# 第1章习题

1．1956年达特茅斯会议上，学者们首次提出“Artificial Intelligence”（人工智能）这个概念时，所确定的人工智能研究方向不包括（ ）。

A. 研究人类大脑结构和智能起源

B. 研究如何用计算机表示人类知识

C. 研究如何用计算机来模拟人类智能

D. 研究智能学习的机制

* 1. 在现阶段，下列哪项尚未成为人工智能研究的主要方向和目标？（ ）

A．研究如何用计算机模拟人类智能的若干功能，如会听、会看、会说

* + 1. 研究如何用计算机延伸和扩展人类智能

C．研究机器智能与人类智能的本质差别

D．研究如何用计算机模拟人类大脑的网络结构和部分功能

3．下列关于人工智能的说法中不正确的是（ ）。

A. 人工智能是关于知识的学科--怎样表示知识以及怎样获得知识并使用知识的学科

B. 人工智能研究如何使计算机去做过去只有人才能做的智能工作

C. 自1956年人工智能学科诞生以来，经过多年的发展，它的理论已趋于成熟，得到了充分应用。

D. 人工智能不是人的智能，但能像人那样思考。甚至也可能超过人的智能

4．强人工智能强调人工智能的完整性。以下哪个不属于强人工智能？（ ）

A. 类人机器的思考和推理就像人的思维一样

B. 非类人机器产生了和人完全不一样的知觉和意识

C. 智能机器看起来像是智能的，其实并不真正拥有智能，也不会有自主意识

D. 有可能制造出真正能推理和解决问题的智能机器

5．从人工智能研究流派来看，西蒙和纽厄尔提出的“逻辑理论家”方法，应该属于（ ）。

A. 行为主义 B. 符号主义 C. 连接主义 D. 理性主义

6. 从人工智能研究流派来看，麦卡锡提出的“状态空间搜索法”，应该属于（ ）。

A. 行为主义 B. 符号主义 C. 连接主义 D. 理性主义

7. 下列关于人工智能未来发展趋势的描述，哪些是错误的？（ ）

A. 人工智能目前仅适用于特定的、专用的问题

B. 人工智能受到越来越多的关注，许多国家出台了支持人工智能发展的战略计划

C. 通用人工智能的发展正处于起步阶段

D.人工智能将脱离人类控制，并最终毁灭人类

8. 考察人工智能的一些应用，目前（ ）任务可以通过人工智能来解决。

A．在市场上购买一周的食品杂货

B. 以竞技水平玩德州扑克游戏

C. 在Web上购买一周的食品杂货

D. 将英文口语实时翻译为中文口语

9. （ ）的本质就是用符号体系来描述知识，再对所得到的符号表示进行计算，从而求解问题。

A.行为主义 B.符号主义 **C.**连接主义 **D.**主观主义

10. 以下说法中，错误的是（ ）。

**A.** 智能，是“智慧”和“能力”的综合表现

**B.** 人工智能学科诞生于1956年的达特茅斯会议

**C.** 目前的人工智能研究是在有限条件下对人类智能的某种能力在某些具体问题上的模拟

**D.** 人工智能最开始的定义是研究或制造人类的智能

11. 下列（ ）属于人工智能的研究领域。

**A.** 图像理解 **B.** 人脸识别 **C.**专家系统 **D.**机器学习 E. 以上都是

1. 在达特茅斯会议上，提出了人工智能研究的主要方法，包括（ ）。
2. 西蒙和纽维尔的逻辑理论家
3. 麦卡锡的状态空间搜索法
4. 明斯基的人工神经网络
5. 以上三种

答案：

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 题号 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 答案 | A | C | C | C | B | B | D | BCD | B | D | E | D |

# 第2章习题

1. 什么是知识? 它有哪些特性? 有哪几类知识表示方法?

答：知识是人类在长期的生活、社会实践及科学实验中经过总结、提升与凝练的对客观世界（包括人类自身）的认识和经验，也包括对事实和信息的描述或在教育和实践中获得的技能。

知识具有相对正确性和不确定性。

知识表示方法分为两类：陈述式知识表示法和过程式知识表示法。

1. 用产生式表示：如果一种微生物的染色斑是革兰氏阴性，其形状呈杆状，病人是中间宿主，那么该生物是绿杆菌的可能性有6成。

答：IF 本微生物的染色斑是革兰氏阴性∧本微生物的形状呈杆状∧病人是中间宿主

THEN 该生物是绿杆菌(0.6)

3.请用状态空间表示法描述下列问题：有5个传教士和5个野人来到河边，准备从左岸渡河到右岸，河岸有一条船,每次至多可供3人乘渡，所有人都会划船。为安全起见，在河的两岸以及船上的野人数目总是不可以超过传教士的数目（但允许在河的某一岸只有野人而没有传教士）。

答：（1）设计**状态空间**表示，令

* 未过河的修道士数量为 m
* 未过河的野人数量为 c，
* 未过河的船的数量为 b（=1表示船未过河，=0表示船已过河）。
* 则状态空间可以用向量：S = { m , c , b } 表示。
* 理论上，过河的所有状态数量为：6\*6\*2=72个
* **初始状态**为(5,5,1) ，**目标状态**为(0,0,0)。

（2）设计**状态转移**形式

* 状态转移即过河操作。
* 用P 代表船从左岸到右岸；用Q 代表船从右岸到左岸。
* 船上：修道士人数为 i， 野人人数为 j ，满足:
  1. 1≦ i + j ≦ 3; 2) i≠0时， i≧ j

根据以上限制，可能的操作共有16种：

* 船从左岸到右岸：P01; P02; P03; P10; P11; P20; P21; P30;（不能有P12）
* 船从右岸到左岸：Q01; Q02; Q03; Q10; Q11; Q20; Q21; Q30;

4. 使用 RDF 三元组表示事实知识：葡萄牙籍的C罗效力于西班牙皇家马德里足球队。

答：（皇家马德里足球队，属于，西班牙）

（C罗，效力，皇家马德里足球队）

（C罗，国籍，葡萄牙）

* + 1. 产生式系统哪几部分组成?分别简述各部分的用途，其中知识库是指什么？

答：一个产生式系统由规则库、综合数据库、控制系统3部分组成。

（1）规则库：用于描述相应领域内知识的产生式集合称为规则库。（1分）

（2）综合数据库又称为事实库，它是用于存放问题求解过程中各种当前信息(例如问题的初始状态、原始证据、推理中得到的中间结论及最终结论)的数据结构。

（3）控制系统又称为推理机，包括控制程序和推理程序两部分，负责整个产生式系统的运行，实现对问题的求解。

知识库是是指由综合数据库和规则库共同组成的数据库。

# 第3章习题

1 名词解释

搜索策略、盲目搜索策略、启发式搜索策略、启发信息、最优解、算法的完备性、算法的最优性

**答：**略

2盲目搜索策略和启发式搜索策略的区别是什么？有哪几种常用的盲目搜索算法? 有哪几种常用的启发式搜索算法?

**答：**

盲目搜索策略和启发式搜索策略的区别是：盲目搜索策略是在不具备给定问题的特定知识的情况下，系统按照某种固定的规则，依次或随机地调用操作算子；启发式搜索策略是考虑给定问题领域的可应用知识，动态地确定调用操作算子的顺序，优先选择当前最合适的操作算子，利用启发式信息来进行搜索，尽量减少试探的次数,提高搜索效率。

常用的盲目搜索策略有：深度优先搜索和宽度优先搜索。

常用的启发式搜索策略有：A搜索和A\*搜索。

3 深度优先搜索与宽度优先搜索的区别是什么？分析两个算法的优缺点。

**答：**深度优先搜索和宽度优先搜索都是常用的盲目搜索算法，其区别是：节点在OPEN表中的排列顺序是不同的，即扩展节点的顺序不同。

深度优先搜索总是选择深度最深的节点进行扩展，若有多个相同深度的节点，则按照指定的规则从中选择一个。若该节点没有子节点，则选择一个其他深度最深的节点进行扩展。依次类推，直到找到问题的解为止；或直到找不到可扩展的节点，结束。

宽度优先搜索总是优先扩展深度最浅的节点。先扩展根节点，再扩展根节点的所有后继，然后再扩展它们的后继，以此类推。如果有多个节点深度是相同的，则按照事先约定的规则，从深度最浅的几个节点中选择一个。

优缺点：

（1）深度优先搜索算法的**优点**：比宽度优先搜索节省大量时间和空间。**缺点：**深度优先搜索不具有最优性，且不一定完备。对于状态空间有限的问题，深度优先搜索是完备的；但在无限状态空间中，若沿着一个“错误”的路径搜索下去而陷入“深渊”，则会导致无法到达目标节点，深度优先搜索是不完备的。

（2）宽度优先搜索算法的**优点**：是完备的。若路径代价是节点深度的非递减函数，或者每步代价都相等，宽度优先搜索还具有最优性。**缺点：**需占用较大的存储空间。

