个不相交的子集，每个子集有 m/k 个实例。然后，运行 k 次交叉验证过程，每一次使用不同的子集作为验证集合，并合并其他的子集作为训练集合。于是，每一个样例会在一次实验中被用作验证集合的成员，在 k-1 次实验中用作训练集合的成员。在每次试验中，都使用上面讨论的交叉验证过程，来决定在验证集合上取得最佳性能的迭代次数 i。然后计算这些 i 的均值 $\bar{i}$，最后运行一次反向传播算法，训练所有 m 个[①]实例并迭代 $\bar{i}$ 次，此时没有验证集合。这个过程与第 5 章描述的基于有限数据比较两种学习方法的过程很相近。

4.7 示例：人脸识别

为了说明反向传播算法应用中的一些实际的设计问题，这一节讨论把这个算法应用到人脸识别的学习任务。这一节用来产生这个例子的所有图像数据和代码都可以从以下网址得到：http://www.cs.cmu.edu//~tom/mlbook.html，同时还有如何使用这些代码的完整文档。读者可以自己进行试验。

4.7.1 任务

这里的学习任务是分类不同人的不同姿态的摄影图像。我们收集了 20 个不同的人的摄影图像，每个人大约有 32 张图像，对应这个人不同的表情（快乐，沮丧，愤怒，中性）；他们看的不同方向（左，右，正前，上）；和他们是否戴太阳镜。从图 4-10 的示例图像中可以看到，人后面的背景、穿的衣服、和人脸在图像中的位置也都有差异。我们共收集了 624 幅灰度图像，每一幅的分辨率为 120×128，图像的每个像素使用 0（黑色）到 255（白色）的灰度值描述。

从这些图像数据中可以学习很多不同的目标函数。例如，我们可以训练一个 ANN，使给定一幅图像输入时输出这个人的惟一标识（identity）、脸的朝向、性别、是否带太阳镜等。所有这些目标函数可以以很高的精度从这些数据中学习到，鼓励读者们自行试验。在本节后面的部分，我们考虑一个特定的任务：学习图像中人脸的朝向（左，右，正前，还是上）。

插图——原书页码：113

30×32resolution input images- 30×32 分辨率的输入图像

Network weights after 1 iteration through each training example- 对每个训练样例迭代 1 次后的网络权值

Network weights after 100 iteration through each training example- 对每个训练样例迭代 100 次后的网络权值

left: 左

straight: 前

---

① 译注：原书此处误为 n

right:　　右

up:　　上

图 4-10 学习识别人脸朝向的人工神经网络

这里使用人脸的灰度图像（见最上一行）训练一个 960×3×4 的网络，来预测一个人是在向左、向右、向前还是向上看。在使用了 260 幅这样的图像训练后，这个网络对于独立的验证集合达到了 90%的精度。图中也显示了使用训练样例迭代 1 次后和迭代 100 次后的网络权值。每个输出单元（左，前，右，上）有四个权值，用暗（负）和明（正）的方块显示。最左侧的方块对应权 w_0，它决定单元的阈值，右面的三个方块对应从三个隐藏单元输入的权。图中也显示了每个像素输入到每个隐藏单元的权值，被画在对应像素的位置。

4.7.2 设计要素

应用反向传播算法到一个给定任务时，必须决定几个设计要素。下面我们归纳出了学习人脸朝向这个学习任务的一些设计要素。尽管我们没有打算去选择精确的最优设计，但这里描述的设计对目标函数学习得相当好。在训练了 260 幅图像样例之后，对于独立测试集合的精度达到 90%。相对而言，如果随机猜测四个脸朝向中的一个，只能达到 25%的正确率。

