第 11 章

第 1 节　迁移学习

迁移学习是把一个领域（即源领域）的知识，迁移到另外一个领域（即目标领域），使得目标领域能够取得更好的学习效果；也可以把为任务 A 开发的模型作为初始点，重新使用在为任务 B 开发模型的过程中。形成知识或模型的重复利用的一种机器学习方法。

1. 迁移学习基本概念

1.1 绪论

人工智能（Artificial Intelligence，AI）是艾伦 • 图灵受其提出的“机器会思考吗?”这一著名图灵测试问题启发而得到的愿景。该图灵测试问题激励了几代研究人员探索机器智能运行的方法。然而纵观最近二三十年的发展，人工智能研究经历了多次起伏，其中大部分都围绕着机器如何从外部世界获取知识这一核心问题而演变。

从人工建立基于规则的知识库到从数据中进行机器学习，使机器像人类一样思考的尝试已经走了很长的路。目前，机器学习（machine learning）已经从一个模糊的学科发展成为一种推动工业和社会发展的重要力量，可以实现从电子商务和广告投放到教育和医疗等领域的自动化决策。由于机器学习具有使机器通过在标注和未标注的数据中进行学习和适应来获得知识的强大能力，因此它正在成为一种世界通用的技术。机器学习根据数据生成预测模型，因此往往需要高质量的数据作为"老师"来帮助调控统计模型。这种对未来事件进行准确预测的能力是基于对任务领域的观察和理解。训练样本中的数据通常是被标注的，也就是说训练样本中的观察和预测结果是相互耦合和相关的。之后，这些样本就能够被机器学习算法当成“老师”来“训练”可应用到新数据的模型。

现实生活中有许多应用机器学习的成功例子。基于计算机的图像分析领域中的人脸识别是一个很好的例子。假设我们已经获得了大量的医疗影像照片。那么，机器学习系统可以使用这些照片训练模型，从而判断新照片中是否有某种疾病的可能性。机器学习模型还可以应用于公司的保系统中，以判断访客是否是公司的员工。

尽管机器学习模型可以具有高质量，但它也有可能出错，尤其是当模型应用于有别于训练环境的场景中时。例如，如果从具有不同光照强度和不同程度的噪声（如阴影、不同角度的光照、路人的遮挡等）的室外环境拍摄照片，那么系统的识别能力将会显著下降。这是由于由机器学习系统训练的模型被应用于“不同”的场景。性能下降表明模型可能已过时，当出现新情况时需要及时更新。正是这种将模型从一种场景更新或者迁移到另一场景的需求体现了本书主题的重要性。

对迁移学习（transfer learning）的需求不仅仅局限于图像理解，通过自然语言处理（Natural Language Processing，NLP）技术理解 Twitter 文本消息是另一个例子。假设我们希望将 Twitter 消息根据用户的不同情绪（如高兴、伤心等）进行分类。那么，当使用来自青少年群体的一组 Twitter 消息构建一个模型然后将其用于成年人的新数据时，模型的性能就会急剧下降，因为不同群体很可能会以不同的方式表达他们的观点。

如上面的例子所述，在许多应用中使用机器学习的一个重要挑战是，现有的模型不能很好地适应到新的领域中。究其原因，训练数据量小、场景变化、任务变化等都可能导致这一问题。例如，在医学诊断和医学图像领域，短期内往往无法获得新病症的大量高质量训练数据用于模型的重训练。没有充足的训练数据的支撑，机器学习模型的性能往往不尽如人意。获得和标注新应用场景下的数据通常要花费很多精力和资源，这已然是现实生活中实现人工智能的一个主要障碍。打个比方，拥有一个精心设计却没有训练数据的 AI 系统就像一辆没有油或电的跑车。

上述讨论显示了将机器学习应用于实际场景的一个主要障碍：在应用机器学习算法前，我们无法获得各个领域的大量训练数据。除此之外，还有其他几个重要原因：

（1）许多应用场景数据量小。当前机器学习的成功应用依赖于大量有标签数据的可用性。然而，高质量有标签数据总是供不应求。传统的机器学习算法常常因为数据量小而产生过拟合问题，因而无法很好地泛化到新的场景中。

（2）机器学习模型需要强鲁棒性。传统的机器学习算法假设训练和测试数据来自相同的数据分布。然而，这种假设对于许多实际应用场景来说太强。在许多情况下，数据分布不仅会随着时间和空间而变化，也会随着不同的情况而变化，因此我们可能无法使用相同的数据分布来对待新的训练数据。在不同于训练数据的新场景下，已经训练完成的模型需要在使用前进行调整。

（3）个性化和定制问题。根据个人喜好和需求为每个用户提供个性化的服务是至关重要且具有经济效应的。在许多实际应用中，我们只能从单个用户收集到非常少的个人数据。因此，当我们尝试将通用的模型应用到特定的场景时，传统的机器学习算法会遇到冷启动问题。

（4）用户隐私和数据安全。在实际应用中，我们常常需要和其他组织合作，从而需要利用多个数据集。这些数据集通常属于不同的所有者，并且出于隐私或者安全考虑不能彼此泄露。当利用多个数据集构建同一模型时，我们希望提取每个数据集的"本质"并在构建模型中拟合它们。例如，如果能够在网络设备的"边缘"调整通用模型，那么就不需要上传存储在设备上的数据来增强该通用模型，因此边缘设备的隐私将得以保证。

智能系统的上述目标促进了迁移学习的发展。简而言之，迁移学习是一种机器学习范式，其算法能够从一个或多个应用场景中提取知识以帮助提高目标场景中的学习性能。与需要大量精心准备的训练数据作为输人的传统机器学习技术相比，迁移学习可以被理解为一种新的学习范式，我们将在本书对其进行详细介绍。除此之外，迁移学习也是解决许多大规模线上应用中数据稀疏性和冷启动问题的一种方式，例如，线上推荐场景中可能因为有标签用户评分数据太少而无法构建高质量的推荐系统。

迁移学习能够在应用开发初期以及技术上发展较少的地理场景中帮助提升 AI 的性能，即使在这些场景中没有太多的有标签数据。例如，假设我们希望在一个新的线上购物应用中构建书籍推荐系统，并且假设在这个全新的书籍购物领域没有太多的交易记录。那么，如果遵循监督学习（supervised learning）方法在这个全新的书籍领域中用不充分的训练数据构建预测模型，我们就无法对用户的下一次购买行为建立可信的预测。然而，通过迁移学习，可以找到相关的、开发完善的但是不同的领域寻求帮助，比如现有的电影推荐领域。利用迁移学习，可以找到书籍和电影两个领域之间的相似性和差异性。例如，一些作者会将他们的书拍成电影，因此这样的电影和书籍会吸引类似的用户群体。值得一提的是，领域之间的相似性允许模型聚焦于书籍推荐领域的独特之处进行优化，从而有效地利用数据集之间的潜在相似性。那么，书籍领域的分类和用户偏好模型就能够从电影领域进行优化。

基于迁移学习的方法，一旦我们在一个领域中获得了训练好的模型，就可以将这个模型引入其他类似的领域。因此，为了设计一个合理的迁移学习方法，找到不同领域任务间准确的“距离”度量方式是必需的。如果两个领域间的“距离”过大，那么我们可能不希望应用迁移学习技术，因为这样的学习可能产生一些负面影响。另一方面，如果两个领域非常“靠近”，则可以有效地应用迁移学习。

在机器学习中，领域之间的距离通常根据描述数据的特征来度量。在图像分析中，特征可以是图像中的像素或者区域，例如颜色和形状。在自然语言处理中，特征可以是单词或者短语。一旦了解两个领域非常接近，我们就能确保 AI 模型可以从一个已开发好的领域迁移到一个欠开发的领域，从而使 AI 应用更少地依赖数据。这对于成功的迁移学习应用来说是一个好的预兆。

能够将知识从一个领域迁移到另一个领域说明机器学习系统能够将其适用范围扩展到其源域外。这种泛化能力使得在 AI 能力或者计算能力、数据和硬件等资源相对匮乏的领域内，更加容易实现 AI 且 AI 更加鲁棒。在某种程度上，迁移学习可以促进 AI 成为一种更为包容的、为每个人服务的技术。

为了给出一个直观的例子，我们可以使用类比的方式突出迁移学习背后的关键要素。考虑在世界不同国家开车的情况。在美国和中国，驾驶员位置在汽车的左侧并且汽车靠右行驶。在英国，驾驶位置在汽车

右侧并且汽车靠左行驶。对于习惯在美国开车的人来说，在英国开车时，其驾驶习惯的转换尤为困难。然而，迁移学习能够帮助我们找到两个驾驶领域中的不变性并将其作为一种共同特征。仔细观察可以发现，无论驾驶员坐在哪边，其离道路中心始终是最近的。换言之，驾驶员坐在离路边最远的位置。这一事实能够使驾驶员将驾驶习惯顺利地从一个国家“迁移”到另一个国家。因此，迁移学习背后的关键要素是寻我不同领域和任务之间的“不变性”。

在人工智能领域，迁移学习已在知识重用、基于案例的推理、类比学习、领域自适应、预训体和微调等不同术语下得到了广泛研究。在教育和学习心理学领域，学习迁移与机器学习中的迁移学习有类似的概念。具体地说，学习迁移是指从之前的源任务中获得的历史经验可用于影响目标情境中的未来学习和表现[L. Thorndike and S. Woodworth，1901]。教育领域中的学习迁移和机器学习中的迁移学习有一个共同目标，即在一个场景中处理学习的过程，然后将学习应用到另一个场景中。在这两个领域中，学习到的知识或者模型都在进行了一定程度的适应后应用到后继的目标任务中。深入研究教育理论和学习心理学的文献[Eli，1965；Pugh and Bergin，2006;Schunk，1965；Cree and Macaulay，2000]可以发现，尽管机器学习中的迁移学习旨在赋予机器适应场景的能力，教育领域中的学习迁移试图研究人在教育中的适应性，但是二者在迁移的过程或者处理方式上却是相似的。

最后关于迁移学习的好处需要提及的是模拟技术。在诸如机器人和药物设计等复杂任务中，在真实的环境中进行试验的成本通常是非常昂贵的。在机器人领域，移动机器人或自动驾驶汽车需要收集大量的训练数据。例如，汽车碰撞可能有多种方式，但在现实生活中造成汽车碰撞的成本太高。相反，研究人员通常建立复杂的模拟器，以便在模拟环境训练出来的模型可以通过迁移学习应用到实际环境中。迁移学习的作用是考虑模拟环境中许多未知的未来情况，并使模拟环境中得到的预测模型（请如汽车自动型驶中的躲避障碍物模型）适应不可预见们未来情况。

1.2 迁移学习定义

首先，我们遵循文献[Pan and Yang，2010]中的符号定义“域”“任务”以及“迁移学习”，域$\mathbb{D}$由两部分组成：特征空间$\mathcal{X}$和边缘概率分布$\mathbb{P}^X$，其中每一个输人样本为$x \in \mathcal{X}$。一般来说，两个不同的域具有不同的特征空间或者边缘概率分布。在一个特定的域$\mathbb{D} = \{\mathcal{X}, \mathbb{P}^X\}$中，任务$\mathbb{T} = \{\mathcal{Y}, f(\cdot)\}$由标签空间罗和函数$f(\cdot)$组成。函数$f(\cdot)$是预测函数，能够对未知的样本{x"}进行标签预测。从概率的角度看，$f(\mathrm{x})$等价于$P(y|x)$。在分类问题中，标签可以是二值的，即$\mathcal{Y} = \{-1, +1\}$，也可以是离散值，即多个类。在回归问题中，标签具有连续值。

为简单起见，我们现在只关注只有一个源域（source domain）$\mathbb{D}$，和一个目标域（target do main）$\mathbb{D}_t$，的情况，两个域的场景是目前为止在文献中最普遍的研究对象。具体地说，我们定义$\mathcal{D}_s = \{(x_{s_i}, y_{s_i})\}_{i=1}^{n_s}$为源域有标签数据，其中$x_{s_i} \in \mathcal{X}_s$和$y_{s_i} \in \mathcal{Y}_s$分别为数据样本和对应的类别标签。类似地$\mathcal{D}_t = \{(x_{t_i}, y_{t_i})\}_{i=1}^{n_t}$为目标域有标签数据，其中$x_{t_i} \in \mathcal{X}_t$和$y_{t_i} \in \mathcal{Y}_t$分别为目标域中的输入和输出。在多数情况下，$0 \leq n_t \leq n_s$。基于以上定义的符号，迁移学习定义如下[Pan and Yang，2010]。

定义 1.1（迁移学习）.给定源域$\mathbb{D}_s$和学习任务$\mathbb{T}_s$、目标域$\mathbb{D}_t$和学习任务$\mathbb{T}_t$，迁移学习的目的是获取源域$\mathbb{D}_s$和学习任务$\mathbb{T}_s$中的知识以帮助提升目标域中的预测函数$f(\cdot)$的学习，其中$\mathbb{D}_s \neq \mathbb{D}_t$或者$\mathbb{T}_s \neq \mathbb{T}_t$。

1.3 迁移学习主要研究问题

要设计一个迁移学习算法，我们需要考虑以下三个主要的研究问题：何时迁移、迁移什么、如何迁移。

1.3.1 何时迁移

何时迁移寻求的是在什么情况下应该完成迁移技术。同样，我们对在哪些情况下不应该讦老知识也保持兴趣。在某些情况下，当源域和目标域彼此不相关时，强制迁移可能不会成功。在最坏的情况下，它甚至可能损害目标域中的学习性能。这种情况通常被称为负迁移（negative transfer）。目前关于迁移学习的大多数研究都集中在“迁移什么”和“如何迁移”上，隐含地假设源域和目标域彼此相关。但是，如何避免负迁移是一个引起越来越多关注的重要的开放性问题。

1.3.2 迁移什么

何时迁移寻求的是在什么情况下应该完成迁移技术。同样，我们对在哪些情况下不应该讦老知识也保持兴趣。在某些情况下，当源域和目标域彼此不相关时，强制迁移可能不会成功。在最坏的情况下，它甚至可能损害目标域中的学习性能。这种情况通常被称为负迁移（negative transfer）。目前关于迁移学习的大多数研究都集中在“迁移什么”和“如何迁移”上，隐含地假设源域和目标域彼此相关。但是，如何避免负迁移是一个引起越来越多关注的重要的开放性问题。

迁移什么决定了哪些部分的知识可以跨域或者跨任务进行迁移。某些知识是针对单个域或者任务的特定知识；某些知识在不同域之间也许是通用的，这样它们也许可以帮助提升目标域或目标任务的性能。请注意，术语“知识”是非常笼统的。因此，在实际中，需要根据不同的环境进行规范确定。

1.3.3 如何迁移

如何迁移指定迁移学习方法所采用的方式。根据对“如何迁移”问题的不同考量可以将迁移学习算法进行以下分类：

(1)基于样本的算法，其中迁移的知识对应于源样本中的权重；

(2)基于特征的算法，其中迁移的知识对应于源域和目标域中特征所共享的子空间；

(3)基于模型的算法，其中迁移的知识嵌入源域模型的一部分中；

2 迁移学习的流程

整个迁移学习过程如图 2.1 所示。图中左侧的过程是传统的机器学习过程，右侧为迁移学习过程。我们可以发现，迁移学习不仅利用目标任务中的数据作为学习算法的输入，还利用源域中的所有学习过程（包括训练数据、模型和任务）作为输入。该图显示了迁移学习的一个关键概念通过从源域获得更多知识来解决目标域中缺少训练数据的问题。

因为每个域由两部分组成，即$\mathbb{D} = \{\mathcal{X}, \mathbb{P}^X\}$，所以条件$\mathbb{D}_s \neq \mathbb{D}_t$意味着$\mathcal{X}_s \neq \mathcal{X}_t$或者$\mathbb{P}^{X_s} \neq \mathbb{P}^{X_t}$。类似地，任务也被定义为一对分量$\mathbb{T} = \{\mathcal{Y}, \mathbb{P}^{Y|X}\}$，所以条件$\mathbb{T}_s \neq \mathbb{T}_t$，意味着$\mathcal{Y}_s \neq \mathcal{Y}_t$或者$\mathbb{P}^{Y_s|X_s} \neq \mathbb{P}^{Y_t|X_t}$。当目标域和源域相同时，即$\mathbb{D}_s = \mathbb{D}_t$，并且其学习任务也相同时，即$\mathbb{T}_s = \mathbb{T}_t$，学习问题就成为一个传统的机器学习问题。

基于上述定义，我们可以制定不同的方式将现有的迁移学习方法进行分类归纳。例如，基于特征空间和/或标签空间是否同构，可以将迁移学习分为两类：同构迁移学习（homogeneous transfer learning）和异构迁移学习（heterogeneous transfer learning）。其定义描述如下[Pan，2014]。

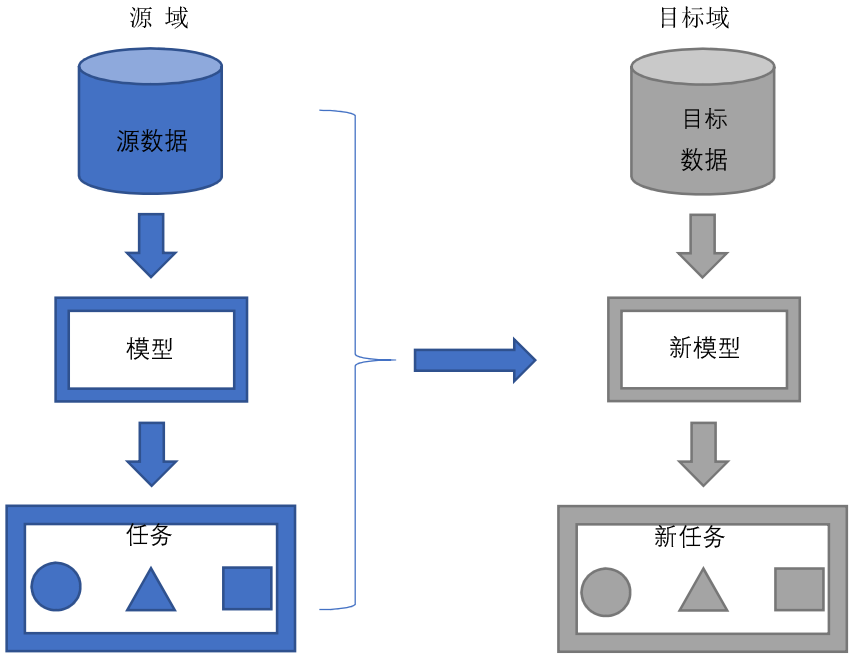

图 2.1 迁移学习过程示例

除了使用特征空间和标签空间的同构性外，还可以通过考虑目标域中是否有标签数据将现有的迁移学习方法分为以下三类：监督迁移学习（supervised transfer learning）、半监督迁移学习（semi-supervised transfer learning）、无监督迁移学习（unsupervised transfer learning）。在有监督迁移学习中，在目标域中仅有一些标签数据可用来训练，并且不使用未标注的数据进行训练。在无监督迁移学习中，目标域中只有无标签的数据可用。在半监督迁移学习中，假定在目标域中可获得足够的无标签数据和一些有标签数据。

2.1、同构迁移学习

定义 1.2（同构迁移学习）.给定源域$\mathbb{D}_s$和学习任务$\mathbb{T}_s$、目标域$\mathbb{D}_t$和学习任务$\mathbb{T}_t$，同构迁移学习的目的是获取源域$\mathbb{D}_s$和学习任务$\mathbb{T}_s$中的知识以帮助提升目标域中的预测函数$f_t(\cdot)$的学习，其中$\mathcal{X}_s \cap \mathcal{X}_t \neq \emptyset$且$\mathcal{Y}_s = \mathcal{Y}_t$，但$\mathbb{P}^{X_s} \neq \mathbb{P}^{X_t}$或$\mathbb{P}^{Y_s|X_s} \neq \mathbb{P}^{Y_t|X_t}$。

2.2、异构迁移学习

定义 1.3（异构迁移学习）.给定源域$\mathbb{D}_s$和学习任务$\mathbb{T}_s$、目标域$\mathbb{D}_t$和学习任务$\mathbb{T}_t$，异构迁移学习的目的是获取源域$\mathbb{D}_s$和学习任务$\mathbb{T}_s$中的知识以帮助提升目标域中的预测函数$f_t(\cdot)$的学习，其中$\mathcal{X}_s \cap \mathcal{X}_t = \emptyset$或$\mathcal{Y}_s \neq \mathcal{Y}_t$。

3 迁移学习常用方法

每一种迁移算法分别对应干知识的哪一部分被视为知识迁移的载体。具体而言，基于样本的迁移学习方法背后的普遍动机是，虽然源域中的有标签数据由于域差异而无法直接使用，但在重新加权或者重采样后，一部分数据能够被目标域重新使用。通过这种方式，权重大的源域有标签样本被视为跨域迁移的“知识”。基于样本的方法背后的隐含假设是源域和目标域具有许多重叠特征，这意味着域共享相同或者相似的支持。

但是，在许多实际应用中，源域和目标域中只有一部分特征空间重叠，这意味着许多特征不能被直接用于知识迁移的桥梁。因此，一些基于样本的算法可能无法有效地用于知识迁移。在这种情况下，基于特征的迁移学习方法更加实用。基于特征的方法背后的一个常见想法是为源域和目标域学习“良好”的特征表示，通过将数据映射到一个新表示形式上，可以重用源域中的有标签数据来精确地训练出目标域的分类器。在这种方式中，跨域迁移的知识可以被认为是学习到的特征表示。

基于模型的迁移学习方法假设源域和目标域共享学习方法的一些参数或者超参数。基于模型的方法的动机是预训练好的源模型已经捕获了许多有用的结构，这些结构具有通用性并且可以被迁移以学习更精确的目标模型，在这种方式中，被迁移的知识指的是模型参数内含的域不变结构。最近广泛使用的基于深度学习（deep learning）的迁移学习的预训练技术就是一种基于模型的方法。具体而言，预训练的想法是首先使用足够的可能与目标域数据不尽相同的源域数据训练深度学习模型。然后在模型被训练后，使用一些有标签的目标域数据对预训练的深度模型的部分参裂进行微调，例如，在固定其他层参数的同时精细调整若干层的参数。

3.1 基于样本的迁移学习

3.1.1 引言

直观上，基于样本的迁移学习重复使用源域中的有标签数据，以帮助训练出一个用于目标任务的更准确的模型。如果源域和目标域过于相似，那么可以将源域和目标域进行合并，将其转化为标准的单领域机器学习问题。然而，在许多应用中，这种“直接采用”源域样本的策略并不能很好地解决目标任务。

有些源域有标签数据有利于为目标域训练出更准确的模型，而有些数据则无法提升甚至损害模型，基于样本的迁移学习的主要动机就在于这些有用的数据。我们可以通过偏差和方差分析来理解这个动机。若目标域的数据集小，则模型的方差偏高、泛化误差大。通过加入部分源域数据作为辅助数据集，可以减小模型的方差。然而，如果源域和目标域的数据分布差距很大，则新模型的偏差会很高。因此，我们需要从源域中筛选出符合目标域数据的相似分布的样本，将它们运用于训练以降低新模型的偏差和方差。

简要来说，基于样本的迁移学习有两个关键问题。第一个问题是如何筛选出源域中与目标域数据具有相似分布的有标签样本；第二个问题是如何利用这些“相似”的数据训练出一个更准确的目标域上的学习模型。

一个域$\mathbb{D}=\{\mathcal{X},\mathbb{P}^X\}$由两个部分组成：特征空间$\mathcal{X}$和边缘分布$\mathbb{P}^X$。给定$\mathbb{D}$，一个任务$\mathbb{T}=\{\mathcal{Y},\mathbb{P}^{Y|X}\}$也由两个部分组成：标签空间$\mathcal{Y}$和条件概率分布$\mathbb{P}^{Y|X}$。大多数基于样本的迁移学习方法通常假设作为训练输人的源域样本和目标域样本有着相同或相似的支持，即大部分的样本有着相似的特征。同时，源任务和目标任务的输出标签需要一致。这个假设保证了知识可以通过样本进行跨领域迁移。结合域和任务的定义，这意味着在基于样本的迁移学习中，域间或任务间的不同之处分别在于域特征的边缘分布（$\mathbb{P}_s^X\neq\mathbb{P}_t^X$）或任务的条件概率分布（$\mathbb{P}_s^{Y|X}\neq\mathbb{P}_t^{Y|X}$）。

若$\mathbb{P}_s^X\neq\mathbb{P}_t^X$但$\mathbb{P}_s^{Y|X}\neq\mathbb{P}_t^{Y|X}$，则将问题称为非归纳式迁移学习（non-inductive transfer learning）。例如，一家医院希望利用该院病人的电子病历训练一个特定疾病的预测模型，在这里，每家医院就是一个域。由于在不同医院（域）就诊的病人群体的结构不同，不同域的边缘分布$\mathbb{P}^X$是不同的。然而，由于各家医院研究的是同一种疾病，不同域对应的任务的条件概率$\mathbb{P}^{Y|X}$是相同的。若$\mathbb{P}_s^{Y|X}\neq\mathbb{P}_t^{Y|X}$，则将问题称为归纳式迁移学习（inductive transfer learning）。假设之前的例子中的疾病是禽流感病毒，禽流感病毒会演变出具有不同发病原因的禽流感亚型（如 H1N1、H5N8）。将一家医院独立地为每种禽流感病毒亚型训练预测模型视为一个任务，由于不同亚型的发病原因不同，因此不同任务的条件概率$\mathbb{P}^{Y|X}$不同。