4 A搜索与A\*搜索的区别是什么?

**答：**A算法对启发函数未做出任何限制，A搜索是不完备，也不具有最优性。

而A\*搜索算法则对启发函数h(n)加以限制，要求h(n)满足如下条件：h(n) ≤ h\*(n)。A\*算法是完备的，且具有最优解。

5 请用 A\* 算法求解图3.18中的八数码问题：

（1）说明所采用的启发函数，并证明其满足A\*算法的条件；

（2）给出移动数码牌的过程，并在每一步都标出此状态的*f*、 *g*、*h*值。

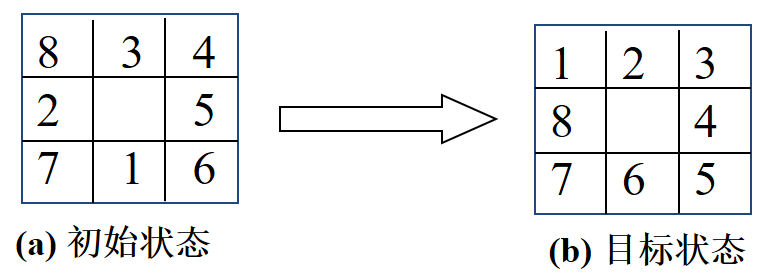


图3.18八数码问题

**答：**

令启发函数h=错位将牌数

证明其满足A\*算法的条件：

* 令d(n) =已移动将牌的步数，即结点n在搜索树中的深度

w(n)=结点n所表示的状态中“错位”的将牌数

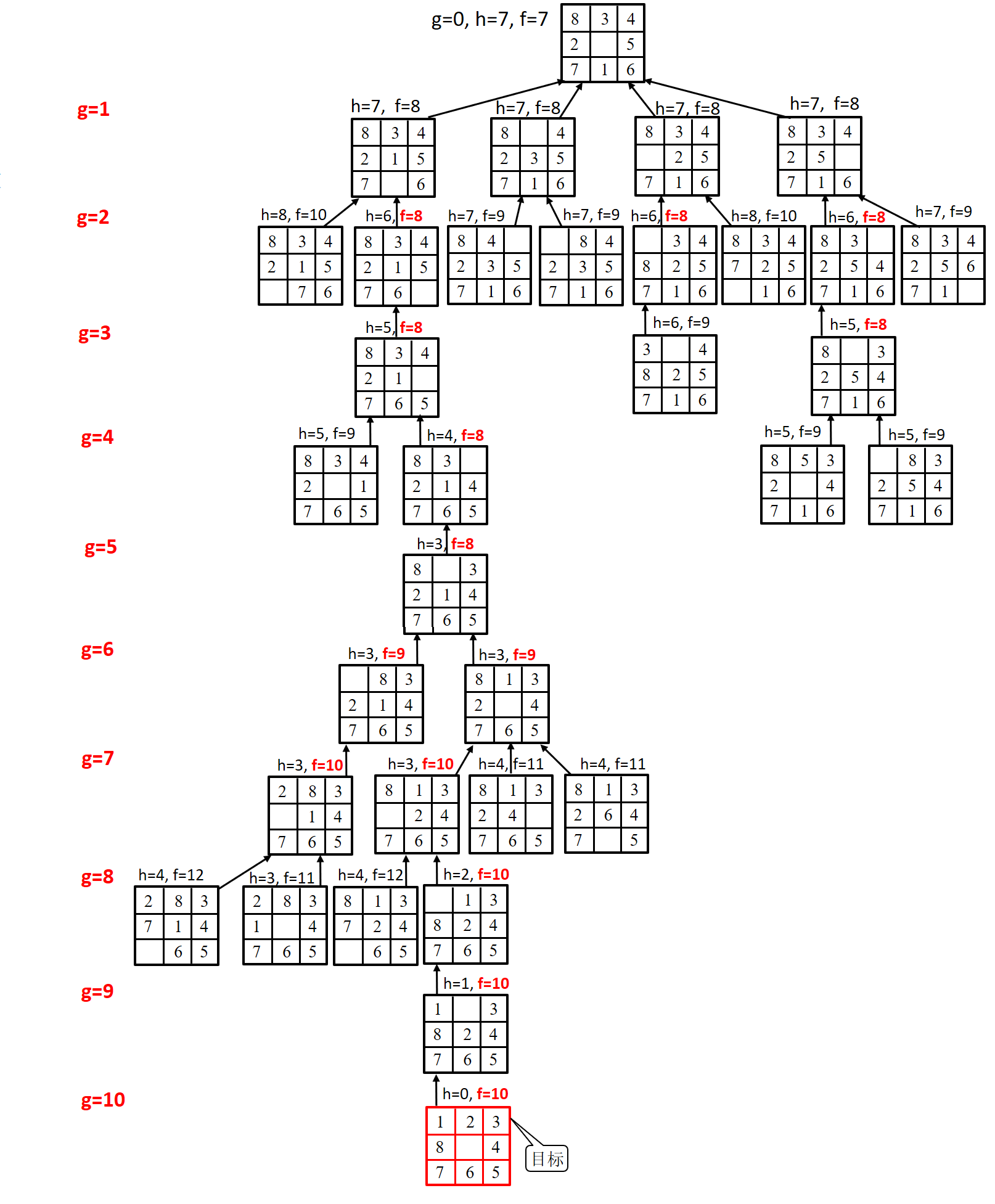
* 将w(n)个**“不在位”的将牌放在其各自的目标位置上，至少需要移动**w(n)步。
* **h\*(*n*)** 是结点n从当前位置移动到目标结点的最少实际步数，

显然 w(n) ≤ h\*(n).

* 以w(*n*)作为启发式函数h(*n*)，可以满足对h(*n*) 的下界的要求，

即有 **h(n)= w(n) ≤ h\*(n).**

因此，当选择w(*n*)作为启发式函数解决8数码问题时，A算法就是A\*算法。



6. 针对同一待求解问题，我们可以定义不同的启发函数。例如，在八数码问题中，还可以选择启发函数为：

*h*2(*n*)= 所有错位数码牌与其目标位置的距离之和

此处的距离采用曼哈顿距离，曼哈顿距离就是计算两点之间横纵坐标的距离之和。在此定义下，图3.6所示初始状态中*h* = 5，则*f* = 5。读者可以自己证明：采用*h*2(*n*)作为启发函数，仍满足A\*算法的要求，并用它解决八数码问题。

**答：** *h*2(*n*)= 所有错位数码牌与其目标位置的距离之和

证明其满足A\*算法的条件：

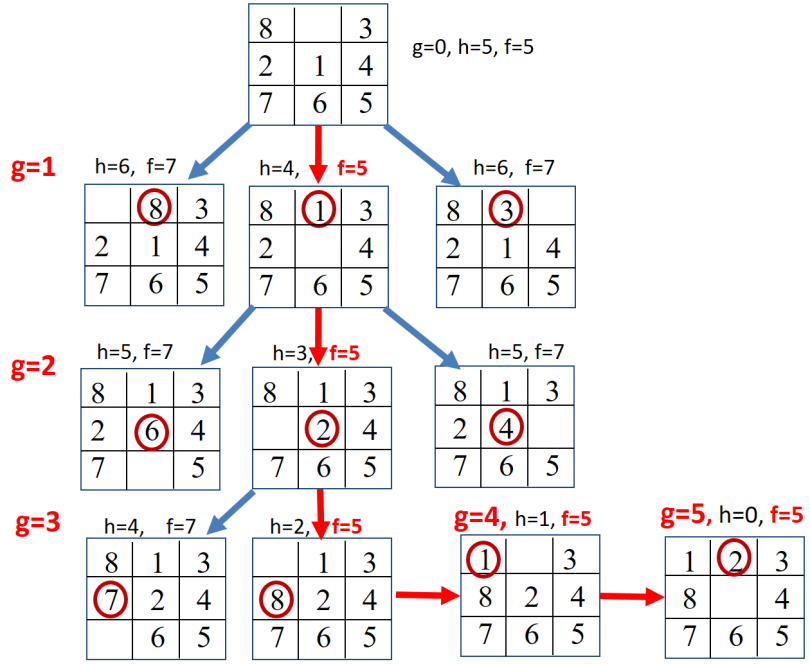
* 令**d(*n*) =已移动将牌的步数，即结点n在搜索树中的深度**

**w(*n*)=结点n所表示的状态中“错位”的将牌数**

M**(*n*) =结点n所表示的状态中错位**将牌与其目标位置之间的曼哈顿距离之和

* 将w(n)个**“错位”的将牌放在其各自的目标位置上，至少需要移动M**(n)步。
* **h\*(*n*)** 是结点n从当前位置移动到目标结点的最少实际步数，显然 M(n) ≤ h\*(n).
* 以M(*n*)作为启发式函数h2(*n*)，可以满足对h2(*n*) 的下界的要求，即有 **h**2**(n)= M(n) ≤ h\*(n).**

因此，当选择M(*n*)作为启发式函数解决8数码问题时，A算法就是A\*算法。



7. 爬山法的基本思路和主要特点是什么?有哪些变种的爬山法？其特点是什么？

**答：**爬山法的基本思路：在每个状态，都是不断地向启发函数值增加最快的方向持续移动，即登高。

爬山法的主要特点：不完备、不具有最优性。

变种的爬山法有：随机爬山法、随机重启爬山法。

随机爬山法、随机重启爬山法均不完备，但随机重启爬山法能以逼近1的概率接近完备。

8. 模拟退火法的基本思路和主要特点是什么?