输入编码。已经知道 ANN 的输入必然是图像的某种表示，那么设计的关键是如何编码这幅图像。例如我们可以对图像进行预处理，来分解出边缘、亮度一致的区域或其他局部图像特征，然后把这些特征输入网络。这种设计的一个问题是会导致每幅图像有不同数量的特征参数（例如边缘的数量），然而 ANN 具有固定数量的输入单元。对于这种情况，我们的设计是把图像编码成固定的 30×32 像素的亮度值，每个像素对应一个网络输入。并且把范围是 0 到 255 的亮度值按比例线性缩放到 0 到 1 的区间内，以使网络输入与隐单元和输出单元在同样的区间取值。实际上这里的 30×32 像素图像就是原来 120×128 像素的图像的低分辨率概括，每个低分辨率像素根据对应的若干高分辨率像素亮度的均值计算得到。使用这样的低分辨率图像，把输入个数和权值的数量减少到了一个更易于处理的规模，从而降低了运算要求，但同时也保留了足够的分辨率以正确分类图像。回忆图 4-1 中 ALVINN 系统使用了相似的的分辨率图像作为网络的输入。一个有趣的差别是，在 ALVINN 中，每一个低分辨率像素的亮度等于从高分辨率图像对应的区域中随机取一个像素的亮度，而不是取这个区域中所有像素亮度的均值。其动机是为了明显地减少从高分辨率图像产生低分辨率图像所需的运算。这个效率对于 ALVINN 系统是特别重要的，因为在自动驾驶车辆的过程中，ALVINN 系统的网络必须在每秒钟处理很多幅图像。

输出编码。ANN 必须输出四个值中的一个来表示输入图像中人脸的朝向（左，右，上，前）。注意我们可以使用单一的输出单元来编码这四种情况的分类，例如指定输出值 0.2，0.4，0.6 和 0.8 来编码这四个可能值。不过这里我们使用 4 个不同的输出单元，每一个对应四种可能朝向中的一种，取具有最高值的输出作为网络的预测值。这种方法经常被称为 *n* 取 1（1-of-*n*）输出编码。选择 *n* 取 1 输出编码而不用单个单元有两个动机。第一，这为网络表示目标函数提供了更大的自由度（即在输出层单元中有 *n* 倍的可用权值）。第二，在 *n* 取 1 编码中，最高值输出和次高值输出间的差异可以作为对网络预测的置信度（不明确的分类可能导致结果相近或相等）。进一步的设计问题是“ 这 4 个输出单元的目标值应该是什么？ ”

一个显而易见的办法是用 4 个目标值<1，0，0，0>来编码脸朝向左，<0，1，0，0>来编码脸朝向正前，依此类推。我们这里使用 0.1 和 0.9，而不是 0 和 1，即<0.9，0.1，0.1，0.1>表示脸朝向左的目标输出向量。避免使用 0 和 1 作为目标值的原因是 sigmoid 单元对于有限权值不能产生这样的输出。如果我们企图训练网络来准确匹配目标值 0 和 1，梯度下降将会迫使权值无界增长。另一方面，值 0.1 和 0.9 是 sigmoid 单元在有限权值情况下可以完成的。

网络结构图。正如前面所描述的，反向传播算法可以被应用到任何有向无环 sigmoid 单元的网络。所以，我们面临的另一设计问题是，这个网络包含多少个单元以及如何互连。最普遍的一种网络结构是分层网络，一层的每个单元向前连接到下一层的每一个单元。目前的设计选择这样的标准结构，使用两层 sigmoid 单元（一个隐藏层和一个输出层）。使用一或两层 sigmoid 单元是很普遍的，偶尔使用三层。使用更多的层是不常见的，因为训练时间会变得很长，而且三层 sigmoid 单元的网络已经能够表示数量相当大的目标函数（见 4.6.2 节）。我们已经确定选择一个分层的前馈网络，那么其中应该包含多少个隐藏单元呢？在图 4-10 报告的结果中，仅使用了三个隐藏单元，达到了对测试集合 90%的精度。在另一个使用 30 个隐藏单元的实验中，得到的精度提高了一到两个百分点。尽管这两个实验得到的泛化精度相差很小，但后一个试验明显需要更多的训练时间。使用 260 幅图像的训练样例，30 个隐单元的网络在 Sun Sparc5 工作站上的训练时间大约是一个小时。相对而言，三个隐藏单元的网络大约是 5 分钟。人们已经发现在很多应用中需要某个最小数量的隐单元来精确地学习目标函数，并且超过这个数量的多余的隐单元不会显著地提高泛化精度，条件是使用交叉验证方法来决定应该进行多少次梯度下降迭代。如果没有使用交叉验证，那么增加隐藏单元数量经常会增加过度拟合训练数据的倾向，从而降低泛化精度。