在非归纳式迁移学习中，由于不同域的条件概率是相同的，即$\mathbb{P}_s^{Y|X}\neq\mathbb{P}_t^{Y|X}$，可以证明，即使目标域的数据全都没有标签，利用源域中的有标签数据和目标域中的无标签数据仍然可以训练出最优的预测模型。在归纳式迁移学习中，由于不同任务的条件概率不同，为了进行条件概率或判别函数从源域到目标域的迁移，目标域中的部分数据需要被标注。

由于非归纳式迁移学习和归纳式迁移学习的假设不同，这两种迁移学习的设计也是不同的。接下来，我们将从细节上探讨两种迁移学习的动机、基本思想和代表性方法。

3.1.2 基于样本的非归纳式迁移学习

正如前文所述，在非归纳式学习中，假设源任务和目标任务是相同的，源域和目标域的输人样本的支持也应该是相同或相似的，即$\mathcal{X}_s=\mathcal{X}_t$。两个域的不同之处只在于输入样本的边缘分布，即$\mathbb{P}_s^X\neq\mathbb{P}_t^X$。在这些假设下，我们有源域的有标签数据集$\mathcal{D}_s=\{(x_{s_i},y_{s_i})\}_{i=1}^{n_s}$和目标域的无标签数

据集$\mathcal{D}_t=\{(x_{t_i})\}_{i=1}^{n_t}$。我们需要为目标域的未见数据训练出一个准确的预测模型。

接下来我们将证明在非归纳式迁移学习的假设下，可以在目标域没有任何有标签数据的条件下为目标域训练出一个最优的预测模型。假设我们需要为目标域训练一个以θ_t为参数的预测模型。根据经验风险最小化（empirical risk minimization）的框架[Vapnik，1998]，θ_t的最优解可以通过下述优化问题得到：

$$\theta_t^*=arg\min_{\theta_t\in\Theta}\mathbb{E}_{(x,y)\in\mathbb{P}_t^{X,Y}}(\ell(x,y,\theta)) \tag{3.1}$$

其中$\ell(x,y,\theta)$是θ_t的损失函数。由于目标域中没有有标签数据，式（3.1）并不能直接被优化。[Pan, 2014]已经证明，通过贝叶斯定理和期望的定义，式（3.1）可以被重写成

$$\theta_t^*=arg\min_{\theta_t\in\Theta}\mathbb{E}_{(x,y)\sim\mathbb{P}_s^{X,Y}}(\frac{P_t(x,y)}{P_s(x,y)}\ell(x,y,\theta_t)) \tag{3.2}$$

式（3.2）表示θ_t^*可以利用源域上的有标签数据通过最小化加权期望风险学习得到。在非归纳式迁移学习中，由于$\mathbb{P}_s^{Y|X}\neq\mathbb{P}_t^{Y|X}$，通过分解联合概率分布$\mathbb{P}^{X,Y}=\mathbb{P}^{Y|X}\mathbb{P}^X$，可以得到$\frac{P_t(x,y)}{P_s(x,y)}=\frac{P_t(x)}{P_s(x)}$。因此，式（3.2）可以进一步写成

$$\theta_t^*=arg\min_{\theta_t\in\Theta}\mathbb{E}_{(x,y)\sim\mathbb{P}_s^{X,Y}}(\frac{P_t(x)}{P_s(x)}\ell(x,y,\theta_t)) \tag{3.3}$$

源城样本 x 的权重为目标域和源域在数据点 x 处的输入样本的边缘分布比。给定源域有标签数据集$\{(x_{s_i},y_{s_i})\}_{i=1}^{n_s}$，通过定义$\beta(x)=\frac{P_t(x)}{P_s(x)}$，式（3.3）可以被近似写成

$$\theta_t^* = arg\min_{\theta_t \in \Theta} \sum_{i=1}^{n_s} \beta(x_{s_i})\, \ell\left(x_{s_i}, y_{s_i}, \theta_t\right) \tag{3.4}$$

因此，若要使用源域有标签数据来学习目标模型，则需要估计权重$\{\ \beta(x_{s_i})\}$，即密度比。如（3.4）所示，估计$\{\ \beta(x_{s_i})\}$并不需要有标签样本。一个简单的估计方法是首先估计$\mathbb{P}_t^X$和$\mathbb{P}_s^X$，然后对源域的每个样本 x，计算密度比$\frac{P_t(x_{s_i})}{P_s(x_{s_i})}$。然而，密度估计（density estimation）本身就是一个非常难的任务[Tsuboi et al, 2009]，尤其是数据维度较高时。这种情况下，由密度估计引起的错误将会影响密度比估计。

[Quionero-Candel et al, 2009]提出了一些跳过密度估计而直接估计比值$\frac{\mathbb{P}_t^X}{\mathbb{P}_s^X}$的方法。接下来的几节将会介绍如何通过几种典型方法直接估计密度比。

3.1.2.1、判别区分源数据和目标数据

学习权重的一个简单有效的方法是把估计边缘密度比的问题转化为判别一个样本是来自源域还是目标域的问题。这可以被视为一个二分类问题：来自源域的数据标注为 1，来自目标域的数据标注为 0。

例如，[Zadrozny, 2004]提出了一种基于拒绝采样（reject sampling）的方法来校正样本选择偏差。拒绝采样的过程定义如下。引入选择变量$\delta \in \{1, 0\}$，以$P_t(x)$的概率从边缘分布为$\mathbb{P}_t^X$的目标域抽取样本 x，其中$P_t(x) = P(x|\delta = 0)$。同样，$P_s(x)$可以写成$P_s(x) = P(x|\delta = 1)$。x 以$P(\delta = 1|x)$的概率被源域接受或以$P(\delta = 0|x)$的概率被源域拒绝。每一个数据样本 x 处的密度比可以写为

$$\frac{P_t(x)}{P_s(x)} = \frac{P(\delta=1)}{P(\delta=0)} \frac{P(\delta=0)}{P(\delta=1)} \frac{P_t(x)}{P_s(x)} \tag{3.5}$$

其中$P(\delta)$是δ在源域和目标域的联合数据集内的先验概率。通过贝叶斯定理，式（3.5）可以进一步写成

$$\frac{P_t(x)}{P_s(x)} = \frac{P(\delta=1)}{P(\delta=0)} \left(\frac{1}{P(\delta = 1|x)} - 1 \right) \tag{3.6}$$

因此，源域中的样本处的密度比的估算为$\frac{P_t(x)}{P_s(x)} \propto \frac{1}{P_{s,t}(\delta = 1|x)}$。我们把计算$P(\delta = 1|x)$看成一个二分类问题并且为其训练一个分类器。计算完成每一个源域样本处的密度比后，通过对源数据样本进行重加权或对源域数据集进行重要性采样即可训练出一个新模型。

按照[Zadrozny, 2004]的想法，[Bicke et al, 2007]提出了一个合并密度估计和模型训练及重加权的源数据样本的框架。$\mathbb{P}^X$表示在源域和目标域联合数据集内的样本 x 的密度比。我们可以用任何分类器来估计$P(\delta = 1|x)$。假设 v 表示分类器的参数，w 表示在重加权源域数据上训练的最终学习模型的参数，那么所有的参数可以通过最大后验估计（Maximum A Posterior，MAP）进行优化：

$$[w, v]_{MAP} = arg\max_{w,v} P(w, v|\mathcal{D}_s, \mathcal{D}_t) \tag{3.7}$$

其中$\mathcal{D}_s$和$\mathcal{D}_t$表示源域和目标域数据集。$P(w, v|\mathcal{D}_s, \mathcal{D}_t)$和$P(\mathcal{D}_s|w, v)\ \ P(\mathcal{D}_s, \mathcal{D}_t|v)\ \ P(w)\ \ P(v)$成正比。因此，可以通过最大化$P(\mathcal{D}_s|w, v)\ \ P(\mathcal{D}_s, \mathcal{D}_t|v)\ \ P(w)\ \ P(v)$找出 MAP 的解。

3.1.2.2、核平均匹配

分布的核嵌入（kernel embedding of distribution）是另一种估算密度比的有效方法[Smola et al, 2007a]。例如，[Huang et al, 2006]提出了核平均匹配（Kernel Mean Matching，KMM）方法，通过在再生核希尔伯特空间（Reproducing Kernel Hilbert Space，RKHS）内将源域数据样本的均值与目标域数据样本的均值对齐，直接学习密度比。

特别地，我们用β_i表示$\frac{P_t(x_i^s)}{P_s(x_i^s)}$，$x_i^s$为源域数据样本。定义$\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_{n_s})$，$n_s$是源域数据集的大小。KMM 利用最大均值差异[Gretton et al，2007]计算不同分布间的距离。给定两个样本，根据 MMD，两个样本的分布距离等于两个分布的均值元素在 RKHS 中的距离。因此，KMM 通过匹配重加权的源域和目标域在 RKHS 上的样本均值来学习源域样本的权重

$$\min_{\beta} \left\| \mu(\mathbb{P}_t^X) - \mathbb{E}_{\mathbb{P}_s^X}[\beta(x)\phi(x)] \right\| \qquad s.t. \qquad \beta(x) \geq 0, \mathbb{E}_{P_s^X}[\beta(x)\phi(x)] = 1 \tag{3.8}$$

其中ϕ将源域样本转化到 RKHS$\mathcal{F}$上，$\mu(\mathbb{P}_t^X)$表示目标域样本在 RKHS 上的期望，即$\mu(\mathbb{P}_t^X)=\mathbb{E}_{\mathbb{P}_t^X}[\phi(x)]$。

实际中，可以优化下面的经验目标：

$$\min_{\beta}\left\|\frac{1}{n_s}\sum_{i=1}^{n_s}\beta_i\phi(x_i^s)-\frac{1}{n_t}\sum_{i=1}^{n_t}\phi(x_i^t)\right\|^2 \qquad s.t. \qquad \beta_i\geq 0,\left|\frac{1}{n_s}\sum_{i=1}^{n_s}\beta_i-1\right|\leq\varepsilon \tag{3.9}$$

其中ε是正实数。优化$\boldsymbol{\beta}$后，将其代入式（3.4）中，通过特定的损失函数为目标域学习预测模型θ_t^*。

3.1.3　基于样本的归纳式迁移学习

与非归纳式迁移学习不同，在归纳式迁移学习中，源任务和目标任务的条件概率分布是不同的，即$\mathbb{P}_s^{Y|X}\neq\mathbb{P}_t^{Y|X}$。由于不同任务的条件概率不同，如果目标域中没有有标签数据，就很难通过$\mathbb{P}_s^{Y|X}$构造准确的$\mathbb{P}_t^{Y|X}$（如果可能的话）。因此，在大部分的归纳式迁移学习方法中，除了源域有标签数据集$\mathbb{D}_s=\{(x_{s_i},y_{s_i})\}_{i=1}^{n_s}$，还需要输人一个小的目标域有标签数据集$\mathbb{D}_t=\{(x_{t_i},y_{t_i})\}_{i=1}^{n_t}$。目标依然是为目标域中的未知数据学习一个准确的预测模型。

3.1.3.1　集成源损失和目标损失

为了同时利用源域和目标域中的有标签数据为目标域训练模型，一个直观的方法是将损失函数分为两个部分：一个用于源域有标签数据，另一个用于目标域有标签数据。通常引入协调参数平衡两种损失的影响。

在早期的代表性工作中，[Wu and Dietterich, 2004]提出了一种基于样本的 KNN 分类器对源域和目标域上的分类的准确性进行优化。在传统 KNN 分类器中，$h(x)$被定义为 k 个最邻近于测试样本 x 的训练样本。根据基于 KNN 的归纳式迁移学习方法，首先为源域的测试数据样本x_i^t定义K_s个最邻近源域样本和K_t个最邻近目标域样本。接下来对于每个类别标签 y，$V(y)$表示样本x_i^t对 y 的总投票，即$V(y)=\theta\left(\frac{V_t(y)}{K_t}\right)+(1-\theta)\left(\frac{V_s(y)}{K_s}\right)$，其中$V_t(y)$和$V_s(y)$分别表示$K_t$和$K_s$个最邻近样本对 y 的投票数，$\theta$是控制源域近邻和目标域近邻相对重要性的协调参数。

这样的思想可以被用于其他基分类器中。[Wu and Diecterich, 2004]也提出了一种基于支持向量机（Support Vector Machine, SVM）[Smola and Scholkopf, 2004]的归纳式迁移学习方法。SVM 的目标函数为

$$\min\textstyle\sum_j a_j+C\sum_j\varepsilon_j \qquad s.t.\,y_i\left(\sum_j y_j\alpha_j K(x_j,x_i)+b\right)\geq 1-\varepsilon_i\forall i,\alpha_j\geq 0\forall j \tag{3.10}$$

其中α_j是 SVM 的模型参数，ε_j是处理离群点的松弛变量，C 是表示对离群点重视程度的惩罚因子。在归纳式迁移学习框架中，考虑到源域和目标域中的有标签数据的不同，[Wu and Dietterich, 2004]提出调整目标函数和限制条件。假设α_j^s和ε_j^s分别表示源域样本x_j^s(j∈{1，…，n_s})的模型参数和松弛变量。相应地，α_j^t和ε_j^t表示目标域样本x_j^t(j∈{1，…，n_t})的模型参数和松弛变量，C_s和C_t是惩罚因子。修改后的 SVM 目标函数为

$$min\sum_{j=1}^{n_s}\alpha_j^s+\sum_{j=1}^{n_t}\alpha_j^t+C_s\sum_{j=1}^{n_s}\varepsilon_j^s+C_t\sum_{j=1}^{n_t}\varepsilon_j^t$$

$$s.t. \quad y_i^t\left(\sum_{j=1}^{n_t}y_j^t\alpha_j^tK(x_j^t,x_i^t)+\sum_{j=1}^{n_s}y_j^s\alpha_j^sK(x_j^s,x_i^t)+b\right)\geq 1-\varepsilon_i^t \qquad i\in\{1,\dots,n_t\},$$

$$y_i^s\left(\sum_{j=1}^{n_t}y_j^t\alpha_j^tK(x_j^t,x_i^s)+\sum_{j=1}^{n_s}y_j^s\alpha_j^sK(x_j^s,x_i^s)+b\right)\geq 1-\varepsilon_i^s \qquad i\in\{1,\dots,n_s\},$$

$$\alpha_j^t\geq 0 \quad j\in\{1,\dots,n_t\}, \quad \alpha_j^s\geq 0 \quad j\in\{1,\dots,n_s\},$$

总的来说，修改后的 SVM 能够同时优化源域和目标域的有标签数据的损失。

[Liao et al, 2005]进一步将上述想法扩展到逻辑回归中，并提出了迁移逻辑回归（Migratory Logit）算法。迁移逻辑回归通过为每个源域数据样本(x_i^s,y_i^s)引入新的“辅助变量”μ_i来建模两个域之间的差异。μ_i可以

被视为几何上的“截距项”，使得x_i^s对可以在目标域中向y_i^s迁移。它衡量了x_i^s与目标域分布$\mathbb{P}_t^X$的不匹配程度，并以此控制源域数据实例的重要性。对于目标域数据样本(x_i^t, y_i^t)，其标签y_i^t的后验概率与传统的逻辑回归是相同的，即$P(y_i^t | x_i^t; w) = \delta(y_i^t w^T x_i^t)$，其中$w$是参数向量，$\delta(a) = \frac{1}{1+\exp(-a)}$是 sigmoid 函数，对于源域样本$(x_i^s, y_i^s)$，$y_i^s$的后验概率定义为

$$P(y_i^s | x_i^s; w, \mu_i) = \delta(y_i^s w^T x_i^t + y_i^s \mu_i)$$

通过定义$\boldsymbol{\mu} = (\mu_1, \dots, \mu_m)^T$，对数似然估计为

$$\mathcal{L}(w, \boldsymbol{\mu}; \mathcal{D}_s \cup \mathcal{D}_t) = \sum_{i=1}^{n_t} \ln \delta(y_i^t w^T x_i^t) + \sum_{i=1}^{n_s} \ln \delta(y_i^s w^T x_i^s + y_i^s \mu_i)$$

然后所有的参数可以通过最大化对数似然估计得到，此时优化问题表示为

$$\max_{w,\mu} L(w, \boldsymbol{\mu}; \mathcal{D}_s \cup \mathcal{D}_t) \qquad s.t. \quad \frac{1}{n_s} \sum_{i=1}^{n_s} y_i^s \mu_i \le \mathrm{C}, \qquad y_i^s \mu_i \ge 0, \quad \forall i \in \{1,2,\dots,n_s\}$$

其中 C 是控制源域数据集整体重要性的超参数。

上述方法假设在目标域中只有有标签数据可以作为迁移学习算法的输入。在许多场景中，也可以在目标域中得到大量的无标签数据，[Jiang and Zhai, 2007]针对基于样本的归纳式迁移学习提出了一种半监督框架，其中目标域中的有标签数据和无标签数据都可以被用于训练目标预测模型。

[Jiang and Zhai, 2007]为源域样本$(x_i^s, y_i^s) \in \mathcal{D}_s$，引入了参数$\alpha_i$用于衡量$P_s(y_i^s | x_i^s)$和$P_t(y_i^s | x_i^s)$的差异。他们还引入了参数$\beta_i$用于估算密度比$\frac{P_t(x_i^s)}{P_s(x_i^s)}$。对于目标域中的无标签样本$x_i^{t,u} \in \mathcal{D}_t$和每一个可能的标签 y，参数$\gamma_i(y)$用于衡量$x_i^{t,u}$的真实标签为 y 的可能性。设集合$\mathcal{D}_t = \mathcal{D}_l \cup \mathcal{D}_u$其中$\mathcal{D}_l = \{(x_j^{t,l}, y_j^{t,l})\}_{j=1}^{n_{t,l}}$表示目标域有标签样本的子集，$\mathcal{D}_u = \{(x_k^{t,u})\}_{k=1}^{n_{t,u}}$表示目标域无标签样本的子集。为了对以$\theta$为参数的分类器进行优化，[Jiang and Zhai, 2007]提出了以下的优化问题：

$$\theta = arg \max_{\theta} \frac{\lambda_s}{C_s} \sum_{i=1}^{n_s} \alpha_i \beta_i \log P(y_i^s | x_i^s; \theta) + \frac{\lambda_{t,l}}{C_{t,l}} \sum_{j=1}^{n_{t,l}} \log P\left(y_j^{t,l} | x_j^{t,l}; \theta\right)$$

$$+ \frac{\lambda_{t,u}}{C_{t,u}} \sum_{k=1}^{n_{t,u}} \sum_{y \in y} \gamma_k(y) \log P(y | x_k^{t,u}; \theta) + \log P(\theta)$$

其中$C_s = \sum_{i=1}^{n_s} \alpha_i \beta_i$，$C_{t,l} = n_{t,l}$，$C_{t,u} = \sum_{k=1}^{n_{t,u}} \sum_{y \in y} \gamma_k(y)$是归一化因子。正规化参数$\lambda_s$、$\lambda_{t,l}$和$\lambda_{t,u}$控制每一部分的相对重要性，其和为 1。$P(\theta)$表示$\theta$的标准先验。这样，源域中的有标签数据、目障中的有标签数据、目标域中的无标签数据都将被用于学习θ的最优解。

3.1.3.2 Boosting 风格方法

基于 Boosting 的算法是另外一种归纳式迁移学习方法，即通过迭代更新源域样本权重来找到那些误导的源域样本。例如，[Dai et al, 2007b]提出的 TrAdaBoost 算法是第一个应用于基于样本的归纳式迁移学习的 Boosting 风格算法。

为了找到源域中的有用样本，TrAdaBoost 使用与 AdaBoost 中相似的重加权策略，TrAdaBoost 首先在$\mathcal{D}_s$和$\mathcal{D}_t$的联合数据集上训练一个模型 h。接看它利用 h 对目标域数据进行预测并计算在目标域上的平均损失，即$\varepsilon = \frac{\sum_{i=1}^{n_t} w_i^t l(h(x_i^t), y_i^t)}{\sum_{i=1}^{n_t} w_i^t}$，其中$w_i^t$是$x_i^t$的权重，$l(\cdot\ ,\ \cdot)$是损失函数。对于每一个目标域样本，其权重更新为$w_i^t = w_i^t \beta^{-l(h(x_i^t), y_i^t)}$，其中$\beta = \varepsilon/(1-\varepsilon)$。如果某个目标域数据样本有着较高的损失，那么它的权重将在下一次迭代中增加，这与 AdaBoost 是一致的。

对于每一个源域样本，如果它损失较高，则会不利于目标任务，因此在下一次迭代中它的权重会被降低。源域样本的权重将被更新为$w_i^s = w_i^s \theta^{l(h(x_i^s), y_i^s)}$，其中$\theta = 1/(1 + \sqrt{2 \ln n_s/(n_s + n_t)})$。

通过以上的权重更新方法，TrAdaBoost 可以减小源域样本中的误导数据样本的影响，并为目标域学习

一个集成分类器。

3.2 基于特征的迁移学习

3.2.1 引言

如前一章所述，基于样本的迁移学习方法有一个普遍的假设，即源域数据和目标域数据有相似或者相同的支持。但是，这样的假设可能过于苛刻而无法满足，因为在许多真实场景中，源域数据和目标域数据的特征往往不重叠。例如，考虑客户对不同产品的评论的情感分类问题。在这个问题下，对每种产品的评价都可以被称为一个域，其中客户可能使用常用词或者某领域的特定词来表达他们的意见。举例来说，在DVD 领域中，“无聊”这个词会用来表达负面情绪，但是在家具领域却从来不使用该词。因此，一些词或特征只会出现在一些特定的域中，而不会出现在其他域中。这意味着某些特征是源（目标）域特定使用的，而在其对应的目标（源）域中不会使用。在这种情况下，重新加权或者重新采样的样本并不能减少域之间的差异。为了解决上述问题，在本章中，我们引入了另一种迁移学习的方法，即基于特征的迁移学习。该方法在抽象的“特征空间”实现迁移，而非原始输人空间。值得注意的是，在某些极端情况下，源域和目标域之间可能没有重叠的部分，但在这样的两个特征空间之间可能存在一些“转换器”来成功实现迁移学习。

在基于特征的迁移学习方法中有一个常见的想法，即学习一对映射函数$\{\varphi_s(\cdot),\varphi_t(\cdot)\}$，将来自源域和目标域的数据映射到共同的特征空间，从而使域之间的差异性减少。然后使用映射之后的源域和目标域数据在新的特征空间上训练目标分类器。为了测试目标域上的未见数据，首先需要将新的数据映射到新的特征空间上，然后执行训练好的目标分类器进行预测。

对于基于特征的迁移学习，不同方法背后关于学习特征映射函数的动机和假设是不同的。在本章中，我们将这些方法总结为三类。第一类方法旨在通过最小化域间差异来学习给定目标域和源域的可迁移特征。第二类方法旨在学习所有域都通用的高质量特征。第三类方法基于跨域的“特征增强”方法，它通过考虑从数据中学习到的额外相关性来扩展特征空间。

3.2.2 最小化域间差异

在许多实际应用中，观察到的高维数据样本通常由一组隐变量或组成部分控制，这些因素被称为特征。域之间的差异可能是仅由这些特征中的一个子集导致的。如果可以识别不会导致域间差异的隐特征，并且使用它们来表示跨域的数据样本，那么就可以利用具有新的特征表示的源域训练数据来训练目标域的准确分类器。因此，对于基于特征的迁移学习来说，如何学习这样的域不变特征，或者等效地，如何学习域间的特征映射函数$\{\varphi_s(\cdot),\varphi_t(\cdot)\}$以将不同域的样本映射到由域不变特征组成的公共空间，是十分重要的。学习这种域不变特征的一个关键问题是如何测量“域不变性”。到目前为止，研究者已经提出了若干个度量标准来测量学习特征的域不变性，这将在下面的章节中进行讨论。

3.2.2.1 最大均值差异

最大均值差异是一种非参数度量，用于在再生核希尔伯特空间[Gretton et al]中基于核嵌入来度量分布之间的距离。给定两个分别来自两个分布的域样本X_s(源)和X_t(目标)，其 MMD 距离根据经验估计如下：

$$MMD(X_s,X_t)=\left\|\frac{1}{n_s}\sum_{i=1}^{n_s}\phi(x_i^s)-\frac{1}{n_t}\sum_{i=1}^{n_t}\phi(x_i^t)\right\|_{\mathcal{K}} \tag{3.11}$$

其中，$\phi(x)$将每个实例映射到与核$k(x_i,x_j)=\phi(x_i)^T\phi(x_j)$相关联的希尔伯特空间$\mathcal{K}$，$n_s$和$n_t$分别是源域和目标域的样本大小。通过使用核函数（kernel function），式（3.11）中的 MMD 距离可似化简为

$$MMD(X_s,X_t)=tr(KL) \tag{3.12}$$

其中K $=\begin{bmatrix}K_{s,s} & K_{s,t}\\ K_{s,t}^T & K_{t,t}\end{bmatrix}\in\mathbb{R}^{(n_s+n_t)\times(n_s+n_t)}$是一个复合核矩阵，由分别在源域、目标域和交叉域中的核矩阵$K_{s,s}$、$K_{t,t}$、$K_{s,t}$组成。L 是一个矩阵，其元素l_{ij}定义如下：

$$l_{ij}=\begin{cases}\frac{1}{n_s^2} & x_i,x_j\in X_s\\ \frac{1}{n^2} & x_i,x_j\in X_t\\ -\frac{1}{n_s n_t} & \text{其他}\end{cases}$$

3.2.2.2 MMD 的深度架构

在深度学习的背景下，研究者提出使用深度神经网络来近似由核函数引起的特征映射$\varphi(\cdot)$，例如，[Tzeng et al, 2014]提出编码 MMD 来测量在卷积神经网络（Convolutional Neural Network，CNN）中学习到的隐藏特征之间的距离。通过这种方式，网络通过最大化标签依赖性同时最小化域不变性来自动学习跨城表示。基本深度架构如图 3.1 所示，其中源域的输入数据$x_s\in X_s$，和目标域的输入数据$x_t\in X_t$为由 CNN 前几层转换后的数据。前几层的转换可以被认为是 MMDE 中$\psi(\cdot)$的一种近似。