**答：**模拟退火法是一种逼近全局最优解的概率方法，它是允许“下山”的随机爬山法。模拟退火法的**基本思路**是：在退火初期，“下山”（即“变坏”）移动容易被采纳，以便摆脱局部极值；但随着时间的推移，“下山”的次数越来越少，即逐渐减少向“坏”的方向移动的频率。

模拟退火法的特点：可能会让算法跳出局部最优解，最终找到全局最优解。

9. 遗传算法的基本思路和主要特点是什么?

**答：**遗传算法的**基本思路**：通过模仿自然进化过程来搜索复杂问题的最优解。

**特点：**在求解较为复杂的组合优化问题时，遗传算法通常能比一些常规优化算法更快地获得较好的优化结果。在遗传算法中，后继节点是由两个父辈状态组合生成的，而不是对单一状态修改而得到的。其处理过程是有性繁殖，而不是无性繁殖。

10. 适应度函数在遗传算法中的作用是什么?试举例说明如何设计适应度函数。

**答：**适应度函数是遗传算法实现优胜劣汰的主要依据。个体适应度的值越大，说明该个体的状态越好，竞争能力越强，被选择参与遗传操作来产生新个体的可能性就越大，以此体现生物遗传中适者生存的原理。

例如，在八皇后问题中，采用“不相互攻击的皇后对的数目”作为适应度函数。

11. 说明选择、交叉、变异操作的作用是什么。

**答：**（1）选择操作是从当前种群中按照一定概率选出的优良个体，使它们有机会作为父代繁殖下一代。其作用是使种群优胜劣汰、不断进化，并且提高种群的收敛速度和搜索效率。

1. 交叉操作是遗传算法中的核心操作。交叉的作用是：获得比父代更优秀的个体。
2. 变异的作用：保持种群的多样性，对选择、交叉过程中可能丢失的某些遗传基因进行修复和补充。当发生早熟收敛时，变异还能跳出局部最优解的陷阱。变异操作不仅可以保证实现搜索的目标，而且可以提高搜索的效率。

12. 遗传算法中避免局部最优解的关键技术是什么?

**答：** 变异操作。

# 第4章习题

1. 什么是机器学习？人工智能、机器学习、深度学习之间的关系是什么？

**答：**机器学习是研究如何使机器模拟或实现人类的学习行为，以获取知识和技能，并不断改善系统自身性能的学科。

人工智能、机器学习、深度学习之间的关系是人工智能包含机器学习，而机器学习又包含深度学习。

1. 机器学习的任务有哪些？应该分别采用哪类学习范式去解决？

**答：**机器学习中的典型任务包括分类、回归、聚类、排名、降维等。

分类、回归、排名采用监督学习方法解决。

聚类、降维采用无监督学习方法解决。

1. 聚类与分类的区别是什么？有什么相似性？

**答：**聚类与分类的相似性是：均把样本分成若干组或若干簇；

区别是：聚类是无监督学习，分类是有监督学习。聚类是事先不知道多少类，也不知道组的名称。分类是给定输入，将样本分成事先定义好名称的类别。

1. 逻辑回归与线性回归有什么区别？

**答：**区别：线性回归用于预测连续值，其输出的值域是实数集，其模型是线性的；逻辑回归主要用于解决分类问题，其输出的值域为[0,1]，其模型是非线性的。

1. KNN算法和K-Means算法的区别是什么？各有什么局限性？

**答：**区别：KNN用于解决分类或回归问题，属于监督学习；K-means用于解决聚类问题，属于非监督学习。

共同的局限性为：需要确定k值。若k值不合适，两个算法的效果不好。

1. 强化学习与监督学习、无监督学习的区别是什么？

**答：**

（1）强化学习与[监督学习](https://baike.baidu.com/item/%E7%9B%91%E7%9D%A3%E5%AD%A6%E4%B9%A0/9820109?fromModule=lemma_inlink" \t "_blank)的不同之处在于：①监督学习输入的训练数据是包含样本及其标签的强监督信息，而强化学习从外部环境接收的反馈信息是一种弱监督信息；②监督学习是采用正确答案来训练模型，给予的指导是即时的；而强化学习是采用“试错”的方式来训练模型，外部环境给予它的指导有时是延迟的。

（2）强化学习与无[监督学习](https://baike.baidu.com/item/%E7%9B%91%E7%9D%A3%E5%AD%A6%E4%B9%A0/9820109?fromModule=lemma_inlink" \t "_blank)的不同之处在于：无监督学习输入的是没有任何监督信息的无标注数据，而强化学习从环境获得的评价信息（不是正确答案）是一种弱监督信息，尽管监督信息很弱，但总比没有要好。

1. 主动学习与强化学习的区别是什么？

**答：**虽然强化学习和主动学习都可以减少模型所需的标注数据量，但它们是非常不同的概念。

1. 主动学习侧重于主动选择最具信息量的样本进行标注，以提高模型性能。强化学习通过与环境的交互学习，通过观察环境的反馈（奖励或惩罚）来调整行为，以达到最大化长期奖励的目标。
2. 主动学习注重数据选择和标注，需要人类的参与；而强化学习注重决策和行为优化，未必需要人的参与。
3. 主动学习与半监督学习的区别是什么？

**答：**主动学习需要人参与，半监督学习不需要人参与。

1. 分别在什么场景需要采用不完全监督学习、不确切监督学习、不准确监督学习？

**答：**

1. 不完全监督学习适用于在训练数据中只有一小部分样本被标注，而大多数样本是未标注的情况。例如：在图像分类任务中，可能只有少数图像被手工标注，而大量图像没有标注。不完全监督学习可用于有效地使用未标注图像提高分类性能。
2. 不确切监督学习适用于只有粗粒度标签的情况。例如，在图像处理中，可能只有图像级别的标签，而没有像素级别的准确标注。
3. 不准确监督学习适用于标注数据标签可能存在噪声或错误的情况。例如，图像的标签是错的。
4. 举例说明迁移学习可以应用于哪些任务？

**答：**图像分类、机器翻译、生成不同风格的绘画等。

1. 针对表1，试编程实现KNN算法，令*k*=3，预测一个新样本点（3，4）的类别。

表4-1个样本的坐标与类别

|  |  |  |  |
| --- | --- | --- | --- |
| 点的编号 | x坐标 | y坐标 | 类别 |
| 1 | 1 | 1 | C3 |
| 2 | 5 | 3 | C1 |
| 3 | 4 | 4 | C2 |
| 4 | 3 | 5 | C2 |
| 5 | 4 | 5 | C2 |
| 6 | 4 | 7 | C1 |
| 7 | 3 | 2 | C1 |
| 8 | 2 | 1 | C3 |
| 9 | 0 | 0 | C3 |
| 10 | 6 | 4 | C1 |

**答：**

1. 将新样本点(3,4)记为T，分别计算T与10个已知样本P的欧氏距离平方：

|T-P1|2 = (3-1)2 + (4-1)2 = 13； |T-P2|2 = (3-5)2 + (4-3)2 = 5；

|T-P3|2 = (3-4)2 + (4-4)2 = 1； |T-P4|2 = (3-3)2 + (4-5)2 = 1；

|T-P5|2 = (3-4)2 + (4-5)2 = 2； |T-P6|2 = (3-4)2 + (4-7)2 = 10；

|T-P7|2 = (3-3)2 + (4-2)2 = 4； |T-P8|2 = (3-2)2 + (4-1)2 = 10；

|T-P3|2 = (3-0)2 + (4-0)2 = 25； |T-P4|2 = (3-6)2 + (4-4)2 = 9；

1. 找出其中距离T最小的3个样本，分别是P3，P4，P5，其中类别为C2的最多。

给T标记为C2类别。

# 第5章习题

* + - 1. 请简述Sigmoid函数的优缺点。

**答：**

**sigmoid函数的优点：**

平滑、易于求导，其导数可直接用函数的输出计算，简单高效。sigmoid函数很好地解释了神经元在受到刺激的情况下是否被激活和向后传递的情景。当取值接近0时，几乎没有被激活；当取值接近1时，几乎完全被激活。

**sigmoid函数的缺点：**

1. 当输入的绝对值大于某个阈值时，会快速进入饱和状态（即函数值趋于1或 1，不再有显著的变化，梯度趋于0），会出现梯度消失的情况，权重无法再更新，会导致算法收敛缓慢，甚至无法完成深层网络的训练。因此在一些现代的神经网络中，sigmoid函数逐渐被ReLU激活函数取代。

（2）sigmoid函数公式中有幂函数，计算耗时长，在反向传播误差梯度时，求导运算涉及除法。

（3）sigmoid函数的输出恒大于 0，非零中心化，在多层神经网络中，可能会造成后面层神经元的输入发生偏置偏移，导致梯度下降变慢。

* + - 1. 请简述BP学习算法的不足之处。

**答：**

（1）BP 学习算法是有监督学习，需要大量带标签的训练数据。

（2）BP神经网络中的参数量大，收敛速度慢，需要较长的训练时间，学习效率低。

（3）BP 学习算法采用梯度下降法更新学习参数，容易陷入局部极值，从而找不到全局最优解。

（4）尚无理论指导如何选择网络隐藏层的层数和神经元的个数，一般是根据经验或通过反复实验确定。因此，网络往往存在很大的冗余性，在一定程度上也增加了网络学习的负担。

* + - 1. 请简述卷积神经网络的结构。

**答：**一个卷积神经网络可以包含多个网络层，每层网络由多个独立的神经元组成。卷积神经网络中的网络层主要分为3种类型：卷积层、池化层和全连接层。通常，每个卷积层与最后一个全连接层之后都会采用激活函数。卷积神经网络可以像搭积木一样，通过叠加多个网络层来组装。

* + - 1. 按照神经元之间的连接范围进行划分，神经网络可以分为哪两大类？

**答：**全连接神经网络和卷积神经网络

* + - 1. 按照网络层之间的连接方式，神经网络可以分为哪两大类？

**答：**前馈神经网络和反馈神经网络。

* + - 1. 假设有一个3层的前馈全连接神经网络，其输入层有5个神经元，第一个隐层有10个神经元，第二个隐层有10个神经元，输出层有2个神经元，请计算神经网络中的连接总数。

**答：**170

* + - 1. 请写出Sigmoid函数的导数公式。

**答：**（

* + - 1. Sigmoid函数 与 Softmax函数是否有关联？若有，请说明两者的关联。

**答：**Sigmoid 是 Softmax的特例。当类别数为2时, Sigmoid 就是 Softmax。Sigmoid用于隐层，Softmax用于输出层。

* + - 1. 若输入数据的大小为240×240×3，对其采用大小为4×4、不重叠的池化操作，则池化操作的步长以及池化操作后特征图的大小分别是多少?