学习算法的其他参数。在这个实验中，学习速率η被设定为 0.3，冲量α被设定为 0.3。赋予这两个参数更低的值会产生大体相当的泛化精度，但需要更长的训练时间。如果这两个值被设定得太高，训练将不能收敛到一个具有可接受误差（在训练集合上）的网络。在整个试验中我们使用完全的梯度下降（和表 4-2 算法中随机近似的梯度下降不同）。输出单元的网络权值被初始化为小的随机值。然而输入单元的权值被初始化为 0，因为这样可以使学习到的权值的图像化（见图 4-10）更易于理解，而对泛化精度没有明显的影响。训练的迭代次数的选择可以通过分割可用的数据为训练集合和独立的验证集合。梯度下降方法被用于最小化训练集合上的误差，并且每隔 50 次梯度下降迭代根据验证集合评估一次网络的性能。最终选择的网络是对验证集合精度最高的网络。可以参见 4.6.5 节得到关于这个过程的解释和依据。最终报告的精度（也就是 90%，对于图 4-10 中的网络）是在没有对训练产生任何影响的第三个集合——测试集合上测量得到的。

4.7.3 学习到的隐藏层表示

有必要分析一下网络中学习得到的 2899 个[①]权值。图 4-10 描绘了对所有训练样例进行一次权值更新后的每个权值，和 100 次更新后的权值。

为了理解这些图像，先考虑图中紧邻人脸图像下的四个矩形。每一个矩形描绘了网络中四个输出单元（编码了左、前、右和上）中的一个权值。每个矩形中的四个小方形表示和这

[①] 译注：2899=输入单元与三个隐单元间连接对应的权（960×3）+三个隐单元与四个输出单元间连接对应的权（3×4）+三个隐单元和四个输出单元的 w_0 权（3+4）

个输出单元关联的四个权值——最左边是权 w_0，它决定单元的阈值，然后是连接三个隐藏单元到这个输出的三个权值。方形的亮度表示权值，亮白表示较大的正权值，暗黑表示较大的负权值，介于中间的灰色阴影表示中等的权值。例如，标为“上”的输出单元的阈值权 w_0 接近 0，从第一个隐藏单元来的权值为较大的正值，从第二个隐藏单元来的权值为较大的负值。

隐藏单元的权值显示在输出单元的下边。回忆一下，每个隐藏单元接受所有 30×32 个像素输入。与这些输入关联的 30×32 个权值被显示在它们对应的像素的位置（阈值权 w_0 被重叠显示在图像阵列的左上角）。非常有趣的是，可以看到权的取值通常对人脸和身体出现的图像区域的特别敏感。

针对每一个训练样例梯度下降迭代 100 次后的网络权值显示在图的下部。注意最左边的隐藏单元的权值和迭代一次时的权值有很大不同，另两个隐藏单元的权值也有所变化。现在可以分析一下这个最终权值集合中的编码。例如，考虑输出单元指出一个人是在向右看。这个单元与第二个隐藏单元间具有一个较大的正权值，与第三个隐单元间具有一个大的负权值。分析这两个隐单元的权值，容易看到如果一个人的脸是转向他的右面（也就是我们的左面），那么他的亮度高的皮肤会大致与这个隐藏单元中的较大正值对齐，同时他的亮度低的头发会大致与负权值对齐，这导致此单元输出一个较大的值。同样的图像会使第三个隐单元输出一个接近 0 的值，因为亮度高的脸部倾向于与大的负权对齐。

4.8 人工神经网络的高级话题

4.8.1 其他可选的误差函数

正如前面所指出的，只要函数 E 相对参数化的假设空间可微，那么就可以执行梯度下降。虽然基本的反向传播算法以网络误差平方和的形式定义 E，但也有人提出其他的定义，以便把其他的约束引入权值调整法则。如果定义了一个新的 E，那么就必须推导出一个新的权值调整法则供梯度下降使用。E 的其他可选定义包括：

