第 10 章　模型独立的学习方式

在前面的章节中，介绍了机器学习的几种学习方式，包括监督学习、无监督学习等。这些学习方式分别可以由不同的模型和算法实现，比如神经网络、线性分类器等。针对一个给定的任务，首先要准备一定规模的训练数据，这些训练数据需要和真实数据的分布一致，然后设定一个目标函数和优化方法，在训练数据上学习一个模型。此外，不同任务的模型往往都是从零开始来训练的，一切知识都需要从训练数据中得到。这也导致了每个任务都需要准备大量的训练数据。在实际应用中，我们面对的任务往往难以满足上述要求，比如训练任务和目标任务的数据分布不一致，训练数据过少等。这时机器学习的应用会受到很大的局限。因此，人们开始关注一些新的学习方式的任务，现在的深度学习因为无法快速适应新的任务，就没办法来替代人类的工作。

本章介绍一些“模型独立的学习方式”，比如集成学习、协同学习、自学习、多任务学习、迁移学习、终身学习、小样本学习、元学习等。这里“模型独立”是指这些学习方式不限于具体的模型，不管是前馈神经网络、循环神经网络还是其他模型。然而一种学习方式往往会对符合某种特性的模型更加青睐，比如集成学习往往和方差大的模型组合时效果显著。

10.1　集成学习

给定一个学习任务，假设输入 $\mathbf{x}$ 和输出 $\mathbf{y}$ 的真实关系为 $\mathbf{y} = h(\mathbf{x})$。对于 M 个不同的模型 $f_1(\mathbf{x}), \cdots, f_M(\mathbf{x})$，每个模型的期望错误为

$$\mathcal{R}(f_m) = \mathbb{E}_{\mathbf{x}}\Big[\Big(f_m(\mathbf{x}) - h(\mathbf{x})\Big)^2\Big] \tag{10.1}$$

$$= \mathbb{E}_{\mathbf{x}}\Big[\epsilon_m(\mathbf{x})^2\Big], \tag{10.2}$$

其中 $\epsilon_m(\mathbf{x}) = f_m(\mathbf{x}) - h(\mathbf{x})$ 为模型 m 在样本 $\mathbf{x}$ 上的错误。

那么所有的模型的平均错误为

$$\bar{\mathcal{R}}(f) = \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}_{\mathbf{x}}[\epsilon_m(\mathbf{x})^2]. \tag{10.3}$$

集成学习（Ensemble Learning）就是通过某种策略将多个模型集成起来，通过群体决策来提高决策准确率。集成学习首要的问题是如何集成多个模型。比较常用的集成策略有直接平均、加权平均等。

最直接的集成学习策略就是直接平均，即“投票”。基于投票的集成模型 $f^{(c)}(\mathbf{x})$ 为

$$F(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} f_m(\mathbf{x}). \tag{10.4}$$

定理 10.1：对于 M 个不同的模型 $f_1(\mathbf{x}), \cdots, f_M(\mathbf{x})$，其平均期望错误为 $\bar{\mathcal{R}}(f)$。基于简单投票机制的集成模型 $F(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} f_m(\mathbf{x})$，$F(\mathbf{x})$ 的期望错误在 $\frac{1}{M}\bar{\mathcal{R}}(f)$ 和 $\bar{\mathcal{R}}(f)$ 之间。

证明. 根据定义，集成模型的期望错误为

$$\mathcal{R}(F) = \mathbb{E}_{\mathbf{x}}\Big[\Big(\frac{1}{M}\sum_{m=1}^{M} f_m(\mathbf{x}) - h(\mathbf{x})\Big)^2\Big] \tag{10.5}$$

$$= \frac{1}{M^2}\mathbb{E}_{\mathbf{x}}\Big[\Big(\sum_{m=1}^{M} \epsilon_m(\mathbf{x})\Big)^2\Big] \tag{10.6}$$

$$= \frac{1}{M^2}\mathbb{E}_{\mathbf{x}}\Big[\sum_{m=1}^{M}\sum_{n=1}^{M} \epsilon_m(\mathbf{x})\epsilon_n(\mathbf{x})\Big] \tag{10.7}$$

$$= \frac{1}{M^2}\sum_{m=1}^{M}\sum_{n=1}^{M} \mathbb{E}_{\mathbf{x}}\Big[\epsilon_m(\mathbf{x})\epsilon_n(\mathbf{x})\Big], \tag{10.8}$$

其中 $\mathbb{E}_{\mathbf{x}}[\epsilon_m(\mathbf{x})\epsilon_n(\mathbf{x})]$ 为两个不同模型错误的相关性。如果每个模型的错误不相关，即 $\forall m \neq n, \mathbb{E}_{\mathbf{x}}[\epsilon_m(\mathbf{x})\epsilon_n(\mathbf{x})] = 0$。如果每个模型的错误都是相同的，则 $\forall m \neq n, \epsilon_m(\mathbf{x}) = \epsilon_n(\mathbf{x})$。并且由于 $\epsilon_m(\mathbf{x}) \geq 0, \forall m$，可以得到

$$\bar{\mathcal{R}}(f) \geq \mathcal{R}(F) \geq \frac{1}{M}\bar{\mathcal{R}}(f), \tag{10.9}$$

即集成模型的期望错误是大于等于所有模型的平均期望错误的 $1/M$，小于等于所有模型的平均期望错误。 □

从定理10.1可知，为了得到更好的集成效果，要求每个模型之间具备一定的差异性。并且随着模型数量的增多，其错误率也会下降，并趋近于 0。

集成学习的思想可以用一句古老的谚语来描述：“三个臭皮匠赛过诸葛亮”。但是一个有效的集成需要各个基模型的差异尽可能大。为了增加模型之间的差异性，可以采取 Bagging 类和 Boosting 类两类方法。

Bagging 类方法 Bagging 类方法是通过随机构造训练样本、随机选择特征等方法来提高每个基模型的独立性，代表性方法有 Bagging 和随机森林等。

Bagging（Bootstrap Aggregating）是一个通过不同模型的训练数据集的独立性来提高不同模型之间的独立性。我们在原始训练集上进行有放回的随机采样，得到 M 比较小的训练集并训练 M 个模型，然后通过投票的方法进行模型集成。

随机森林（Random Forest）[Breiman, 2001] 是在 Bagging 的基础上再引入了随机特征，进一步提高每个基模型之间的独立性。在随机森林中，每个基模型都是一棵决策树。

Boosting 类方法 Boosting 类方法是按照一定的顺序来先后训练不同的基模型，每个模型都针对前序模型的错误进行专门训练。根据前序模型的结果，来调整训练训练样本的权重，从而增加不同基模型之间的差异性。Boosting 类方法是一种非常强大的集成方法，只要基模型的准确率比随机猜测好，就可以通过集成方法来显著地提高集成模型的准确率。Boosting 类方法的代表性方法有 AdaBoost[Freund et al., 1996] 等。

10.1.1 AdaBoost 算法

Boosting 类集成模型的目标是学习一个加性模型（additive model）

$$F(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m f_m(\mathbf{x}), \tag{10.10}$$

其中 $f_m(\mathbf{x})$ 为*弱分类器*（weak classifier），或*基分类器*（base classifier），α_m 为弱分类器的集成权重，$F(\mathbf{x})$ 称为*强分类器*（strong classifier）。

Boosting 类方法的关键是如何训练每个弱分类器 $f_m(\mathbf{x})$ 以及对应的权重 α_m。为了提高集成的效果，应当尽量使得每个弱分类器的差异尽可能大。一种有效的算法是迭代的方法来学习每个弱分类器，即按照一定的顺序依次训练每个弱分类器。在学习了第 m 个弱分类器后，增加其分错样本的权重，使得第 $m+1$ 个弱分类器“更关注”于前面弱分类器分错的样本。这样增加每个弱分类器的差异，最终提升的集成分类器的准确率。这种方法称为 *AdaBoost*（Adaptive Boosting）算法。

AdaBoost 算法是一种迭代式的训练算法，通过改变数据分布来提高弱分类器的差异。在每一轮训练中，增加分错样本的权重，减少分对样本的权重，从而得到一个新的数据分布。