**答：**4，60\*60\*3

* + - 1. 请指出人工神经网络中的超参数。

**答：**网络层数；每个隐藏层中滤波器的尺寸、步长、个数；学习率；

# 第6章习题

1. 说明CNN结构中1×1卷积的作用，哪个网络率先使用了1×1卷积？

答：（1）1×1卷积的作用是放缩特征图的通道数，若是降维，可同时减少参数量和计算量。

（2）Inception结构

1. 试计算AlexNet的总参数量，给出详细的计算过程。

答：8层参数共计60965224个

（1）第一个卷积层的参数量= 11\*11\*3\*96+96（偏置）=34944

（2）第二个卷积层的96个特征图分为两组，各自在一个独立的GPU上进行运算的参数量= (5×5×48×128+128)×2=307456

（3）第三个卷积层的参数数量：3×3×256×384+384（偏置）=885120

（4）第四个卷积层的参数数量：(3×3×192×192+192)×2=663936

（5）第五个卷积层的参数数量：(3×3×192×128+128)×2=442624

（6）第1个全连接层的参数量= (6×6×256)×4096+4096（偏置） = 37752832

（7）第2个全连接层的参数量= 4096×4096+4096=16781312

（8）第3个全连接层的参数量= 4096×1000+1000=4097000

1. 请说明InceptionV1模块中的分支MaxPool 3×3+1**(S)** 表示什么操作，为什么1×1卷积要放在MaxPool 3×3+1(S)之后执行？

答：

（1）InceptionV1模块中的分支MaxPool 3×3+1**(S)** 表示：采用3×3的重叠最大池化，步长为1，**padding=same** (pad=1)，保证输出与输入的特征图尺寸相同。

Inception模块中所有的池化操作并非都为了下采样，分支MaxPool 3×3+1**(S)**只是为了能提取最大特征且无需学习卷积中的参数。

（2）MaxPool 3×3+1(S)没有参数，故1×1卷积放在其前与放在其后的参数量是一样的。之所以放在其后，是要在不增加参数量的前提下，尽可能保留最显著（最大值）特征，先做1×1卷积，会损失特征图的深度信息（变薄了）。

1. 请说明Batch Normalization（BN）的作用。

答：

1. 在使用梯度下降法求最优解时，防止梯度爆炸或弥散，加快收敛速度；
2. 可以提高训练时模型对于不同超参（学习率、初始化）的鲁棒性；
3. 可以使大部分激活函数能够远离其饱和区域。
4. 请描述Inception (4b)模块的结构细节。

答：Inception (4b)模块分为四个分支，采用不同尺度的卷积核来进行处理

1）第1个分支Conv **1**×**1+1(s) 表示：**160个1×1×512的卷积核，步长为1，**padding=same** (pad=0)，采用ReLU激活函数，输出14×14×160。

2）第2个分支Conv **1**×**1+1(s) 表示：112**个1×1×512的卷积核（**reduction layers**），作为3×3卷积核之前的降维，步长为1，padding=same(pad=0)，ReLU计算，变成14×14×112；然后，Conv **3**×**3+1(s) 表示：**224个3×3×112的卷积，步长为1，padding=same (pad=1)，ReLU计算后，输出14×14×224。

3）第3个分支Conv **1**×**1+1(s) 表示：24**个1×1×512的卷积核（**reduction layers**），作为5×5卷积核之前的降维，步长为1，padding=same(pad=0)，ReLU计算，变成14×14×24；然后，Conv **5**×**5+1(s) 表示： 64**个5×5×24的卷积，步长为1，padding=same (pad=2)，ReLU计算后，输出14×14×64。

4）第4个分支MaxPool 3×3+1**(S)** 表示：采用3×3的重叠最大池化，步长为1**，padding=same** (pad=1)，输出14×14×512；然后，Conv **1**×**1+1(s) 表示：64**个1×1×512的卷积（**projection layer**），步长为1，padding=same(pad=0)，ReLU计算，输出14×14×64。

DepthContat表示将四个分支的特征图层依次拼接起来，通道数为160+224+64+64=512，形成14×14×**512**的特征图层。

1. 请说明ResNet和DenseNet中短路连接机制的不同之处。

答：

**（1）两个网络中短路连接的密集程度不同**。由于ResNet的一个残差模块一般只包含2~3个卷积层，所以，每层只跨越2~3层与其前面的某一层直接连接，形成短路（或捷径）。而DenseNet的一个Dense Block模块一般包含6~64个卷积层，同一模块中的所有层均互相连接。显然，DenseNet网络中的短路连接比ResNet中的更为密集，更好地实现了特征重用，增强了特征在各个层之间的传播。

**（2）两个网络中短路连接的方式不同**。在ResNet网络的同一残差模块中，对于残差映射特征图与恒等映射特征图，执行对应位置上的元素级相加（element-wise addition）操作。而在DenseNet网络的同一Dense Block模块中，每层都会与其前面所有层在通道维度上作拼接（channel-wise concatenation）操作，并作为下一层的输入。

1. 假设输入图像为W×W×3，要求输出大小为W×W×256的特征图，请给出两种**升维**的方法，并分别计算两种方法的参数量。

答：

第一种方法：直接用256个3×3×3的滤波器，padding=same

**参数量**为：3×3×3 (深度)×256 = 6,912

第二种方法，分两步完成：

* 第1步：先用1个3×3×3滤波器做卷积操作，padding=same

得到W×W×1。参数量= 3×3×3=27.

* 第2步：用256个1×1×1滤波器做卷积操作，得到W×W×256

**总参数量**为：3×3×3 + 3×1×1×256 = 795，是6912的 **1/9**！

1. 假设上一网络层的输出为 100×100×128，要求得到100×100×256的特征图，请给出两种方法，并分别计算两种方法的参数量和计算量。

答：第一种方法：经过具有 256 个通道的 5×5 卷积层之后 (stride=1，pad=2)，输出数据为 100×100×256，其中，参数量为 128×5×5×256= 819200。

第二种方法：上一层输出先经过 32 个 1x1×128 滤波器，得到100×100×32特征图；再经过 256 个 5×5×32 (stride=1，pad=2)滤波器，那么输出数据仍为 100×100×256，但参数量已经减少为 1x1x128 x 32 + 5x5 x 32x256= 204800。

参数量大约减少了 4 倍。

# 第7章习题

1. 请简述数字图像的定义，并说明常用的数字图像类型。

答：数字图像是指用数字摄像机、扫描仪等成像设备经过采样和数字化得到的一个二维数组或矩阵，该数组或矩阵的元素称为像素（pixel），像素值均为整数，称为灰度值。

常用的数字图像有二值图像、灰度图像、彩色图像、RGBD图像等。

2. 请简要描述图像处理的任务。

答：图像处理的任务包括：图像采集与获取、图像变换、图像去噪、图像增强、图像复原、图像重建、图像编解码、图像压缩、图像边缘检测、图像分割、图像表示与描述、图像特征提取与分析、图像质量评价、图像目标检测、图像分类、图像识别、图像理解等。

3. 请简述HOG图像特征提取的过程。

答：HOG图像特征提取主要分为5个环节：彩色图像的灰度化、灰度值的Gamma校正、梯度计算、细胞单元的梯度直方图绘制、块特征向量表示。

第一步：利用加权平均或其他方式将彩色的RGB图像转化为灰度图像；

第二步：通过对像素值进行归一化、预补偿和反归一化完成图像的Gamma校正；

第三步：使用差分方式计算每个像素点的水平和垂直梯度，综合得像素点的梯度；

第四步：将图像划分为一个个细胞单元后，梯度按照角度不同分为K个组，据此细胞单元内的像素点归入不同的梯度组，并累计像素点的梯度幅值作为该组权重，最终统计得细胞单元得梯度直方图，作为细胞单元的K维特征向量；

第五步：将相邻的m×m细胞单元组合为块，随之，K维特征向量组合为m×m×K维特征向量，并将各维除以特征长度完成归一化。通过块在图上的范围移动可以计算出所有的块特征向量，组合即得此图的HOG特征表示。