- 为权值增加一个惩罚项。如同前面讨论的，我们可以加入一个随着向量幅度增长的项到 E 中。这导致梯度下降搜寻较小的权值向量，从而减小过度拟合的风险。一种办法是按照下面的等式重新定义 E：

$$E(\vec{w}) \equiv \frac{1}{2}\sum_{d\in D}\sum_{k\in outputs}(t_{kd}-o_{kd})^2+\gamma\sum_{i,j}w_{ji}^2$$

  这得到了一个与反向传播法则基本一致的权更新法则，只是在每次迭代时为每个权乘以常量 $(1-2\gamma\eta)$。因此选择这种 E 的定义和使用权衰减策略（见练习 4.10）是等价的。
- 对误差增加一项目标函数的斜率（slope）或导数。某些情况下，训练信息中不仅有目标值，而且还有关于目标函数的导数。例如，Simard et al.（1992）描述了一个字符识别的应用，在这个应用中使用了一些训练导数来强迫网络学习那些在图像平移中不变的字符识别函数。Mitchell and Thrun（1993）描述了根据学习器以前的知识计算训练导数的方法。在这两个系统中（在第 12 章中描述），

误差函数都被增加了一项，用来衡量这些训练导数和网络的实际导数间的差异。这样的误差函数的一个例子是

$$E(\vec{w}) \equiv \frac{1}{2}\sum_{d\in D}\sum_{k\in outputs}\left[(t_{kd}-o_{kd})^2+\mu\sum_{j\in inputs}\left(\frac{\partial t_{kd}}{\partial x_d^j}-\frac{\partial o_{kd}}{\partial x_d^j}\right)^2\right]$$

这里，x_d^j 表示对于训练实例 d 第 j 个输入单元的值。于是 $\frac{\partial t_{kd}}{\partial x_d^j}$ 是描述目标输出值 t_{kd} 应该如何随输入值 x_d^j 变化的训练导数。类似的，$\frac{\partial o_{kd}}{\partial x_d^j}$ 表示实际的学习网络的对应导数。常数μ决定匹配训练值对于匹配训练导数的相对权值。

- 使网络对目标值的交叉熵（cross entropy）最小化。考虑学习一个概率函数，比如预测一个借贷申请者会否还贷，根据是这个申请者的年龄和存款余额。尽管这里的训练样例仅提供了布尔型的目标值（要么是 1，要么是 0，根据这个申请者是否还贷），但基本的目标函数最好以申请者还贷的概率的形式输出，而不是对每个输入实例都企图输出明确的 0 或 1 值。在这种情况下，我们希望网络输出一个概率估计，可以证明最小化交叉熵（cross entropy）的网络可以给出最好的（也就是最大似然）概率估计，交叉熵的定义如下：

  $$-\sum_{d\in D} t_d \log o_d + (1-t_d)\log(1-o_d)$$

  这里 o_d 是网络对于训练样例 d 输出的概率估计，t_d 是对于训练样例 d 的目标值（0 或 1）。第 6 章讨论了何时及为什么最可能的网络假设就是使交叉熵最小化的假设，并推导了相应的 sigmoid 单元的梯度下降权值调整法则。第 6 章也描述了在什么条件下最可能的假设就是使误差平方和最小化的假设。
- 改变有效误差函数也可以通过权值共享（weight sharing）完成，也就是把与不同单元或输入相关联的权“捆绑在一起”。这里的想法是强迫不同的网络权值取一致的值，通常是为了实施人类设计者事先知道的某个约束。例如，Waibel et al.（1989）和 Lang et al.（1990）描述了神经网络在语音识别方面的一个应用，其中网络的输入是在一个 144 毫秒的时间窗中不同时间的语音频率分量。在这个应用中可以做的一个假定是：一个特定语音（例如“eee”）的频率分量的识别是与这个语音在 144 毫秒时间窗中出现的确切时间无关的。为了实施这个约束，必须强迫接收这个时间窗不同部分的不同单元共享权值。这样做的效果是约束了假设的潜在空间，从而减小了过度拟合的风险，提高了准确泛化到未见过情形的可能性。权值共享通常这样实现：首先在共享权值的每个单元分别更新各个权值，然后取这些权值的平均，再用这个平均值替换每个需共享的权值。这个过程的结果是被共享的权值与没有被共享的权值相比使用了不同的误差函数。

