层处理启发的模型。这些模型为简单的分类任务产生了有用的特征。**但是，这种纯粹自底向上的方法能否扩展到更具挑战性的问题，例如场景理解或自然语言理解？**

为了让问题有个参照系，考虑图 28.4(a) 中用于手写数字分类的 DBN。它大约有 160 万个自由参数 ( $28 \times 28 \times 500 + 500 \times 500 + 510 \times 2000 = 1,662,000$ )。虽然这很多，但与大脑中的神经元数量相比却微不足道。正如 Hinton 所说：

“这大约等同于0.002 立方毫米的小鼠皮层中的参数量，而数百个这种复杂度的网络都能装进一次高分辨率 fMRI 扫描的单个体素里。这暗示要与人类的形状识别能力竞争，可能需要更大的网络。”—— (Hinton 等, 2006, 第 1547 页)

为了扩展到更具挑战性的问题，各个研究组正在使用 GPU (例如参见 Raina 等, 2009) 和/或并行计算。**但也许更高效的方法是在更高的抽象层次上工作，即在对象或其部件的空间中进行推断，而不是在比特和像素的空间中进行推断。**也就是说，我们想要跨越“信号到符号”的鸿沟；这里的“符号”指的是可组合、可以与其它符号以组合方式结合的原子化事物。

**如何把低层信号转化为更有结构/“语义”的表示**这一问题被称为**符号奠基问题 (symbol grounding problem)** (Harnard, 1990)。传统上，这样的符号与自然语言中的词相关联，但我们不太可能直接从低层信号跳到高层语义概念。相反，我们需要一个**中间层次的符号化或原子化部件**。

从**实值信号** (如图像) 中创建这种部件的一种非常简单的方法是应用**矢量量化 (vector quantization, VQ)**。这会生成一组**视觉词 (visual words)**。之后可以用第 27 章中用于建模\*\*词袋 (bag of words) \*\*的一些技术进行建模。然而，这类模型仍然相当“浅”。

可以**定义并学习使用离散潜在部件的深层模型**。这里我们仅提到一些近期方法，给出可能性的一个感觉。 (Salakhutdinov 等, 2011) 将 RBM 与**分层潜在狄利克雷分配 (hierarchical LDA) 方法结合，并以无监督方式训练。** (Zhu 等, 2010) 使用**潜在的 and-or 图**，其训练方式类似于**潜在结构 SVM**。一种类似的方法，基于**文法 (grammars)**，见 (Girshick 等, 2011)。这些技术有趣之处在于：它们将**数据驱动的机器学习方法应用到丰富的、结构化/符号化的“AI 风格”模型上**。这似乎是机器学习一个很有前景的未来方向。

---

## 通俗而详尽的解释

## 1) 纯“自下而上”有多远的边界?

- 我们前面讲的深度特征学习，多数是在**像素/比特层面**“往上堆”，确实能把**简单分类**做好。
- 但要搞定**场景理解、自然语言理解**这类高层任务，仅靠**像素级别的逐层堆叠**可能不够。

## 2) 参数规模 vs. 大脑规模 (算一笔)

- 文中那个 MNIST 的 DBN 规模：
  - $28 \times 28 = 784; 784 \times 500 = 392,000$
  - $500 \times 500 = 250,000$
  - $510 \times 2000 = 1,020,000$
  - 总计  $392,000 + 250,000 + 1,020,000 = 1,662,000$  (约 166 万)
- Hinton 的意思：这点参数量在生物大脑里只是**皮毛**；要接近人类识别能力，**网络可能要大得多**。

## 3) 两条“变强”路线

- **硬件扩容**：用 GPU/并行把同样的像素级模型做得更大、更深。
- **抽象跃迁**（作者更看好）：把推理搬到**更高层的单位上——对象 & 部件**的层面，而不是像素。
  - 这就是**跨越“信号→符号”鸿沟**：让模型操纵\*\*“能组合的原子单元”（**符号**），像搭句子\*\*一样组合出复杂含义。