4. 请简述传统图像目标检测的流程，并给出目标检测的评价指标。

答：传统图像目标检测的流程一般分为4个阶段: 预处理图像、生成目标候选区域、提取目标候选区域的特征、训练目标分类器。

第一步：首先对图像执行降噪、增强等预处理操作，提高图像信息的有效性；之后按需要将数据划分为训练集、测试集、验证集；

第二步：通过滑动窗口遍历的方式获取目标候选区域；

第三步：利用HOG、SIFT等特征表示方法描述各目标候选区域；

第四步：首先选择适配的分类器和训练算法，然后将训练集图像中各候选区域的特征向量和对应标签匹配，并输入分类器按优化算法训练模型。新图像在获取候选区域后便可利用所得模型进行目标检测。

目标检测的评价指标有交并比、精确度和召回率、AP、MAP等。

5. 图像分割包含哪几种？请简述它们之间的区别和联系。

答：图像分割包括：语义分割、实例分割和全景分割。语义分割是指对图像进行像素级分类，将同类像素点合并成一个区域，以实现对图像的分割；实例分割比语义分割要求高，除了需要完成语义分割任务之外，还需要识别图像中每个目标实例对应的像素区域，获得每个目标的轮廓；全景分割则是结合了语义分割和实例分割，不仅要对所有像素点进行分类，还应识别出图像中存在的所有目标实例。

6. 请简述迁移学习的定义，并说明在图像分类中是如何应用迁移学习的。

答：迁移学习是将在一个领域（称为源领域）的知识迁移到另外一个领域（称为目标领域），使得在目标领域能够取得更好的学习效果。迁移学习是模仿人类“举一反三”的能力。

一般是基于模型进行迁移，即在已用其他图像数据训练好的分类模型上利用新的图像数据进行微调，而不是从零开始训练深度学习图像分类模型。

7. 请简述图像分类的评价指标。

答：图像分类评价指标：Precision、Recall、Specificity、Accuracy、F-measure、AUC、AP、以及MAP等。

8. 请分别列举几个典型的图像分类、图像目标检测和图像分割常用的深度学习模型。

答：图像分类常用的深度学习模型：AlexNet、VGGNet、GoogLeNet和ResNet等；图像目标检测常用的深度学习模型：R-CNN、Fast R-CNN、Faster R-CNN、YOLO、SSD等。图像分割常用的深度学习模型：FCN、U-Net、 SegNet、DeepLab、Mask R-CNN等。

# 第8章习题

1. 请简述利用Scikit-learn库进行模型超参数选择的过程。

**答：**可先预设需要优化的超参数候选数值集合，以及模型性能比较的指标，然后构建model\_selection模块中的GridSearchCV对象进行模型筛选，最后访问best\_params\_属性获取最佳的超参数组合。

2. 对于二分类的机器学习任务，请简述利用ROC曲线选择合适分类阈值的过程。

**答：**ROC曲线显示了不同分类阈值下模型的真正例率（TPR）和假正例率（FPR），可以通过观察约登指数（Youden index）的大小来选择合适的分类阈值，利用敏感度和特异度之和减去1：Youden index = TPR + (1 -FPR) – 1，约登指数范围在0-1之间，越接近1表示分类效果很好。最佳阈值就是使得约登指数最大的那个点。

3. 请简述自动求导机制。

**答：**自动求导机制是一种在神经网络计算框架中实现梯度计算的技术，它可以根据网络结构和损失函数构建动态计算图，然后再通过反向传播进行梯度计算，从而实现参数的更新。

4. 请简述PyTorch与Scikit-learn之间的差异。

**答：**PyTorch是一个深度学习框架，主要用于构建和训练神经网络。而Scikit-learn是一个通用的机器学习库，提供了很多常用的机器学习算法和数据处理方法。两者有时也可以结合使用，例如在使用PyTorch框架之前，可以用Scikit-learn对数据进行预处理。

5. 请简述PyTorch中Tensor和Function的关系。

**答：**PyTorch中的Tensor是一种类似于Numpy的多维数组，可以用于存储和操作数据。Function是一种用于定义Tensor之间的运算和变换的对象，可以实现自动求导功能。两者的关系通过一个属性grad\_fn来体现，grad\_fn表示了当前Tensor是如何从其他Tensor计算得到的。这样在我们需要计算梯度时，PyTorch可以通过反向追踪grad\_fn来构建动态计算图，并使用链式法则来自动求导。

6. 请简述PyTorch中如何利用动态计算图实现自动求导。

**答：**可分为以下四步：

（1）在我们创建一个Tensor时，可以设置它的requires\_grad属性为True，表示我们需要对它进行求导。

（2）当我们对Tensor进行各种运算和变换时，PyTorch会自动记录这些操作，并构建一个由Function对象组成的有向无环图（DAG），表示Tensor之间的关系。

（3）当我们调用Tensor的backward()方法时，PyTorch会从当前Tensor开始。沿着动态计算图反向传播，根据链式法则计算每个Tensor的梯度，并将结果保存在它们的grad属性中。

（4）通过访问Tensor的grad属性即可获取梯度值。

7. 请描述PyTorch中model.train()和model.eval()的作用。

**答：**model.train()的作用是启用Batch Normalization和Dropout，在训练时有助于防止过拟合，但在测试时会影响模型的输出。model.eval()的作用是不启用Batch Normalization和Dropout，这样可以保证模型在测试时的输出是确定的，不受随机因素的影响。一般来说，在训练时我们需要调用model.train()，在测试或预测时我们需要调用model.eval()。

8. 请描述PyTorch中选择深度学习模型的超参数的过程。

**答：**可以分为以下几步进行：

（1）定义一个超参数空间，即需要调整的超参数的范围和分布，例如学习率、批量大小等。

（2）定义一个目标函数，即需要优化的指标，例如准确率、损失函数等。

（3）选择一个搜索算法，即用于探索超参数空间的方法，例如网格搜索、随即搜索、贝叶斯优化等。

（4） 运行多次实验，每次实验根据搜索算法从超参数空间采样一组超参数，并使用PyTorch构建和训练模型，记录目标函数的值。

（5）分析实验结果，找出最优的一组超参数。

9. 请简述如何使用PyTorch构建和加载自定义数据集。

**答：**（1）定义一个继承自torch.utils.data.Dataset的类，并实现三个方法：\_\_init\_\_,\_\_len\_\_和\_\_getitem\_\_。在\_\_init\_\_函数中，需要加载所有样本和标签数据；在\_\_len\_\_函数中需要返回样本的数量；在\_\_getitem\_\_函数中，需要根据输入的样本索引返回对应的样本特征张量和对应的标签。

（2）创建一个torch.utils.data.DataLoader对象，传入数据集对象和其他参数，如batch\_size，shuffle等。

（3）迭代DataLoader对象来获取批量的样本和标签。

10. 请简述Scikit-learn模型训练过程。

**答：**Scikit-learn模型训练过程大致分为以下几个步骤：

（1）导入Scikit-learn模块和所需的算法，如from sklearn import linear\_model。

（2）准备数据集，将特征和标签分开，如X = mnist.data，y = mnist.target。

（3）划分训练集和测试集，如from sklearn.model\_selection import train\_test\_split，X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y)。

（4）创建模型，如model = linear\_model.LinearRegression()。

（5）训练模型，使用fit方法，如model.fit(X\_train, y\_train)。

（6）评估模型，使用score或predict方法，如model.score(X\_test, y\_test), y\_pred = model.predict(X\_test)。

# 第9章习题

1. 请简述构建机器学习模型的基本步骤。

**答：**

（1）数据预处理：对原始数据进行清洗、数学表示、数学变换、数据整合和特征选择等操作，使其符合模型输入的要求。

（2）数据集构建：根据根据数据分布规律和任务特点设计合理的样本选择和划分策略，以特定比例将样本分为训练集、验证集或测试集。

（3）模型构建：根据目标和数据选择合适的机器学习算法（如决策树、支持向量机等），并设置相关的超参数（如学习率、正则化系数等）。

（4）模型训练：使用训练集数据对模型进行拟合，优化模型参数，提高模型性能。

（5）模型性能评估及测试：根据机器学习任务类型和目标选择合适的单个或多个性能评价指标（AUC、敏感度、特异度、精度及F1等），并且通过比较模型在训练集和验证或测试集上性能的差别评估是否存在过拟合问题。根据模型性能评估结果可对预测误差较大的样本进行分析，寻找模型改进策略。

（6）模型超参数优化：当模型存在多个超参数时，可以基于搭建的模型训练和评估流程，通过枚举或搜索等方式比较不同超参数组合下模型训练和评估的性能指标差异，寻找最佳的超参数组合。

2. 请简述paddle.nn.Sequential类和paddle.nn.Layer类的作用与区别。

**答：**paddle.nn.Sequential类是一个顺序容器，它可以将多个子Layer按照构造函数参数的顺序添加到容器中，并按照顺序依次执行。paddle.nn.Layer类是所有神经网络层的基类，它定义了一些通用的属性和方法，例如name\_scope、parameters、sublayers等，用户可以通过继承Layer类来自定义自己的神经网络层。