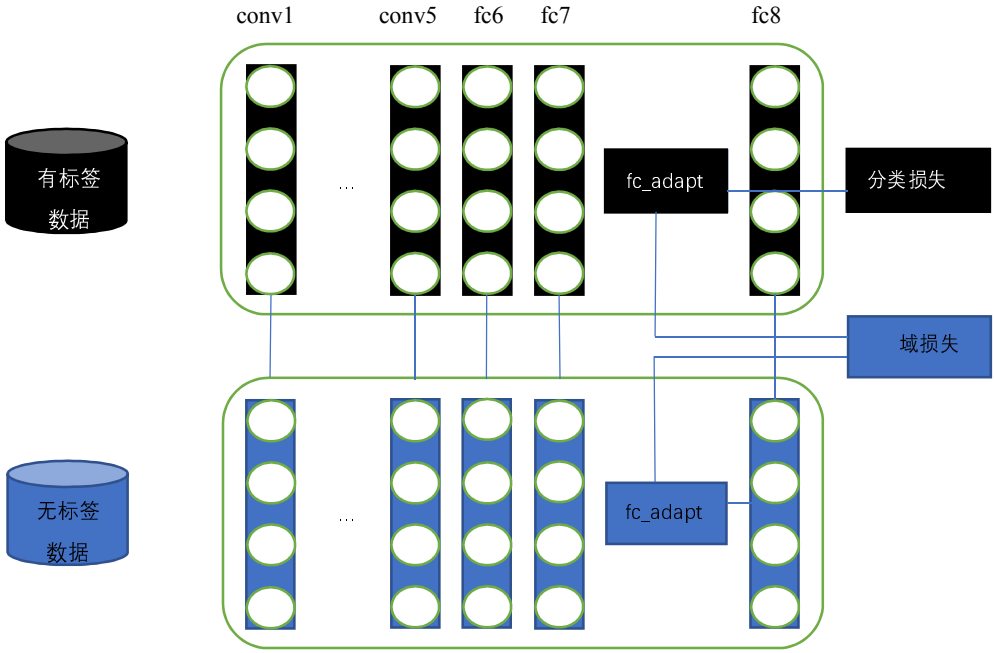

图 3.1 用于分类损失和域不变性的深度卷积神经网络（改编自[Tzeng et al., 2014]的论文，虚线表示权值共享）

作为后续工作，[Long et al, 2015]提出了一种多核 MMD（Multi-Kernel MMD，MK-MMD）度量作为计算 MMD 距离的替代方案，来测量深度学习模型的域差异。其基本思想是使用多个 PSD 内核来计算 MMD 距离，这应该能够为神经网络提供一个更灵活、更鲁棒的距离测量方法来学习跨域特征表示。

为了在测量域差异时考虑标签信息，[Long et al, 2017]根据源域的联合分布函数$P(X_s,X_s)$和目标域的联合分布函数$P(X_t,Y_t)$提出了联合分布散度（Joint Distribution Discrepancy，JDD）。其得到的联合自适应网络（Joint Adaptation Networks，JAN）的架构如图 3.2 所示。从图中可以看出，不仅最后的隐藏层参与了 JAN 中 JDD 标准的计算，所有的全连接层和输出层也参与其中。

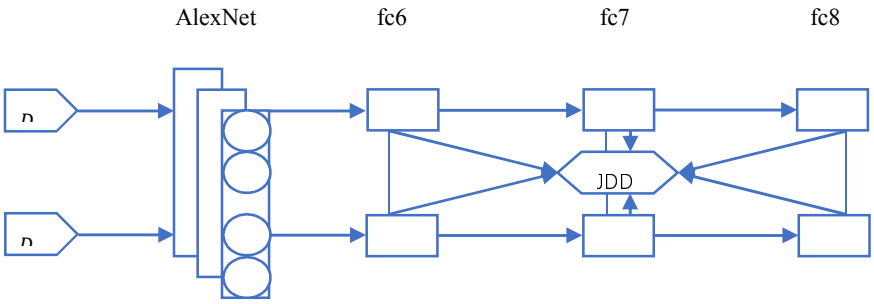

图 3.2 基于 AlexNet 的 JAN 的架构（改编自 Long 等人（2017）的论文）

3.2.2.3 使用特定分布假设的度量

通过假设数据遵循高斯分布，[Castrejon et al, 2016]研究使用特征激活的统计信息来学习迁移学习方法中的跨模态场景表示。他们提出的方法将用于具有不同风格的图像的跨模态卷积神经网络和用于语言模型的多层感知机（Muti-Layer Perceptron，MLP）进行正则化，使得它们具有模态不可知的共享表示。

此外，其提出的方法进一步引入了关于激活函数的正则化项，使激活函数在中间隐藏层中具有跨模态的相似统计。设$P_i(h)$为第 i 隐藏层的激活函数的分布，正则化项则可以通过负对数来计算：$\mathcal{R}_i = -\ln P_i\,(h;\theta_i)$，其中$\theta_i$表示超参数。通过将$P_i$的正态分布实例化为$P_i(h;\mu,\Sigma)\sim N(\mu,\Sigma)$，正则化项$R_i(h)$计算如下：

$$P_i(h;\mu_i,\Sigma_i) = \frac{1}{2}(h-\mu_i)^T\Sigma_i^{-1}(h-\mu_i)$$

另外，可以使用混合高斯分布来定义P_i，它比单一的高斯分布更灵活。

3.2.3 学习通用特征

上一节中介绍的大部分工作旨在学习源域和目标域中的域不变性特征，这些特征需要事先给出。另外一种特征学习方法的分支旨在从若干个域中学习通用特征表示。由于这些特征适用于所有纳入考虑范围的领域，因此被称为“通用的”。这一想法部分受到自学习（self-taught learning）[Raina et al., 2007]的启发，其目的在于从大量无标签数据中学习通用特征表示，其中无标签数据的真实标签可能和目标任务的数据标签不同。自学习被应用于图像分类问题中。大多数这样的方法包含三个步骤：从源域或者辅助域的无标签数据中学习高级特征；用学习到的高级特征表示目标域的有标签数据；利用目标域中有标签数据的新表示来训练分类器。

请注意，给定多个源或辅助域，也可以通过调整多任务特征学习方决来学习通用特征表示[Argyriou et al., 2006;Zhang and Yang, 2017b]。在多任务特征学习中，可以跨不同任务学习共同特征。这些共同特征可以被视为其他任务的通用特征。

3.2.3.1 深度通用特征

受到通过稀疏编码（sparse coding）来学习通用特征的后发，[Glorot et al., 2011]提出应用深度自动编码器（deep autoencoder）来学习高级特征作为通用特征。具体来说，给定一个输人 x，深度编码器$f(\cdot)$将其映射为隐藏编码$h = f(x)$，而深度解码机$g(\cdot)$旨在使用隐藏编码通过$\hat{x} = g(h)$重构输入。由于编码器和解码器使用不同的辅助域进行训练，编码器的输出h被认为是对于每个输入样本的一种通用特征表示。[Chen et al,2012b]进一步提出了自动编码器的变体即边缘化堆叠去噪自动编码器（marginalized Stacked Denoising Autoencoder，mSDA），以提高跨域学习通用特征的效率和有效性。

除了使用稀疏编码和自动编码器中的重构损失来学习通用特征外，一些研究者提出在辅助任务上使用聚类来学习通用特征。与重构损失相比，聚类是在复杂性方面轻量级的无监督学习方法。它还可以增加学习到的特征表示的可解释性。

假设神经网络中一层神经层的表示为一个 4 维张量$Y \in R^{N\times C\times H\times W}$，其中 N、C、H 和 W 分别为批大小、隐藏单元的数量、特征表示的高度以及相应的宽度。具体来说，通过将每个数据实例展开成一个矩阵$T^{\{N\}\times\{H,W,C\}}$，样本聚类的损失函数定义如下：

$$\mathcal{R}_{spatial}(Y,\mu) = \frac{1}{2NCHW}\sum_{n=1}^{N}\left\|T^{\{N\}\times\{H,W,C\}}(Y)_n - \mu_{z_n}\right\|^2 \tag{3.13}$$

实例的表示可以认为是一个 C 通道的“图像”。C 通道包含的像素可以通过空间聚类聚类如下：

$$\mathcal{R}_{spatial}(Y,\mu) = \frac{1}{2NCHW}\sum_{i=1}^{NHW}\left\|T^{\{N,H,W\}\times\{C\}}(Y)_i - \mu_{z_i}\right\|^2 \tag{3.14}$$

另外，可以通过使用以下损失函数在通道上执行聚类。

$$\mathcal{R}_{spatial}(Y,\mu) = \frac{1}{2NCHW}\sum_{i=1}^{NC}\left\|T^{\{N,C\}\times\{H,W\}}(Y)_i - \mu_{z_i}\right\|^2 \tag{3.15}$$

[Liao et al., 2016]着重研究了聚类的表示是否适用于没有训练过的类别，这是一个零样本学习同题。给定使用式（3.13）中的损失函数训练的特征，可以通过结构化 SVM 来学习输出嵌入 E 面

无须正则化：

$$\min_{E} \frac{1}{N}\sum_{n=1}^{N} \max_{y\in Y} \left(0, \Delta\left(y_n, y\right) + x_n^T E\left(\phi(y) - \phi(y_n)\right)\right) \tag{3.16}$$

其中x_n和y_n分别是第 n 个样本的特征和类标签，Δ是 0-1 损失函数，ϕ是 CUB 数据集提供的类属性矩阵，每个输入表示一个属性在给定类别中存在的可能性。

3.2.4　特征增强

[Daume, 2007]提出了一种简单的域适应方法，该方法使用特定域的信息来增强源域和目标域数据的特征向量，并将其视为学习算法的新输入。

将$\mathcal{X}$和$\mathcal{Y}$分别定义为输入和输出空间。假设原始输入空间为$\tilde{\mathcal{X}} \in \mathbb{R}^F$，则其提出的方法将原始输入空间增强到$\tilde{\mathcal{X}} \in \mathbb{R}^{3F}$。源域和目标域的映射函数$\Phi^s$、$\Phi^t$：$\mathcal{X} \to \tilde{\mathcal{X}}$定义为

$$\Phi^s(x) = \langle x, x, \mathbf{0}\rangle，\ \Phi^t(x) = \langle x, \mathbf{0}, x\rangle \tag{3.17}$$

其中，**0** 表示 F 维空间中的零向量。增强特征的第一部分为原始特征，第二部分和第三部分分别表示源域和目标域的特定特征。

可以容易地将上述方法泛化为核函数版本。假设每个数据点 x 被投射到具有相应内核 k：$\mathcal{X} \times \mathcal{X} \to \mathcal{R}$的 RKHS 中，k 可以写作两个向量的点乘形式，即$k(x, x') \le \langle \Phi(x), \Phi(x')\rangle_x$那么$\Phi^s$和$\Phi^t$定义如下

$$\Phi^s(x) = \langle\ \Phi(x), \Phi(x), 0\rangle,\quad \Phi^t(x) = \langle\ \Phi(x), 0, \Phi(x)\rangle \tag{3.18}$$

用$\tilde{k}(x, x')$表示扩展的内核。当x和x'来自同一域中时，$\tilde{k}(x, x') = \langle \Phi(x), \Phi(x')\rangle_x + \langle \Phi(x), \Phi(x')\rangle_x = 2k(x, x')$。当$x$和$x'$来自不同域时，宸$\tilde{k}(x, x') = \langle \Phi(x), \Phi(x')\rangle_x = k(x, x')$。

考虑到以核作为相似性的度量方法，上述核公式直观可见，因为来自相同域的数据点本质是跨域点的两倍。考虑到对目标数据的测试，目标域的训练数据的影响应该是源点的两倍。

请注意，上述特征增强方法将假设分解为三个子假设，即$h = \langle h_c, h_s, h_t\rangle$，这相当于学习两个特定域的假设$w_s = h_c + h_s$和$w_t = h_c + h_t$。通过假设$w_s$和$w_t$对每个无标签目标域实例$x_i$达成致，该方法可以自然地扩展到半监督学习：

$$w_s \cdot x_i \approx w_t \cdot x_i \Leftrightarrow \langle h_c, h_s, h_t\rangle \cdot \langle 0, x_i, -x_i\rangle \approx 0 \tag{3.19}$$

通过这种方式，可以构建无标签数据的特征映射：

$$\Phi^u(x) = \langle 0, x, -x\rangle$$

在此之后，任何半监督学习分类器都可以应用于为源域有标签数据、目标域有标签数据和目标域无标签数据定义的特征映射。

3.3　基于模型的迁移学习

3.3.1　引言

基于模型的迁移学习（mode-based transfer learning）也称为基于参数的迁移学习（parameter-based transfer learning），其假设是在模型层次上源任务和目标任务共享部分通用知识。顾名思义，所迁移的知识被编码到模型参数、模型先验知识、模型架构等模型层次上。因此，基于模型的迁移学习的核心目标是明确源域模型的何种部分有助于目标域模型的学习。

与基于样本的迁移学习（instance-based transfer learning）和基于特征的迁移学习（feature-based transfer learning）一样，基于模型的迁移学习也利用源域的知识。然而，三者在利用知识的层面上存在显著差别：基于模型的迁移学习利用模型层面的知识，而基于样本的迁移学习与基于特征的迁移学习分别利用样本和特征层面的知识。直观地，重新使用从源域中学习到的模型以避免再次抽取训练数据或再对复杂的数据表示进行关系推理，这使得基于模型的迁移学习更同效，更能抓住源域的高层级知识。

假设通过良好训练，源域模型θ_s已经从数据中学习到了大量的结构知识。通过迁移该结构识到与源域相似的目标域中，在使用少量的目标域有标签数据情况下，可获得更为精准的目标域模型θ_t。在目标域仅有少量训练样本的情况下，避免过拟合风险的唯一方法是学习简单模型。然而，如果借助于源域已训练模

型，那么尽管目标域中仅有有限的训练样本，仍可获得性能更强的模型。

大部分基于模型的迁移学习算法都是在可归纳的过移学习设置中提出的，其假设目标域有有标签的样本。值得关注的是，通过调整基于模型的多任务学习方法可获得相应的基于模型的迁移学习算法。再次重申多任务学习与迁移学习的区别在于，多任务学习试图同时优化多个目标任务的性能，而迁移学习只关注通过利用辅助住芳的知识来提高一个目标域的性能。例如，[Evgeniou and Pontil, 2004]提出了一种基于支持向量机的正则化多任务学习方法。在该方法中，优化目标是最小化所有任务的平均损失，这使得最终模型获得在所有任务中达到平衡的最佳总体性能。然而，这一结果并不能保证理想目标任务上的最优表现。而迁移学习仅关注了目标任务的表现。通过修改多任务学习中目标函数的不同任务的权重分配，可以消除其中的差异。

第一类方法包含了如下一类方法：通过重新利用源域中的模型成分或者源域中的超参数来确定目标域模型。

第二类方法是通过正则化来迁移知识。正则化是一种解决不适定的机器学习问题的技术，也是一种通过限制模型灵活性来防止模型过拟合的技术。该类方法中，在一些先验假设下，正则化约束了模型的超参数。其中，SVM 由于具有良好的计算性能以及在一些应用中有良好的预测表现，目前已被广泛应用于基于正则化的知识迁移中。随着深度模型的引入，一些方法将模型参数从辅助任务转移到预先训练的深度学习模型中，以初始化目标域模型。

3.3.2 基于共享模型成分的迁移学习

先验概率分布也称为先验，是一种概率分布，指在看到任何证据之前对一些不确定事件概率的判断。例如，想象你正在和朋友玩抛硬币游戏，其游戏规则是：如果硬币正面朝上则你赢，否则你输。在掷硬币之前，你对结果是正面或者反面朝上进行下注。由于你知道正面和反面朝上的概率是均等的，你很可能任意地选择一面。然而，如果你知道正面朝上的可能性较大，你就更有可能把赌注押在正面。在这个例子中，先验是硬币正面朝上的概率。

先验能够在你做决策之前给你一个更好、更高效的评估，所以你不必通过抛多次硬币来判断哪一面更可能朝上。同样，在现实应用中，如果能够将一些先验知识应用到一个新任务中，那么即便新任务仅拥有少量的训练数据，也能够获得一个在性能上令人满意的模型。

3.3.2.1 利用高斯过程进行迁移学习

本小节将首先简要介绍高斯过程（Gaussian Process，GP），随后介绍一些文献[Lawrence and Platt, 2004；Schwaighofer et al., 2005；Bonilla et al., 2007]中提出的方法如何利用高斯过程在不同任务间共享知识。

高斯过程是使用高斯先验建模数据分布的一种通用工具。它是一种随机过程，随机变量的每个有限子集都服从多元正态分布。在监督学习中，依靠训练数据间的相似性度量，高斯过程能够预测未见数据的标签。具体地，设有标签的数据集为$X = [x_1, x_2, \dots, x_N,]^T$，定义潜变量$z = [z_1, z_2, \dots, z_N,]^T$，该潜变量的先验分布为如下形式的高斯先验：

$$p(z|\boldsymbol{X}, \theta) = N(\boldsymbol{0}, \boldsymbol{K}) \tag{3.20}$$

其中，θ是参数。$\boldsymbol{K}$是协方差函数（称为核（kernel）），描述一个多元正态分布。$\boldsymbol{K}$可以采取不同的形式，如线性核$K(x, x') = x^T x'$和平方指数核$K(x, x') = \sigma^2 \exp(-\frac{\|x-x'\|^2}{2\ell^2})$。在平方指数核中，参数$\sigma$和$\ell$包含在预估的$\theta$中。

在一个高斯过程中，给定变量 z，y 和$\boldsymbol{X}$是条件独立的，其整体数据的联合似然函数可以表示为

$$p(y, z|\boldsymbol{X}, \theta) = p(z|\boldsymbol{X}, \theta) \prod_{i=1}^{N} p(y_i|z_i) \tag{3.21}$$

其中，条件概率$p(y_i|z_i)$给出了观测值与潜变量间的关系。式（3.21）由先验和似然$p(y_i|z_i)$两部分组成。

假设我们有 m 个相关但不同的任务，每个任务在对应的训练集$\{(\boldsymbol{X}_m, \boldsymbol{y}_m)\}$下由一个 GP 来建模，则$y = (y_1^T, y_2^T, \dots, y_m^T)$的概率分布为

$$p(y|\boldsymbol{X}, \theta) = \prod_{m=1}^{N} p(y_m|\boldsymbol{X}_m, \theta) \tag{3.22}$$

[Lawrence and Platt, 2004]通过约束协方差矩阵 K 为分块对角矩阵，利用式（3.22）定义了一个多任务

高斯过程，并利用信息向量机（Informative Vector Machine，IVM）寻找稀疏表示以降低计算开销并加快模型训练。其中协方差矩阵为

$$K=\begin{bmatrix}K_1 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & K_m\end{bmatrix}$$

[Schwaighofer et al., 2005]将层次贝叶斯学习和高斯过程相结合用于多任务学习。在该算法中，层次贝叶斯模型本质上学习了高斯过程的均值和协方差函数，其算法分为两个步骤：

(1) 通过一个简单有效的 EM 算法，从数据中学习一个通用的协同核矩阵；

(2) 利用广义 Nystrom 方法推广协方差矩阵。

[Bonilla et al., 2007]利用数据上的共享协方差函数和不同任务上的自由形式协方差矩阵来建模任务间的依赖关系，从而放宽了[Lawrence and Platt, 2004]的方法对协方差矩阵的约束，表现出了更好的灵活性。

3.3.2.2 利用深度模型的模型迁移

随着深度学习的发展，人工智能的研究者利用深度学习强大的表达能力来抽取和迁移诸如类别间的关系等知识。其中，知识蒸馏（knowledge distilling）技术就是一个很好的例子，它涉及了教师网络和学生网络。知识蒸馏，也称为软标签，最初是为处理模型压缩而提出的。在处理压缩的过程中，源域与目标域被认为是一致的。

在基于模型的迁移学习中，通过计算类别k所有源样本的激励函数的 softmax 的平均值，可提取类别k的软标签l，其中该平均值用$l^{(k)}$表示。如果简单的 softmax 生成了一个较陡的分布，我们则使用一个具有高温度的 softmax 来保留足够多的类别关系信息。

基于软标签的类别间关系知识的损失可形式化定义为

$$\ell_{\text{softlabel}}(x_t,y_t;\theta_{\text{repr}},\theta_c)=-\sum_i l^{(k)}\text{softmax}\left(\frac{\theta_c^T f(x_T;\theta_{\text{repr}})}{\tau}\right) \tag{3.23}$$

3.3.3 深度模型中的微调方法

随着深度学习成为广泛应用的流行的机器学习技术，研究人员开始赋予深度模型迁移学习的能力。参数微调是一种简单有效的、涉及模型参数的知识迁移技术。

3.3.3.1 贪心逐层预训练和微调

贪心逐层预训练思想已广泛应用于深度置信网络（Deep Brief Network，DBN）和自编码的模型训练中。其中，[Bengio, 2012]基于非监督学习算法训练参数，并利用该参数初始化特殊的分类任务。该算法假设无监督学习任务（例如样本重构）能够表现良好的表示。因此，利用该形式获得的初始化参数将适合后续任务。该初始化策略可以被看作学习模型参数的一种正则化方法。

贪婪逐层算法的第一阶段是利用非监督学习对每层进行训练，也称为预训练阶段。具体地，该阶段利用第$l-1$层输出的训练样本$h_{l-1}(x)$来训练非监督模型，进而重新生成第l层的表示$h_l(x)=R_l(h_{l-1}(x))$。

第二阶段是利用监督信号对后续任务（如分类）进行微调。现有几种微调的变体，其中最常见的一种方法是通过利用第一阶段输出的$h_l(x)$作为输入来初始化线性或者非线性监督模型，并根据监督训练损失微调模型参数。

3.3.3.2 监督学习下的参数微调

贪心逐层预训练算法在深度学习的早期比较流行，但是随后被 dropout 和批处理规范化所取代，其中 dropout 和批处理规范化采用端到端的方式训练所有层。利用稳定优化算法和大量有标签数据，可以直接从零开始训练深度监督模型。然而，实现该目标的关键问题是如何在不同的监督任务间传递由监督学习训练出来的参数。

针对 CNN，[Yosinski et al., 2014]通过大量实验评估了其预训练模型的不同层的迁移能力。实验将 ImageNet 数据集分为 A 和 B 两部分。前两行的模型作为基础模型在 A 和 B 的数据上训练。在后两行中，该模型的前几层由所学数值进行初始化，其余层随机初始化。XnY代表前n层，由基础模型X复制而来，并且被冻结用于Y中的迁移学习。XnY^+代表被迁移的前n层，可通过Y进行微调。

同样，[Mou et al., 2016]基于大量实验评估了循环神经网络（Recurrent Neural Network，RNN）模型在

处理自然语言分类任务时其参数的迁移能力。

为了分析每层的迁移能力，[Mou et al., 2016]在大型的影评数据集 IMDb、小型的影评数据集 MR 和小型的六向问题数据集 QC 中进行了相关实验，并检测了在迁移学习的冻结、微调和任务迁移设置下的结果。RNN 的实验结果与 CNN 的相似。具体地，高层（例如隐藏层和输出层）不适合进行迁移。甚至当从 IMDb 迁移到 MR 时，在相同语义设置中，如果冻结所有层，则性能也会下降。在从 IMDb 到 QC 等不同任务的情况下，冻结隐藏层会导致性能急剧下降。如果利用源域模型的参数来初始化模型，然后继续微调模型参数，则所获得的模型性能通常比基础模型的高，或至少与基础模型的相当。

4 迁移学习具体应用

4.1 隐私保护的迁移学习

4.1.1 引言

机器学习技术越来越广泛应用于社交网络、银行、供应链管理和医疗保健等领域。随着放些应用的出现，各种数据（如个人医疗记录和金融交易信息）中的敏感信息越来越多。这就提出了一个关键问题：如何保护用户的隐私信息?