## 4) 符号奠基问题 (symbol grounding)

- **问题**：怎样把**连续、嘈杂的低层信号**（像素、波形）转成**离散、可组合的符号**（如“轮子”“车门”“猫耳朵”，再到“猫在垫子上”）？
- **不现实的跳跃**：直接从像素 → “猫坐垫子上”**太远**。
- **可行的路线**：先学出**中间层的“部件型”符号**（可复用的小块），再逐步组成高层语义。

## 5) 一个简单的中间层：视觉词汇 ( $VQ \rightarrow$ visual words)

- **矢量量化** ( $VQ$ )：把相似的局部特征聚成**离散码本**，每个码字=一个“**视觉词**”。
- **词袋** (BoW)：统计图片里有哪些视觉词、各出现多少。
- **局限**：BoW 不看结构和关系（谁在谁上面、部件如何拼），所以被称为比较**浅**。

## 6) 更“深”的离散部件模型

- **RBM + 层次 LDA**（无监督）：同时学“**部件主题**”和它们的层次组织。
- **潜在 and-or 图**（像“与/或”语法树）：

- AND 组合：一个整体由若干部件**同时存在**；
- OR 选择：某个部件可能有**不同外观/姿态的替代**。
- 训练类似**潜在结构 SVM**（把未观测的部件当潜变量）。
- **文法式方法**：用语法规则来表达“**部件如何组成对象**”，再让数据驱动去学参数/结构。
- **共同点**：让**数据驱动与符号结构结合**——既保留**学习**（从数据中自动发现规律），又引入**可组合的结构**（便于表达复杂关系）。

## 7) 为什么这条路有前景？

- **鲁棒性/可组合性**：符号化部件可在新场景**重用与组合**，更利于**迁移与可解释**。
  - **数据效率**：高层结构能减少对纯粹“堆参数”的依赖，不必一味靠巨量数据和算力“硬推”。
  - **贴合认知**：与人类“**从部件到整体**”的理解方式更接近。
- 

## 要点总结

- 仅靠**自底向上的低层特征**，在**简单分类**上有效，但面对**场景/语言理解**等高层任务可能**不够**。
- 让模型变强有两路：
  1. **做更大** (GPU/并行)；
  2. **做更抽象** (在**对象/部件/符号**层面推理，跨越“信号→符号”鸿沟）。
- **符号奠基**：把连续信号变成可组合的离散**符号/部件**，不应从像素直接跳语义，需**中间层**。
- VQ → **视觉词** → **词袋**是简便的中间层，但较浅，缺结构关系。
- **更深的离散部件模型**：如RBM+**层次 LDA**、**潜在 and-or 图**、**文法式方法**；它们把**数据驱动学习与符号化结构结合**，是**有前景**的方向。

如果你想，我可以画一张“信号→部件→对象→语义”的层级小图，或给你一个 and-or 图的简易例子，帮你把“可组合符号”这件事更直观地记住。

基于本对话里我贴过的整章原文，输出：  
 ①知识点总结  
 (分点+关键词+用途) ②子章节关联 (依赖/对比/递进+关系句，附Mermaid可选) ③本章总结。

已思考 16s

好嘞！我把整章浓缩成一份**考试/面试友好型速读包**。分三部分给你：①知识点总结（分点+关键词+用途）②子章节关联（依赖/对比/递进+关系句，附 Mermaid）③本章总结。

---

## ① 知识点总结（分点 + 关键词 + 用途）

### 28.1 引言（大脑启发与分层表征）

- **核心：**从边缘→图块→表面→物体的多层抽象
    - **关键词：**分层、抽象、表征学习、视觉皮层
    - **用途：**为“深”模型正名——用多层结构自动学特征，而非手工造特征。
- 