以两类分类为例，AdaBoost 算法的训练过程如算法10.1所示。最初赋予每个样本同样的权重。在每一轮迭代中，根据当前的样本权重训练一个新的弱分

类器。然后根据这个弱分类器的错误率来计算其集成权重，并调整样本权重。

算法 10.1: AdaBoost 算法

输入: 训练集 $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$，迭代次数 M

初始样本权重：$w_1^{(n)} \leftarrow \frac{1}{N}, \forall n \in [1, N]$;

for $m = 1 \cdots M$ **do**

  按照样本权重 $w_m^{(1)}, \cdots, w_m^{(N)}$，学习弱分类器 f_m;

  计算弱分类器 f_m 在数据集上的错误为 ϵ_m;

  计算分类器的集成权重：

$$\alpha_m \leftarrow \frac{1}{2} \log \frac{1 - \epsilon_m}{\epsilon_m}.$$

  调整样本权重：

$$w_{m+1}^{(n)} \leftarrow w_m^{(n)} \exp(\alpha_m), \forall n \in [1, N]$$

end

输出: α_m 和 f_m，$1 \leq m \leq M$

AdaBoost 算法的统计学解释 AdaBoost 算法可以看做是一种分步（stage-wise）优化的加性模型 [Friedman et al., 2000]，其损失函数定义为

$$\mathcal{L}(F) = \exp\big(-yF(\mathbf{x})\big) \tag{10.11}$$

$$= \exp\big(-y \sum_{m=1}^{M} \alpha_m f_m(\mathbf{x})\big), \tag{10.12}$$

其中 $y, f_m(\mathbf{x}) \in \{+1, -1\}$。

假设经过 $m-1$ 次迭代，得到

$$F_{m-1}(\mathbf{x}) = \sum_{t=1}^{m-1} \alpha_t f_t(\mathbf{x}), \tag{10.13}$$

则第 m 次迭代的目标是找一个 α_m 和 $f_m(\mathbf{x})$ 使得下面的损失函数最小。

$$\mathcal{L}(\alpha_m, f_m(\mathbf{x})) = \sum_{n=1}^{N} \exp\Big(-y^{(n)}\big(F_{m-1}(\mathbf{x}^{(n)}) + \alpha_m f_m(\mathbf{x}^{(n)})\big)\Big). \tag{10.14}$$

令 $w_m^{(n)} = \exp\big(-y^{(n)} F_{m-1}(\mathbf{x}^{(n)})\big)$，则损失函数可以写为

$$\mathcal{L}(\alpha_m, f_m(\mathbf{x})) = \sum_{n=1}^{N} w_m^{(n)} \exp\big(-y^{(n)} \alpha_m f_m(\mathbf{x}^{(n)})\big). \tag{10.15}$$

因为$y, f_m(\mathbf{x}) \in \{+1, -1\}$，有

$$yf_m(\mathbf{x}) = 1 - 2I(y \neq f_m(\mathbf{x})), \tag{10.16}$$

其中$I(x)$为指示函数。

将损失函数在$f_m(\mathbf{x}) = 0$处进行二阶泰勒展开，有

指数函数$\exp(x)$在$x = 0$处的二阶泰勒展开公式为$1 + x + \frac{x^2}{2!}$。

$$\mathcal{L}(\alpha_m, f_m(\mathbf{x})) = \sum_{n=1}^{N} w_m^{(n)} \big(1 - y^{(n)} \alpha_m f_m(\mathbf{x}^{(n)}) + \frac{1}{2}\alpha_m^2\big) \tag{10.17}$$

$$\propto \alpha_m \sum_{n=1}^{N} w_m^{(n)} I\big(y^{(n)} \neq f_m(\mathbf{x}^{(n)})\big). \tag{10.18}$$

从上式可以看出，当$\alpha_m > 0$时，最优的分类器$f_m(\mathbf{x})$为使得在样本权重为$w_m^{(n)}, 1 \leq n \leq N$时的加权错误率最小的分类器。

在求解出$f_m(\mathbf{x})$之后，公式(10.15)可以写为

$$\mathcal{L}(\alpha_m, f_m(\mathbf{x})) = \sum_{y^{(n)}=f_m(\mathbf{x}^{(n)})} w_m^{(n)} \exp(-\alpha_m) + \sum_{y^{(n)}\neq f_m(\mathbf{x}^{(n)})} w_m^{(n)} \exp(\alpha_m) \tag{10.19}$$

$$\propto (1 - \epsilon_m) \exp(-\alpha_m) + \epsilon_m \exp(\alpha_m), \tag{10.20}$$

其中ϵ_m为分类器$f_m(\mathbf{x})$的加权错误率，

$$\epsilon_m = \frac{\sum_{y^{(n)}\neq f_m(\mathbf{x}^{(n)})} w_m^{(n)}}{\sum_n w_m^{(n)}}. \tag{10.21}$$

求上式关于α_m的导数并令其为0，得到

$$\alpha_m = \frac{1}{2} \log \frac{1 - \epsilon_m}{\epsilon_m}. \tag{10.22}$$

10.2 自训练和协同训练

监督学习往往需要大量的标注数据，而标注数据的成本比较高，因此如何利用大量的无标注数据来提高监督学习的效果，有十分重要的意义。这种利用少量标注数据和大量无标注数据进行学习的方式称为半监督学习（Semi-supervised Learning）。

本节介绍两种半监督学习算法：自训练和协同训练。

10.2.1 自训练

这里的 bootstrapping 和统计中的概念不同。

自训练（Self-Training），也叫自训练（Self-Teaching）或自举法（bootstrapping），是一种非常简单的半监督学习算法 [Scudder, 1965, Yarowsky, 1995]。

自训练是首先使用标注数据来训练一个模型，并使用这个模型来预测无标注样本的标签，把预测置信度比较高的样本及其预测的伪标签加入训练集，然后重新训练新的模型，并不断重复这个过程。算法10.2给出了自训练的训练过程。

算法 10.2: 自训练的训练过程

输入: 标注数据集 $\mathcal{L} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$;
无标注数据集 $\mathcal{U} = \{\mathbf{x}^{(m)}\}_{m=1}^{M}$;
迭代次数 T; 每次迭代增加样本数量 P;

for $t = 1 \cdots T$ **do**
  根据训练集 $\mathcal{L}$，训练模型 f;
  使用模型 f 对未标记数据集 $\mathcal{U}$ 的样本进行预测，选出预测置信度高的 P 个样本 $\mathcal{P} = \{(\mathbf{x}^{(p)}, f(\mathbf{x}^{(p)}))\}_{p=1}^{P}$;
  更新训练集：

$$\mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{P}, \qquad \mathcal{U} \leftarrow \mathcal{U} - \mathcal{P}.$$

end

输出: 模型 f

参见习题10-2。

自训练和密度估计中 EM 算法有一定的相似之处，通过不断地迭代来提高模型能力。但自训练的缺点是无法保证每次加入训练集的样本的伪标签是正确的。如果选择样本的伪标签是错误的，反而会损害模型的预测能力。因此，自训练最关键的步骤是如何设置挑选样本的标准。

10.2.2 协同训练

协同训练（Co-Training）是自训练的一种改进方法，通过两个基于不同视角（view）的分类器来相互促进。很多数据都有相对独立的不同视角。比如互联网上的每个网页都由两种视角组成：文字内容（text）和指向其它网页的链接（hyperlink）。如果要确定一个网页的类别，可以根据文字内容来判断，也可根据网页之间的链接关系来判断。

假设一个样本 $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2]$，其中 $\mathbf{x}_1$ 和 $\mathbf{x}_2$ 分别表示两种不同视角 V_1 和 V_2 的特征，并满足下面两个假设：（1）条件独立性：给定样本标签 y 时，两种特征条件独立 $p(\mathbf{x}_1, \mathbf{x}_2|y) = p(\mathbf{x}_1|y)p(\mathbf{x}_2|y)$；（2）充足和冗余性：当数据充分时，每种视角的特征都可以足以单独训练出一个正确的分类器。令 $y = g(\mathbf{x})$ 为需要学习