如今，现代社会越来越需要隐私问题的解决方案。最著名的法律之一是欧洲的通用数据保护条例（General Data Protection Regulation，GDPR），该条例规定了对私人用户数据的保护，并限制了组织间的数据传输。

因此，如何保证用户隐私和数据机密性已经成为机器学习中一个引起极大关注的问题。到前为止，研究人员已经尝试从几个角度来解决这个问题[Dwork et al., 2006a;Chaudun et al.,2011;Dwork and Roth., 2014;Abadi et al., 2016b]。

在各种方法中，数据匿名化是保护用户数据中敏感信息的一种基本方法。然而，仅进行数量匿名化不足以保护用户隐私。事实上，通过使用额外的外部信息，攻击者可以识别匿名记录。众所周知的案例是，马萨诸塞州州长 William Weld 的个人健康信息在一个据称是匿名的公共数定中被发现[Sweeney, 2002;Ji et al., 2014]。通过合并健康数据库和选民登记之间的重叠记录，研究人员能够确定州长的个人健康记录。

在过去的几十年中，差分隐私（differential privacy）[Dwork et al., 2006a;Dwork, 2008]已经发展成为隐私保护的标准之一。为了设计差分隐私算法，通常需要在原始数据中加入精心设过的噪声来对解析算法进行歧义化。通过注入随机噪声，单个样本不能显著影响差分隐私算法的输出，这限制了攻击者获得的信息。

近年来，许多机器学习算法被修改以实现差分隐私，包括逻辑回归（logistic regression）[Chaudhuri et al. , 2011]和深度神经网络（deep neural network）[Abadi et al., 2016b]等。

然而，在应用迁移学习时，这个问题变得更加至关重要，因为迁移学习模型通常跨不同的域和数据集进行映射，并关联不同的组织机构。因此，迁移学习也面临着用户隐私挑战，尤其是当它被跨组织机构应用时。设计差分隐私保护机制来提取和迁移知识成为一个挑战。在本章中，我将首先介绍差分隐私的定义以及相关的差分隐私算法，然后介绍一些隐私保护迁移学习的先进方法。

4.1.2 差分隐私

4.1.2.1 定义

差分隐私[Dwork et al., 2006b;Dwork and Roth, 2014]已经被确立为一个严格的标准，以保证访问私有数据的算法的隐私。直观地说，给定隐私预算（privacy budget）ε，如果更政数据集中的一个样本不会使算法任意输出对象的对数似然值变化超过ε，则该算法保持ε-差分隐私。它的正式定义如下。

定义 15.1（差分隐私）如果对于任何两个输入数据集$\mathcal{D}_1$，$\mathcal{D}_2$，随机机制，μ的任一输出t满足$P(\mu(\mathcal{D}_1) = t) \leqslant e' \mathbb{P}(\mathcal{U}(\mathcal{D}_2) = t)$，则随机机制，$\mu$是$\varepsilon$-差分隐私的。

为了满足ε-差分隐私保证，通常需要在学习算法中添加细致的扰动或噪声。较小的ε提供更严格的隐私保证，但代价是噪声较大，导致性能下降[Chaudhuri et al.，2011;Bas, 2014]为了解决这个问题，最近 Dwork 和 Roth（2014）提出了一种松弛版本的ε-差分陷入为（ε, δ）差分隐私，其中δ度量隐私中的损失，其定义如下。

定义 15.2（(ε,δ)-差分隐私）如果对于任何两个输入数据集$\mathcal{D}_1$，$\mathcal{D}_2$，随机机制μ的 e 输出 t 满足$P(\mathcal{H}(\mathcal{D}_1)=t)\leqslant e'P(\mathcal{M}(\mathcal{D}_2)=t)+\delta$，则随机机制$\mu$是（$\varepsilon,\delta$)-差分隐私的。

4.1.2.2 隐私保护的正则化经验风险最小化

对于训练数据集$\mathcal{D}$，正则化经验风险最小化选择一个假设空间$\mathcal{X}$中的预测器f，将正则化经验损失最小化为

$$\min_{f\in\mathcal{X}} J(f,\mathcal{D})=\frac{1}{n}\sum_{j=1}^{n} l\left(f\left(x_j\right),y_j\right)+\lambda r(f) \tag{4.1}$$

其中正则化项$r(f)$防止过拟合，λ是正则化参数。正则化 ERM 方法在实践中被广泛使用，例如逻辑回归和支持向量机。为简单起见，在下文中我们只关注线性函数，即$f（x）=w^{T}x$。

在接下来的章节中，我们将介绍一些创建隐私保护 ERM 算法的技术，包括输出扰动、目标扰动和梯度扰动。

4.1.2.2.1 输出扰动

输出扰动（output perturbation）方法[Chaudhuri et al., 2011]源自[Dwork et al., 2006]用出的灵敏度方法（sensitivity method），是一个用于对任何函数生成隐私保护近似的通用方法，对于最小化的$w^{*}=\underset{w}{\operatorname{argmin}}(w,\mathcal{D})$，它输出一个预测器

$$w_{piv}=w^{*}+b$$

其中 b 是随机噪声，其密度为

$$\mathbb{P}(\boldsymbol{b})=\frac{1}{\alpha}\mathrm{e}^{-\boldsymbol{\beta}\|\boldsymbol{b}\|_2} \tag{4.2}$$

其中 a 是归一化常数，$\beta=\frac{n\varepsilon\lambda}{2}$。[Chaudhuri et al., 2011]证明：如果正则化项$r(.)$是可微和 1-强凸的，损失l是凸和可微的，并且对于所有的 z 满足$\left|l\ ''(z)\right|\leq 1$，那么输出扰动方法提供了$\varepsilon$-差分隐私。

4.1.2.2.2 目标扰动

与输出扰动法不同，目标扰动（objective perturbation）法[Chaudhuri et al., 2011]在目标高教中添加了噪声项。它不是最小化J，而是通过求解以下目标函数来学习预测器。

$$w_{\text{priv}}=\underset{w}{\operatorname{argmin}} J(w,\mathcal{D})+\frac{1}{n}b^{\mathrm{T}}w+\frac{1}{2}\Delta\parallel w\parallel_2^2$$

其中 b 根据式（4.2）进行采样，$\beta=\varepsilon'/2$，ε'和Δ按照如下方法进行计算：

（1）$\varepsilon'=\varepsilon^{-}\log\left(1+\frac{2c}{n\lambda}+\frac{c^2}{n^2\lambda^2}\right)$

（2）如果$\varepsilon'>0$，则 Δ= 0，否则$\Delta=\frac{c}{n(\mathrm{e}^{\epsilon/4}-1)}-\lambda$且$\varepsilon'=\varepsilon/2$

其中 c 是常数项。类似地，如$r(.)$是 1-强凸和二阶可微的，$l(.)$是凸和二阶可微的，且对于所有的 z 满足$\left|l\ ''(z)\right|\leq 1$和$\left|l\ '''(z)\right|\leq c$，那么目标扰动方法是$\varepsilon'$-差分隐私的。具体地，如果使用正则化逻辑回归作为 ERM 模型，即$r(w)=\frac{1}{2}\parallel w\parallel_2^2$并且$l(z)=\log(1+e^{-z})$，那么 c=$\frac{1}{4}$。

4.1.2.2.3 梯度扰动

如前所述，输出扰动和目标扰动方法要求目标函数是凸的，并且具有强凸的正则化项。在一些算法（如

深度模型）中，这一前提条件是不满足的。为了解决这个问题，[Abadi et al., 2016b]提出了一种梯度扰动（gradient perturbation）方法，以保证深度学习算法中的差分隐私。具体地在第t次迭代中，一组样本L_t被随机选中，然后调整其梯度$g_t(x_i)$到一个上界 C，即

$$\bar{g}_t(x_i) = g_t(x_i)/max\left(1, \|g_t(x_i)\|_2/C\right)$$

其中$x_i \in L$。在这组样本中添加随机高斯噪声：

$$\tilde{g}_t = \frac{1}{L}\left(\sum_i \bar{g}_t(\boldsymbol{x}_i) + \mathcal{N}(0, \sigma^2 C^2 \boldsymbol{I})\right)$$

中L是L_t的大小，σ是一个常数。我们来用梯度$\tilde{g}_t$更新模型。[Abadi et al., 2016b]证明了利用仔细选择的样本采样率σ和迭代次数，梯度扰动方法保证了（ε, δ)-差分隐私。

4.1.3　隐私保护的迁移学习

4.1.3.1　问题设置

许多迁移学习应用中，源域和目标域或许位于不同的组织机构中，这样由于隐私问题，信息就不能直接相互传输。下面我们将讨论用户隐私问题至关重要的几种场景。

目标提升（target improvement）：在大多数迁移学习应用中，一个或多个源数据集被用于帮助提高在目标域中训练的模型性能。因此，有必要设计一种隐私保护机制，用于从源域提取和迁移知识，以阻止敏感信息的泄露。

多任务学习：在多任务学习中，每个任务都互相借用信息以改进甘学习横利，因此，息机制需要保证每一方都不会泄露自己的隐私。

4.1.3.2　目标提升

4.1.3.2.1 差分隐私假设迁移学习

[Wang et al,. 2018d]提出训练局部差分隐私逻辑模型，并将其迁移到目标域，如图 4.2 所示。要做到这一点，需要一个可以同时被源域和目标域访问的公共数据集（不一定有标签）来充当信息中介。具体来说，该模型需要以下几步：

(1)基于参数 e，每个源域使用其有标签的样本来训练差分隐私逻辑回归模型$w_{piv}^{S_i}$。然后所有假设$\left\langle w_{\mathrm{piv}}^{S_i}\right\rangle_{\mathrm{i=1}}^{\mathrm{m}}$被发送到目标域。

(2)每个源域获取公共数据集，并使用其无标签的样本和公共数据集来计算差分隐私"重要性权重"向量v^{S_i}，然后发送$\langle v^{S_i}\rangle_{\mathrm{i=1}}^{\mathrm{m}}$到目标域。

(3)目标域获取公共数据集，计算非隐私"重要性权重"向量 v。

(4)目标域计算"假设权重"向量$v_H \in \mathbb{R}^m$，使 v与由v_H加权的v^{S_i}的线性组合的 KL 散度最小化。

(5)目标模型用v_H和源域中的$\left\langle w_{\mathrm{piv}}^{S_i}\right\rangle_{\mathrm{i=1}}^{\mathrm{m}}$，来构造一个带信息量的高斯先验。

(6)目标域通过[Marx et al,. 2008]的方法利用有限的有标签目标数据和带信息量的高斯先验来训练一个贝叶斯逻辑回归模型，并返回参数w_{piv}。

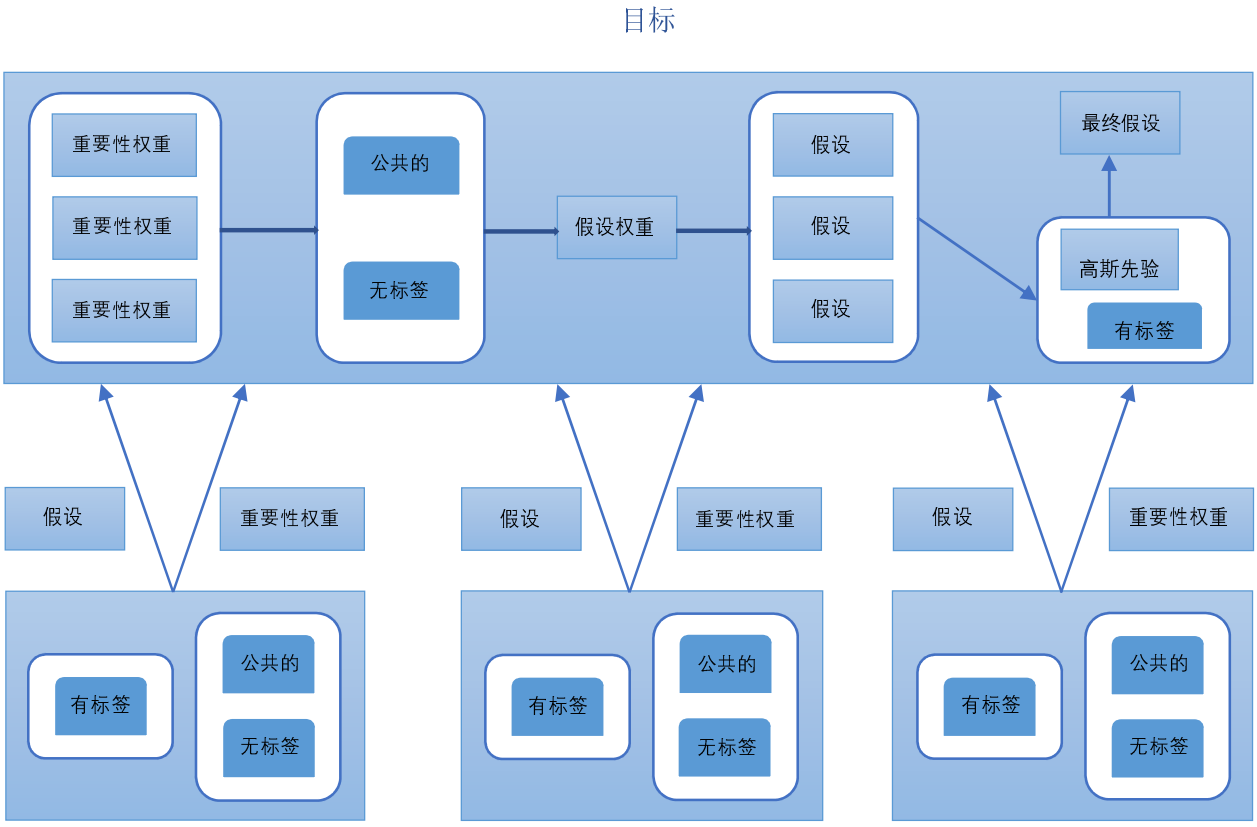

图 4.2 多源迁移学习系统示意图（改编自[Wang et al,. 2018d]）

4.1.3.3 多任务学习

在学习过程中，当所有任务相互帮助时应考虑到隐私，因此地存在隐私问题。[Xie et al,. 2017]提出了一种差分隐私保护多任务学习方法。假设模型参数w^i 被分解为两个分量，即 $w^i = p^i + q^i$，其中$p^i \in \mathbb{R}^d$是学习任务相关性的分量，$\boldsymbol{q}^i \in \mathbb{R}^d$是特定于任务的分量，从而$W = P + Q$。因此，提出的目标函数公式为

$$\min_{\boldsymbol{P},\boldsymbol{Q}} \sum_{i=1}^{m} \left(\frac{1}{n_i} \sum_{j=1}^{n_i} l_j^i(\boldsymbol{p}^i + \boldsymbol{q}^i) \right) + \lambda_1 r_P(\boldsymbol{P}) + \lambda_2 r_Q(\boldsymbol{Q})$$

其中$l_j^i(w^i)$定义了第i个任务中模型参数为w^i时的第j个数据的损失，$r_p(P)$表示知识迁移，

$r_Q(Q)$是模型复杂度的惩罚项。这里假设$r_Q(Q)$是按照任务分解的，即$r_Q(Q)= \sum_{i=1}^{m} r_Q(\boldsymbol{q}_i)$。这样，$Q$中的每个列$q_i$，能被分发到计算节点，并且能进行本地更新。对于$P$，先计算$\boldsymbol{p}^i$'的梯度信息，然后中心服务器收集该梯度信息并更新 P。这是因为只有$\boldsymbol{p}^i$被传递到中心服务器，噪声才能被添加进去来保证差分隐私。

4.2 城市计算中的迁移学习

4.2.1 引言

如今，移动电话、公共交通和基础设施（如交通摄像头和空气质量监测站）不断地产生与城市相关的大量异构数据（例如 GPS 点、线上发帖、道路状况和天气状况）。这为我们从不同角度了解城市的动态打开了一扇新的大门，并促进了各种城市计算的发展，这种计算可以应用于交通监控、社会安全、城市规划、医疗保健等。目前的解决方案可以采用如下方式帮助简化城市规划和决策：

细粒度数据推理：在许多城市监测任务中，所获得的数据无法覆盖整个城市区域。一个典型的例子是空气质量监测，其中空气质量传感站仅在城市中稀疏分布。因此，如何基于收集的稀疏数据推断出更细粒度的数据分布将成为一个重要的研究课题。

未来现象预测：另一个重要且热门的研究领域是城市事件预测问题，例如空气质量和交通预测。在传统的统计学中，这种问题通常可以建模为时间序列预测问题，并通过统计模型（如整合移动平均自回归（Auto Regressive Integrated Moving Average，ARIMA））来解决。然而，在复杂的城市计算应用中，为了获得更准

确的预测，通常需要构建更复杂的机器学习模型以考虑异构数据源。例如，在空气质量预测中，诸如道路地图、天气条件和交通状况等许多数据都可以帮助预测。

事件检测：检测异常事件在城市计算应用中非常重要。例如，在诸如台风和飓风等破坏性气条件下，以实时方式检测道路障碍物（例如倒下的树木和积水）是一个关键问题。城市管理部分可以及时恢复道路运输以减少损失。

设施部署：寻找适当的位置来部署新设施（例如购物中心、电动车充电站或环境监测站）是另一个主要的研究课题。值得注意的是，大多数设施一旦建成，将难以被移至其他地点。因此，设施部署的相关方案和机制通常不能米用试错法，这使得这项任务更具挑战性。

尽管在城市计算方面已经有了广泛的研究，但大多数现有研究都是在假设与服务相关的数据量足够且容易获得的情况下构建其应用，例如交通预测应用的交通流量记录。然而，实际情况并非总是如此。许多城市可能刚刚启动城市数字化进程，并没有太多的历史服务相关数据。因此，城市计算中的一个关键问题出现了（虽然目前很少有人研究），即如何在数据稀缺的问题下冷启动新的城市计算服务。例如，假设我们想要建立设施部署辅助系统来推荐各种设施的选址，如大型购物中心或五星级酒店，但是城市中还没有这样的设施。在这种情况下，我们怎样才能以这个城市的少量数据为基础完成推荐?

在本章的其余部分，我们将研究在存在数据稀缺问题时，迁移学习如何帮助构建城市计算应用。首先，我们将介绍在城市计算应用中"迁移什么"。通常，这个问题取决于具体的应用，但考虑到城市计算应用的共同特征，我们可以将此问题分为三类，即跨模态迁移、跨区域或跨城市迁移以及跨应用迁移。然后，我们将介绍城市计算迁移学习中的关键问题，包括确定适当的源域、连接源域和目标域以及评估知识的可迁移性。最后，我们将阐述两个实际应用，以说明经典的迁移学习技术在城市计算中的应用。

4.2.2 城市计算中的“迁移什么”

要利用迁移学习来构建具有数据稀缺性问题的城市计算应用，需解决的第一个关键问题是找到适当的源域知识进行迁移，即"迁移什么"。在城市计算应用中，这些知识主要有以下来源：

跨模态迁移。城市计算应用的一个关键特征是它通常依赖于异构数据模态。例如在空气质量预测任务中，诸如城市道路地图、车辆 GPS 轨迹、兴趣点分布、天气信息等各种数据模态可以带提高预测准确度。然而，一些城市或地区可能缺失一些数据模态，因此预测准确度不太令人清意。在这种情况下，如果可以通过从其他模态迁移知识来首先推断缺失模态的信息，我们就有那会提高目标应用的性能。

跨区域或跨城市迁移。在区域或城市构建新应用的辅助知识源是来自其他已经建立了相同（或类似）应用的区域或城市的经验。在这种情况下，我们还可以将源区域或城市称为数据丰富的区域或城市，将目标区域或城市称为数据稀缺的区域或城市。虽然跨区域或跨城市迁移的基本概念是直观的，但它实际上面临着许多困难。例如，不同的城市具有不同的发展水平，这使得直接迁移通常无效，并可能导致"负迁移"。

跨应用迁移。对于将要开发的新城市计算应用，迁移学习的另一个重要知识来源是已经收集了大量数据的相关应用。例如，假设我们想在一个城市开设一个新的共享汽车业务，但我们没有关于共享汽车行为的任何数据。那么为了实现与共享汽车相关的城市计算应用（如供需预测），我们可以利用来自出租车相关应用的现有数据。

值得注意的是，上述知识源可以在城市计算中的迁移学习应用中被组合在一起。例如，我们希望为目标应用在城市的不同数据模态之间建立可靠且有用的关联，那么可以首先在所有数据模态足够的源城市中学习这些关联，然后将其迁移到缺少某些数据模态和目标应用数据的目标城市。

4.2.3 城市计算中迁移学习的关键问题

在本节中，我们将总结将迁移学习应用于城市计算时应考虑的几个关键问题。

（1）识别适当的源域。虽然方详细阐述了获取城市计算应用的源域知识的一些方法，但我到合适的源域仍然是实践中最困难的部分。首先，对于许多应用，可能很难找到包含构建目标应用所需的所有信息的完美源域。更糟糕的是，我们可能无法找到任何能用于迁移的数据模态、区域或城市，或者应用。在这种情况下，我们可能需要依靠仿真软件来生成数据以作为源域，或尝试从网站和应用程序中爬取数据。在许多情况下，Facebok 和 Twiter 之类的社交媒体平台是潜在的数据源，因为用户在这些平台上的活动（例如，

签到）通常可以映射到现实生活中的物理空间并反映城市动态。

（2）连接源域和目标域。在确定源域后，第二步是学习可以从源域迁移到目标域的知识。换句话说，需要在两个域之间提取知识的一“不变”部分。虽然这部分通常是应用特定的，因而没有一种始终有效的方法，但我们在此将提供一些指导和建议。

■以系统的方式构建隐私保护的解决方案。在城市计算应用中，许多数据只能以隐私保护的方式被收集和共享。这可能导致源域数据和目标域数据之间的不一致。例如，许多城市八有出粗车轨迹记录仅包括粗略的乘客上车和下车的区域，而不是详细的 GPS 坐标 那么，当我们想通过跨城市迁移学习是立与出担年相关的城市计算应用时，可能只能从源城市获取隐私保护的出租车数据，而可以从目标城市检索更细粒度的数据。因此，可能客一些处理这种隐私保护数据的系统方法来促进迁移学习。

■通过神经网络学习共同表示以实现可迁移性。随着深度神经网络的发展，神经网络已成出一种可以自动学习大范围任务的特征表示的强大工具。类似地，在城市计算中的迁移学习应用中，它也成了一种流行的方法。例如，我们可以使用神经网络来学习城市区域的新特征表示，而其原始特征可以包括兴趣点分布、温度、交通状况等。然后，可以将其他有用的先验知识（例如两个城市中的两个区域是彼此相似的，如 CBD）添加到神经网络中，以帮助其学习具有更好的可迁移性的新特征表示。

（3）评估可迁移性。另一个基本问题是定量衡量源域和目标域之间的可迁移性。例如，已知几个候选源城市，评估可迁移性将有助于选择适当的城市作为源城市。我们可能会考虑规模、人口、文化、经济等因素，来量化城市间的相似性以及城市间的可迁移性。然而，到目前为止，仍然很少有工作侧重于从数学角度量化这种城市间的可迁移性。因为这将极大地推动城市计算中迁移学习的应用，所以我们相信这将是未来重要的研究方向。在接下来的小节中，我们将介绍城市计算中的实际问题，即连空气质量预测问题，以阐述迁移学习如何有效地解决城市计算中的数据稀缺问题。

4.2.4 空气质量预测

由于越来越多的工厂、车辆、人类活动等，空气污染成为世界上许多地方的城市生活中的一个严重问题。准确预测城市每个地区的空气质量对市民来说非常重要，这可以帮助他们提前计划其户外活动。显然，一个地区的空气质量受到许多因素的影响，例如：兴趣点、交通、生产工厂等。因此，污染程度很大程度上随位置而变化。在污染预测系统中，我们希望利用这些多模态数据提前估算一个城市的细粒度空气质量。

更具体地说，我们的任务是将空气质量分为较好、中等、不健康等类别，所以可以将空气质量预测看作分类问题。给定某时间内每个地区的多模态数据，我们希望将相应的空气质量划分为某个类别。我们注意到传统的分类模型无法很好地解决这个问题，其原因有两个。首先，空气质量数据非常稀缺，因为许多城市只有少数空气质量监测站，这导致了标签的稀缺问题。其次，存在数据不足的问题，因为某些地区或某些时期可能缺少甚至完全缺失一些关于重要影响因素的多模态数据，例如，可能会缺少几个小时的上海气象数据。同样，上海某些地区以及特定地区的某些时段的出租车轨迹数据可能无法获得。

因此，我们考虑是否可以将知识从一个城市迁移到另一个城市以帮助预测空气质量。接下来我们将阐述一种称为 FLORAL 模型[Wei et al.，2016b]的解决方案，以便在城市之间迁移多模态数据。

4.2.4.1 问题设置

假设在源域北京已经收集了足够的有标签和无标签数据来构建交通预测模型，而目标境上少量有标签和一些无标签数据。每个数据实例代表一段时间内的一个区域，表示为多模态元舌有关空气质量的各种影响因素的数据。我们假设存在 M 个模态，并将目标域中有标答标签实例分别表示为$T_l = \{t_{li}^1, t_{li}^2, \cdots, t_{li}^M\}$和$T_u = \{t_{ui}^1, t_{ui}^2, \cdots, t_{ui}^M\}$。类似地，源域中有标签和无标签的实例用$S_l = \{s_{li}^1, s_{li}^2, \cdots, s_{li}^M\}$和$S_u = \{s_{ui}^1, s_{ui}^2, \cdots, s_{ui}^M\}$表示。由于$|S_l| \gg |T_l|$，且$T_l$中的某些实例缺少一些模态，我们希望利用$S_u$和$S_l$来帮助更有效且高效地学习目标域的分类器

在我们的问题中存在四种数据模态，即来自 Bing 地图的路网和 POI 数据、从公共网站取象数据，以及出租车轨迹数据。请注意，仅可获得北京的出租车轨迹数据，并且一些用例的某些模态在上海是缺失的。表 4.1 总结了其他三种模态的数据详情。

表 4.1 模态统计[Wei et al., 2016b]

模态	北京	上海
#公路段	249080	313736
高速公路	994km	2016km
公路	24643km	40944km
#POI	379022	433016
时间跨度（2014 年）	2 月 1 日～5 月 31 日	8 月 1 日～9 月 10 日

4.2.4.2 FLORAL 模型

在本小节中，我们将介绍用于城市计算的 FLORAL 模型[Wei et al.，2016b]。FLORAL 模型包括两个主要组件，一个组件用于学习源域中多个模态的语义相关字典，另一个组件用于将字典和实例从源域迁移到目标域。FLORAL 模型基于来自源域和目标域的每个实例的稀疏编码，学习分类模型以预测目标城市的空气质量。下面我们依次介绍每个组件。