4.8.2 其他可选的误差最小化过程

虽然梯度下降是搜寻使误差函数最小化的假设的最通用的搜索方法之一，但它不总是最高效的。当训练复杂的网络时，不难见到反向传播算法要进行上万次的权值更新迭代。由于这个原因，人们探索并提出了很多其他的权值优化算法。为了领会其他的可能方法，我们不妨把权值更新方法就看作是要决定两个问题：选择一个改变当前权值向量的方向；选择要移

动的距离。在反向传播算法中，这个方向是通过取梯度的负值来选择的，距离是通过常量的学习速率η决定的。

一种被称为“**线搜索**（line search）”的优化方法，采用了不同的方法选择权值更新的距离。确切地讲，每当选定了一条确定权值更新方向的路线，那么权更新的距离是通过寻找沿这条线的误差函数的最小值来选择的。注意这可能导致很大幅度也可能是很小幅度的权值更新，要看沿这条线的最小误差点的位置。另一种方法，是根据“线搜索”的思想建立的，被称为**共轭梯度**（conjugate gradient）法。这种方法进行一系列线搜索来搜索误差曲面的最小值。这一系列搜索的第一步仍然使用梯度的反方向作为方向。在后来的每一步，选择使误差梯度分量刚好为 0 并保持为 0 的方向。

虽然其他的误差最小化方法提高了训练网络的效率，但象共轭梯度这样的方法对于最终网络的泛化误差没有明显的影响。对最终误差惟一可能的影响是，不同的误差最小化过程会陷入不同的局部极小值。Bishop（1996）包含了关于训练网络的几种参数优化方法的一般性讨论。

4.8.3 递归网络（Recurrent Networks）

直到现在我们考虑的只是有向无环的网络拓扑结构。递归网络是有如下特征的人工神经网络：适用于时序数据；使用网络单元在时间 t 的输出作为其他单元在时间 t+1 的输入。以这种方式，递归网络支持在网络中使用某种形式的有向环（directed cycles）。为了演示递归网络，考虑一个时序预测任务——根据当天的经济指标 $x(t)$，预测下一天的股票平均市值 $y(t+1)$。给定了这样的时序数据，一个显而易见的办法是根据输入值 $x(t)$训练一个前馈网络预测输出 $y(t+1)$。一个这样的网络显示在图 4-11（a）中。

这样的网络的缺点是仅依赖 $x(t)$作出对 $y(t+1)$预测，而不能捕捉 $y(t+1)$对 x 的以前值的依赖性。而这可能是必需的，例如，明天的股票平均市值可能依赖于今天的经济指标和昨天的经济指标的差异。当然我们可以通过把 $x(t)$和 $x(t-1)$都作为前馈网络的输入，来弥补这个不足。但是如果我们希望这个网络预测 $y(t+1)$时考虑任意过去的时间窗内的信息呢？那么就需要用不同的解决方案了。图 4-11（b）显示的递归网络提供了一个这样的解决方案。这里我们向隐藏层加了一个新的单元 b 和新的输入单元 $c(t)$。$c(t)$的值被定义为单元 b 在时间 t-1 的值；也就是说，网络在某一个时间步（time step）的输入值 $c(t)$拷贝自单元 b 在前一时间步的值。注意这实现了一种递归关系，其中 b 表示关于网络输入的历史信息。因为 b 既依赖于 $x(t)$又依赖于 $c(t)$，所以 b 可能概括了 x 以前任意时间距离的值。很多其他的网络拓扑也可以用来表示递归网络。例如，我们可以在输入和单元 b 间插入若干层单元，也可以在加入单元 b 和输入单元 c 的地方再并行插入几个单元。