3. 请简述百度飞桨框架中Model.fit方法的作用及其对应的运算过程。

**答：**Model.fit方法是百度飞桨框架中的一个高层API，用于完成模型的训练和保存。它的运算过程大致如下：

（1）将模型封装为 paddle.Model对象，并指定优化器、损失函数和评估指标。

（2）调用 Model.fit方法，传入训练数据、训练轮数、批次大小、保存路径等参数。

（3） Model.fit方法在每个批次上执行前向传播、反向传播和参数更新，并在每个轮数结束时计算评估指标。

（4） Model.fit方法在指定的保存路径下保存模型的参数和优化器状态，并输出训练日志。

4. 请采用非线性拟合的LSTM模型（paddle.nn.LSTM）完成波士顿房价预测任务。

**答：**import paddle  
  
paddle.device.set\_device("gpu")  
  
#设置默认的全局dtype为float64  
paddle.set\_default\_dtype("float64")  
#下载数据  
train\_dataset = paddle.text.datasets.UCIHousing(mode='train')  
test\_dataset = paddle.text.datasets.UCIHousing(mode='test')  
  
# 定义LSTM网络结构  
class LSTM(paddle.nn.Layer):  
 def \_\_init\_\_(self):  
 super(LSTM, self).\_\_init\_\_()  
 self.lstm= paddle.nn.LSTM(input\_size=13 ,hidden\_size=32 ,num\_layers=1)  
 self.linear= paddle.nn.Linear(in\_features=32 ,out\_features=1)  
  
 def forward(self,x):  
 x = paddle.unsqueeze(x, axis=1) # 在中间维度上增加一个维度，变成三维张量  
 x,\_= self.lstm(x) # x.shape: (batch\_size ,seq\_len ,hidden\_size )  
 x= x[:, -1, :] # 取最后一个时刻的输出作为全连接层的输入，x.shape: (batch\_size ,hidden\_size )  
 x= self.linear(x) # x.shape: (batch\_size ,output\_size )  
 return x  
# 模型训练配置  
model = paddle.Model(LSTM())  
model.prepare(optimizer=paddle.optimizer.Adam(learning\_rate = 0.001, parameters =model.parameters()) , loss = paddle.nn.MSELoss())  
# 训练并验证模型  
model.fit(train\_data = train\_dataset, eval\_data = test\_dataset, epochs=10, batch\_size=8, verbose=1)

5. 请采用LeNet模型完成手写体数字识别任务，实现完整的模型训练和预测过程。

**答：**from paddle.metric import Accuracy  
from paddle.vision.transforms import Compose, Normalize  
import numpy as np  
import paddle  
import matplotlib.pyplot as plt  
  
paddle.device.set\_device("gpu")  
  
transform = Compose([Normalize(mean=[127.5],std=[127.5],data\_format='CHW')])  
# 使用transform对数据集做归一化  
train\_dataset = paddle.vision.datasets.MNIST(mode='train', transform=transform)  
test\_dataset = paddle.vision.datasets.MNIST(mode='test', transform=transform)  
  
# 定义模型,使用paddle里自带的LeNet函数  
LeNet = paddle.vision.models.LeNet(num\_classes=10)  
  
# 用Model封装模型  
model = paddle.Model(LeNet)  
optimizer = paddle.optimizer.Adam(learning\_rate=0.001, parameters=model.parameters())  
# 配置模型  
model.prepare(optimizer,paddle.nn.CrossEntropyLoss(),Accuracy())  
  
#模型训练  
#model.fit(train\_dataset,test\_dataset,epochs=2,batch\_size=64,save\_dir='multilayer\_perceptron',verbose=1)  
model.fit(train\_dataset,test\_dataset,epochs=2,batch\_size=64,verbose=1)  
  
#获取测试集的第一个图片  
test\_data0, test\_label\_0 = test\_dataset[0][0],test\_dataset[0][1]  
test\_data0 = test\_data0.reshape([28,28])  
plt.figure(figsize=(2,2))  
#展示测试集中的第一个图片  
print(plt.imshow(test\_data0, cmap=plt.cm.binary))  
plt.show()  
print('test\_data0 的标签为: ' + str(test\_label\_0))  
#模型预测  
result = model.predict(test\_dataset, batch\_size=1)  
print('test\_data0 预测的数值为：%d' % np.argsort(result[0][0])[0][-1])

6. 请采用飞桨框架构建多层卷积神经网络，完成鸢尾花分类任务。

**答：**import numpy as np  
import paddle.nn.functional as F  
import paddle  
from sklearn import model\_selection  
  
paddle.device.set\_device("gpu")  
  
#设置默认的全局dtype为float64  
paddle.set\_default\_dtype("float64")  
# 类别标签转化  
def Iris\_type(s):  
 it = {b'Iris-setosa': 0, b'Iris-versicolor': 1, b'Iris-virginica': 2}  
 return it[s]  
  
# 数据准备  
# 加载数据，本地若没有鸢尾花数据集需下载到本地  
data = np.loadtxt('./datasets/Iris.data', # 数据文件路径  
 dtype = float, # 数据类型  
 delimiter = ',', # 数据分割符  
 converters = {4: Iris\_type}) # 将第五列使用函数Iris\_type进行转换  
# 数据分割  
x, y = np.split(data, (4,), axis=1) # 数据分组 第五列开始往后为y 代表纵向分割按列分割  
x\_train, x\_test, y\_train, y\_test = model\_selection.train\_test\_split(x, y, random\_state=1, test\_size=0.2)  
x\_train = paddle.to\_tensor(x\_train)  
x\_test = paddle.to\_tensor(x\_test)  
y\_train = paddle.to\_tensor(y\_train)  
y\_test = paddle.to\_tensor(y\_test)  
  
class MyDataset(paddle.io.Dataset):  
 def \_\_init\_\_(self,feature,label):  
 # 初始化数据集  
 self.data = feature  
 self.label = label  
  
 def \_\_getitem\_\_(self, index):  
 # 根据索引返回样本和标签  
 return self.data[index], self.label[index]  
  
 def \_\_len\_\_(self):  
 # 返回数据集大小  
 return len(self.data)  
  
train\_dataset = MyDataset(feature=x\_train, label=y\_train)  
test\_dataset = MyDataset(feature=x\_test, label=y\_test)  
  
# 构建多层卷积神经网络  
class CNN(paddle.nn.Layer):  
 def \_\_init\_\_(self):  
 super(CNN, self).\_\_init\_\_()  
 # 创建卷积和池化层块，每个卷积层后面接着2x2的池化层  
 #卷积层L1  
 self.conv1 = paddle.nn.Conv1D(in\_channels=1,  
 out\_channels=16,  
 kernel\_size=2)  
 #池化层L2  
 self.pool1 = paddle.nn.MaxPool1D(kernel\_size=2)  
 #卷积层L3  
 self.conv2 = paddle.nn.Conv1D(in\_channels=16,  
 out\_channels=32,  
 kernel\_size=2)  
 #池化层L4  
 self.pool2 = paddle.nn.MaxPool1D(kernel\_size=4)  
 #线性层L5  
 self.fc=paddle.nn.Linear(64,3)  
  
 #正向传播过程  
 def forward(self, x):  
 x = paddle.unsqueeze(x, axis=1) # 在最后一个维度上增加一个维度，变成三维张量  
 x = self.conv1(x)  
 x = F.relu(x)  
 x = self.pool1(x)  
 x = F.relu(x)  
 x = self.conv2(x)  
 x = self.pool2(x)  
 x = paddle.flatten(x, start\_axis=1,stop\_axis=-1)  
 out = self.fc(x)  
 return out  
  
model=paddle.Model(CNN())  
  
model.prepare(paddle.optimizer.Adam(parameters=model.parameters()),  
 paddle.nn.CrossEntropyLoss(),  
 paddle.metric.Accuracy())  
  
model.fit(train\_dataset,  
 epochs=5,  
 batch\_size=10,  
 verbose=1)  
  
model.evaluate(x\_test,y\_test,verbose=1)

7. 请在9.3节手写体数字识别的模型训练和测试过程中，增加其他性能指标的计算和比较（如AUC、敏感度、特异度等）。

**答：**

from paddle.metric import Accuracy  
from paddle.vision.transforms import Compose, Normalize  
from sklearn.metrics import roc\_curve, auc, confusion\_matrix  
from sklearn.preprocessing import OneHotEncoder  
import numpy as np  
import paddle.nn.functional as F  
import paddle.nn as nn  
import paddle as paddle  
import matplotlib.pyplot as plt  
  
paddle.device.set\_device("gpu")  
  
transform = Compose([Normalize(mean=[127.5],std=[127.5],data\_format='CHW')])  
# 使用transform对数据集做归一化  
train\_dataset = paddle.vision.datasets.MNIST(mode='train', transform=transform)  
test\_dataset = paddle.vision.datasets.MNIST(mode='test', transform=transform)  
test\_loader = paddle.io.DataLoader(test\_dataset, batch\_size=100)  
  
class mnist(paddle.nn.Layer):  
 def \_\_init\_\_(self):  
 super(mnist,self).\_\_init\_\_()  
 self.fc1 = nn.Linear(in\_features=28 \* 28, out\_features=100)  
 self.fc2 = nn.Linear(in\_features=100, out\_features=100)  
 self.fc3 = nn.Linear(in\_features=100, out\_features=10)  
  
 def forward(self, input\_):  
 x = paddle.reshape(input\_, [input\_.shape[0], -1])  
 x = self.fc1(x)  
 x = F.relu(x)  
 x = self.fc2(x)  
 x = F.relu(x)  
 x = self.fc3(x)  
 y = F.softmax(x)  
 return y  
  