的真实映射函数，f_1 和 f_2 分别为两个视角的分类器，有

$$\exists f_1, f_2, \quad \forall \mathbf{x} \in \mathcal{X}, \qquad f_1(\mathbf{x}_1) = f_2(\mathbf{x}_2) = g(\mathbf{x}), \tag{10.23}$$

其中 $\mathcal{X}$ 为样本 $\mathbf{x}$ 的取值空间。

由于不同视角的条件独立性，在不同视角上训练出来的模型就相当于从不同视角来理解问题，具有一定的互补性。协同训练就是利用这种互补性来来进行自训练的一种方法。首先在训练集上根据不同视角分别训练两个模型 f_1 和 f_2，然后用 f_1 和 f_2 在无标记数据集上进行预测，各选取预测置信度比较高的样本加入训练集，重新训练两个不同视角的模型，并不断重复这个过程。

算法10.3给出了协同训练的训练过程。

算法 10.3: 协同训练的训练过程

输入: 标注数据集 $\mathcal{L} = \{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$;
无标注数据集 $\mathcal{U} = \{\mathbf{x}^{(m)}\}_{m=1}^{M}$;
迭代次数 T; 候选池大小 K; 每次迭代增加样本数量 $2P$;

for $t = 1 \cdots T$ **do**
  根据训练集 $\mathcal{L}$ 的视角 V_1 训练训练模型 f_1;
  根据训练集 $\mathcal{L}$ 的视角 V_2 训练训练模型 f_2;
  从无标记数据集 $\mathcal{U}$ 上随机选取一些样本放入候选池 $\mathcal{U}'$，使得 $|\mathcal{U}'| = K$;
  for $f \in f_1, f_2$ **do**
    使用模型 f 预测候选池 $\mathcal{U}'$ 中的样本的伪标签;
    for $p = 1 \cdots P$ **do**
      根据标签分布，随机选取一个标签 y;
      从 $\mathcal{U}'$ 中选出伪标签为 y，并且预测置信度最高的样本 $\mathbf{x}$;
      更新训练集：

$$\mathcal{L} \leftarrow \mathcal{L} \cup \{(\mathbf{x}, y)\}, \qquad \mathcal{U}' \leftarrow \mathcal{U}' - \{(\mathbf{x}, y)\}.$$

    end
  end
end

输出: 模型 f_1,f_2

协同算法要求两种视图是条件独立的。如果两种视图完全一样，则协同训练退化成自训练算法。

10.3 多任务学习

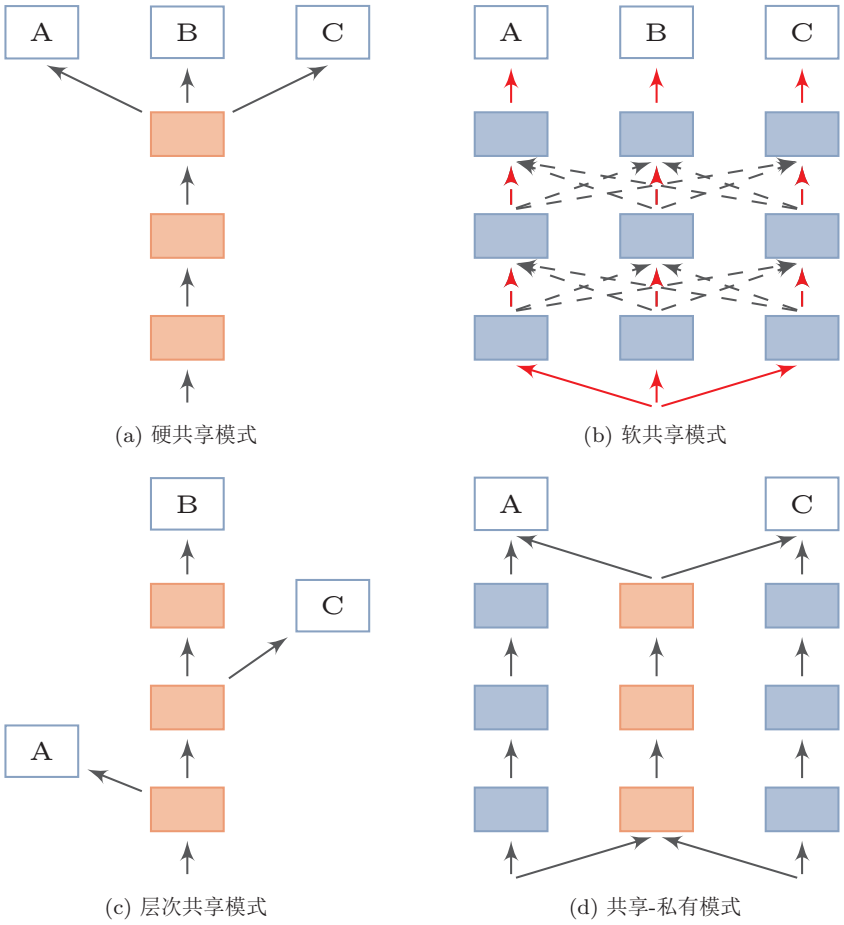
一般的机器学习模型都是针对单一的特定任务，比如手写体数字识别、物体检测等。不同任务的模型都是在各自的训练集上单独学习得到的。如果有两个任务比较相关，它们之间会存在一定的共享知识，这些知识对两个任务都会有所帮助。这些共享的知识可以是表示（特征）、模型参数或学习算法等。目前，主流的多任务学习方法主要关注于表示层面的共享。

多任务学习（Multi-task Learning）是指同时学习多个相关任务，让这些任务在学习过程中共享知识，利用多个任务之间的相关性来改进模型在每个任务的性能和泛化能力。多任务学习可以看作是一种*归纳迁移学习*（Inductive Transfer Learning），即通过利用包含在相关任务中的信息作为*归纳偏置*（Inductive Bias）来提高泛化能力 [Caruana, 1997]。

共享机制 多任务学习的主要挑战在于如何设计多任务之间的共享机制。在传统的机器学习算法中，引入共享的信息是比较困难的，通常会导致模型变得复杂。但是在神经网络模型中，模型共享变得相对比较容易。深层神经网络模型提供了一种很方便的信息共享方式，可以很容易地进行多任务学习。多任务学习的共享机制比较灵活，有非常多种的共享模型。图10.1给出了多任务学习中四种常见的共享模式。

- 硬共享模式：让不同任务的神经网络模型共同使用一些共享模块（一般是低层）来提取一些通用特征，然后再针对每个不同的任务设置一些私有模块（一般是高层）来提取一些任务特定的特征。
- 软共享模式：不显式地设置共享模块，但每个任务都可以从其它任务中“窃取”一些信息来提高自己的能力。窃取的方式包括直接复制使用其它任务的隐状态，或使用注意力机制来主动选取有用的信息。
- 层次共享模式：一般神经网络中不同层抽取的特征类型不同。底层一般抽取一些低级的局部特征，高层抽取一些高级的抽象语义特征。因此如果多任务学习中不同任务也有级别高低之分，那么一个合理的共享模式是让低级任务在底层输出，高级任务在高层输出。
- 共享-私有模式：一个更加分工明确的方式是将共享模块和任务特定（私有）模块的责任分开。共享模块捕捉一些跨任务的共享特征，而私有模块只捕捉和特定任务相关的特征。最终的表示由共享特征和私有特征共同构成。

学习步骤 在多任务学习中，每个任务都可以有自己单独的训练集。为了让所有任务同时学习，我们通常会使用交替训练的方式来“近似”地实现同时学习。



(a) 硬共享模式 (b) 软共享模式

(c) 层次共享模式 (d) 共享-私有模式

图 10.1 多任务学习中四种常见的共享模式

假设有 M 个相关任务，第 m 个任务的训练集为 $\mathcal{D}_m$，包含 N_m 个样本。

$$\mathcal{D}_m = \{(\mathbf{x}^{(m,n)}, y^{(m,n)})\}_{n=1}^{N_m}, \tag{10.24}$$

其中 $\mathbf{x}^{(m,n)}$ 和 $y^{(m,n)}$ 表示第 m 个任务中的第 n 个样本以及它的标签。

假设这 M 任务对应的模型分别为 $f_m(\mathbf{x}, \theta), 1 \leq m \leq M$，多任务学习的联合目标函数为所有任务损失函数的线性加权。

$$\mathcal{L}(\theta) = \sum_{m=1}^{M} \sum_{n=1}^{N_m} \eta_m \mathcal{L}_m \Big(f_m(x^{(m,n)}, \theta), y^{(m,n)} \Big), \tag{10.25}$$