插图——原书页码：120

Feedforward network-前馈网络

Recurrent network-递归网络

Recurrent network unfolded in time-按时间展开的递归网络

图 4-11 递归网络

如何训练这样的递归网络呢？递归网络有多种变体，因此人们也分别提出了不同的训练方法（例如参见 Jordan 1986; Elman 1990; Mozer 1995; Williams & Zipser 1995）。有趣的是，象图 4-11（b）那样的递归网络可以简单使用反向传播算法的变体来训练。为了理解如何实施，考虑图 4-11（c），显示了递归网络按照时间展开的数据流。这里我们把递归网络拷贝成几份，用不同拷贝间的连接替换掉反馈环。注意这个大的网络不再包含回路。所以展开网络的权值可以直接使用反向传播算法来训练。当然实践中我们希望仅保留一份递归网络和权值集合的拷贝。所以，在训练了展开的网络后，可以取不同拷贝中权值 w_{ji} 的平均值作为最终网络的对应的权值 w_{ji}。Mozer（1995）非常详细地描述了这个训练过程。实践中，递归网络比没有反馈环的网络难以训练，泛化的可靠性也不如后者。然而它们仍然因较强的表征力保持着重要性。

4.8.4 动态修改网络结构

直到现在我们考虑的神经网络学习问题是调整一个固定网络结构中的权值。为了改善泛化精度和训练效率，人们提出了很多动态增长或压缩网络单元和单元间连接数量的方法。

一种想法是从一个不包含隐藏单元的网络开始，然后根据需要增加隐单元增长网络，直到训练误差下降到某个可接受的水平。级联相关（Cascade-Correlation）算法（Fahlman & Lebiere 1990）就是这样一种算法。级联相关算法从创建一个没有隐单元的网络开始。例如，对于我们的人脸朝向的学习任务，它会建立一个仅包含四个输出单元全连接到 30×32 个输入结点的网络。在这个网络被训练了一段时间后，我们可以很容易地发现还有较大的残留误差，因为事实上这个目标函数不可能被一个单层结构的网络理想地表示。在这种情况下，算法增加一个隐藏单元，选择它的权值使这个隐藏单元的值和整个网络的残留误差的相关性最大化。现在一个新的单元被安装进了网络，它的权值保持不变，并且增加这个新单元到每一个输出单元间的连接。重复这个过程。原始的权值被再次训练（保持隐藏单元的权值不变），检查残留误差，如果残留误差还高于阈值就加入第二个隐单元。每当加入一个新的隐藏单元，它的输入包括所有原始的网络输入和已经存在的隐藏单元的输出。网络以这种方式增长，积聚隐藏单元，直到网络的残余误差下降到某个可接受的水平。Fahlman & Lebiere（1990）报告了级联相关算法显著减少训练时间的例子，原因是每一步仅有一层网络在被训练。这个算法的一个实际困难是因为算法可以无限制地增加单元，它就很容易过度拟合训练数据，所以必须采取避免过度拟合的预防措施。

动态修改网络结构的第二个想法是使用相反的途径。不再从可能的最简单网络开始增加复杂性，而是从一个复杂的网络开始修剪掉某些无关紧要的连接。判断某个权是否无关紧要的一种方法是看它的值是否接近 0。第二种看来在实践中更加成功的方法是考虑这个权值的一个小的变化对误差 E 的影响。变化 w 对 E 的影响（也就是 $\frac{\partial E}{\partial w}$）可以被看作衡量这个连接的显著性（salient）的尺度。LeCun et al.（1990）描述了一个网络被训练的过程，最不显著的连接被拆除，重复这个过程直到遇到某个终止条件。他们称这种方法为“最优脑损伤（optimal brain damage）”法，因为在每一步算法都试图去除最没有用的连接。他们报告了在一个字符识别应用中这种方法将一个大的网络中权值减少到四分之一，对泛化精度有微小的改善，并且大大改善了后来的训练效率。

一般而言，动态修改网络结构的方法已经取得了一些成功，但也有不足。这种方法是否能稳定地提高反向传播算法的泛化精度还有待研究。然而已经证明在一些情形下它可以显著地降低训练时间。