# 用Model封装模型  
model = paddle.Model(mnist())  
optimizer = paddle.optimizer.Adam(learning\_rate=0.001, parameters=model.parameters())  
# 配置模型  
model.prepare(optimizer,paddle.nn.CrossEntropyLoss(),Accuracy())  
  
#模型训练  
model.fit(train\_dataset,test\_dataset,epochs=2,batch\_size=64,verbose=1)  
  
y\_true = [] # 真实标签列表  
y\_score = [] # 预测得分列表  
fpr = []  
tpr = []  
roc\_auc = [] # auc值列表  
  
for data in test\_loader:  
 x, y = data # x是输入，y是真实标签  
 out = model.predict(x) # out是输出，包含预测得分  
 y\_true.extend(y.numpy().tolist()) # 将真实标签添加到列表中  
 y\_score.extend(out) # 将预测得分添加到列表中  
  
# 创建一个 one-hot 编码器实例  
encoder = OneHotEncoder()  
# 对真实标签进行 one-hot 编码  
y = encoder.fit\_transform(y\_true).toarray()  
y\_true = y.reshape(-1,10)  
y\_score = np.array(y\_score)  
y\_score = y\_score.reshape(-1,10)  
  
# 单独计算每一类对应的AUC值  
for i in range(10):  
 FPR, TPR, \_ = roc\_curve(y\_true[:, i], y\_score[:, i])  
 fpr.append(FPR)  
 tpr.append(TPR)  
 Roc\_auc = auc(fpr[i], tpr[i])  
 roc\_auc.append(Roc\_auc)  
  
# plot ROC curves for each class  
colors = ['blue', 'red', 'green', 'black', 'gray', 'gold', 'yellow', 'orange', 'pink', 'teal']  
for i in range(10):  
 plt.plot(fpr[i], tpr[i], color=colors[i],  
 label='ROC curve of class {0} (area = {1:0.2f})'  
 ''.format(i+1 , roc\_auc[i]))  
print('AUC值: {}'.format(roc\_auc))  
  
# 计算分类阈值为0.5时模型的敏感度特异度  
y\_predict = []  
for i in range(10):  
 row = []  
 for j in range(len(y\_score)):  
 if y\_score[j,i] > 0.5:  
 row.append(1)  
 else:  
 row.append(0)  
 y\_predict.append(row)  
# 使用混淆矩阵计算敏感度、特异度  
y\_predict = np.array(y\_predict)  
y\_predict = y\_predict.reshape(-1,10)  
for i in range(10):  
 tn, fp, fn, tp = confusion\_matrix(y\_true[:, i], y\_predict[:, i]).ravel()  
 TPR = tp / (tp + fn)  
 FPR = fp / (tn + fp)  
 print('数字{}对应的敏感度特异度分别为：{} {}'.format(i,TPR,1-FPR))  
  
#获取测试集的第一个图片  
test\_data0, test\_label\_0 = test\_dataset[0][0],test\_dataset[0][1]  
test\_data0 = test\_data0.reshape([28,28])  
plt.figure(figsize=(2,2))  
#展示测试集中的第一个图片  
print(plt.imshow(test\_data0, cmap=plt.cm.binary))  
plt.show()  
print('test\_data0 的标签为: ' + str(test\_label\_0))  
#模型预测  
result = model.predict(test\_dataset, batch\_size=1)  
print('test\_data0 预测的数值为：%d' % np.argsort(result[0][0])[0][-1])

8. 请在9.3节手写体数字识别程序中增加模型超参数选择的过程，寻找最优的模型学习率和全连接层层数。

**答：**

from paddle.metric import Accuracy  
from paddle.vision.transforms import Compose, Normalize  
import numpy as np  
import paddle.nn.functional as F  
import paddle.nn as nn  
import paddle as paddle  
import matplotlib.pyplot as plt  
  
paddle.device.set\_device("gpu")  
  
transform = Compose([Normalize(mean=[127.5],std=[127.5],data\_format='CHW')])  
# 使用transform对数据集做归一化  
train\_dataset = paddle.vision.datasets.MNIST(mode='train', transform=transform)  
test\_dataset = paddle.vision.datasets.MNIST(mode='test', transform=transform)  
test\_loader = paddle.io.DataLoader(test\_dataset, batch\_size=100)  
  
# 先列出几种模型超参数对应的集合，为方便只列出几种  
learningrates = [0.001,0.005,0.01]  
FC1 = [200,100]  
FC2 = [100,50]  
  
class mnist(paddle.nn.Layer):  
 def \_\_init\_\_(self):  
 super(mnist,self).\_\_init\_\_()  
 self.fc1 = nn.Linear(in\_features=28 \* 28, out\_features=m)  
 self.fc2 = nn.Linear(in\_features=m, out\_features=n)  
 self.fc3 = nn.Linear(in\_features=n, out\_features=10)  
  
 def forward(self, input\_):  
 x = paddle.reshape(input\_, [input\_.shape[0], -1])  
 x = self.fc1(x)  
 x = F.relu(x)  
 x = self.fc2(x)  
 x = F.relu(x)  
 x = self.fc3(x)  
 y = F.softmax(x)  
 return y  
  
# 从所列超参数集合中选取参数，观察每次组合对应在测试数据集上的准确率以此选取最优超参数  
for i in learningrates:  
 for m in FC1:  
 for n in FC2:  
 # 用Model封装模型  
 model = paddle.Model(mnist())  
 optimizer = paddle.optimizer.Adam(learning\_rate=i, parameters=model.parameters())  
 # 配置模型  
 model.prepare(optimizer, paddle.nn.CrossEntropyLoss(), Accuracy())  
 # 返回模型预测值对应的标签  
 predict = paddle.to\_tensor(model.predict(test\_dataset))  
 predict = paddle.reshape(predict, (-1, 10))  
 label = paddle.argmax(predict, axis=1)  
 #print(label)  
 labels = paddle.to\_tensor(test\_dataset.labels)  
 # 模型训练  
 model.fit(train\_dataset, epochs=2, batch\_size=64, verbose=1)  
 acc = paddle.fluid.layers.accuracy(predict, labels)  
 print('学习率为{}，全连接层第一层数为{}，第二层为{}时，模型在测试集上的准确率为：{}'.format(i,m,n,acc))  
 # 获取测试集的第一个图片  
 test\_data0, test\_label\_0 = test\_dataset[0][0], test\_dataset[0][1]  
 test\_data0 = test\_data0.reshape([28, 28])  
 plt.figure(figsize=(2, 2))  
 # 展示测试集中的第一个图片  
 print(plt.imshow(test\_data0, cmap=plt.cm.binary))  
 plt.show()  
 print('test\_data0 的标签为: ' + str(test\_label\_0))  
 # 模型预测  
 result = model.predict(test\_dataset, batch\_size=1)  
 print('test\_data0 预测的数值为：%d' % np.argsort(result[0][0])[0][-1])

9. 请在9.4节动物图像分类程序中，改用深度残差网络ResNet重新构建模型，完成图像分类任务。

**答：**

from paddle.vision.models.resnet import BasicBlock  
import os  
import paddle  
import numpy as np  
from PIL import Image  
import matplotlib.pyplot as plt  
from paddle.io import Dataset  
import random  
  
paddle.device.set\_device("gpu")  
  
train\_paramters = {  
 'class\_dim': 3, #图像类别数量  
 'train\_dir': './datasets/afhq/train/', #训练数据集目录  
 'train\_list\_path': './datasets/train\_img\_list.txt', # 训练图片信息  
 'test\_dir': './datasets/afhq/val/', # 测试数据集目录  
 'test\_list\_path': './datasets/test\_img\_list.txt', # 测试图片信息  
 'num\_epochs': 5, # 训练轮数  
 'train\_batch': 128, # 批次大小  
 'lr': 0.0001 # 学习率  
}  
  
def get\_data(data\_dir, img\_file):  
 img\_list = []  
 for k, psubd in enumerate(['cat', 'dog', 'wild']):  
 for perf in os.listdir(data\_dir + '/' + psubd):  
 if not perf.endswith('jpg'):  
 continue  
  
 ppath = data\_dir + '/' + psubd + '/' + perf  
 img\_list.append([ppath, k])  
 print(len(img\_list))  
 random.shuffle(img\_list)  
  
 with open(img\_file, 'w') as tf:  
 for per in img\_list:  
 tf.write('{} {}\n'.format(per[0], per[1]))  
  
get\_data(train\_paramters['train\_dir'], train\_paramters['train\_list\_path'])  
get\_data(train\_paramters['test\_dir'], train\_paramters['test\_list\_path'])  
  
class MyDataset(Dataset):  
 def \_\_init\_\_(self, mode = 'train'):  
 super(MyDataset, self).\_\_init\_\_()  
 self.data = []  
 self.label = []  
 img\_file = train\_paramters['train\_list\_path']  
 if mode == 'val':  
 img\_file = train\_paramters['test\_list\_path']  
  
 with open(img\_file, 'r') as tf:  
 for per in tf:  
 img\_path, lab = per.strip().split(' ', 1)  
 img = Image.open(img\_path)  
 if img.mode != 'RGB':  
 img = img.convert('RGB')  
 img = img.resize((128, 128), Image.BILINEAR)  
 img = np.array(img).astype('float32')  
 img = img.transpose((2, 0, 1)) # HWC to CHW  
 img = img/255 # 像素值归一化  
 self.data.append(img)  
 self.label.append(int(lab))  
  