其中 $\mathcal{L}_m(\cdot)$ 为第 m 个任务的损失函数，η_m 是第 m 个任务的权重，θ 表示包含了共享模块和私有模块在内的所有参数。权重可以根据不同任务的重要程度来赋

值，也可以根据任务的难易程度来赋值。通常情况下，所有任务设置相同的权重，即 $\eta_m = 1, 1 \leq m \leq M$。

多任务学习的流程可以分为两个阶段：（1）联合训练阶段：每次迭代时，随机挑选一个任务，然后从这个任务中随机选择一些训练样本，计算梯度并更新参数；（2）单任务精调阶段：基于多任务的学习到的参数，分别在每个单独任务进行精调。其中单任务精调阶段为可选阶段。当多个任务的差异性比较大时，在每个单任务上继续优化参数可以进一步提升模型能力。

多任务学习中联合训练阶段的具体过程如算法10.4所示。

算法 10.4: 多任务学习中联合训练过程

输入: M 个任务的数据集 $\mathcal{D}_m, 1 \leq m \leq M$;
每个任务的批量大小 $K_m, 1 \leq m \leq M$;
最大迭代次数 T，学习率 α;

随机初始化参数 θ_0;
for $t = 1 \cdots T$ **do**
  // 准备 M 个任务的数据
  for $m = 1 \cdots M$ **do**
    将任务 m 的训练集 $\mathcal{D}_m$ 中随机划分为 $c = \frac{N_m}{K_m}$ 个小批量集合：$\mathcal{B}_m = \{\mathcal{I}_{m,1}, \cdots, \mathcal{I}_{m,c}\}$;
  end
  合并所有小批量样本 $\bar{\mathcal{B}} = \mathcal{B}_1 \cup \mathcal{B}_2 \cup \cdots \cup \mathcal{B}_M$;
  随机排序 $\bar{\mathcal{B}}$;
  foreach $\mathcal{I} \in \bar{\mathcal{B}}$ **do**
    计算小批量样本 $\mathcal{I}$ 上的损失 $\mathcal{L}(\theta)$; // 只计算 $\mathcal{I}$ 在对应任务上的损失
    更新参数：$\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \nabla_\theta \mathcal{L}(\theta)$;
  end
end
输出: 模型 $f_m, 1 \leq m \leq M$

多任务学习通常可以获得比单任务学习更好的泛化能力，主要由于以下几个原因：

1. 多任务学习在多个任务的数据集上进行训练，比单任务学习的训练集更大。由于多个任务之间有一定的相关性，因此多任务学习相当于是一种隐式的数据增强，可以提高模型的泛化能力。
2. 多任务学习中的共享模块需要兼顾所有任务，这在一定程度上避免了模型过拟合到单个任务的训练集，可以看作是一种正则化。

3. 既然一个好的表示通常需要适用于多个不同任务，多任务学习的机制使得它会比单任务学习可以获得一个更好的表示。

参见第1.4节。

4. 在多任务学习中，每个任务都可以“选择性”利用其他任务中学习到的隐藏特征，从而提高自身的能力。

10.4 迁移学习

标准机器学习的前提假设是训练数据和测试数据的分布是相同的。如果不满足这个假设，在训练集上学习到的模型在测试集上的表现会比较差。而在很多实际场景中，经常碰到的问题是由标注数据的成本十分高，无法为一个目标任务准备足够多相同分布的训练数据。因此，如果有一个相关任务已经有了大量的训练数据，虽然这些训练数据的分布和目标任务不同，但是由于训练数据的规模比较大，我们假设可以从中学习某些可以泛化的知识，那么这些知识对目标任务会有一定的帮助。如何将相关任务的训练数据中的可泛化知识迁移到目标任务上，就是*迁移学习*（Transfer Learning）要解决的问题。

具体而言，假设一个机器学习任务$\mathcal{T}$的样本空间为$\mathcal{X} \times \mathcal{Y}$，其中$\mathcal{X}$为输入空间和$\mathcal{Y}$为输出空间，其概率密度函数为$p(\mathbf{x}, y)$。简单起见，这里设$\mathcal{X}$为$d$维实数空间的一个子集，$\mathcal{Y}$为一个离散的集合。

$p(\mathbf{x}, y) = P(X = \mathbf{x}, Y = y)$.

一个样本空间及其分布可以称为一个*领域*（Domain）：$\mathcal{D} = (\mathcal{X}, \mathcal{Y}, p(\mathbf{x}, y))$。给定两个领域，如果它们的输入空间、输出空间或概率分布中至少一个不同，那么这两个领域就被认为是不同的。从统计学习的观点来看，一个机器学习任务$\mathcal{T}$定义为在一个领域$\mathcal{D}$上的条件概率$p(y|\mathbf{x})$的建模问题。

*迁移学习*是指两个不同领域的知识迁移过程，利用*源领域*（Source Domain）$\mathcal{D}_S$中学到的知识用来帮助*目标领域*（Target Domain）$\mathcal{D}_T$上的学习任务。源领域的训练样本数量一般远大于目标领域。

表10.1给出了迁移学习和标准机器学习的比较。

学习类型	样本空间	概率分布
标准机器学习	$\mathcal{X}_S = \mathcal{X}_T, \mathcal{Y}_S = \mathcal{Y}_T$	$p_S(\mathbf{x}, y) = p_T(\mathbf{x}, y)$
迁移学习	$\mathcal{X}_S \neq \mathcal{X}_T$ 或 $\mathcal{Y}_S \neq \mathcal{Y}_T$ 或 $p_S(\mathbf{x}, y) \neq p_T(\mathbf{x}, y)$	

表 10.1 迁移学习类型

迁移学习根据不同的迁移方式又分为两个类型：*归纳迁移学习*（Inductive Transfer Learning）和*推导迁移学习*（Transductive Transfer Learning）。这两

个类型分别对应两个机器学习的范式：*归纳学习*（Inductive Learning）和*转导学习*（Transductive Learning）[Vapnik, 1998]。一般的机器学习都是指*归纳学习*，即希望在训练数据集上学习到使得期望风险（即真实数据分布上的错误率）最小的模型。而*转导学习*的目标是学习一种在给定测试集上错误率最小的模型，在训练阶段可以利用测试集的信息。

期望风险参见第2.2.2节。

归纳迁移学习是指在源领域和任务上学习出一般的规律，然后将这个规律迁移到目标领域和任务上；而转导迁移学习是一种从样本到样本的迁移，直接利用源领域和目标领域的样本进行迁移学习。

10.4.0.1 归纳迁移学习

在归纳迁移学习中，源领域和目标领域有相同的输入空间 $\mathcal{X}_S = \mathcal{X}_T$，输出空间可以相同也可以不同，源任务和目标任务一般都不相同 $\mathcal{T}_S \neq \mathcal{T}_T$，即 $p_S(y|\mathbf{x}) \neq p_T(y|\mathbf{x})$。一般而言，归纳迁移学习要求源领域和目标领域是相关的，并且源领域 $\mathcal{D}_S$ 有大量的训练样本，这些样本可以是有标注的样本也可以是无标注样本。

1）当源领域只有大量无标注数据时，源任务可以转换为无监督学习任务，比如*自编码*和*密度估计*任务。通过这些无监督任务学习一种可迁移的表示，然后在将这种表示迁移到目标任务上。这种学习方式和*自学习*（Self-Taught Learning）[Raina et al., 2007] 以及*半监督学习*比较类似。比如在自然语言处理领域，由于语言相关任务的标注成本比较高，很多自然语言处理任务的标注数据都比较少，这导致了在这些自然语言处理任务上经常会受限于训练样本数量而无法充分发挥深度学习模型的能力。同时，由于我们可以低成本地获取大规模的无标注自然语言文本，因此一种自然的迁移学习方式是将大规模文本上的无监督学习（比如语言模型）中学到的知识迁移到一个新的目标任务上。从早期的预训练词向量（比如 word2vec [Mikolov et al., 2013] 和 GloVe [Pennington et al., 2014] 等）到句子级表示（比如 ELMO [Peters et al., 2018]、OpenAI GPT [Radford et al., 2018] 以及 BERT[Devlin et al., 2018] 等）都对自然语言处理任务有很大的促进作用。