学习语义相关字典。FLORAL 模型中的字典是通过聚类获得的，共有三个步骤，即图构造、图聚类和字典推断。第一步，构造相似性图 G=（V，E），其中顶点集包括源域中每个实例的所有模态，边描述顶点之间的成对关系，即每个模态内的内部边（intra-edge）和不同模态间的外部边（inter-edge）。对于第 m 个模态中的每对顶点s_i^m和s_j^m，我们用欧式距离衡量其在特征表示上的相似性。如果它们都属于与彼此最相似的 k 个顶点，则用内部边连接它们。每个内部边的权重用它们之间的高斯核计算。对于不同模态的一对顶点s_i和s_j，如果已知两个实例s_i^m和s_j^m，是相关的，例如它们是两个相邻区域，那么我们用权重为 1 的外部边连接它们。第二步，设计一个子模块图聚类算法来将获得的相似度图聚为 K 个类，我们通过保证一些特性来实现聚类：有标签实例的数量在不同类上的分布是均匀的以及每个类中的模态足够分散。第三步，基于所获得的簇来推断每种模态的字典。也就是说，对于每个簇 k，字典中的元素d_k^m是模态 m 中顶点的中心，因此，最终模态 m 的字典结合了从 K 个簇中推断的 K 个字典元素，即$D^m = [d_1^m, d_2^m, \cdots, d_k^m], m = 1,2, \cdots, M$。这 M 个字典具有相同的大小并且在语义上相关。

将字典和实例迁移到目标域。可以将从源域获得的子典在自标域中重用，然后计算源实例和目标实例的稀疏编码。通过利用来自源域和目标域的有你签实例的稀疏编码，设计使用基于 T-AdaBoost 的多模态迁移 AdaBoost 算法[Dai et al., 2007b]作为目标域的分类器。作为 T-A Boost 的扩展，该系统还可以学习不同模态的权重。

4.3 基于DANN迁移学习实验

4.3.1 模型介绍

本章基于 DANN 模型[Muhammad Ghifary, 2014]。该模型通过在损失函数中加入 MMD 距离的方式来缩短源域和目标域的距离，达成从知识从源域迁移至目标域的目的。模型结构如图 4.3 所示。

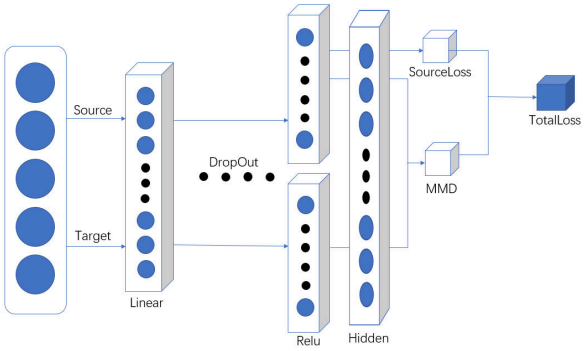

图 4.3 DANN 模型

4.3.2 实验环境

该模型的实现是基于 python3.6 和 pytorch1.4 实现的。该实验的数据集选择的是 office_caltech_10，其

中源域选择的是 office_caltech_10 的 amazon 数据集，目标域选择的是 office_caltech_10 的 webcam 数据集。如图 4.4 所示。

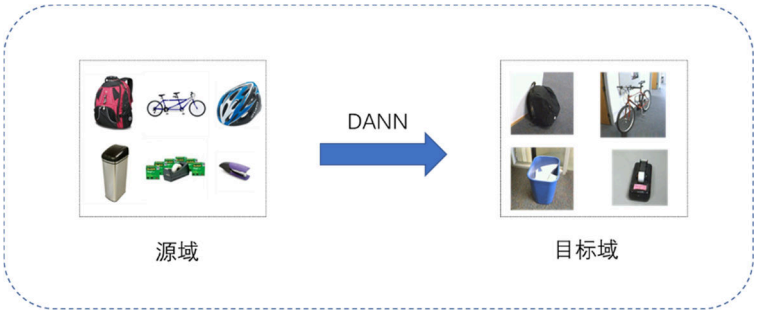

图 4.4

4.3.3 模型搭建

```
import torch.nn as nn

class DaNN(nn.Module):
    def __init__(self, n_input=28 * 28, n_hidden=256, n_class=10):
        super(DaNN, self).__init__()
        self.layer_input = nn.Linear(n_input, n_hidden)
        self.dropout = nn.Dropout(p=0.5)
        self.relu = nn.ReLU()
        self.layer_hidden = nn.Linear(n_hidden, n_class)

    def forward(self, src, tar):
        x_src = self.layer_input(src)
        x_tar = self.layer_input(tar)
        x_src = self.dropout(x_src)
        x_tar = self.dropout(x_tar)
        x_src_mmd = self.relu(x_src)
        x_tar_mmd = self.relu(x_tar)
        y_src = self.layer_hidden(x_src_mmd)
        return y_src, x_src_mmd, x_tar_mmd
```

4.3.4 训练过程构建

```
def train(model, optimizer, epoch, data_src, data_tar):
    total_loss_train = 0
    criterion = nn.CrossEntropyLoss()
    correct = 0
    batch_j = 0
    list_src, list_tar = list(enumerate(data_src)), list(enumerate(data_tar))
    for batch_id, (data, target) in enumerate(data_src):
        _, (x_tar, y_target) = list_tar[batch_j]
        data, target = data.data.view(-1, 28 * 28).to(DEVICE), target.to(DEVICE)
        x_tar, y_target = x_tar.view(-1, 28 * 28).to(DEVICE), y_target.to(DEVICE)
        model.train()
        y_src, x_src_mmd, x_tar_mmd = model(data, x_tar)
        loss_c = criterion(y_src, target)
        loss_mmd = mmd_loss(x_src_mmd, x_tar_mmd)
        pred = y_src.data.max(1)[1]  # get the index of the max log-probability
        correct += pred.eq(target.data.view_as(pred)).cpu().sum()
        loss = loss_c + LAMBDA * loss_mmd
        optimizer.zero_grad()
        loss.backward()
```

```
        optimizer.step()
        total_loss_train += loss.data
        res_i = 'Epoch: [{}/{}], Batch: [{}/{}], loss: {:.6f}'.format(
                epoch, N_EPOCH, batch_id + 1, len(data_src), loss.data
        )
        batch_j += 1
        if batch_j >= len(list_tar):
                batch_j = 0
total_loss_train /= len(data_src)
acc = correct * 100. / len(data_src.dataset)
res_e = 'Epoch: [{}/{}], training loss: {:.6f}, correct: [{}/{}], training accuracy: {:.4f}%'.format(
        epoch, N_EPOCH, total_loss_train, correct, len(data_src.dataset), acc
)
tqdm.write(res_e)
log_train.write(res_e + '\n')
RESULT_TRAIN.append([epoch, total_loss_train, acc])
return model
```

4.3.5 测试过程构建

```
def tst(model, data_tar, e):
        total_loss_test = 0
        correct = 0
        criterion = nn.CrossEntropyLoss()
        with torch.no_grad():
                for batch_id, (data, target) in enumerate(data_tar):
                        data, target = data.view(-1,28 * 28).to(DEVICE),target.to(DEVICE)
                        model.eval() #不启用 BatchNormalization 和 Dropout
                        ypred, _, _ = model(data, data)
                        loss = criterion(ypred, target)
                        pred = ypred.data.max(1)[1]    # get the index of the max log-probability
                        correct += pred.eq(target.data.view_as(pred)).cpu().sum()
                        total_loss_test += loss.data
                accuracy = correct * 100. / len(data_tar.dataset)
                res = 'Test: total loss: {:.6f}, correct: [{}/{}], testing accuracy: {:.4f}%'.format(
                        total_loss_test, correct, len(data_tar.dataset), accuracy
                )
        tqdm.write(res)
        RESULT_TEST.append([e, total_loss_test, accuracy])
        log_test.write(res + '\n')
```

4.3.6 主程序构建

```
if __name__ == '__main__':
    rootdir = 'D:/works/pywork/pycharm/DaNN/data/office_caltech_10/'
    torch.manual_seed(1)
    data_src = data_loader.load_data(
        root_dir=rootdir, domain='amazon', batch_size=BATCH_SIZE[0])
    data_tar = data_loader.load_test(
        root_dir=rootdir, domain='webcam', batch_size=BATCH_SIZE[1])
    model = DaNN.DaNN(n_input=28 * 28, n_hidden=256, n_class=10)
    model = model.to(DEVICE)
    optimizer = optim.SGD(
        model.parameters(),
        lr=LEARNING_RATE,
        momentum=MOMEMTUN,
        weight_decay=L2_WEIGHT
    )
    for e in tqdm(range(1, N_EPOCH + 1)):
        model = train(model=model, optimizer=optimizer,
                      epoch=e, data_src=data_src, data_tar=data_tar)
        tst(model, data_tar, e)
    torch.save(model, 'model_dann.pkl')
    log_train.close()
    log_test.close()
    res_train = np.asarray(RESULT_TRAIN)
    res_test = np.asarray(RESULT_TEST)
    np.savetxt('res_train_a-w.csv', res_train, fmt='%.6f', delimiter=',')
    np.savetxt('res_test_a-w.csv', res_test, fmt='%.6f', delimiter=',')
```

4.3.7 源域与目标域的距离选择

本实验使用了 mix_rbf_mmd2 作为衡量源域和目标域距离的标准。此外这一节也附上其他距离的实现以供同学们自己实现基于多种不同距离的 DANN 模型。

```
import torch
min_var_est = 1e-8
def linear_mmd2(f_of_X, f_of_Y):
    loss = 0.0
    delta = f_of_X - f_of_Y
    loss = torch.mean(torch.mm(delta, torch.transpose(delta, 0, 1)))
    return loss
```

```
def poly_mmd2(f_of_X, f_of_Y, d=1, alpha=1.0, c=2.0):
    K_XX = (alpha * (f_of_X[:-1] * f_of_X[1:]).sum(1) + c)
    K_XX_mean = torch.mean(K_XX.pow(d))
    K_YY = (alpha * (f_of_Y[:-1] * f_of_Y[1:]).sum(1) + c)
    K_YY_mean = torch.mean(K_YY.pow(d))
    K_XY = (alpha * (f_of_X[:-1] * f_of_Y[1:]).sum(1) + c)
    K_XY_mean = torch.mean(K_XY.pow(d))
    K_YX = (alpha * (f_of_Y[:-1] * f_of_X[1:]).sum(1) + c)
    K_YX_mean = torch.mean(K_YX.pow(d))
    return K_XX_mean + K_YY_mean - K_XY_mean - K_YX_mean

def _mix_rbf_kernel(X, Y, sigma_list):
    m = X.size(0)
    Z = torch.cat((X, Y), 0)
    ZZT = torch.mm(Z, Z.t())
    diag_ZZT = torch.diag(ZZT).unsqueeze(1)
    Z_norm_sqr = diag_ZZT.expand_as(ZZT)
    exponent = Z_norm_sqr - 2 * ZZT + Z_norm_sqr.t()
    K = 0.0
    for sigma in sigma_list:
        gamma = 1.0 / (2 * sigma**2)
        K += torch.exp(-gamma * exponent)
    return K[:m, :m], K[:m, m:], K[m:, m:], len(sigma_list)

def mix_rbf_mmd2(X, Y, sigma_list, biased=True):
    K_XX, K_XY, K_YY, d = _mix_rbf_kernel(X, Y, sigma_list)
    return _mmd2(K_XX, K_XY, K_YY, const_diagonal=False, biased=biased)

def mix_rbf_mmd2_and_ratio(X, Y, sigma_list, biased=True):
    K_XX, K_XY, K_YY, d = _mix_rbf_kernel(X, Y, sigma_list)
    return _mmd2_and_ratio(K_XX, K_XY, K_YY, const_diagonal=False, biased=biased)

def _mmd2(K_XX, K_XY, K_YY, const_diagonal=False, biased=False):
    m = K_XX.size(0)
    if const_diagonal is not False:
        diag_X = diag_Y = const_diagonal
        sum_diag_X = sum_diag_Y = m * const_diagonal
    else:
        diag_X = torch.diag(K_XX)
        diag_Y = torch.diag(K_YY)
        sum_diag_X = torch.sum(diag_X)
```

```
    sum_diag_Y = torch.sum(diag_Y)
    Kt_XX_sums = K_XX.sum(dim=1) - diag_X
    Kt_YY_sums = K_YY.sum(dim=1) - diag_Y
    K_XY_sums_0 = K_XY.sum(dim=0)
    Kt_XX_sum = Kt_XX_sums.sum()
    Kt_YY_sum = Kt_YY_sums.sum()
    K_XY_sum = K_XY_sums_0.sum()
    if biased:
        mmd2 = ((Kt_XX_sum + sum_diag_X) / (m * m)
            + (Kt_YY_sum + sum_diag_Y) / (m * m)
            - 2.0 * K_XY_sum / (m * m))
    else:
        mmd2 = (Kt_XX_sum / (m * (m - 1))
            + Kt_YY_sum / (m * (m - 1))
            - 2.0 * K_XY_sum / (m * m))
    return mmd2
def _mmd2_and_variance(K_XX, K_XY, K_YY, const_diagonal=False, biased=False):
    m = K_XX.size(0)    # assume X, Y are same shape
    if const_diagonal is not False:
        diag_X = diag_Y = const_diagonal
        sum_diag_X = sum_diag_Y = m * const_diagonal
        sum_diag2_X = sum_diag2_Y = m * const_diagonal**2
    else:
        diag_X = torch.diag(K_XX)
        diag_Y = torch.diag(K_YY)
        sum_diag_X = torch.sum(diag_X)
        sum_diag_Y = torch.sum(diag_Y)
        sum_diag2_X = diag_X.dot(diag_X)
        sum_diag2_Y = diag_Y.dot(diag_Y)

    Kt_XX_sums = K_XX.sum(dim=1) - diag_X
    Kt_YY_sums = K_YY.sum(dim=1) - diag_Y
    K_XY_sums_0 = K_XY.sum(dim=0)
    K_XY_sums_1 = K_XY.sum(dim=1)

    Kt_XX_sum = Kt_XX_sums.sum()
    Kt_YY_sum = Kt_YY_sums.sum()
    K_XY_sum = K_XY_sums_0.sum()

    Kt_XX_2_sum = (K_XX ** 2).sum() - sum_diag2_X
    Kt_YY_2_sum = (K_YY ** 2).sum() - sum_diag2_Y
    K_XY_2_sum  = (K_XY ** 2).sum()
```

```
        if biased:
            mmd2 = ((Kt_XX_sum + sum_diag_X) / (m * m)
                    + (Kt_YY_sum + sum_diag_Y) / (m * m)
                    - 2.0 * K_XY_sum / (m * m))
        else:
            mmd2 = (Kt_XX_sum / (m * (m - 1))
                    + Kt_YY_sum / (m * (m - 1))
                    - 2.0 * K_XY_sum / (m * m))

        var_est = (
            2.0 / (m**2 * (m - 1.0)**2) * (2 * Kt_XX_sums.dot(Kt_XX_sums) - Kt_XX_2_sum + 2 *
Kt_YY_sums.dot(Kt_YY_sums) - Kt_YY_2_sum)
            - (4.0*m - 6.0) / (m**3 * (m - 1.0)**3) * (Kt_XX_sum**2 + Kt_YY_sum**2)
            + 4.0*(m - 2.0) / (m**3 * (m - 1.0)**2) * (K_XY_sums_1.dot(K_XY_sums_1) +
K_XY_sums_0.dot(K_XY_sums_0))
            - 4.0*(m - 3.0) / (m**3 * (m - 1.0)**2) * (K_XY_2_sum) - (8 * m - 12) / (m**5 * (m - 1)) *
K_XY_sum**2
            + 8.0 / (m**3 * (m - 1.0)) * (
                1.0 / m * (Kt_XX_sum + Kt_YY_sum) * K_XY_sum
                - Kt_XX_sums.dot(K_XY_sums_1)
                - Kt_YY_sums.dot(K_XY_sums_0))
            )
        return mmd2, var_est
```

本实验详细代码可以在https://github.com/jindongwang/transferlearning/tree/master/code/deep/DaNN找到。

5 迁移学习历史与展望

迁移学习解决了 AI 领域面临的一个主要问题，即数据通常非常短缺。有时这种数据短缺是由于在该领域中收集数据很困难，比如在医疗领域中，为了确认一个完整的案例需要多年的治疗和手术。有时因为社会需要对数据的所有权进行更多的管理和控制，所以更多的法律法规实施在与第三方数据的共享上。因此，在很多领域中的数据获取将更加困难。此外，随着社会越来越多的领域转向数字化和数据化，对预测模型的需求也越来越大。在应用领域的"长尾"中，可用数据过少，只有头部受益于机器学习和人工智能。如果我们不能让"穷人"享受到"富人"的福利那么社会将变得更加两极分化。

迁移学习可以成为这种"小数据挑战"的技术解决方案。如果我们能够将这些模型从数据丰富的领域迁移到数据匮乏的领域，那么就有可能使这些数据匮乏的领域更快地迈向一个以信息和知识为基础的社会。事实上，通过本书给出的许多应用实例，我们已经看到迁移学习可以有效地缓解小数据问题。

未来需要探索的领域之一是继续探索终身机器学习和自动迁移学习。人类的智力来源之一在于其能够迅速而毫不费力地适应新任务和新环境的能力。事实上，人类不仅可以将知识迁移到一个新的领域，还可以在给定的新任务和新环境下学习如何自动迁移。这实际上是自然界中的一个奇妙的谜题，仅靠计算手段是无法解决的。神经科学和实验神经学有可能揭示这种能力的本质我们希望人工智能特别是迁移学习在总体上能从这些洞察中受益。

当我们正在见证人类历史上最基本的人工智能革命之一时，迁移学习作为一个深人的研究领域脱颖而出，它激发了新的想法和思想，使之深入到智能的本质。在回答图灵的问题“机器会思考吗”时，我们希

望通过回答“机器在新环境中和新任务下如何思考”来开始揭示该问题的答案。

参考文献

Abadi, Martín, Chu, A. , Goodfellow, I. , Mcmahan, H. B. , Mironov, I. , & Talwar, K. , et al. (2016). Deep learning with differential privacy.

Argyriou, A. , Evgeniou, T. , & Pontil, M. . (2006). Multi-task feature learning. *Proceedings of the 19th International Conference on Neural Information Processing Systems*. MIT Press.

Bengio, Y. . (2011). Deep Learning of Representations for Unsupervised and Transfer Learning. *JMLR: Workshop and Conference Proceedings 7*.

Bickel, S. , Brückner, Michael, & Scheffer, T. . (2007). Discriminative learning for differing training and test distributions. *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*.

Bollegala, D. , Maehara, T. , & Kawarabayashi, K. I. . (2015). Unsupervised cross-domain word representation learning. *Computer Science, 65*, 60–66.

Bonilla, E. V. , Chai, K. M. A. , & Williams, C. K. I. . (2008). Multi-task gaussian process prediction. *Advances in neural information processing systems, 20*, 153-160.

Castrejon, L. , Aytar, Y. , Vondrick, C. M. , Pirsiavash, H. , & Torralba, A. . (2016). Learning aligned cross-modal representations from weakly aligned data.

Chaudhuri, K. , Monteleoni, C. , & Sarwate, A. D. . (2011). Differentially private empirical risk minimization. *Journal of Machine Learning Research Jmlr, 12*(2), 1069.

Dai, W. , Yang, Q. , Xue, G. R. , & Yu, Y. . (2007). Boosting for Transfer Learning. *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*.

Daumé III, Hal. (2009). Frustratingly easy domain adaptation. *ACL*.

Dwork, C. . (2008). Differential privacy: a survey of results.

Dwork, C. , Kenthapadi, K. , Mcsherry, F. , Mironov, I. , & Naor, M. . (2006). Our Data, Ourselves: Privacy Via Distributed Noise Generation. *International Conference on Advances in Cryptology-eurocrypt*. DBLP.

Dwork, C. . (2012). Calibrating noise to sensitivity in private data analysis. *Lecture Notes in Computer ence, 3876*(8), 265-284.

Evgeniou, T. , & Pontil, M. . (2004). Regularized multi--task learning. *Tenth Acm Sigkdd International Conference on Knowledge Discovery & Data Mining*. ACM.

Glorot, X. , Bordes, A. , & Bengio, Y. . (2011). Domain Adaptation for Large-Scale Sentiment Classification: A Deep Learning Approach. *ICML*. Omnipress.

Gretton, A. , Sriperumbudur, B. , Sejdinovic, D. , Strathmann, H. , & Kenji, F. . (2012). Optimal kernel choice for large-scale two-sample tests. *Advances in Neural Information Processing Systems 25 (NIPS 2012)*.

Hamm, J. , Cao, P. , & Belkin, M. . (2016). Learning privately from multiparty data.

Huang, J. . (2006). Correcting sample selection bias by unlabeled data. *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*. MIT Press.

Jiang, J. , & Zhai, C. X. . (2007). Instance Weighting for Domain Adaptation in NLP. *Meeting of the Association of Computational Linguistics*.

Long, M. , & Wang, J. . (2015). Learning transferable features with deep adaptation networks.

Long, M. , Zhu, H. , Wang, J. , & Jordan, M. I. . (2016). Deep transfer learning with joint adaptation networks.

Mark, Dredze, Alex, KuleszaKoby, & Crammer. (2010). Multi-domain learning by confidence-weighted parameter combination. *Machine Learning*.

Mou, L. , Meng, Z. , Yan, R. , Li, G. , & † Yan Xu. (2016). How Transferable are Neural Networks in NLP Applications?. *EMNLP*.

Quionero-Candela, J. , Sugiyama, M. , Schwaighofer, A. , & Lawrence, N. D. . (2009). *Dataset Shift in Machine Learning*. The MIT

Press.

Raina, R. , Battle, A. , Lee, H. , Packer, B. , & Ng, A. Y. . (2007). Self-taught learning. 759-766.

Schwaighofer, A. , Tresp, V. , & Yu, K. . (2004). Learning Gaussian Process Kernels via Hierarchical Bayes. DBLP.

Smola, A. J. , & Schlkopf, B. . (2004). A tutorial on support vector regression. *Stats and Computing, 14*(3), 199-222.

Smola, A. , Gretton, A. , Song, L. , & Schlkopf, B. . (2007). A Hilbert Space Embedding for Distributions. *International Conference on Algorithmic Learning Theory*. Springer, Berlin, Heidelberg.

Sweeney, L. . (2002). K-anonymity: a model for protecting privacy. *INTERNATIONAL JOURNAL OF UNCERTAINTY FUZZINESS AND KNOWLEDGE BASED SYSTEMS*.

Tsuboi, Y. , Kashima, H. , Hido, S. , Bickel, S. , & Sugiyama, M. . (2009). Direct density ratio estimation for large-scale covariate shift adaptation. *Information and Media Technologies*.

Vapnik, V. N. . (1998). Statistical learning theory. *Encyclopedia of the ences of Learning, 41*(4), 3185-3185.

Wang, Y. , Gu, Q. , & Brown, D. . (2018). Differentially Private Hypothesis Transfer Learning. *European Conference on Machine Learning & Principles & Practice of Knowledge Discovery in Databases*.

Wei, Y. , Zheng, Y. , & Yang, Q. . (2016). Transfer knowledge between cities. 1905-1914.

Oregon State University. Dept. of Computer Science, Wu, P. , & Dietterich, T. G. . (2004). Improving svm accuracy by training on auxiliary data sources. *Icml*.

Xie, L. , Baytas, I. M. , Lin, K. , & Zhou, J. . (2017). Privacy-Preserving Distributed Multi-Task Learning with Asynchronous Updates. *Acm Sigkdd International Conference on Knowledge Discovery & Data Mining*. ACM.

Yosinski, J. , Clune, J. , Bengio, Y. , & Lipson, H. . (2014). How transferable are features in deep neural networks?. *International Conference on Neural Information Processing Systems*. MIT Press.

Zhang, Y. , & Yang, Q. . (2017). A survey on multi-task learning.