### 28.2 深层生成模型（无监督为先）

#### 28.2.1 深层有向网络 DDN / Sigmoid Belief Net

- **核心：**有向图，层层生成；二值结点 + logistic CPD
  - **关键词：**有向、解释互斥（explaining away）、Mean Field、MCMC
  - **用途：**建模“因果式”生成过程；用于无监督特征学习与生成。

#### 28.2.2 深层玻尔兹曼机 DBM

- **核心：**无向能量模型；层间对称连接；可分块 Gibbs/Mean Field
  - **关键词：**无向、能量函数、配分函数  $Z$ 、按层采样
  - **用途：**高效按层推断的深层生成模型；支持自顶向下与自底向上信息交互。

#### 28.2.3 深度置信网络 DBN（顶层无向 + 底层有向）

- **核心：**顶层 RBM 当联想记忆/互补先验；底层有向生成
  - **关键词：**RBM 等价性、互补先验、因子化后验、近似自底向上推断
  - **用途：**快速推断 + 无监督预训练，兼顾生成与判别使用。

#### 28.2.4 DBN 的逐层贪婪学习

- **核心：**RBM→展开→冻结→下一层，层层叠加，最后up-down微调
    - **关键词：**贪婪层-wise、预训练、变分下界提升、上-下过程（up-down）
    - **用途：**让很深的生成/混合模型可训练，并提供良好初始化。
-

## 28.3 深度神经网络（直接判别式训练）

### 28.3.1 深层 MLP（反向传播）

- 核心：分类为主；早期受梯度消失/平台困扰，GPU+二阶优化改进
  - 关键词：反向传播、梯度消失、平台、二阶优化
  - 用途：端到端判别；作为预训练后的微调主力。

### 28.3.2 深度自编码器（AE/DAE）

- 核心：输入→瓶颈→重建；线性 AE  $\approx$  PCA；深 AE 需预训练+微调
  - 关键词：降维、重建误差、瓶颈、非线性表征、RBM 预训练
  - 用途：无监督特征学习、可视化、作为下游任务输入。

### 28.3.3 堆叠去噪自编码器（SDAE）

- 核心：加噪训练以防恒等映射；可过完备；与 RBM 的 score matching 有关
    - 关键词：去噪、稀疏性、过完备、汉明鲁棒、score matching（近似 MLE）
    - 用途：学鲁棒表征；堆叠后可判别式微调。
- 

## 28.4 应用（跨视觉/文本/音频）

### 28.4.1 DBN 做 MNIST（置换不变）

- 核心：无监督预训练 + 顶层监督 + up-down 微调  $\rightarrow$  1.25% 错误率
  - 关键词：CD、自由能、置换不变
  - 用途：验证预训练→微调范式的有效性。

### 28.4.2 自编码器做可视化/特征

- 核心：2D 瓶颈嵌入能呈现语义结构
  - 关键词：低维嵌入、LSA 对比、无标签语义
  - 用途：文档/样本可视化；下游监督学习更好起点。

### 28.4.3 语义哈希（基于深 AE）

- 核心：中间层二值码（如 20 位）；哈希表 + 汉明邻域检索
  - 关键词：二值嵌入、哈希、汉明距离、常数时间
  - 用途：超快召回文本/图像；查询时间与库规模解耦。

### 28.4.4 1D 卷积 DBN（音频）

- 核心：卷积 RBM + 池化（参数共享、时移不变）；胜过 MFCC

- 关键词：卷积、最大池化、音素样表征、DNN+HMM
- 用途：语音/音乐表征学习与识别。

### 28.4.5 2D 卷积 DBN (图像)

- **核心：**层 1 Gabor、层 2 通用部件、层 3 类/实例专属 (“祖母细胞”)
    - 关键词：分层特征、部件-整体
    - 用途：图像分层抽象与识别。
- 

## 28.5 讨论 (走向符号与结构)

- **核心：**仅自底向上在像素/比特层面**不易扩展**到场景/语言；需**更大网络或更高抽象**(对象/部件/符号)
    - 关键词：符号奠基 (symbol grounding) 、VQ→视觉词→词袋、and-or 图、层次LDA、语法模型
    - 用途：把**数据驱动与符号结构结合**，通往**可组合、可迁移的高层理解**。
- 