自编码器参见第9.1.3节。
概率密度估计参见第9.2节。

2）当源领域有大量的标注数据时，可以直接将源领域上训练的模型迁移到目标领域上。比如在计算机视觉领域有大规模的图像分类数据集 ImageNet [Deng et al., 2009]。由于在 ImageNet 数据集上有很多预训练的图像分类模型，比如 AlexNet[Krizhevsky et al., 2012]、VGG [Simonyan and Zisserman, 2014] 和 ResNet[He et al., 2016] 等，我们可以将这些预训练模型迁移到目标任务上。

在归纳迁移学习中，由于源领域的训练数据规模非常大，这些预训练模型通常有比较好的泛化性，其学习到的表示通常也适用于目标任务。归纳迁移学习一般有下面两种迁移方式：

1. 基于特征的方式：将预训练模型的输出或者是中间隐藏层的输出作为特征直接加入到目标任务的学习模型中。目标任务的学习模型可以是一般的浅层分类器（比如支持向量机等）或一个新的神经网络模型。
2. 精调的方式：在目标任务上复用预训练模型的部分组件，并对其参数进行精调（fine-tuning）。

假设预训练的模型是一个深层神经网络，不同层的可迁移性也不尽相同[Yosinski et al., 2014]。通常来说，网络的低层学习一些通用的低层特征，中层或高层学习抽象的高级语义特征，而最后几层一般学习和特定任务相关的特征。因此，根据目标任务的自身特点以及和源任务的相关性，可以针对性地选择预训练模型的不同层来迁移到目标任务中。

将预训练模型迁移目标任务上通常会比从零开始学习的方式更好，主要体现在以下三点[Torrey and Shavlik, 2010]：（1）初始模型的性能一般比随机初始化的模型要好；（2）训练时模型的学习速度比从零开始学习要快，收敛性更好；（3）模型的最终性能更好，具有更好的泛化性。

归纳迁移学习和多任务学习也比较类似，但有下面两点区别：（1）多任务学习是同时学习多个不同任务，而归纳迁移学习是通常分为两个阶段，即源任务上的学习阶段，和目标任务上的迁移学习阶段；（2）归纳迁移学习是单向的知识迁移，希望提高模型在目标任务上的性能，而多任务学习是希望提高所有任务的性能。

10.4.0.2 转导迁移学习

*转导迁移学习*是一种从样本到样本的迁移，直接利用源领域和目标领域的样本进行迁移学习[Arnold et al., 2007]。转导迁移学习可以看作一个种特殊的*转导学习*（Transductive Learning）[Joachims, 1999]。转导迁移学习通常假设源领域有大量的标注数据，而目标领域没有（或有少量）标注数据，但是有大量的无标注数据。目标领域的数据在训练阶段是可见的。

转导迁移学习的一个常见子问题是*领域适应*（Domain Adaptation）。在领域适应问题中，一般假设源领域和目标领域有相同的样本空间，但是数据分布不同$p_S(\mathbf{x}, y) \neq p_T(\mathbf{x}, y)$。

根据贝叶斯公式，$p(\mathbf{x}, y) = p(\mathbf{x}|y)p(y) = p(y|\mathbf{x})p(\mathbf{x})$，因此数据分布的不一致通常由三种情况造成：

1. *协变量偏移*（Covariate Shift）：源领域和目标领域的输入边际分布不同$p_S(\mathbf{x}) \neq p_T(\mathbf{x})$，但后验分布相同$p_S(y|\mathbf{x}) = p_T(y|\mathbf{x})$，即学习任务相同$\mathcal{T}_S = \mathcal{T}_T$。

2. 概念偏移（Concept Shift）：输入边际分布相同$p_S(\mathbf{x}) = p_T(\mathbf{x})$，但后验分布不同$p_S(y|\mathbf{x}) \neq p_T(y|\mathbf{x})$，即学习任务不同$\mathcal{T}_S \neq \mathcal{T}_T$。

3. 先验偏移（Prior Shift）：源领域和目标领域中的输出y先验分布不同$p_S(y) \neq p_T(y)$，条件分布相同$p_S(\mathbf{x}|y) = p_T(\mathbf{x}|y)$。在这样情况下，目标领域必须提供一定数量的标注样本。

广义的领域适应问题可能包含上述一种或多种偏移情况。目前，大多数的领域适应问题主要关注于协变量偏移，这样领域适应问题在关键就在于如何学习领域无关（Domain-Invariant）的表示。假设$p_S(y|\mathbf{x}) = p_T(y|\mathbf{x})$，领域适应的目标是学习一个模型$f : \mathcal{X} \to \mathcal{Y}$使得

$$\mathcal{R}_T(\theta_f) = \mathbb{E}_{(\mathbf{x},y)\sim p_T(\mathbf{x},y)}[\mathcal{L}(f(\mathbf{x},\theta_f),y)] \tag{10.26}$$

$$= \mathbb{E}_{(\mathbf{x},y)\sim p_S(\mathbf{x},y)}\frac{p_T(\mathbf{x},y)}{p_S(\mathbf{x},y)}[\mathcal{L}(f(\mathbf{x},\theta_f),y)] \tag{10.27}$$

$$= \mathbb{E}_{(\mathbf{x},y)\sim p_S(\mathbf{x},y)}\frac{p_T(\mathbf{x})}{p_S(\mathbf{x})}[\mathcal{L}(f(\mathbf{x},\theta_f),y)], \tag{10.28}$$

其中$\mathcal{L}(\cdot)$为损失函数，θ_f为模型参数。

如果我们可以学习一个映射函数$g : \mathcal{X} \to \mathbb{R}^d$，将$\mathbf{x}$映射到一个特征空间中，并在这个特征空间中使得源领域和目标领域的边际分布相同$p_S\big(g(\mathbf{x},\theta_g)\big) = p_T\big(g(\mathbf{x},\theta_g)\big), \forall \mathbf{x} \in \mathcal{X}$，其中$\theta_g$为映射函数的参数，那么目标函数可以近似为

$$\mathcal{R}_T(\theta_f,\theta_g) = \mathbb{E}_{(\mathbf{x},y)\sim p_S(\mathbf{x},y)}\left[\mathcal{L}\Big(f\big(g(\mathbf{x},\theta_g),\theta_f\big),y\Big)\right] + \gamma d_g(S,T) \tag{10.29}$$

$$= \mathcal{R}_S(\theta_f,\theta_g) + \gamma d_g(S,T), \tag{10.30}$$

其中$\mathcal{R}_S(\theta_f,\theta_g)$为源领域上的期望风险函数，$d_g(S,T)$是一个分布差异的度量函数，用来计算在映射特征空间中源领域和目标领域的样本分布的距离，γ为一个超参数用来平衡两个子目标的重要性比例。这样，学习的目标是优化参数θ_f,θ_g使得提取的特征是领域无关的，并且在源领域上损失最小。

令

$$\mathcal{D}_S = \{(\mathbf{x}_S^{(n)}, y_S^{(n)})\}_{n=1}^N \sim p_S(\mathbf{x},y), \tag{10.31}$$

$$\mathcal{D}_T = \{\mathbf{x}_T^{(m)}\}_{m=1}^M \sim p_T(\mathbf{x},y), \tag{10.32}$$

分别为源领域和目标领域的训练数据，我们首先用映射函数$g(\mathbf{x},\theta_g)$将两个领域中训练样本的输入$\mathbf{x}$映射到特征空间，并优化参数θ_g使得映射后两个领域的输入分布差异最小。分布差异一般可以通过一些度量函数来计算，比如 MMD（Maximum Mean Discrepancy）[Gretton et al., 2007]、CMD（Central Moment

Discrepancy）[Zellinger et al., 2017] 等，也可以通过领域对抗学习来得到领域无关的表示 [Bousmalis et al., 2016, Ganin et al., 2016]。