杨强，张宇，戴文渊，潘嘉林.(2020). 迁移学习. 中国：机械工业出版社.

https://github.com/jindongwang/transferlearning/tree/master/code/deep/DaNN

第 11 章

第 2 节　对抗网络

对抗网络是一种深度学习模型,是近年来复杂分布上无监督学习最具前景的方法之一。

1　对抗网络的历史

对抗网络即生成式对抗网络，自从 Goodfellow 在 14 年发表了论文 Generative Adversarial Nets [Goodfellow et al., 2014]以来，生成式对抗网络 GAN 就广受关注，随后便成为了当时深度学习最具代表性的发展成果，与此同时诸多学者也针对 GAN 现存的一些问题，提出了许多具有显著特点的改进模型。在学术界围绕 GAN 所产生的研究成果可谓是百花齐放、万紫千红，如同生成的图片一样，在深度学习领域绽放出五彩斑斓的花朵，下面本文将通过由浅至深、从理论到实践的过程带你进入生成式对抗网络的万花筒。

2　对抗生成网络基本概念

生成式对抗网络，英文名称为 Generative Adversarial Nets，它是一种生成模型，核心思想是从训练样本中学习所对应的概率分布，以期根据概率分布函数获取更多的“生成”样本来实现数据的扩张。

生成式对抗网络基于博弈论场景[Mirza et al., 2014]，其中生成网络必须要与对手竞争，生成器网络直接产生样本$\mathrm{x} = \mathrm{g}\left(z;\theta^{(g)}\right)$。其对手，判别器网络(discriminator network)试图区分从训练数据抽取的样本和从生成器抽取的样本。判别器发出由$\mathrm{d}\left(x;\theta^{(d)}\right)$给出的概率值，指示x是真实训练样本而不是从模型抽取的伪造样本的概率。

2.1　GANs的目的

生成式对抗网络的启发主要源于博弈论中的二人零和博弈，即指参与博弈的双方，在严格竞争下，一方的收益必然意味着另一方的损失，博弈双方的收益和损失相加总和永远为“零”，双方不存在合作。对于非合作、纯竞争型博弈，例如两个人打乒乓球，一个人赢则意味着另一人输；抽象后的博弈问题为：已知参与者集合(双方)，策略集合(乒乓球技术水平)和盈利集合(胜负)，能否找到一个理论上的平衡点，即对参与双方来说都最合理，最优的具体策略?冯·诺依曼已经从数学上证明，对二人零和博弈问题，可以通过一定的线性运算操作(即竞争双方以概率分布的形式随机使用某类最优策略中的各个策略)，找到一个最小最大的平衡点，这个著名的最小最大定理的思想是抱最好的希望，做最坏的打算。

众所周知，深度学习的主要驱动力为可利用的数据量(即输入与输出) [Lecun et al., 2015]，数据量越充分，训练得到的模型泛化能力(测试性能)越好。但在实际应用中，带有标记的数据很少，且代价昂贵；除了常用的统计扩张数据方式(例如裁剪、滑块和旋转角度、多分辨非下采样处理、加入服从不同分布下的随机噪声等，本质上，得到的样本可视为对抗样本且以多分辨特性、容许旋转不变性和鲁棒性等融入至模型内，但模型的预测(外插)能力受限于这种扩充方式)外，生成式对抗网络也可以无监督学习的方式实现数据的扩张，如图 2.1 所示。

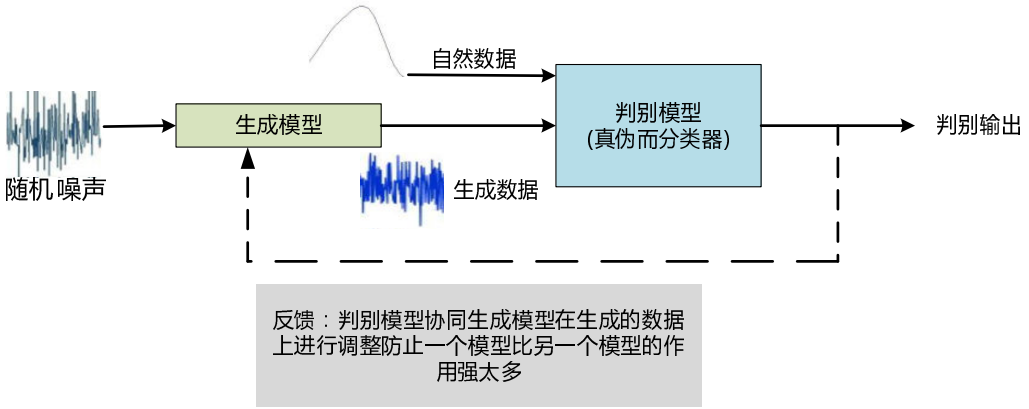

图 2.1 基于生成式对抗网络的数据扩张

2.2 GANs的数学描述

下面基于上述模型给出生成式对抗网络的数学原理及物理解释，首先，符号描述规定为：随机噪声$z \in \mathbb{R}^m$，自然数据$x \in \mathbb{R}^m$，生成数据为$\tilde{x} \in \mathbb{R}^n$；判别器为一个二分类器，所以$y \in [0, 1]^2$。

$$GAN\begin{cases} G: \tilde{x} = g(z, \ \theta^G) \in \mathbb{R}^n \\ D: \begin{cases} X = d(x, \ \theta^d) \\ \tilde{X} = d(\tilde{x}, \ \theta^d) \end{cases} \end{cases} \tag{2.1}$$

其中的 G 表示 Generator，即生成模型或称生成器，待优化参数为 θ^G，需进一步量化非线性映射函数 g(·)；另外 D 表示 Discriminator，即判别模型或判别器，待优化的参数为 θ^G；同样需量化映射 D(·)。

优化目标函数

GAN 的判别模型定义为：

$$y = \begin{bmatrix} D(x) \\ D(\tilde{x}) \end{bmatrix} = \begin{pmatrix} D(x) \\ D(G(z)) \end{pmatrix} \in \mathbb{R}^2 \tag{2.2}$$

其中$D(x) \in [0, 1]$为将 x 判断为真样本的概率。在固定生成模型时所对应判别器的损失函数为：

$$loss \ \ of \ \ Discriminator: min\left\{-\left[\sum_{x \sim P(x)} log(D(x, \ \ \theta^d)) + \sum_{\tilde{x} \sim P(\tilde{x})} log(1 - D(\tilde{x}, \ \ \theta^d))\right]\right\} \tag{2.3}$$

这里$x \sim P(x)$为服从自然数据分布$p(x)$下的采样，即式(2.1)中的自然数据集，对应着$\tilde{x} \sim P(\tilde{x})$为服从生成分布概率$P(\tilde{x})$下的采样，即式(2.1)中的生成数据集。其中$-\log(D(x))$解释为将x判断为真的不确定性越小越好，其最佳状态为 0，即$D(x) = 1$；另外$1 - \log(D(x))$的物理解释为将x判断为伪的不确定性越小越好，即$1 - D(\tilde{x})$为将$\tilde{x}$判断为伪的概率要越大越好，此意味着$D(\tilde{x})$为将$\tilde{x}$判断为真的概率越小越好；将所有采样样本的不确定性(也称信息量)进行求和，便得到熵的概念。简言之，判别模型的设计要求为：将自然数据判断为真的概率要高，将生成数据判断为伪的概率要高。

另外，对生成模型的要求是：在判别模型的固定时，生成数据的分布特性尽最大可能与自然数据的一致，即在$P(\tilde{x})$尽可能与$p(x)$一致的情形下，最大化如下目标函数:

$$\max_{\theta^G} \sum_{\tilde{x} \sim P(\tilde{x})} log(D(\tilde{x})) \tag{2.4}$$

对应着，将$\tilde{x} = G(z)$代入便有生成模型的优化目标函数：

$$\max_{\theta^g} \sum_{\tilde{x}\sim P(\tilde{x})} log(D(G(z,$$

$$\theta^g))) \tag{2.5}$$

即在z~P(z)的条件下，所有关于z的$\log\left(D\left(G(z)\right)\right)$的和越大，意味着：

$$D\left(G(z)\sim P(\tilde{x})\right) \longrightarrow d\left(G(z),\ x\right)$$

$$\longrightarrow \left(D\left(G(z)\right)\sim \mathrm{P}(x)\right) \tag{2.6}$$

生成数据与自然数据之间的差距$\mathrm{d}\left(G(z),\ x\right)$越小，即最为理想的状态是：关于所有的$z$，若都有$\log\left(D\left(G(z)\right)\right)=0$，则意味着$\mathrm{D}\left(G(z)\right)=1$，即将生成数据判别为自然数据(注意这是在生成模型阶段的要求)，即$\mathrm{D}\left(G(z)\right)$服从于自然数据的分布概率$P(\tilde{x})$，最终达到这两个分布概率$\mathrm{d}\left(P(\tilde{x}),\ P(x)\right)$尽可能接近。

最后，依据符号描述的定义，结合式(1.3)的损失函数，得到基于判别模型的优化目标函数为：

$$\max_{\theta^d} J(\theta^d,\ \theta^g) = \mathrm{E}_{x\sim p(x)}[log(D(x))] + \mathrm{E}_{\tilde{x}\sim p(\tilde{x})}[log(1-D(G(z,$$

$$\theta^g)))] \tag{2.7}$$

在优化目标公式(2.7)的基础上，然后融入生成模型的要求，得到最后优化目标函数：

$$\min_{\theta^g}\max_{\theta^d} J(\theta^d,$$

$$\theta^g) \tag{2.8}$$

随后利用梯度下降方法进行参数$\left(\theta^{\mathrm{d}},\ \theta^{\mathrm{g}}\right)$交替优化。

3 对抗生成网络的流程架构

3.1 GANs的几种基本结构

经典对抗生成网络模型 GAN

在 GAN 中，第一个网络通常被称为生成器并且以 G(z)表示，第二个网络通常被称为判别器并且以 D(x)表示，如图 3.2 所示为经典对抗生成网络模型的流程架构图[Goodfellow et al., 2014]。

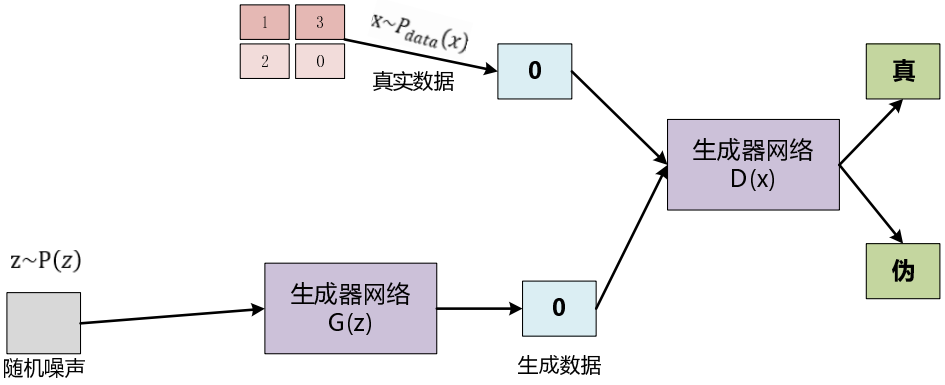

图 3.2 经典对抗生成网络模型流程架构

- 生成器

生成器网络以随机的噪声作为输入并试图生成样本数据。 在上图中，我们可以看到生成器 G(z)从概率分布 p(z)中接收输入 z，并且生成数据提供给判别器网络 D(x)。

- 判别器

判别器网络以真实数据或者生成数据为输入，并试图预测当前输入是真实数据还是生成数据。其中一个输入 x 从真实的数据分布 P(x)中获取，接下来解决一个二分类问题并产生一个范围在 0 和 1 之间的标量。

深度卷积神经网络架构下的生成式对抗网络 DCGAN

深度卷积生成对抗神经网络(Deep Convolutional Generative Adversarial Network， DCGAN) [Denton et al., 2015]是将卷积神经网络和对抗网络结合起来的用于图像生成的网络模型，如图 3.3 所示为深度卷积对抗生成网络的结构。

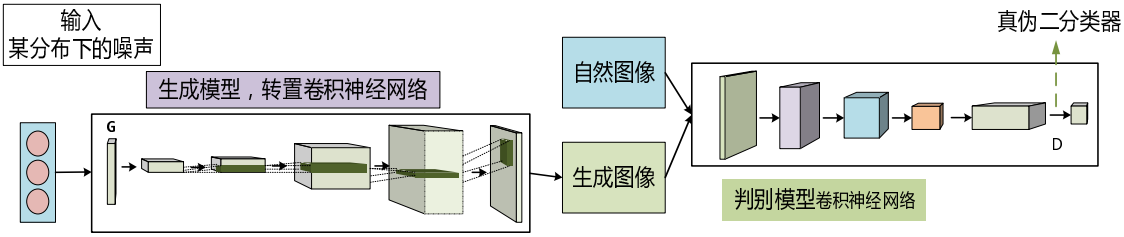

图 3.3 深度卷积对抗网络结构图

上图中的生成模型与判别模型可以采纳传统卷积神经网络的架构，下面通过例子来说明生成模型和判别模型。

- 生成模型

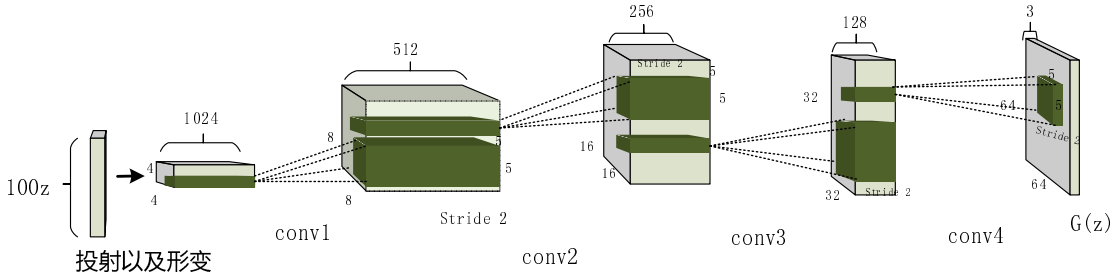

图 3.4 深度卷积对抗网络的生成模型

假设模型的输入噪声为服从均匀分布下的随机采样，即：

$$z \in \mathbb{R}^{100}$$

$$\sim P(z) \qquad (3.1)$$

如何利用输入的噪声通过卷积神经网络得到输出，即生成的图像，故模型引入了转置卷积层通过对噪声进行转置卷积操作来生成相应的图像。其中转置卷积也称微步卷积，它可以视为传统卷积操作的一种“逆向”传递过程。通常，转置卷积受“正向”卷积的参数约束，即步长 Stride 和填充方式(Zero-Padding 和 Full Padding)。下面将通过一个简单的例子来说明转置卷积与卷积的关系。

- 例 1：无填充，步长不为 1 的情况下，卷积与转置卷积的关系如下图 3.5 所示。

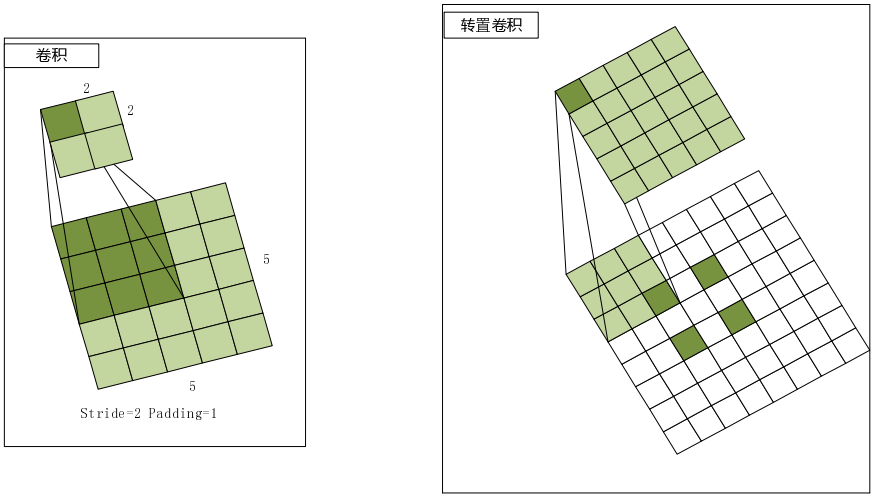

图 3.5 卷积与转置卷积的关系

首先，对于卷积操作而言，输入为$x \in \mathbb{R}^{5\times5}$，且步长 Stride 为 2，填充 padding 为 0；滤波器 w 的尺寸大小为 3×3，即 kernel 的尺寸为 3，那么输出可以通过如下的公式计算：

$$Output = \left\lfloor \frac{Input - kernel + 2 \cdot padding}{Stride} \right\rfloor + 1 \tag{3.2}$$

可知输出为$y = Conv(x，w，'valid') \in \mathbb{R}^{2\times2}$。

其次，对于转置卷积而言，它能够回答如何由$y \in \mathbb{R}^{2\times2}$得到 $x \in \mathbb{R}^{5\times5}$，即卷积的“逆向”过程，这里先需要计算新的步长与填充参数，利用公式有：

$$\begin{cases} Stride^{(New)} = 1 \\ padding^{(New)} = kernel - 1 \end{cases} \tag{3.3}$$

注意此时的输入为$y \in \mathbb{R}^{2\times2}$，即新的输入尺寸，则新的输出尺寸计算如下：

$$\begin{cases} Output^{(New)} = Stride \cdot (Input^{(New)} - 1) + kernel^{(New)} \\ kernel^{(New)} = kernel \end{cases} \tag{3.4}$$

可以得到$x \in \mathbb{R}^{5\times5}$。

- 判别模型

判别模型的某一网络架构如下图 3.6 所示。

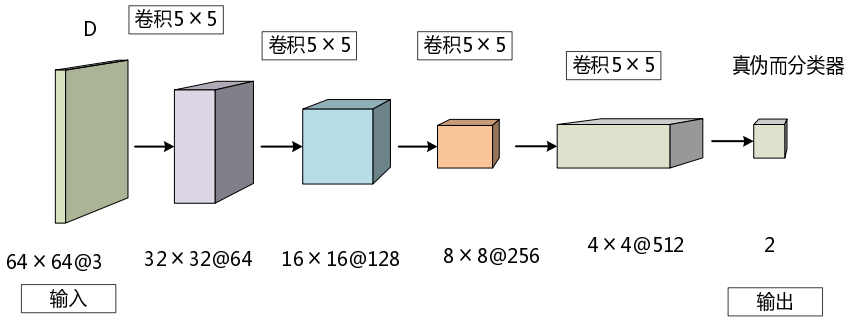

图 3.6 判别模型的网络架构

判别模型的输入为自然数据集，以及生成模型的输出(生成数据)，输出所对应的类别即为真(自然数据为 1)伪(生成数据为 0)，其中该模型可以使用传统的卷积神经网络如 LeNet、AlexNet、GoogleNet、VGGNet 等经典的网络模型，不过需要注意的是：一是真伪二分类器仍可沿用 Softmax 分类器(即退化为 Logistic 分类器)，也可以使用非线性分类器(非线性函数为 Tanh 或 Sigmoid 函数);二是该模型中所有的池化层被卷积

(融入步长)操作所替代;三是除真伪二分类器这一层外，所有隐层使用的非线性函数为修正线性单元(ReLU)的改进版 Leaky ReLU。

3.2 GANs的基本流程

根据上述章节描述，对抗生成网络的训练流程可表述如下[Goodfellow et al., 2014]。

for 训练的迭代数 **do**

  for 每迭代 k 步 **do**

    从概率分布$p_g(z)$中生成 m 个随机噪声$\{z^{(1)}, \ldots, z^{(m)}\}$；

    从概率分布$p_{data}(x)$中选择 m 个真实样本$\{x^{(1)}, \ldots, x^{(m)}\}$；

    利用如下梯度公式，通过梯度下降法更新判别器参数：

$$\nabla\theta_d \frac{1}{m}\sum_{i=1}^{m}\left[logD_{\theta_d}\left(x^{(i)}\right) + log\left(1 - D_{\theta_d}\left(G_{\theta_g}\left(z^{(i)}\right)\right)\right)\right];$$

  End for

    从概率分布$p_g(z)$中生成 m 个随机噪声$\{z^{(1)}, \ldots, z^{(m)}\}$；

    利用如下梯度公式，通过梯度上升法更新生成器参数；

$$\nabla\theta_d \frac{1}{m}\sum_{i=1}^{m} log\left(D_{\theta_d}\left(G_{\theta_g}\left(z^{(i)}\right)\right)\right);$$

End for

3.3 模型的理论分析

众所周知，经典的生成式对抗网络 GAN 有着严格的系统理论分析与收敛性证明，即假设生成模型和判别模型都有足够的性能的条件下，如果在迭代过程中的每一步，判别模型都可以达到当下在给定生成模型时的最优值，并在这之后再更新生成模型，那么最终生成数据的概率分布函数就一定会收敛于自然数据的概率分布函数。lan Goodfellow 等人最初假设判别模型具有无限区分能力，即不论生成数据以任意小的误差或准则接近自然数据，判别模型均可有效地识别。可问题是:若生成数据的分布函数与自然数据的分布函数接近，但其支撑集互不相交或重叠(特别当这两个分布函数是低维流型的时候，容易发生)，则生成模型所对应的优化目标函数(即詹森香农散度，它给出了生成数据与自然数据所对应分布函数之间的差异性)关于参数的偏导数退化为一常数，从而导致梯度消失，发生梯度弥散现象，换言之判别模型越好，生成模型的梯度消失越严重。

为了有效地解决这一问题，目前从理论分析上，学者提出了两个思路，一种是修正生成模型的优化目标函数，即利用沃瑟斯坦距离衡量生成数据与自然数据所对应分布函数之间的差异性，替代传统的詹森香农散度，通过最优传输定理，即生成数据的分布概型与自然数据的分布概型之间存在一个唯一的映射，记为 Wasscrstein-GAN。另一种是关于判别器具有无限可分性的假设，修正为自然数据的概率分布特性具有 Lipschitz 连续且(有限阶)可微性，从而优化目标函数变为:在生成数据的分布函数与自然数据的分布函数尽可能一致的条件下(利用二者差的期望来衡量一致性)，优化判别器带有有限阶 Lipschitz 概型约束的可分能力，记为损失敏感度生成式对抗网络 Loss-Sensitive GAN，本质上，它与 Wasserstein-GAN 具有一定的相似性。

4 对抗网络具体应用与实现

GAN 在多个领域都产生了许多让人振奋的结果[Gauthier et al., 2014]。同时本章将按照 GANs 的定义并利用最新的深度学习框架实现两种对抗生成网络。

基于生成式对抗网络分类任务的应用主要包含两方面的贡献：一是利用生成模型进行数据扩充或利用少数有类别数据对扩充后的数据进行“类标传递”；二是利用判别模型可进行共享计算或特征学习阶段的参数初始化；注意这里的参数初始化与之前基于自编码网络的逐层学习机制有所不同，它是整个(判别模型中的特征学习)阶段的参数初始化。

对抗式生成网络已经在图像分类、检测、分割、高分辨率图像生成等诸多领域取得了突破性的成绩。但是它也存在一些问题，首先，它与传统的机器学习和深度学习方法一样，通常，假设训练数据与测试数据服从同样的分布，或者是在训练数据上的预测结果与在测试数据上的预测结果服从同样的分布。而实际上这两者存在一定的偏差，例如在测试数据上的预测准确率就通常比在训练数据上的要低，这就是过度拟合的问题。下面详细地陈述深度卷积生成对抗网络在分类任务、超分辨任务和分割任务上的应用以及对这一问题所做出的改进。

4.1 GANs的应用

目前，对于分类任务，深度神经网络的范式主流为半监督学习，即利用无类别数据进行参数初始化，有类别数据进行网络的精调[Han et al., 2016]。

基于生成式对抗网络分类任务的应用主要包含两方面的贡献：一是利用生成模型进行数据扩充或利用少数有类别数据对扩充后的数据进行“类别传递”；二是利用判别模型可进行共享计算或特征学习阶段的参数初始化；注意这里的参数初始化与之前基于自编码网络的逐层学习机制有所不同，它是整个(判别模型中的特征学习)阶段的参数初始化。

当然，生成式对抗网络也并不完美，首先，优化过程存在不稳定性，很容易陷入到一个鞍点或局部极值点上；其次，模型的可解释性比较差；再次，需要提高生成式对抗网络模型的延展性，尤其在处理大规模数据的时候。相应的分析及策略如下：由于生成模型的输入为随机噪声，将会对深度卷积对抗网络的收敛产生较大的震荡，避免随机噪声的影响，实际中，常将生成模型的输入更改为自然数据的编码系数(如利用自编码网络，对输入(自然数据)进行编码后的编码系数，这样一方面可以增加生成网络模型的拓扑(几何)结构对应性，另一方面可以提升整体网络的稳定性。另外模型的设计通常需要自然数据的个数要远大于生成模型的参数量，以期保证生成数据的质量。

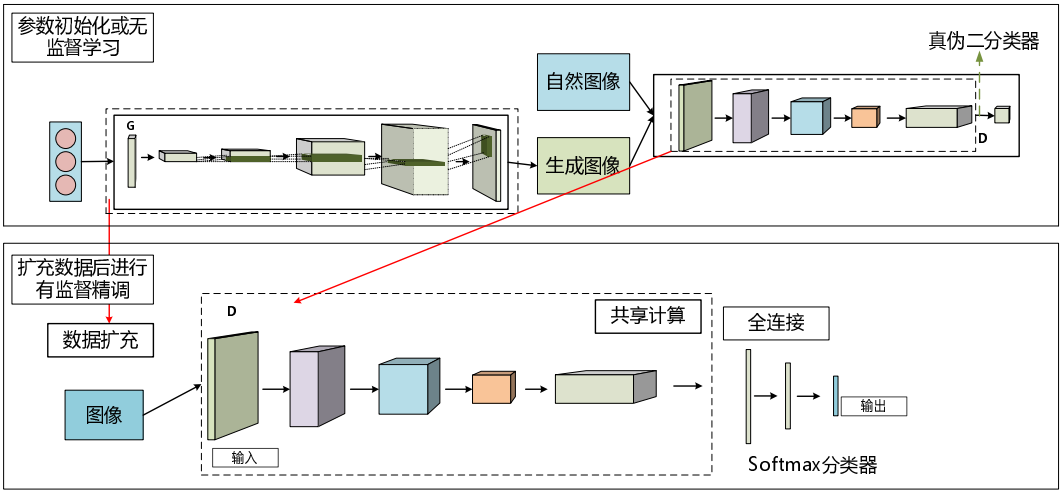

图 4.1 数据扩充与共享计算

- 超分辨任务

超分辨任务的核心是在寻找低分辨图像(高分辨图像的某种退化操作所得到的)与高分辨图像之间的关系[Arjovsky et al., 2017]，期望利用量化后的关系将低分辨图像通过恢复生动纹理和颗粒细节等以达到高分辨图像的过程，如图 4.2 所示。

与稀疏编码和传统深度神经网络方法(寻求在某一特征空间上，低分辨与高分辨图像所蕴含的特性是一致的)不同的是：深度卷积对抗生成网络(这里生成模型与判别模型的设计仍采用卷积架构，只不过生成模型是“上采样”的过程而判别模型是“下采样”的过程)使用零和博弈的策略在生成模型与判别模型之间寻求平衡(非合作纳什均衡点)。这里的生成网络采用具有 16 个残差模块的网络(残差网络仍为一种卷积架构)，而判别模型使用的是 VGG 网络(分类器为高低分辨率二分类器)，与稀疏编码和传统深度学习模型做图像超分辨率的结果相比可以发现深度卷积生成对抗网络的结果能够提供更丰富的细节，这也是该模型做图像生成