## ② 子章节关联 (依赖 / 对比 / 递进)

**关系句 (精炼版) :**

- 28.1 → 28.2：由“**大脑分层**”动机，先走**无监督生成**路线 (DDN/DBM/DBN)，解决**标签稀缺与特征学习**。
- 28.2 ↔ 28.3：从**生成过渡到判别**——DBN常“前馈用”，于是**直接训练深网**，同时用**生成式预训练**缓解训练难点。
- 28.3 → 28.4：把方法**落地应用**：分类、可视化、检索、音频/图像卷积。
- 28.4 → 28.5：从应用反思**可扩展性**：仅像素级自底向上不足，提出**符号层与结构化模型**的方向。
- 28.2.2 (DBM) vs 28.2.3 (DBN) : DBM支持**高效上下行推断**但训练难 (Z) ; DBN易于**快速自底推断/预训练**但顶向下不便。
- 28.3.2 (AE) vs PCA: **线性** AE≈PCA; **非线性/深度** AE学更强表征，需**预训练+微调**。
- 28.3.3 (SDAE) vs 标准 AE: SDAE 通过**加噪/稀疏/过完备防恒等**，更**鲁棒**。

**Mermaid 关系图 (可选) :**

```
graph TD
    A[28.1 引言: 大脑分层表征] --> B[28.2 深层生成: DDN/DBM/DBN]
    B --> C[28.2.4 贪婪层级预训练: RBM堆叠]
    C --> D[28.3 深度神经网络: 直接判别训练]
    D --> E[28.3.2/3 自编码器/去噪AE: 预训练与鲁棒表征]
    E --> F[28.4 应用: 分类/可视化/检索/音频/图像]
    F --> G[28.5 讨论: 向符号与结构化迈进]
    subgraph 对比 [对比]
        B1[DBM: 上下行推断]
    end
```

强, Z难] --- B2[DBN: 快速自底推断, 预训练友好] D1[AE(线性~PCA)] --- D2[SDAE: 加噪/过完备/鲁棒] end

### ③ 本章总结（一屏带走）

- **主线**: 从大脑**分层处理**得到启发, 深度学习用**多层**结构实现**表征学习**。当**标注稀缺**时, 先走**无监督生成** (RBM/DBN/DBM) 与**贪婪层级预训练**, 再**判别式微调**。
- **两条路径**:
  1. **生成式路径** (DDN/DBM/DBN) : 擅长**无监督学表示**与 (在 DBM 中) **上下行推断**;
  2. **判别式路径** (深 MLP/AE/SDAE) : 端到端**任务性能**, 以\*\***预训练** (AE/RBM) \*\*缓解训练难点。
- **落地能力**: 在MNIST分类、语义嵌入/可视化、语义哈希检索、音频/图像卷积等任务上展示**强实绩**, 证明“**先学表示—再微调**”的价值。
- **现实挑战**: 仅在像素/比特层面**自底向上**, 难以直接扩展到**场景理解/自然语言理解**。需要**更大规模**与 (更重要的) **更高层的可组合结构 (符号/部件/语法)**。
- **前瞻方向**: 把**数据驱动学习与结构化/符号化模型结合** (如and-or图、**层次 LDA**、**语法**), 跨越**信号→符号鸿沟**, 获得更**可迁移、可解释**的智能。

记忆钩子: “**无监督打地基, 判别式盖高楼; 应用拉验收, 最后奔符号。**”

请用通俗易懂的话总结本章, 使我能够理解

已思考 14s

好的! 我用尽量“接地气”的方式, 把整章讲成一段好懂的故事。

### 这章在说啥 (一句话)

大脑是分层处理信息的: 从**简单特征到复杂概念**。深度学习就是用多层模型去模仿这点。标签不够时, 先用**无监督**把好特征学出来, 再用少量标签**微调**完成任务; 在图像、文本、音频里都能见到成效。往前走, 还需要把“像素级信号”升级为能组合的“**符号/部件**”, 才能搞定更复杂的理解任务。