以对抗学习为例，我们可以引入一个领域判别器 c 来判断一个样本是来自于哪个领域。如果领域判别器 c 无法判断一个映射特征的领域信息，就可以认为这个特征是一种领域无关的表示。

对于训练集中的每一个样本 $\mathbf{x}$，我们都赋予 $z \in \{1, 0\}$ 表示它是来自于源领域还是目标领域，领域判别器 $c\big(\mathbf{h}, \theta_c\big)$ 根据其映射特征 $\mathbf{h} = g(\mathbf{x}, \theta_g)$ 来预测它自于源领域的概率 $p(z = 1|\mathbf{x})$。由于领域判别是一个两分类问题，$\mathbf{h}$ 来自于目标领域的概率为 $1 - c\big(\mathbf{h}, \theta_c\big)$。

因此，领域判别器的损失函数为：

$$\mathcal{L}_c(\theta_g, \theta_c) = \frac{1}{N}\sum_{n=1}^{N} \log c(\mathbf{h}_S^{(n)}, \theta_c) + \frac{1}{M}\sum_{m=1}^{M} \log\big(1 - c(\mathbf{x}_D^{(m)}, \theta_c)\big), \tag{10.33}$$

其中 $\mathbf{h}_S^{(n)} = g(\mathbf{x}_S^{(n)}, \theta_g), \mathbf{h}_D^{(m)} = g(\mathbf{x}_D^{(m)}, \theta_g)$ 分别为样本 $\mathbf{x}_S^{(n)}$ 和 $\mathbf{x}_D^{(m)}$ 的特征向量。

这样，领域迁移的目标函数可以分解为两个对抗的目标。一方面，要学习参数 θ_c 使得领域判别器 $c(\mathbf{h}, \theta_c)$ 尽可能区分出一个表示 $\mathbf{h} = g(\mathbf{x}, \theta_g)$ 是来自于哪个领域；另一方面，要学习参数 θ_g 使得提取的表示 $\mathbf{h}$ 无法被领域判别器 $c(\mathbf{h}, \theta_c)$ 预测出来，并同时学习参数 θ_g 使得模型 $f(\mathbf{h}, \theta_f)$ 在源领域的损失最小。

$$\min_{\theta_c} \quad \mathcal{L}_c(\theta_f, \theta_c), \tag{10.34}$$

$$\min_{\theta_f, \theta_g} \quad \sum_{n=1}^{N} \mathcal{L}\left(f\Big(g(\mathbf{x}_S^{(n)}, \theta_g), \theta_f\Big), y_S^{(n)}\right) - \gamma \mathcal{L}_c(\theta_f, \theta_c). \tag{10.35}$$

10.5 终生学习

虽然深度学习在很多任务上取得了成功，前提是训练数据和测试数据的分布要相同，一但训练结束模型就保持固定，不再进行迭代更新。并且，要想一个模型同时在很多不同任务上都取得成功依然是一件十分困难的事情。比如在围棋任务上训练的 AlphaGo 只会下围棋，对象棋一窍不通。如果将让 AlphaGo 去学习象棋，可能会损害其下围棋的能力，这显然不符合人类的学习过程。我们在学会了围棋之后，再去学象棋，不会忘记围棋的下法。人类的学习是一直持续的，人脑可以通过记忆不断地累积学习到的知识，这些知识累积可以在不同的任务中持续进行。在大脑的海马系统上，新的知识在以往知识的基础上被快速建立起来；之后经过长时间的处理，在大脑皮质区形成较难遗忘的长时记忆。由于不断的知识累积，人脑在学习新的任务时一般不需要太多的标注数据。

终生学习（Lifelong Learning），也叫**持续学习**（Continuous Learning），是指像人类一样具有持续不断的学习能力，根据历史任务中学到的经验和知识

来帮助学习不断出现的新任务，并且这些经验和知识是持续累积的，不会因为新的任务而忘记旧的知识 [Chen and Liu, 2016, Thrun, 1998]。

在终生学习中，假设一个终生学习算法已经在历史任务 $\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_m$ 上学习到一个模型，当出现一个新任务 $\mathcal{T}_{m+1}$ 时，这个算法可以根据过去在 m 个任务上学习的知识来帮助第 $m+1$ 个任务，同时累积所有的 $m+1$ 个任务上的知识。这个设定和*归纳迁移学习*十分类似，但归纳迁移学习的目标是优化目标任务的性能，而不关心知识的累积。而终生学习的目标是持续的学习和知识累积。另外，也和*多任务学习*的不同之处在于终生学习并不在所有任务上同时学习。多任务学习是在使用所有任务的数据进行联合学习，并不是持续的一个一个的学习。

在终生学习中，一个关键的问题是如何避免*灾难性遗忘*（Catastrophic Forgetting），即按照一定顺序学习多个任务时，在学习新任务的同时不忘记先前学会的历史任务 [French, 1999, Kirkpatrick et al., 2017]。比如在神经网络模型中，一些参数对任务 $\mathcal{T}_A$ 非常重要，如果在学任务 $\mathcal{T}_B$ 时被改变了，就可能给任务 $\mathcal{T}_A$ 造成不好的影响。

在网络容量有限时，学习一个新的任务一般需要遗忘一些历史任务的知识。而目前的神经网络往往都是过参数化的，对于任务 $\mathcal{T}_A$ 而言有很多参数组合都可以达到最好的性能。这样在学习任务 $\mathcal{T}_B$ 时，可以找到一组不影响任务 $\mathcal{T}_A$ 而又能使得任务 $\mathcal{T}_B$ 最优的参数。

解决灾难性遗忘的方法有很多。我们这里介绍一种*弹性权重巩固*（Elastic Weight Consolidation）方法 [Kirkpatrick et al., 2017]。

不失一般性，以两个任务的持续学习为例，假设任务 $\mathcal{T}_A$ 和任务 $\mathcal{T}_B$ 的数据集分别为 $\mathcal{D}_A$ 和 $\mathcal{D}_B$。从贝叶斯的角度来看，我们希望计算给定两个任务时参数的后验分布：

$$\log p(\theta|\mathcal{D}) = \log p(\mathcal{D}|\theta) + \log p(\theta) - \log p(\mathcal{D}), \tag{10.36}$$

其中 $\mathcal{D} = \mathcal{D}_A \cup \mathcal{D}_B$，$\theta$ 为模型参数。根据独立同分布假设，上式可以写为

$$\log p(\theta|\mathcal{D}) = \underline{\log p(\mathcal{D}_A|\theta)} + \log p(\mathcal{D}_B|\theta) + \underline{\log p(\theta)} - \underline{\log p(\mathcal{D}_A)} - \log p(\mathcal{D}_B), \tag{10.37}$$

$$= \log p(\mathcal{D}_B|\theta) + \log p(\theta|\mathcal{D}_A) - \log p(\mathcal{D}_B), \tag{10.38}$$

其中 $p(\theta|\mathcal{D}_A)$ 包含了所有从任务 $\mathcal{T}_A$ 的学习的信息。当顺序地学习任务 $\mathcal{T}_B$ 时，参数在两个任务上的后验分布和其在任务 $\mathcal{T}_A$ 的后验分布有关。

参考 [Bishop, 2007] 中第 4 章中的拉普拉斯近似。

由于后验分布比较难以建模，我们可以通过一个近似的方法来估计。假设 $p(\theta|\mathcal{D}_A)$ 为高斯分布，期望为在任务 $\mathcal{T}_A$ 上学习到的参数 θ_A^*，精度矩阵（即协方差矩阵的逆）是用参数 θ 的 Fisher 信息矩阵来衡量。

$$p(\theta|\mathcal{D}_A) = \mathcal{N}(\theta_A^*, F^{-1}), \tag{10.39}$$

其中F为Fisher信息矩阵。为了提高计算效率，F可以简化为对角阵，由Fisher信息矩阵对角线构成。

Fisher信息矩阵 *Fisher信息矩阵*（Fisher Information Matrix）是一种测量可观察一个似然函数$p(x|\theta)$携带的关于参数θ的信息量的方法。通常一个参数对分布的影响可以通过对数似然函数的梯度来衡量。我们定义打分函数$s(\theta)$为

$$s(\theta) = \nabla_\theta \log p(x|\theta), \tag{10.40}$$

则$s(\theta)$的期望为0。

证明.