4.9 小结和补充读物

这一章的要点包括：

- 人工神经网络学习为学习实数值和向量值函数提供了一种实际的方法，对于连续的和离散值的属性都可以使用，并且对训练数据中的噪声有很好的鲁棒性。反向传播算法是最常见的网络学习算法，已经成功应用到很多学习任务，比如手写识别和机器人控制。
- 反向传播算法考虑的假设空间是固定连接的有权网络所能表示的所有函数空间。包含三层单元的前馈网络能够以任意精度逼近任意函数，只要每一层有足够数量（可能非常多）的单元。即使是一个实际大小的网络也能够表示很大范围的高度非线性的函数，这使得前馈网络成为学习预先未知的一般形式的离散和连续函数的很好选择。
- 反向传播算法使用梯度下降方法搜索可能假设的空间，迭代减小网络的误差以拟合训练数据。梯度下降收敛到训练误差相对网络权值的局部极小值。更一般的，梯度下降是一种有应用潜力的方法，它可用来搜索很多连续参数的假设空间，只要训练误差是假设参数的可微函数。
- 反向传播算法最令人感兴趣的特征之一是，它能够创造出网络输入中没有明确出现的特征。确切地讲，多层网络的内部（隐藏）层能够表示对学习目标函数有用的但隐含在网络输入中的中间特征。这种能力被例子如 4.6.4 节的 8×3×8 网络中创造的数字 1 到 8 的布尔编码；以及 4.7 节人脸识别应用中隐藏层表示的图像特征。
- 过度拟合训练数据是 ANN 学习中的一个重要问题。过度拟合导致网络泛化到新的数据时性能很差，尽管网络对于训练数据表现非常好。交叉验证方法可以用来估计梯度下降搜索的合适终止点，从而最小化过度拟合的风险。
- 尽管反向传播算法是最常见的 ANN 学习算法，人们也提出很多其他的算法，包括对于特殊任务的一些算法。例如，递归网络方法训练包含有向环的网络，类似级联相关的算法改变权的同时也改变网络结构。

本书的其他章节也介绍了一些关于 ANN 学习的其他信息。第 6 章给出了选择最小化误差平方和的贝叶斯论证，以及在其他情况下用最小化交叉熵（cross entropy）代替最小化误差平方和的方法。第 7 章讨论了为可靠学习布尔函数所需要的训练实例数量的理论结果，以及某些类型网络的 Vapnik-Chervonenkis 维。关于过度拟合以及如何避免的讨论可以在第 5 章中找到。第 12 章讨论了使用以前的知识来提高泛化精度的方法。

对人工神经网络的研究可以追溯到计算机科学的早期。McCulloch & Pitts（1943）提出了一个相当于感知器的神经元模型，60 年代的大量工作探索了这个模型的很多变体。60 年代早期 Widrow & Hoff（1960）探索了感知器网络（他们称为“adelines”）和 delta 法则，Rosenblatt（1962）证明了感知器训练法则的收敛性。然而，直到 60 年代晚期，人们开始清楚单层的感知器网络的表征能力很有限，而且找不到训练多层网络的有效方法。Minsky &

Papert（1969）说明即使是象 XOR 这样简单的函数也不能用单层的感知器网络表示或学习，在整个 70 年代 ANN 的研究衰退了。

在 80 年代中期 ANN 的研究经历了一次复兴，主要是因为训练多层网络的反向传播算法的发明（Rumelhart & McClelland 1986；Parker 1985）。这些思想可以被追溯到有关的早期研究（例如 Werbos 1975）。自从 80 年代，反向传播算法就成为应用最广泛的学习方法，而且人们也积极探索出了很多其他的 ANN 方法。在同一时期，计算机变得不再贵重，这允许人们试验那些在 60 年代不可能被完全探索的计算密集性的算法。

很多教科书专门论述了神经网络学习。一本早期的但仍有用的关于模式识别的参数学习方法的书是 Duda & Hart（1973）。Windrow & Stearns（1985）的教科书覆盖了感知器和相关的单层网络以及它们的应用。Rumelhart & McClelland（1986）收编了 80 年代中期开始的重新激发起人们对神经网络方法兴趣的论文。关于神经网络最近出版的书籍包括 Bishop（1996）；Chauvin & Rumelhart（1995）；Freeman & Skapina（1991）；Fu（1994）；Hecht-Nielson（1990）和 Hertz et al.（1991）。