时的一个显著优点，即能够提供更锐利的纹理和颗粒细节。

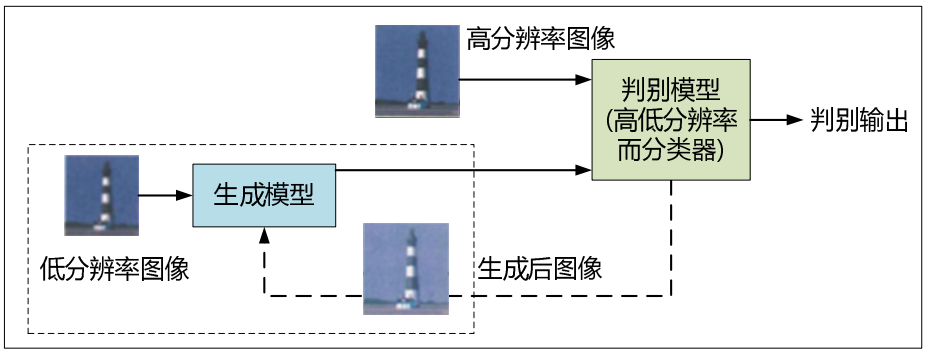

图 4.2 基于深度卷积生成对抗网络的超分辨任务

- 分割任务

图像分割就是把图像分成若干个特定的，具有特定性质的区域并提出感兴趣目标的技术和过程。分割任务流程如图 4.3 所示。

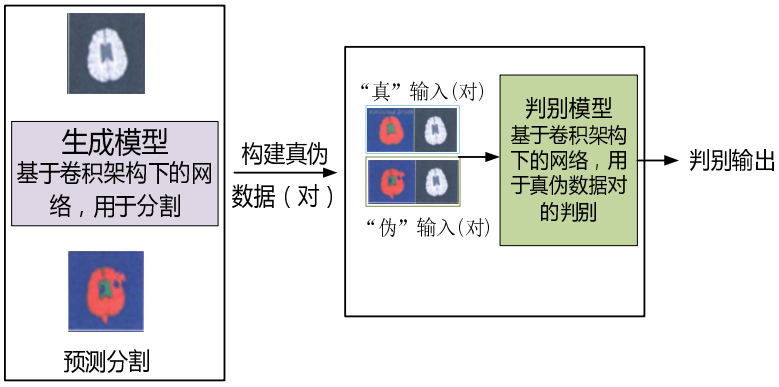

图 4.3 基于深度卷积网络生成对抗网络的分割任务

- 堆栈生成式对抗网络

问题描述：众所周知，通过文本描述来得到实际场景图像一直是计算机视觉中的难题，目前，诸多已知的处理方式所得到的图像仅能够粗略地反映文本所描述的意思，但常失效于必要的细节和清晰的目标。

为了有效地解决这一实际应用问题，Han Zhang 等人提出了堆栈生成式对抗网络[Han Zhang et al., 2015]，模型主要包括两个阶段，第一阶段基于条件生成式对抗网络(C-GAN)，依据文本的描述绘制目标的基本颜色和初始形状，从而产生低分辨图像的描述；第二阶段同样基于条件生成式对抗网络(注意与第一阶段的网络设计不同)，将文本描述与第一阶段的输出作为该阶段的输人，以获取高分辨的图像，以期弥补细节或精细化处理。基于条件生成式对抗网络的低分辨图像描述如图 4.4 所示。

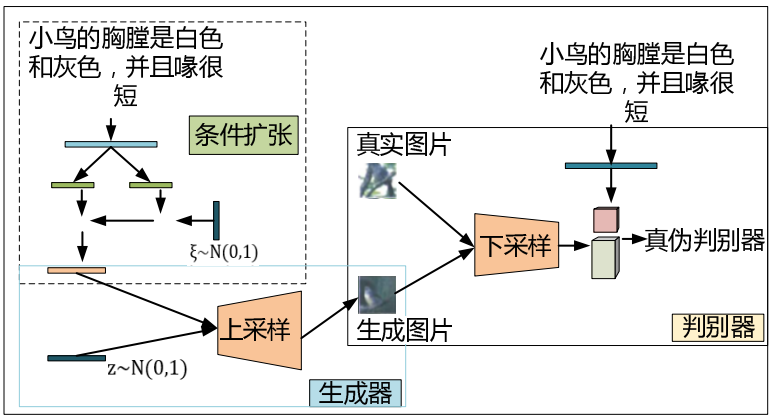

图 4.4 基于条件生成式对抗网络的高分辨任务精修

- 对偶学习范式下生成式对抗网络

从 2016 年开始，无监督学习进入实质性发展阶段，逐层学习生成式对抗网络等为无监督方式下的深度学习注入新的活力，同时对偶学习也为研究过程中所遇到的困难提供了新的思路。针对机器“翻译”任务，包括图片不同风格间的转化、不同语种间的翻译等，对偶学习范式下的生成式对抗网络，充分利用未标注的数据，提高对偶任务中的两个“翻译”模型的性能，如图 4.5 所示。下面针对图片不同风格间的转化任务(如彩色照片与素描图之间的转化)给予详细的描述和数学分析。

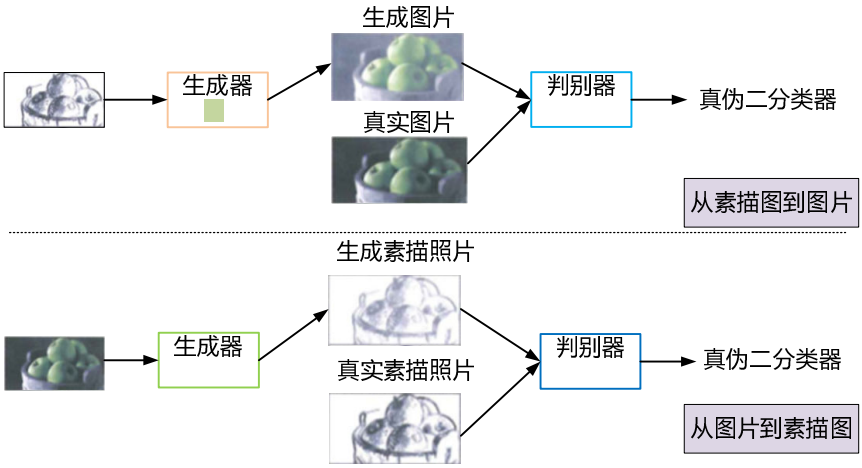

图 4.5 基于对偶生成式对抗网络的图片风格转化任务

4.2 利用TensorFlow实现LSGAN

最小二乘 GAN(Least Squares GAN, LSGAN) [Mao,X et al., 2014]，正如它的名字所说的，针对传统 GAN 使用交叉熵函数作为代价函数，从而容易出现梯度消失和模式崩塌等问题，LSGAN 使用最小二乘损失代替交叉熵损失，解决了传统 GAN 生成的图片质量不高，训练过程不稳定的问题。下面本节将使用 Tensorflow 2.0 深度学习框架实现 LSGAN 模型，并将该模型用于 MNIST 手写数字数据集的图片生成实验。

根据 LSGAN 定义，其生成器的损失为：

$$l_G = \frac{1}{2}E_{z\sim p(z)}\left[\left(D\left(G(z)\right)-1\right)^2\right]$$

其判别器的损失定义为：

$$l_D = \frac{1}{2}E_{x\sim p_{dt}}[(D(x)-1)^2] + \frac{1}{2}E_{z\sim p(z)}\left[\left(D\left(G(z)\right)\right)^2\right]$$

根据如上公式，各个模块的详细代码如下所述：

- 定义生成器的输入，即取值范围为[-1,1]的均匀分布随机向量，产生噪声的函数如下所示。

```
def sample_noise(batch_size, dim):
    """从-1 到 1 范围中生成随机噪声
    输入:
    - batch_size: 产生噪声的批量大小
    - dim: 产生噪声的维度
    输出:
    TensorFlow Tensor 范围 [-1, 1] 尺寸 [batch_size, dim]
    """
    return tf.random.uniform(shape=[batch_size, dim], minval=-1, maxval=1)
```

- 判别器模型的定义，判别器模型由三层全连接层组成，具体的超参数设置如下所示。

```
def discriminator():
    """为一批输入图像计算判别器评分。

    输入:
    - x: 图像平铺后的 TensorFlow Tensor, 尺寸 [batch_size, 784]

    输出:
     TensorFlow Tensor 其尺寸为 [batch_size, 1], 对于每个输入图像，包含图像的真实分数。
     """
    model = tf.keras.models.Sequential([

        tf.keras.layers.Dense(256, input_shape=(784, ), use_bias=True),   # 256 个结点的全连接层
        tf.keras.layers.LeakyReLU(alpha=0.01),
        tf.keras.layers.Dense(256, use_bias=True),        # 同上
        tf.keras.layers.LeakyReLU(alpha=0.01),
        tf.keras.layers.Dense(1, use_bias=True),          # 输出层，输出真假图片评分

    ])
    return model
```

- 生成器的定义，生成器由三层全连接层组成，具体的超参数设置如下所示。

```
def generator(noise_dim=NOISE_DIM):
    """从随机噪声向量生成图片

    输入:
    - z: TensorFlow Tensor 随机噪声，尺寸 [batch_size, noise_dim]

    输出:
    TensorFlow Tensor 生成的图像, 尺寸 [batch_size, 784].
    """
    model = tf.keras.models.Sequential([

        tf.keras.layers.Dense(1024, input_shape=(noise_dim,), use_bias=True, activation='relu'),
        tf.keras.layers.Dense(1024, use_bias=True, activation='relu'),
        tf.keras.layers.Dense(784, use_bias=True, activation='tanh')

    ])
    return model
```

- 定义判别器的 LS 损失，如下所示。

```
def ls_discriminator_loss(scores_real, scores_fake):
    """计算判别器的 LS 损失值

    输入:
    - scores_real: Tensor 尺寸 (N, 1) 真实数据的预测分数。
    - scores_fake: Tensor 尺寸 (N, 1) 假数据的预测分数。

    输出:
    - loss: Tensor LS 损失值。
    """

    loss = (tf.reduce_mean(tf.square(scores_real - 1)) + tf.reduce_mean(tf.square(scores_fake))) / 2

    return loss
```

- 定义生成器的 LS 损失函数，如下所示。

```
def ls_generator_loss(scores_fake):
    """计算生成器的 LS 损失值

    输入:
    - scores_fake: Tensor 尺寸 (N, 1) 假数据的预测分数。

    输出:
    - loss: Tensor LS 损失值。
    """

    loss = tf.reduce_mean(tf.square(scores_fake - 1)) / 2

    return loss
```

- 定义生成器和判别器的优化方法，本次实验使用 Adam 优化算法，其他方法读者可参考最新的 Tensorflow API，实现代码如下所示：

```
def get_solvers(learning_rate=1e-3, beta1=0.5):
    """为 LSGAN 的训练过程创建优化器

    输入:
    - learning_rate: 两个优化器的学习率。
    - beta1: Adam 算法的 beta1 参数

    输出:
    - D_solver: tf.optimizers.Adam 的实例
    - G_solver: tf.optimizers.Adam 的实例
    """

    D_solver = tf.optimizers.Adam(learning_rate=learning_rate, beta_1=beta1)
    G_solver = tf.optimizers.Adam(learning_rate=learning_rate, beta_1=beta1)

    return D_solver, G_solver
```

- 定义数据集迭代器，本次实验使用 MNIST 数据集，代码如下所示。

```
class MNIST(object):
    def __init__(self, batch_size, shuffle=False):
        """定义一个关于 MNIST 数据集的可迭代对象

        输入:
        - batch_size: 每次训练批次的大小
        - shuffle: 是否在每个 epoch 中事先打乱数据
        """
        train, _ = tf.keras.datasets.mnist.load_data(path='mnist.npz')        # 读取 mnist 数据集
        X, y = train
        X = X.astype(np.float32)/255   # 归一化
        X = X.reshape((X.shape[0], -1))       # 将图片三位数据平铺
        self.X, self.y = X, y
        self.batch_size, self.shuffle = batch_size, shuffle

    def __iter__(self):          # 迭代函数，每次返回一个 batch_size 大小的数据
        N, B = self.X.shape[0], self.batch_size
        idxs = np.arange(N)
        if self.shuffle:
            np.random.shuffle(idxs)
        return iter((self.X[i:i+B], self.y[i:i+B]) for i in range(0, N, B))
```

得到图片样例输出如下图 4.6 所示：

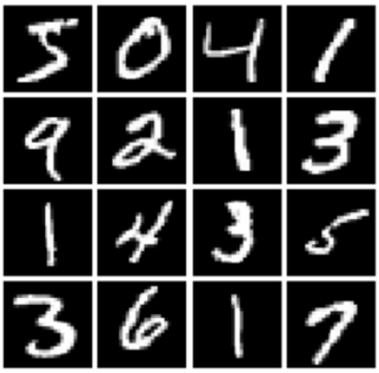

图 4.6 MNIST 数据集输出样例

- 定义训练的主函数，基于 Tensorflow 框架，定义 LSGAN 的训练主函数如下所示。

```
def run_a_gan(D, G, D_solver, G_solver, discriminator_loss, generator_loss,\
                    show_every=20, print_every=20, batch_size=128, num_epochs=10, noise_size=96):
    """训练 LSGAN 模型。

    输入:
    - D: 判别器模型
    - G: 生成器模型
    - D_solver: 判别器的优化器
    - G_solver: 生成器的优化器
    - generator_loss: 生成器损失
    - discriminator_loss: 判别器损失
    """
    mnist = MNIST(batch_size=batch_size, shuffle=True)

    iter_count = 0
    for epoch in range(num_epochs):
        for (x, _) in mnist:
            with tf.GradientTape() as tape:
                real_data = x
                logits_real = D(preprocess_img(real_data))

                g_fake_seed = sample_noise(batch_size, noise_size)
                fake_images = G(g_fake_seed)
                logits_fake = D(tf.reshape(fake_images, [batch_size, 784]))

                d_total_error = discriminator_loss(logits_real, logits_fake)
                d_gradients = tape.gradient(d_total_error, D.trainable_variables)
                D_solver.apply_gradients(zip(d_gradients, D.trainable_variables))
```

```
                with tf.GradientTape() as tape:
                    g_fake_seed = sample_noise(batch_size, noise_size)
                    fake_images = G(g_fake_seed)

                    gen_logits_fake = D(tf.reshape(fake_images, [batch_size, 784]))
                    g_error = generator_loss(gen_logits_fake)
                    g_gradients = tape.gradient(g_error, G.trainable_variables)
                    G_solver.apply_gradients(zip(g_gradients, G.trainable_variables))

                if (iter_count % show_every == 0):
                    print('Epoch: {}, Iter: {}, D: {:.4}, G:{:.4}'.format(epoch,
iter_count,d_total_error,g_error))
                    imgs_numpy = fake_images.cpu().numpy()
                    show_images(imgs_numpy[0:16])
                    plt.show()
                iter_count += 1

        # 随机噪声，用于产生测试用例
        z = sample_noise(batch_size, noise_size)
        # 生成的图片
        G_sample = G(z)
        print('Final images')
        show_images(G_sample[:16])
        plt.show()
```

- 训练主流程，代码如下所示。

```
    # 定义判别器模型
    D = discriminator()

    # 定义生成器模型
    G = generator()

    # 定义判别器和生成器的优化器
    D_solver, G_solver = get_solvers()

    # 开始训练
    run_a_gan(D, G, D_solver, G_solver, discriminator_loss, generator_loss)
```

- 最终模型训练结果，如下所示：

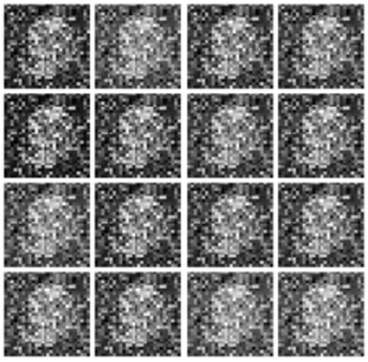

图 4.7 迭代 100 次后生成器生成的图片

可以看到经过简单的 100 次迭代很难让生成器产生有效的图片生成规则。

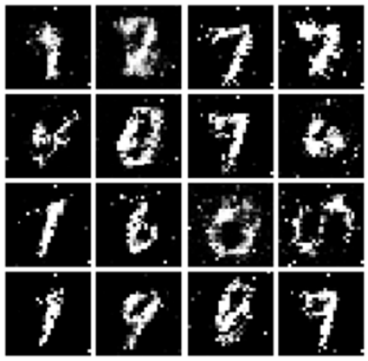

图 4.8 迭代 5000 次后生成器生成的图片

可以看到经过 5000 次迭代，生成器能够生成具有一定意义的手写字符，但由于并未引入卷积和转置卷积层，所以生成图片的能力较差，相应的改进模型请有兴趣的读者自己实现。

4.3 利用TensorFlow实现深度卷积DCGAN

深度卷积生成对抗神经网络(Deep Convolutional Generative Adversarial Network， DCGAN) [Denton et al., 2015]是将卷积神经网络和对抗网络结合起来的用于图像生成的网络模型，其详细的结构介绍上一节已经给出，本节将不再赘述其模型的详细结构。下面本节使用 Tensorflow 2.0 实现深度卷积 GAN，并将该模型应用于 MNIST 手写数字生成任务，实现过程如下所示。

- 定义判别器模型，该判别器模型在原来 GAN 的基础上引入了卷积池化等操作，实现代码如下所示。

```
def discriminator():
    """为一批输入图像计算判别器评分。

    输入:
    - x: 平铺后图片的 TensorFlow Tensor, 尺寸 [batch_size, 784]

    输出:
    TensorFlow Tensor 尺寸 [batch_size, 1], 对于每个输入图像，包含图像的真实分数。
    """
    model = tf.keras.models.Sequential([
        tf.keras.layers.Reshape((28, 28, 1), input_shape=(28,28,1)),
        # 定义卷积层
        tf.keras.layers.Conv2D(32, kernel_size=5, strides=1, padding='VALID', use_bias=True),
```

```
        tf.keras.layers.LeakyReLU(alpha=0.01),
        tf.keras.layers.MaxPool2D(pool_size=2, strides=2), # 最大池化层
        tf.keras.layers.Conv2D(64, kernel_size=5, strides=1, use_bias=True, padding='VALID'),
        tf.keras.layers.LeakyReLU(alpha=0.01),
        tf.keras.layers.MaxPool2D(pool_size=2, strides=2),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(4*4*64, use_bias=True),
        tf.keras.layers.LeakyReLU(alpha=0.01),
        tf.keras.layers.Dense(1, use_bias=True),

    ])
    return model
```

- 定义生成器模型，在原有 GAN 模型的生成器的基础上，引入了转置卷积操作，使得生成器更擅长于图片的生成，实现代码如下所示。

```
def generator(noise_dim=NOISE_DIM):
    """从随机噪声向量中生成对应的图片

    输入:
    - z: 一个批量的随机噪声的 TensorFlow Tensor [batch_size, noise_dim]

    输出:
    生成图片的 TensorFlow Tensor, 尺寸 [batch_size, 784].
    """
    model = tf.keras.models.Sequential()

    layers = [
        tf.keras.layers.Dense(1024, activation='relu', input_shape=(noise_dim,), use_bias=True),
        tf.keras.layers.BatchNormalization(trainable=True),
        tf.keras.layers.Dense(7*7*128, activation='relu', use_bias=True),
        tf.keras.layers.BatchNormalization(trainable=True),
        tf.keras.layers.Reshape((7, 7, 128)),
        # 定义转置卷积层
        tf.keras.layers.Conv2DTranspose(64, kernel_size=4, strides=2, activation='relu',
padding='SAME', use_bias=True),
        tf.keras.layers.BatchNormalization(trainable=True),
        tf.keras.layers.Conv2DTranspose(1, kernel_size=4, strides=2, activation='tanh', padding='SAME',
use_bias=True),
        tf.keras.layers.Flatten()
    ]
    model = tf.keras.models.Sequential(layers)
    return model
```

- 最终模型训练结果，如下所示：

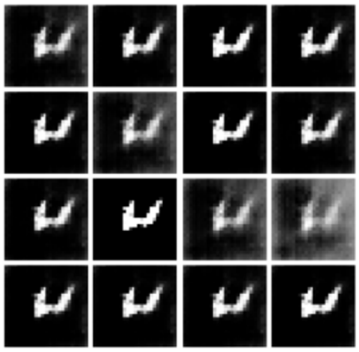

图 4.9 迭代 100 次后生成器生成的图片

可以看到经过简单的 100 次迭代很难让生成器产生有效的图片生成规则，但相比上一节实现的 LSGAN，生成器生成的图片更具有线条感，这于引入的转置卷积层有一定的联系。

图 4.10 迭代 1000 次后生成器生成的图片

可以看到经过 1000 次迭代，生成器能够生成具有一定意义的手写字符，但由于引入了卷积和转置卷积层，所以在训练方面仍然具有一定的复杂性，所以本节实验推荐有条件的读者使用 GPU 来加速整个训练的过程。

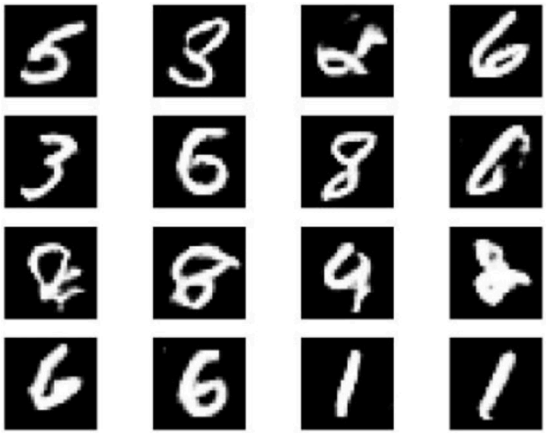

图 4.11 迭代 10000 次后生成器生成的图片

经过 10000 次高强度的迭代，生成器能够生成与人手写数字极为相似的图片，并能够有效的产生随机手写图片。

5 对抗网络的展望与现存问题

随着生成模型的不断发展，不断有学者提出新的技术和方法，GAN 的提出是生成模型里程碑式的发展成果，针对最初的 GAN 网络，学者们提出了诸多 GAN 的变种网络模型，同时也发现了这类生成网络模型存在的问题，本章将进一步讲述 GAN 网络的各种改进模型以及 GANs 模型现存的问题与挑战。

5.1 GANs的爆发年

2017 年对 GAN 是特殊的一年，这一年中很多学者提出了各种 GAN 的改进版本，下面将依次简单介绍 GAN 网络的诸多版本中的几个。

LSGANs 这篇经典的论文主要是把交叉熵损失函数换做了最小二乘损失函数[Mao,X et al., 2014]，这样做作者认为改善了传统 GAN 的两个问题，即传统 GAN 生成的图片质量不高以及训练过程十分不稳定。

使用交叉熵虽然会让我们分类正确，但是这样会导致那些在决策边界被分类为真的、但是仍然远离真实数据的假样本（即生成器生成的样本）不会继续迭代，因为它已经成功欺骗了判别器，更新生成器的时候就会发生梯度弥散的问题。

论文指出最小二乘损失函数会对处于判别成真的那些远离决策边界的样本进行惩罚，把远离决策边界的假样本拖进决策边界，从而提高生成图片的质量。如图 5.1 所示，为 LSGANs 生成的图片样例。

图 5.1 LSGANs 生成的图片

原始版本的 GAN 使用的交叉熵损失最小化，这种方式会产生梯度消失和模式崩塌等问题，WGAN 引入了 Wasserstein 距离[Arjovsky et al., 2017]，由于它相对 KL 散度与 JS 散度具有优越的平滑特性，理论上可以解决梯度消失问题。接着通过数学变换将 Wasserstein 距离写成可求解的形式，利用一个参数数值范围受限的判别器神经网络来最大化这个形式，就可以近似 Wasserstein 距离。在此近似最优判别器下优化生成器使得 Wasserstein 距离缩小，就能有效拉近生成分布与真实分布。WGAN 既解决了训练不稳定的问题，也提供了一个可靠的训练进程指标，而且该指标确实与生成样本的质量高度相关，如图 5.2 所示为 WGAN 生成的图片样例。

图 5.2 WGAN 生成的图片

5.2 GANs模型的挑战

- 初始化问题

如果你仔细思考上面的定义，会发现这正是 GAN 试图做的事情：生成器和判别器最终达到了一个如果对方不改变就无法进一步提升的状态。梯度下降的启动会选择一个减小所定义问题损失的方向，但是我们

并没有一个办法来确保利用 GAN 网络可以进入纳什均衡的状态，这是一个高维度的非凸优化目标。网络试图在接下来的步骤中最小化非凸优化目标，最终有可能导致进入振荡而不是收敛到底层真实目标

在大多数情况下，当判别器损失非常接近于 0 时，那么可以确信你的模型出现了问题，然而更困难的问题是如何找到问题究竟出现在哪里。

一个常见的优化 GAN 训练过程的手段是故意停止某个网络的学习过程，或者降低学习率，使得另一个网络可以追上来。在大多数场景下，生成器是落后的一方，我们需要让判别器进行等待。在特定情况下这样做可能没有问题，但是我们需要记住想要让生成器的质量更好，需要判别器的质量更好，反之亦然。因此，理想情况下我们希望两个网络以同样的速率同时进行改善。 判别器最理想的损失接近于 0.5，在这个情况下对于判别器来说其无法从真实图像中区分出生成的图像。

- 模式坍塌

生成对抗网络中最主要的一种失败模式被称为模型明塌。其中的基本原理是生成器可能会在某种情况下重复生成完全一致的图像，这其中的原因和博弈论中的启动相关。我们可以这样来想象对抗神经网络的训练过程，先从判别器的角度试图最大化，再从生成器的角度试图最小化。如果生成器最小化开始之前，判别器已经完全最大化，所有工作还可以正常运行。然而如果我们通过另一种方式首先尝试最小化生成器，接下来尝试最大化判别器，那么工作就无法正常运行。原因在于如果我们保持判别器不变，它会将空间中的某些点标记为最有可能是真的而不是假的，这样生成器就会选择将所有的噪声输入映射到那些最可能为真的点。