### 展开讲 (用类比帮你记)

## 1) 为啥要“深”

像搭积木：先学**边缘**，再拼**小块**，再成**表面**，最后识别**物体**。多层=每层学更抽象的东西，比一把梭直接从像素猜类别更靠谱。

## 2) 标签不够怎么办？——用“生成模型”先打地基

- **有向的 DBN**：像“食谱一步步做菜”，但不同原因会互相“抢功劳”（explaining away），推断难、慢。
- **无向的 DBM**：像“大家一锅炖”的能量模型，好处是可以**按层并行推断**，但训练要算配分函数，难。
- **折中的 DBN**：顶层用 RBM 做“**联想记忆/先验**”，下面是有向生成。这样**自底向上推断快**，又方便**逐层预训练**。

**训练窍门：**逐层贪婪（一层一个 RBM 先学好，再往上叠），最后再来一次上-下（up-down）微调，把整网调顺。

## 3) 直接上深度神经网络也行，但要“热身”

直接用深层 MLP 做分类会遇到**梯度消失、误差平台**，训练不稳。解决办法：先做**无监督预训练**（让网络**重建输入**），再用**反向传播监督微调**。

- **自编码器 (AE)**：输入→瓶颈→重建；线性 AE≈PCA，非线性 AE 能学更强表征。
- **深度 AE**：直接训难，配合**逐层 RBM 预训练**就顺多了。
- **去噪 AE (SDAE)**：给输入**加噪/挖空**让它学会**补全**，既能**过完备**又不学“**抄作业**”的恒等映射，还更鲁棒。

## 4) 能做什么？（几个拿得出手的应用）

- **手写体 (MNIST)**：DBN 先无监督预训练，再监督+微调，做到当时置換不变设定下约 1.25% 错误率。
- **可视化/特征**：深度 AE 把文本压到2D 也能显出语义结构，常当下游监督的好特征。
- **语义哈希**：把中间表示变成**20位二值码**，像给文档打**条形码**；查相似就按条码表秒查，复杂度几乎**与库大小无关**；图像上也好用。
- **音频 (1D 卷积 DBN)**：卷积+池化=参数共享和时移不变，能学出类似**音素**的单位，音乐/说话人识别优于传统 MFCC。
- **图像 (2D 卷积 DBN)**：第1层像**边缘/Gabor**，第2层学**通用部件**，第3层学到**类别/实例专属**的“祖母细胞”式单元——体现**逐层抽象**。

## 5) 往哪儿走？——从“信号”到“符号”

仅靠像素级**自底向上**要想理解**场景或语言**还不够；要么**更大网络**，要么（更关键）**提高抽象层次**：在**对象、部件、符号**的空间里推理。

- 简单做法：向量量化→“视觉词”→词袋（简单但浅，缺结构）。
  - 更有希望：把数据驱动和结构化模型结合，例如 RBM+层次 LDA、潜在 And-Or 图、语法/文法式模型——用可组合的部件表达复杂关系，贴近人类理解。
- 

## 记忆小抄（5 句压缩版）

1. 多层=逐层抽象：边缘→部件→物体→语义。
2. 无监督打地基：RBM/DBN/DBM 先学表示，再微调做任务。
3. 直接深网要热身：先自编码器/去噪预训练，再反传更稳。
4. 卷积拿不变性：音频用 1D，图像用 2D，卷积+池化=共享+平移不敏感。
5. 走向符号化：从像素到可组合的部件/符号，才能攻克更复杂的理解。

如果你想把这些概念“落地”，我可以给你一个最小实践路线（比如：用去噪自编码器在小数据上做特征→用简单分类器验证→再试语义哈希），一步步做一遍就更清楚了。