$$\begin{aligned}
\mathbb{E}[s(\theta)] &= \int \nabla_\theta \log p(x|\theta) p(x|\theta) \mathrm{d}x && (10.41)\\
&= \int \frac{\nabla_\theta p(x|\theta)}{p(x|\theta)} p(x|\theta) \mathrm{d}x && (10.42)\\
&= \int \nabla_\theta p(x|\theta) \mathrm{d}x && (10.43)\\
&= \nabla_\theta \int p(x|\theta) \mathrm{d}x && (10.44)\\
&= \nabla_\theta 1 = 0. && (10.45)
\end{aligned}$$

□

$s(\theta)$的协方差矩阵称为Fisher信息矩阵，可以衡量参数θ的估计的不确定性。

$$\begin{aligned}
F(\theta) &= \mathbb{E}[s(\theta)s(\theta)^{\mathrm{T}}] && (10.46)\\
&= \mathbb{E}[\nabla_\theta \log p(x|\theta)(\nabla_\theta \log p(x|\theta))^{\mathrm{T}}]. && (10.47)
\end{aligned}$$

通常我们不知道似然函数$p(x|\theta)$的具体形式，Fisher信息矩阵可以用经验的分布来进度估计。给定一个数据集$\{x^{(1)}, \cdots, x^{(N)}\}$，Fisher信息矩阵可以近似为

$$F(\theta) = \frac{1}{N} \sum_{n=1}^{N} \nabla_\theta \log p(x^{(n)}|\theta)(\nabla_\theta \log p(x^{(n)}|\theta))^{\mathrm{T}}. \tag{10.48}$$

Fisher信息矩阵的对角线的值反映了对应参数在通过最大似然进行估计时的不确定性，其值越大表示该参数估计值的方差越小，估计更可靠性，其携带的关于数据分布的信息越多。

因此，对于任务 $\mathcal{T}_A$ 的数据集 $\mathcal{D}_A$，我们可以用 Fisher 信息矩阵来衡量一个参数携带的关于 $\mathcal{D}_A$ 的信息量。

$$F^A(\theta) = \frac{1}{N} \sum_{(\mathbf{x},y)\in\mathcal{D}_A} \nabla_\theta \log p(y|\mathbf{x},\theta)(\nabla_\theta \log p(y|\mathbf{x},\theta))^{\mathrm{T}}. \tag{10.49}$$

通过上面的近似，在训练任务 $\mathcal{T}_B$ 时的损失函数为

$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum_{i=1}^{N} \frac{\lambda}{2} F_i^A \cdot (\theta_i - \theta_{A,i}^*)^2, \tag{10.50}$$

其中 $\mathcal{L}_B(\theta)$ 为任务 $p(\theta|\mathcal{D}_B)$ 的损失函数，F_i^A 为 Fisher 信息矩阵的第 i 个对角线元素，λ 为平衡两个任务重要性的超参数，N 为参数的总数量。

参见习题10-3。

10.6 元学习

根据没有免费午餐定理，没有一种通用的学习算法在所有任务上都有效。因此，当使用机器学习算法实现某个任务时，我们通常需要“就事论事”，根据任务的特定来选择合适的模型、损失函数、优化算法以及超参数。那么，我们是否可以有一套自动方法，根据不同任务来动态地选择合适的模型或动态地调整超参数呢？事实上，人脑中的学习机制就具备这种能力。在面对不同的任务时，人脑的学习机制并不相同。即使面对一个新的任务，人们往往也可以很快找到其学习方式。这种可以动态调整学习方式的能力，称为*元学习*（Meta-Learning），也称为*学习的学习*（Learning to Learn）[Thrun and Pratt, 2012]。

元学习的目的是从已有任务中学习一种学习方法或元知识，可以加速新任务的学习。从这个角度来说，元学习十分类似于*归纳迁移学习*，但元学习更侧重从多种不同（甚至是不相关）的任务中归纳出一种学习方法。

和元学习比较相关的另一个机器学习问题是*小样本学习*（Few-shot Learning），即在小样本上的快速学习能力。每个类只有 K 个标注样本，K 非常小。如果 $K = 1$，称为*单样本学习*（One-shot Learning），如果 $K = 0$，称为*零样本学习*（One-shot Learning）。

这里我们主要介绍两种典型的元学习方法：*基于优化器的元学习*和*模型无关的元学习*。

10.6.1 基于优化器的元学习

目前神经网络的学习方法主要是定义一个目标损失函数 $\mathcal{L}(\theta)$，并通过梯度下降算法来最小化 $\mathcal{L}(\theta)$，

$$\theta_t \leftarrow \theta_{t-1} - \alpha \nabla \mathcal{L}(\theta_{t-1}) \tag{10.51}$$

其中θ_t为第t步时的模型参数，$\nabla\mathcal{L}(\theta_{t-1})$为梯度，$\alpha$为学习率。根据没有免费午餐定理，没有一种通用的优化算法在所有任务上都有效。因此在不同的任务上，我们需要选择不同的学习率以及不同的优化方法，比如动量法，Adam等。这些选择对具体一个学习的影响非常大。对于一个新的任务，我们往往通过经验或超参搜索来选择一个合适的设置。

参见第7.2节。

不同的优化算法的区别在于更新参数的规则不同，因此一种很自然的元学习就是自动学习一种更新参数的规则，即通过另一个神经网络（比如循环神经网络）来建模梯度下降的过程[Andrychowicz et al., 2016, Schmidhuber, 1992, Younger et al., 2001]。图10.2给出了基于优化器的元学习的示例。

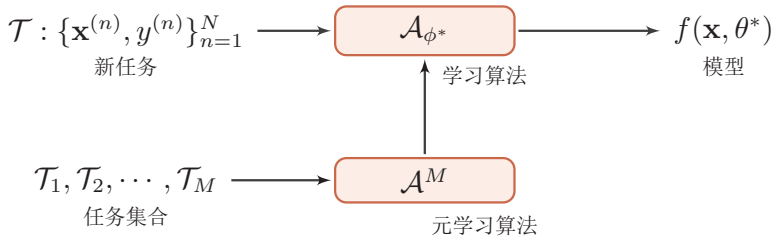


图 10.2 基于优化器的元学习

我们用函数$g_t(\cdot)$来预测第t步时参数更新的差值$\Delta\theta_t=\theta_t-\theta_{t-1}$。函数$g_t(\cdot)$称为优化器，输入是当前时刻的梯度值，输出是参数的更新差值$\Delta\theta_t$。这样第t步的更新规则可以写为

$$\theta_{t+1}=\theta_t+g_t\big(\nabla\mathcal{L}(\theta_t),\phi\big) \tag{10.52}$$

其中ϕ为优化器$g_t(\cdot)$的参数。

学习优化器$g_t(\cdot)$的过程可以看做是一种元学习过程，其目标是找到一个适用于多个不同任务的优化器。在标准梯度下降中，每步迭代的目标是使得$\mathcal{L}(\theta)$下降。而在优化器的元学习中，我们希望在每步迭代的目标是$\mathcal{L}(\theta)$最小，具体的目标函数为

$$\mathcal{L}(\phi)=\mathbb{E}_f\big[\sum_{t=1}^{T}w_t\mathcal{L}(\theta_t)\big], \tag{10.53}$$

$$\theta_t=\theta_{t-1}+\mathbf{g}_t, \tag{10.54}$$

$$[\mathbf{g}_t;\mathbf{h}_t]=\mathbf{LSTM}\big(\nabla\mathcal{L}(\theta_{t-1}),\mathbf{h}_{t-1},\phi\big), \tag{10.55}$$

其中T为最大迭代次数，$w_t>0$为每一步的权重，一般可以设置$w_t=1,\forall t$。由于LSTM网络可以记忆梯度的历史信息，学习到的优化器可以看做是一个高阶的优化方法。

在每步训练时，随机初始化模型参数，计算每一步的$\mathcal{L}(\theta_t)$，以及元学习的损失函数$\mathcal{L}(\phi)$，并使用梯度下降更新参数。由于神经网络的参数非常多，导致LSTM网络的输入和输出都是非常高维的，训练这样一个巨大的网络是不可行的。因此，一种简化的方法是为每个参数都使用一个共享的LSTM网络来进行更新，这样可以使用一个非常小的共享LSTM网络来更新参数。