第 12 章

第 1 节　机器博弈

机器博弈被认为是人工智能领域最具挑战性的研究方向之一。本章学习博弈过程中涉及逻辑推理、形象思维、优化选择等多种人类智能的计算机实现的方法和技术。

6　机器博弈

长期以来，人们认为如果计算机能够掌握难度较大的棋类博弈（如奥赛罗和西洋双陆棋），那么就足以证明它们具有真正的人工智能。经过 50 多年的研究，事实证明，在这些博弈中，计算机展现出了很大的优势（表现方面），例如，肯尼思·汤普森(图 6.1)开发了国际象棋的残局数据库，但这并不一定是研究人员所希望的“强人工智能”。

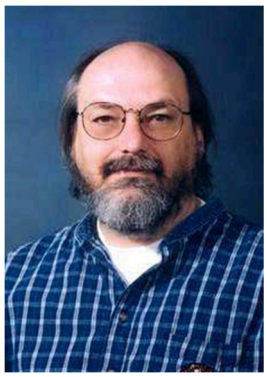

图 6.1 肯尼思·汤普森(Kenneth Lane Thompson)

6.1 机器博弈概述

机器博弈也称计算机博弈（Computer Games），英文直译应该是计算机游戏，其覆盖面非常广泛。然而，从事计算机棋牌竞技研究的科学家们，很早便将 Computer Games 定义为让计算机能够像人一样会思考和决策，能够下棋。为此还成立了 International Computer Games Association（ICGA—国际机器博弈协会），专门组织世界范围内的棋类（后又加入牌类）博弈竞赛和学术交流。为了和计算机游戏区别开来，Computer Games 中文名字便称之为机器博弈，或者计算机博弈。

几个世纪以来，人们沉迷于博弈。他们倾向于努力工作、恪尽职守，然后在挑战和发展自身智力的同时找到时间放松、参与博弈。博弈的部分魅力来自于：你可以在不同的层面上进行竞争；你可以测试自己的知识和能力，并且可以及时得到结果；你可以分析为什么特定的结果会发生（胜利、平局或失败)，并从错误中学习，进行另一场博弈。

机器博弈的第一个里程碑成果是 1997 年 IBM 深蓝战胜世界棋王卡斯帕罗夫。当谷歌的 AlphaGo 战胜了围棋世界冠军李世石，AlphaGo 的升级版 Master 横扫了包括中国在内的 60 位世界顶尖高手，这也是机器博弈的另一个重要里程碑。

机器博弈的产业化前景也是很可观的。AlphaGo 的成功，标志着人工智能进入了新的阶段，深度学习算法得以在各个领域的广泛重视和应用。丰富多彩的博弈搜索算法无疑可以应用到面对决策优化的各种场合。随着不完全信息博弈、随机环境博弈搜索算法的不断完善，也将在兵棋推演和战略、战役和战术博弈中加以应用。博弈是人类经济、政治、军事、反恐、治霾和日常生活中无所不在的内容，机器博弈的概念和技术也必然大有用武之地。

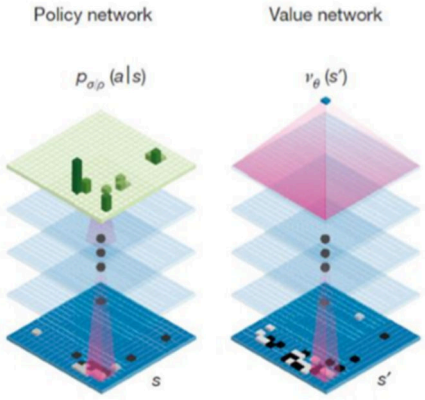

图 6.2 AlphaGo 的策略网络和价值网络

6.2 机器博弈的主要技术

计算机博弈系统中，典型的关键技术主要包括搜索、评估与优化、学习与训练等技术，它们是决定博弈结果的关键因素。以中国象棋、国际跳棋、亚马逊棋等为代表的传统二人零和完备信息博弈，其搜索理论已经很成熟。典型的博弈搜索算法，从搜索方向考虑，可以分为深度优先搜索和宽度优先搜索；从控制策略考虑，可以分为盲目搜索和启发搜索；从搜索范围考虑，可以分为穷尽搜索、裁剪搜索。

近几年来，随着计算机硬件和神经网络、机器学习、大数据等技术的发展，特别是 GPU 并行计算技术的广泛应用，使得深度学习变得更加便宜、快速、实用与有效，机器博弈系统的逻辑思维与计算能力也得到大幅提升。

- 裁剪搜索

裁剪算法也称剪枝算法，是计算机博弈中最常用的主流算法，它包括深度优先的 Alpha-Beta 剪枝搜索[56]和以此为基础改进与增强的算法，如渴望窗口搜索（Aspiration search）、MTD(f) （Memory-enhanced Test Driver with f and n）搜索等。在具体应用中，合理地交叉使用各种搜索方法，可以具有更高的效率。

1. Alpha-Beta 剪枝 Alpha-Beta 剪枝是在极大极小算法基础上的改进算法，是其它剪 枝算法的基础。目前，多数博弈程序都采用负极大值形式的 Alpha-Beta 搜索算法。为保证 Alpha-Beta 搜索算法的效率，需要调整树的结构，即对搜索节点排序，确保尽早剪枝。

2. 渴望搜索是在 Alpha-Beta 搜索算法基础上，缩小搜索范围的改进算法。渴望搜索从一开始就使用小的窗口，从而在搜索之初，就可以进行大量的剪枝。通常，渴望搜索与遍历深化技术结合使用，以提高搜索性能。

3. MTD(f)搜索[58] MTD(f)算法实际上就是不断应用零窗口的 Alpha-Beta 搜索，缩小上界和下界，并移动初始值使其接近最优着法。MTD(f)算法简单高效，在国际象棋、国际跳棋等博弈程序里，MTD(f)算法平均表现出色。

此外，还有各种在 Alpha-Beta 搜索基础上优化的算法，例如，有学者提出在博弈树同层结点中，用广度优先搜索，接力式空窗探测，平均搜索效率高于 MTD (f)搜索。通常，裁剪算法需要与置换表技术相结合，以减少博弈树的规模，提高搜索效率。

- 随机搜索算法

随机搜索有两种算法：拉斯维加斯算法和蒙特卡罗算法。采样越多，前者越有机会找到最优解，后者则越接近最优解。通常，要根据问题的约束条件来确定随机算法，如果对采样没有限制，但必须给出最优解，则采用拉斯维加斯算法。反之，如果要求在有限采样内求解，但不要求是最优解，则采用蒙特卡罗算法。在计算机博弈中，每步走法的运算时间、堆栈空间都是有限的，且仅要求局部优解，所以适合采用蒙特卡罗算法。

蒙特卡洛树搜索(MCTS，Monte Carlo Tree Search)在人工智能的问题中，蒙特卡洛树搜索是一种最优决

策方法，它结合了随机模拟的一般性和树搜索的准确性。由于海量搜索空间、评估棋局和落子行为的难度，围棋长期以来被视为人工智能领域最具挑战的经典游戏。近年来，MCTS 在类似计算机围棋等完备信息博弈、多人博弈以及其它随机类博弈难题上的成功应用而受到快速关注。理论上，MCT 可以被用在以 {状态，行动} 定义并用模拟预测输出结果的任何领域。

基本的 MCTS 算法根据模拟的输出结果，按照节点构造博弈树，其过程如图 6.3 所示，包括路径选择（Selection）、节点扩展 （Expansion）、模拟实验（Simulation）、反向传播（Backpropagation） 四个步骤。

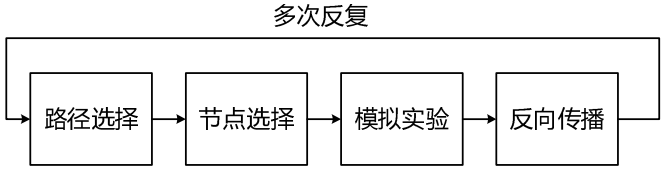

图 6.3 构造 MCTS 博弈树的过程

MCTS 算法适用于非完备信息博弈，也适用于有较大分支因子的博弈程序，例如，AlphaGo 就是采用 MCTS 算法进行搜索。

- 遗传算法

遗传算法是人工智能领域的关键技术，它是一种非数值、并行、 随机优化、搜索启发式的算法，通过模拟自然进化过程随机化搜索最优解。它采用概率化的寻优方法，能自动获取和指导优化的搜索空间， 自适应地调整搜索方向、不需要确定的规则，同时具有内在的隐并行 性和更好的全局寻优能力[90]。

遗传算法是解决搜索问题的一种通用算法，在计算机博弈中，遗 传算法通常被用于搜索、自适应调整和优化局面评估参数。它的基本思想是将博弈树看作遗传操作的种群，博弈树中由根节点到叶子节点 组成的所有子树为种群中的个体。根据优化目标设计评估函数，计算种群中每个个体的适应度函数值，依据适应度函数值的大小确定初始种群，让适应性强（适应度函数值大）的个体获得较多的交叉、遗传机会，生成新的子代个体，通过反复迭代，可得到满意解。

采用遗传算法优化局面估值时，可根据博弈程序与其他程序对弈 的结果，检验某一组参数获胜的几率。经过多次试验，通常可以找到 较好的估值参数。传统的算法一般只能维护一组最优解，遗传算法可以同时维护多组最优解。在实践中，遗传算法被引入了中国象棋、国际象棋、亚马逊棋以及禅宗花园游戏等博弈系统的智能搜索与评估优化中，效果还是很明显的。

- 神经网络

人工神经网络（Artificial Neural Network，即 ANN ），简称为 神经网络或类神经网络。它是一种运算模型，由大量的节点（或称神 经元）之间相互联接构成。每个节点代表一种特定的输出——激励函 数（Activation function）。每两个节点间的连接都代表一个对于通过 该连接信号的加权值，这相当于人工神经网络的记忆。网络的输出则依网络的连接方式，依权值和激励函数的不同而不同。而网络自身通常都是对自然界某种算法或者函数的逼近，也可能是对一种逻辑策略的表达。

近年来，人工神经网络的研究取得了很大的进展，尤其是实现了以超算为目标的并行算法的运行与概念证明后，在机器博弈、计算机 视觉、模式识别等人工智能领域与深度学习相结合，成功地解决 了许多现代计算机难以解决的实际问题（例如围棋、中国象棋博弈中 的估值、学习与训练等），表现出了良好的智能特性。

- 机器学习

与传统为解决特定任务、硬编码的软件程序不同，机器学习是用 大量数据进行训练，使用各种算法来解析数据并从中学习，做出决策 和预测。当前主流机器学习技术包括度量学习、多核学习、多视图学 习、集成学习、主动学习、强化学习、迁移学习、统计关系学习、演 化学习、并行机器学习、哈希学习等，其中强化学习（Reinforcement Learning，也称为增强学习）被列为机器学习的四大研究方向之一。

强化学习研究学习器在与环境的交互过程中，如何学习到一种行为策略，以得到累积利益最大化。在

机器博弈中，强化学习的设定可用图 6.4 来表示，学习器所处的环境为博弈规则，学习器根据当 前博弈状态输出着法，以博弈收益作为每步着法的结果，反馈给学习 器，以期望最终的利益最大化。

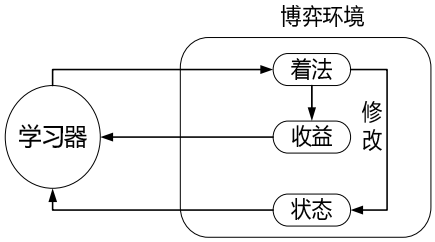

图 6.4 机器博弈强化学习设定

在实际应用中，由于强化学习的优化目标通常涉及多步决策，相 对复杂，且策略的搜索空间巨大，优化比较困难。另外，强化学习还 面临着特征表示、泛化能力等诸多挑战。

深度学习是基于多层网络结构的特征学习方法，把原始数据通过多层神经网络非线性变换，逐层提取抽象特征，完成复杂的目标函数系统逼近。深度学习典型的网络模型包括卷积神经网络、深层玻尔兹曼机和堆叠自动编码器等。利用 GPU 来训练深度神经网络，充分发挥其并行计算能力，大幅缩短海量数据训练所耗费的时间，因此 GPU 并行计算已经成为业界在深度学习模型训练方面的首选解决方案。

相对于传统的机器学习方法，深度学习能够学习多层次抽象的数据表示，能够发现大数据中的复杂结构，对于解决强化学习中策略评估和优化的问题有明显优势。深度学习被成功地用于机器博弈中， 例如采用基于深度学习和 Q-Learning 的 Deep Q-Network 技术的博弈系统已达到人类玩家水平，而 AlphaGo 则可以战胜人类顶级高手。

6.3 机器博弈网络

- 生成对抗网络

生成式对抗网络，英文名称为 Generative Adversarial Nets，2014 年 6 月 Ian Goodfellow 等学者提出，它是一种生成模型，核心思想是从训练样本中学习所对应的概率分布，以期根据概率分布函数获取更多的“生成"样本来实现数据的扩张。

生成式对抗网络的启发主要源于博弈论中的二人零和博弈，即指参与博弈的双方，在严格竞争下，一方的收益必然意味着另一方的损失，博弈双方的收益和损失相加总和永远为“零”，双方不存在合作。对于非合作、纯竞争型博弈，例如两个人打乒乓球，一个人赢则意味着另一人输；抽象后的博弈问题为：已知参与者集合(双方)，策略集合(乒乓球技术水平)和盈利集合(胜负)，能否找到一个理论上的平衡点，即对参与双方来说都最合理、最优的具体策略？冯·诺依曼已经从数学上证明，对二人零和博弈问题，可以通过一定的线性运算操作(即竞争双方以概率分布的形式随机使用某类最优策略中的各个策略)，找到一个最小最大的平衡点，这个著名的最小最大定理的思想是抱最好的希望，做最坏的打算，生成对抗网络的训练过程如图 6.5 所示。

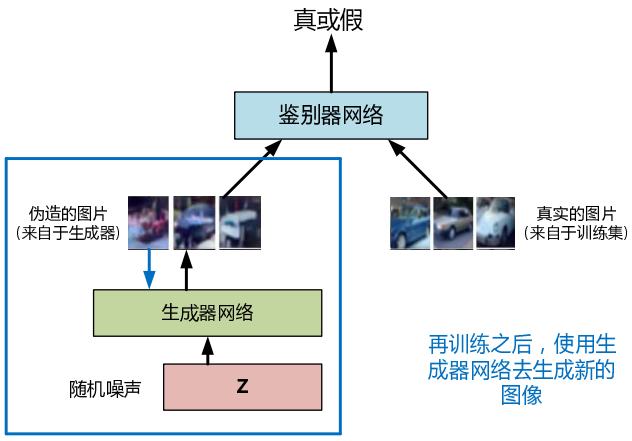

图 6.5 生成对抗网络的训练过程

- 强化学习网络

强化学习是学习如何将当前映射到行动决策上，并最大化数字奖励信号。学习者并没有被告知应该采取哪些行动，而是必须通过尝试来发现哪些行动会带来最大的回报。在一般情况下，行为可能不仅影响眼前的奖励，还会影响下一种情况，并通过这种情况影响所有后续的奖励。试错搜索和延迟奖励是强化学习的两个最重要的特征。

强化学习不同于目前机器学习领域研究的监督学习。监督学习是从一个现有的知识库中提供的一组有标签的示例的训练集进行学习。这种监督学习的目的是让系统推断或总结它的知识，以便在测试集中能够预测正确。这是一种重要的学习，但仅从交互中学习是不够的。在交互问题中，要获得既正确又能代表智能体采取行动的所有情况的期望行为往往是不切实际的。所以人们认为学习是最正确的方法，即 Agent 必须能够从自己的历史经验中学习，强化学习的网络结构如图 6.6 所示。

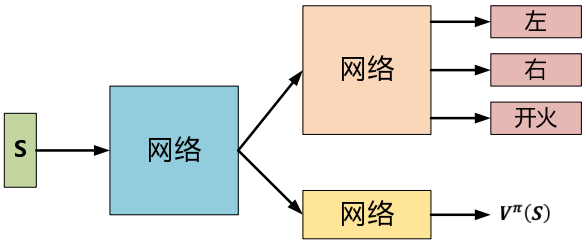

图 6.6 Advantage Actor-Critic 强化学习网络结构

6.4 现存的问题与展望

在过去的几十年里，尽管机器博弈研究成果对于推动人工智能的发展具有重要意义，但在计算机博弈领域仍存在不同程度的局限性。 具体表现为：

（1）在学术研究方面，尽管深度学习等技术在围棋方面取得了前所未有的成功，但在其应用拓展方面，仍有许多值得研究和探索之处。另外，对于具有模糊性和随机性的麻将、桥牌、斗地主、多国军 旗等非完备信息博弈，虽然在基于案例的策略研究方面有了一定进 展，但因其相关研究还不成熟，开发的程序智力有限，目前仍难以战胜人类顶级高手，尚有很大的提升空间。

（2）在相关技术产业化方面，产学研结合仍有不足。表现为：一方面，相关企业缺乏机器博弈领域的专业人才，特别是缺少机器博弈领域顶级专家的技术支持；另一方面，机器博弈领域专家、学者们缺少相关部门、企业给予的研发资金支持。

从学术研究和相关技术产业化来看，我国对机器博弈技术的研发 与应用相对于国外存在较大的差距。开发具有高效自主学习能力与抽象思维能力的智能博弈系统，特别是在非完备信息和不确定性机器博 弈方面，还有很长的路要走。此外，只有将机器博弈技术作为战略核 心予以关注，不断加大投入，在未来竞争中，我国才能处于不败之地。

国务院近期印发的《新一代人工智能发展规划》给我国人工智能 发展指明了道路，为机器博弈发展注入新的活力并带来更多机遇，规划中明确提出：开展综合深度推理与创意人工智能理论与方法、非完全信息下智能决策基础理论与框架、数据驱动的通用人工智能数学模 型与理论等研究；支持开展人工智能竞赛。相信在国家层面政策的支持下，我国计算机博弈领域的研究与应用将进入快速发展的新阶段。未来计算机博弈将呈现多学科技术融合、人机协同、产学研相结合等趋势。具体体现在：

（1）计算机博弈研究的内容将不断拓宽，处理的问题复杂程度越来越高，信息量将越来越大。为解决某类特定问题，技术方法将集成化，计算机博弈技术将与并行计算、大数据、知识工程等相关技术紧密结合。

（2）计算机博弈软件与硬件的结合越来越密切，固化博弈系统 的智能硬件产品将越来越多的出现在人们的生活中，典型的应用包括：有博弈思维能力的机器人、智能决策控制系统的无人驾驶汽车和 无人机等。

（3）计算机博弈将融入各个领域的应用中，在此基础上可以开 展一系列人工智能领域的科学研究。计算机博弈越来越注重实际工程 应用，紧密地结合经济、医疗、航空航天等领域，解决实际问题。特别在航空航天领域的多学科协同综合设计、虚拟现实仿真及人机交互 智能游戏与教育方面，拥有广阔的应用前景。

（4）计算机博弈技术将呈现高度智能化趋势，通过与遗传算法、 人工神经网络、类脑思维等人工智能技术进一步融合，类似基于神经 网络深度学习的智能技术将大量涌现，使得计算机博弈程序的类脑智能越来越高。

（5）合理拓展现有博弈技术，深入研究更加智能的普适算法， 构建一个通用计算机博弈系统，将成为未来计算机博弈研究的重点。

（6）作为计算机博弈技术交流与验证的平台，中国计算机博弈比赛将越来越被社会所认同。各种新技术将会被越来越多地运用到计算机博弈中。

（7）学术界与产业界的结合日趋紧密，计算机博弈研究学术成果加速向产业化转变，助力游戏开发、智能医疗、航空航天企业，促进计算机游戏、智能医疗、航空航天、国防等相关产业发展。 可以预见，在计算机博弈领域越来越多的人机博弈项目中，人类终将被战胜。机器智能的胜利，既是人类创造力与智慧的结晶，也是科学发展的必然，同样也是人类最终的胜利。

参考文献

Lecun， Y. ， Bengio， Y. ， & Hinton， G. . (2015). Deep learning. Nature， 521(7553)， 436.

Han， S. ， Mao， H. ， & Dally， W. J. . (2016). Deep Compression: Compressing Deep Neural Networks with Pruning， Trained Quantization and Huffman Coding. ICLR.

Peng， X. B. ， Berseth， G. ， & Panne， M. V. D. . (2016). Terrain-adaptive locomotion skills using deep reinforcement learning. Acm Transactions on Graphics， 35(4)， 81.

Wu， J. ， Zhang， C. ， Xue， T. ， Freeman， W. T. ， & Tenenbaum， J. B. . (2016). Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling.

Mirza， M.， & Osindero， S. (2014). Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784.

Goodfellow， I. J. ， Pouget-Abadie， J. ， Mirza， M. ， Xu， B. ， Warde-Farley， D. ， & Ozair， S. ， et al. (2014). Generative adversarial networks. Advances in Neural Information Processing Systems， 3， 2672-2680.

Gauthier， J. (2014). Conditional generative adversarial nets for convolutional face generation. Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition， Winter semester， 2014(5)， 2.

Radford， A.， Metz， L.， & Chintala， S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434.

Denton， E. L.， Chintala， S.， & Fergus， R. (2015). Deep generative image models using a laplacian pyramid of adversarial networks. In Advances in neural information processing systems (pp. 1486-1494).

Liu， M. Y.， & Tuzel， O. (2016). Coupled generative adversarial networks. In Advances in neural information processing systems (pp. 469-477).

Chen， M.， & Denoyer， L. (2017， September). Multi-view generative adversarial networks. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases (pp. 175-188). Springer， Cham.

Arjovsky， M.， & Bottou， L. (2017). Towards principled methods for training generative adversarial networks. arXiv preprint arXiv:1701.04862.

Schuster， M.， & Paliwal， K. K. (1997). Bidirectional recurrent neural networks. IEEE transactions on Signal Processing， 45(11)， 2673-2681.

Graves， A. (2012). Supervised sequence labelling. In Supervised sequence labelling with recurrent neural networks (pp. 5-13). Springer， Berlin， Heidelberg.

Graves， A.， Mohamed， A. R.， & Hinton， G. (2013， May). Speech recognition with deep recurrent neural networks. In 2013 IEEE international conference on acoustics， speech and signal processing (pp. 6645-6649). IEEE.

Krueger， D.， & Memisevic， R. (2015). Regularizing rnns by stabilizing activations. arXiv preprint arXiv:1511.08400.

Ji， S.， Vishwanathan， S. V. N.， Satish， N.， Anderson， M. J.， & Dubey， P. (2015). Blackout: Speeding up recurrent neural network language models with very large vocabularies. arXiv preprint arXiv:1511.06909.

Choi， E.， Bahadori， M. T.， Schuetz， A.， Stewart， W. F.， & Sun， J. (2016， December). Doctor ai: Predicting clinical events via recurrent neural networks. In Machine Learning for Healthcare Conference (pp. 301-318).

Talathi， S. S.， & Vartak， A. (2015). Improving performance of recurrent neural network with relu nonlinearity. arXiv preprint arXiv:1511.03771.

Karpathy， A.， Johnson， J.， & Fei-Fei， L. (2015). Visualizing and understanding recurrent networks. arXiv preprint arXiv:1506.02078.

Bashivan，P.，Rish，I.，Yeasin，M.，& Codella，N. (2015). Learning representations from EEG with deep recurrent-convolutional neural networks. arXiv preprint arXiv:1511.06448.

Bell， S.， Lawrence Zitnick， C.， Bala， K.， & Girshick， R. (2016). Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2874-2883).

Collins J， Sohl-Dickstein J， Sussillo D. Capacity and trainability in recurrent neural networks[J]. arXiv preprint arXiv:1611.09913，2016.

He， K.， Zhang， X.， Ren， S.， & Sun， J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

Jou， B.， & Chang， S. F. (2016， October). Deep cross residual learning for multitask visual recognition. In Proceedings of the 24th ACM international conference on Multimedia (pp. 998-1007).

He， K.， Zhang， X.， Ren， S.， & Sun， J. (2016， October). Identity mappings in deep residual networks. In European conference on computer vision (pp. 630-645). Springer， Cham.

Ishikawa， Y.， Washiya， K.， Aoki， K.， & Nagahashi， H. (2016， December). Brain tumor classification of microscopy images using deep residual learning. In SPIE BioPhotonics Australasia (Vol. 10013， p. 100132Y). International Society for Optics and Photonics.

Han， Y. S.， Yoo， J.， & Ye， J. C. (2016). Deep residual learning for compressed sensing CT reconstruction via persistent homology analysis. arXiv preprint arXiv:1611.06391.

Bae， W.， Yoo， J.， & Chul Ye， J. (2017). Beyond deep residual learning for image restoration: Persistent homology-guided manifold simplification. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops (pp. 145-153).

Yang， W.， Feng， J.， Yang， J.， Zhao， F.， Liu， J.， Guo， Z.， & Yan， S. (2017). Deep edge guided recurrent residual learning for image super-resolution. IEEE Transactions on Image Processing， 26(12)， 5895-5907.

Shah， A.， Kadam， E.， Shah， H.， Shinde， S.， & Shingade， S. (2016， September). Deep residual networks with exponential linear unit. In Proceedings of the Third International Symposium on Computer Vision and the Internet (pp. 59-65).

Boyd， S.， Parikh， N.， & Chu， E. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. Now Publishers Inc.

Mao, X. , Li, Q. , Xie, H. , Lau, R. Y. K. , Wang, Z. , & Smolley, S. P. . (2016). Least squares generative adversarial networks.

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein gan. arXiv preprint arXiv:1701.07875.