10.6.2 模型无关的元学习

既然元学习的目标之一是快速学习的能力，即在多个不同的任务上学习学习一个模型，让其在新任务上经过少量的迭代，甚至是单步迭代，就可以达到一个非常好的性能，并且避免在新任务上的过拟合。

模型无关的元学习（Model-Agnostic Meta-Learning，MAML）是一个简单的模型无关、任务无关的元学习算法 [Finn et al., 2017]。假设所有的任务都来自于一个任务空间，其分布为$p(\mathcal{T})$，我们可以在这个任务空间的所有任务上学习一种通用的表示，这种表示可以经过梯度下降方法在一个特定的单任务上进行精调。假设一个模型为$f(\theta)$，如果我们让这个模型适应到一个新任务$\mathcal{T}_m$上，通过一步或多步的梯度下降更新，学习到的任务适配参数为

$$\theta'_m = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_m}(f_\theta), \tag{10.56}$$

其中α为学习率。这里θ'_m可以理解为关于θ的函数，而不是真正的参数更新。

MAML的目标是学习一个参数θ使得其经过一个梯度迭代就可以在新任务上达到最好的性能。

$$\min_\theta \sum_{\mathcal{T}_m \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_m}\big(f(\theta'_m)\big) = \sum_{\mathcal{T}_m \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_m}\Big(f\big(\underline{\theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_m}(f_\theta)}\big)\Big) \tag{10.57}$$

在所有任务上的元优化（Meta-Optimization）也采用梯度下降来进行优化，即

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{m=1}^{M} \mathcal{L}_{\mathcal{T}_m}(f_{\theta'_m}), \tag{10.58}$$

其中β为元学习率，这里为一个真正的参数更新步骤。这里需要计算关于θ的二阶梯度，但用一级近似通常也可以达到比较好的性能。

MAML 的具体过程如算法10.5所示。

算法 10.5: 模型无关的元学习

输入: 任务分布 $p(\mathcal{T})$;
最大迭代次数 T，学习率 α, β;

随机初始化参数 θ;
for $t = 1 \cdots T$ **do**
  根据 $p(\mathcal{T})$ 采样一个任务集合 $\{\mathcal{T}_m\}_{m=1}^{M}$ **for** $m = 1 \cdots M$ **do**
    计算 $\nabla_\theta \mathcal{L}_{\mathcal{T}_m}(f_\theta)$;
    计算任务适配的参数：$\theta'_m \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_m}(f_\theta)$;
  end
  更新参数：$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{m=1}^{M} \mathcal{L}_{\mathcal{T}_m}(f_{\theta'_m})$;
end

输出: 模型 f_θ

10.7 总结和深入阅读

目前，神经网络的学习机制主要是以监督学习为主，这种学习方式得到的模型往往是任务定向的，也是孤立的。每个任务的模型都是从零开始来训练的，一切知识都需要从训练数据中得到，导致每个任务都大量的训练数据。这些学习过程和人脑的学习方式是不同，人脑的学习过程一般不太需要太多的标注数据，并且是持续的学习，可以通过记忆不断地累积学习到的知识。本章主要介绍了一些和模型无关的学习方式。

关于集成学习可以参考《Pattern Recognition and Machine Learning》[Bishop, 2007] 和综述文献 [Zhou, 2012]。在训练神经网络时采用的 Dropout 方法在一定程度上也是一个模型集成。集成学习通过汇总多个模型来提高预测准确率的有效方法，代表性模型有随机森林 [Breiman, 2001] 和 AdaBoost[Freund et al., 1996]。

半监督学习研究的主要内容就是如何高效的利用少量标记数据和大量的未标记数据来训练分类器。相比于监督学习，半监督学习一般需要更少的标注数据，因此在理论和实际在运用中均受到了广泛关注。半监督学习可以参考综述 [Zhu, 2006]。最早在训练中运用未标记数据的方法是自训练（Self-Training）[Scudder, 1965]：在自训练的基础上，Blum and Mitchell [1998] 提出了由两个分类器协同训练的算法 Co-Training。该工作获得了国际机器学习会议ICML 2008 的10年最佳论文。

关于多任务学习是一种利用多个相关任务来提高模型泛化性的方法，可以参考文献 [Caruana, 1997, Zhang and Yang, 2017]。

关于迁移学习是研究如何将在一个领域上训练的模型，迁移到新的领域，使得新模型不用从零开始学习。但在迁移学习中需要避免将领域相关的特征迁移到新的领域 [Ganin et al., 2016, Pan and Yang, 2010]。迁移学习的一个主要研究问题是领域适应 [Ben-David et al., 2010, Zhang et al., 2013]。

终生学习是一种持续地学习方式，学习系统可以不断累积在先前任务中学到的知识，并在未来新的任务中能够利用这些知识 [Chen and Liu, 2016, Goodfellow et al., 2013, Kirkpatrick et al., 2017]。

元学习是主要关注于如何在多个不同任务上学习一种可泛化的快速学习能力 [Thrun and Pratt, 2012]。

上述这些方式都是目前深度学习中的前沿研究问题。

习题

习题 10-1 集成学习是否可以避免过拟合？

习题 10-2 分析自训练和 EM 算法之间的联系。

习题 10-3 根据最大后验估计来推导公式(10.50)。

参考文献

Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989, 2016.

Andrew Arnold, Ramesh Nallapati, and William W Cohen. A comparative study of methods for transductive transfer learning. In *icdmw*, pages 77–82. IEEE, 2007.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

Christopher M. Bishop. *Pattern recognition and machine learning, 5th Edition*. Information science and statistics. Springer, 2007. ISBN 9780387310732.

Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, 1998.

Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In *Advances in Neural Information Processing Systems*, pages 343–351, 2016.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

R. Caruana. Multi-task learning. *Machine Learning*, 28(1):41–75, 1997.

Zhiyuan Chen and Bing Liu. Lifelong machine learning. *Synthesis Lectures on Ar-*

tificial Intelligence and Machine Learning, 10(3):1–145, 2016.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

Robert M French. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135, 1999.

Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *Proceedings of the International Conference on Machine Learning*, volume 96, pages 148–156. Citeseer, 1996.

Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. Additive logistic regression: a statistical view of boosting. *The annals of statistics*, 28(2):337–407, 2000.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35, 2016.

Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

Arthur Gretton, Karsten M Borgwardt, Malte Rasch, Bernhard Schölkopf, and Alex J Smola. A kernel method for the two-sample-problem. In *Advances in neural information processing systems*, pages 513–520, 2007.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Thorsten Joachims. Transductive inference for text classification using support vector machines. In *ICML*, volume 99, pages 200–209, 1999.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114 (13):3521–3526, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative

pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*, 2018.

Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pages 759–766, 2007.

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

H Scudder. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371, 1965.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Sebastian Thrun. Lifelong learning algorithms. In *Learning to learn*, pages 181–209. Springer, 1998.

Sebastian Thrun and Lorien Pratt. *Learning to learn*. Springer Science & Business Media, 2012.

Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, pages 242–264. IGI Global, 2010.

Vladimir Vapnik. *Statistical learning theory*. Wiley, New York, 1998.

David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, pages 189–196, 1995.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

A Steven Younger, Sepp Hochreiter, and Peter R Conwell. Meta-learning with backpropagation. In *Proceedings of International Joint Conference on Neural Networks*, volume 3. IEEE, 2001.

Werner Zellinger, Thomas Grubinger, Edwin Lughofer, Thomas Natschläger, and Susanne Saminger-Platz. Central moment discrepancy (cmd) for domain-invariant representation learning. *arXiv preprint arXiv:1702.08811*, 2017.

Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. Domain adaptation under target and conditional shift. In *International Conference on Machine Learning*, pages 819–827, 2013.

Yu Zhang and Qiang Yang. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017.

Zhi-Hua Zhou. *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2012.

Xiaojin Zhu. Semi-supervised learning literature survey. *Computer Science, University of Wisconsin-Madison*, 2(3):4, 2006.