 # 返回对应索引的图像信息  
 def \_\_getitem\_\_(self, index):  
 data = self.data[index]  
 label = self.label[index]  
 #注：返回标签数据时必须是int64  
 return data, np.array(label).astype('int64')  
  
 def \_\_len\_\_(self):  
 #返回数据总数  
 return len(self.data)  
  
train\_dataset = MyDataset('train')  
print(train\_dataset.\_\_len\_\_())  
test\_dataset = MyDataset('val')  
print(test\_dataset.\_\_len\_\_())  
  
# 定义模型，使用预训练的ResNet50，并修改最后一层为3个类别  
ResNet = paddle.vision.models.ResNet(BasicBlock, 18,num\_classes=3)  
  
print('model')  
model = paddle.Model(ResNet)  
# 定义损失函数  
print('prepare')  
model.prepare(paddle.optimizer.Adam(parameters=model.parameters(), learning\_rate = train\_paramters['lr']),paddle.nn.CrossEntropyLoss(),paddle.metric.Accuracy())  
# 训练可视化VisualDL工具的回调函数  
visualdl = paddle.callbacks.VisualDL(log\_dir='visualdl\_log')  
# 启动模型全流程训练  
print('fit')  
model.fit(train\_dataset, # 训练数据集  
 test\_dataset, # 测试数据集  
 epochs=train\_paramters['num\_epochs'], # 总的训练轮次  
 batch\_size = train\_paramters['train\_batch'], # 批次计算的样本量大小  
 shuffle=False, # 是否打乱样本集  
 verbose=1, # 日志展示格式  
 save\_dir='./result/', # 分阶段的训练模型存储路径  
 save\_freq = 5,  
 callbacks=[visualdl]) # 回调函数使用  
#保存模型  
#model.save('./result/final\_model.pth')  
  
label\_list = ['cat', 'dog', 'wild']  
pred\_res = model.predict(test\_dataset)  
indexs = [1 ,2, 3, 4, 5, 6, 7, 8, 9]  
for idx in indexs:  
 predict\_label = np.argmax(pred\_res[0][idx])  
 pimg, real\_label = test\_dataset.\_\_getitem\_\_(idx)  
 print('样本ID：{}, 真实标签：{}, 预测值：{}'.format(idx, label\_list[real\_label], label\_list[predict\_label]))  
 plt.subplot(3, 3, idx)  
 plt.imshow(pimg.transpose((1,2,0)))  
 plt.title(label\_list[predict\_label])  
 plt.axis('off')  
plt.show()

10. 采用Oxford-IIIT Pet数据集（<https://www.robots.ox.ac.uk/~vgg/data/pets>）中宠物猫的图像，利用U-Net模型，编程实现完整的图像分割模型的训练和测试。

答：

import os  
import io  
import numpy as np  
from paddle.io import Dataset  
from paddle.vision.transforms import transforms as T  
import matplotlib.pyplot as plt  
from PIL import Image as PilImage  
import paddle  
from paddle.nn import functional as F  
  
paddle.device.set\_device("gpu")  
  
IMAGE\_SIZE = (160, 160)  
train\_images\_path = "./datasets/Oxford-IIIT Pet/images/"  
label\_images\_path = "./datasets/Oxford-IIIT Pet/annotations/trimaps/"  
image\_count = len([os.path.join(train\_images\_path, image\_name)  
 for image\_name in os.listdir(train\_images\_path)  
 if image\_name.endswith('.jpg')])  
  
# 对数据集进行处理，划分训练集、测试集  
def \_sort\_images(image\_dir, image\_type):  
 # 对文件夹内的图像进行按照文件名排序  
 files = []  
 for image\_name in os.listdir(image\_dir):  
 if image\_name.endswith('.{}'.format(image\_type)) \  
 and not image\_name.startswith('.'):  
 files.append(os.path.join(image\_dir, image\_name))  
  
 return sorted(files)  
  
def write\_file(mode, images, labels):  
 with open('./datasets/Oxford-IIIT Pet/{}.txt'.format(mode), 'w') as f:  
 for i in range(len(images)):  
 f.write('{}\t{}\n'.format(images[i], labels[i]))  
  
# 第一步是对原始的数据集进行整理，得到数据集和标签两个数组，分别一一对应。  
images = \_sort\_images(train\_images\_path, 'jpg')  
labels = \_sort\_images(label\_images\_path, 'png')  
# 由于数据集中猫类动物占比过少，故选择较少猫类数据作为测试集  
eval\_num = int(image\_count \* 0.05)  
  
write\_file('train', images[eval\_num:], labels[eval\_num:])  
write\_file('test', images[:eval\_num], labels[:eval\_num])  
write\_file('predict', images[:eval\_num], labels[:eval\_num])  
  
class PetDataset(Dataset):  
 *"""  
 数据集定义  
 """* def \_\_init\_\_(self, mode='train'):  
 *"""  
 构造函数  
 """* self.image\_size = IMAGE\_SIZE  
 self.mode = mode.lower()  
 assert self.mode in ['train', 'test', 'predict'], \  
 "mode should be 'train' or 'test' or 'predict', but got {}".format(self.mode)  
 self.train\_images = []  
 self.label\_images = []  
 with open('./datasets/Oxford-IIIT Pet/{}.txt'.format(self.mode), 'r') as f:  
 for line in f.readlines():  
 image, label = line.strip().split('\t')  
 self.train\_images.append(image)  
 self.label\_images.append(label)  
 def \_load\_img(self, path, color\_mode='rgb', transforms=[]):  
 *"""  
 统一的图像处理接口封装，用于规整图像大小和通道  
 """* with open(path, 'rb') as f:  
 img = PilImage.open(io.BytesIO(f.read()))  
 if color\_mode == 'grayscale':  
 # if image is not already an 8-bit, 16-bit or 32-bit grayscale image  
 # convert it to an 8-bit grayscale image.  
 if img.mode not in ('L', 'I;16', 'I'):  
 img = img.convert('L')  
 elif color\_mode == 'rgba':  
 if img.mode != 'RGBA':  
 img = img.convert('RGBA')  
 elif color\_mode == 'rgb':  
 if img.mode != 'RGB':  
 img = img.convert('RGB')  
 else:  
 raise ValueError('color\_mode must be "grayscale", "rgb", or "rgba"')  
  
 return T.Compose([  
 T.Resize(self.image\_size)  
 ] + transforms)(img)  
  
 def \_\_getitem\_\_(self, idx):  
 *"""  
 返回 image, label  
 """* train\_image = self.\_load\_img(self.train\_images[idx],  
 transforms=[  
 T.Transpose(),  
 T.Normalize(mean=127.5, std=127.5)  
 ]) # 加载原始图像  
 label\_image = self.\_load\_img(self.label\_images[idx],  
 color\_mode='grayscale',  
 transforms=[T.Grayscale()]) # 加载Label图像  
  
 # 返回image, label  
 train\_image = np.array(train\_image, dtype='float32')  
 label\_image = np.array(label\_image, dtype='int64')  
 return train\_image, label\_image  
  
 def \_\_len\_\_(self):  
 *"""  
 返回数据集总数  
 """* return len(self.train\_images)  
  
class SeparableConv2D(paddle.nn.Layer):  
 def \_\_init\_\_(self,  
 in\_channels,  
 out\_channels,  
 kernel\_size,  
 stride=1,  
 padding=0,  
 dilation=1,  
 groups=None,  
 weight\_attr=None,  
 bias\_attr=None,  
 data\_format="NCHW"):  
 super(SeparableConv2D, self).\_\_init\_\_()  
  
 self.\_padding = padding  
 self.\_stride = stride  
 self.\_dilation = dilation  
 self.\_in\_channels = in\_channels  
 self.\_data\_format = data\_format  
  
 # 第一次卷积参数，没有偏置参数  
 filter\_shape = [in\_channels, 1] + self.convert\_to\_list(kernel\_size, 2, 'kernel\_size')  
 self.weight\_conv = self.create\_parameter(shape=filter\_shape, attr=weight\_attr)  
  
 # 第二次卷积参数  
 filter\_shape = [out\_channels, in\_channels] + self.convert\_to\_list(1, 2, 'kernel\_size')  
 self.weight\_pointwise = self.create\_parameter(shape=filter\_shape, attr=weight\_attr)  
 self.bias\_pointwise = self.create\_parameter(shape=[out\_channels],  
 attr=bias\_attr,  
 is\_bias=True)  
  
 def convert\_to\_list(self, value, n, name, dtype=np.int):  
 if isinstance(value, dtype):  
 return [value, ] \* n  
 else:  
 try:  
 value\_list = list(value)  
 except TypeError:  
 raise ValueError("The " + name +  
 "'s type must be list or tuple. Received: " + str(  
 value))  
 if len(value\_list) != n:  
 raise ValueError("The " + name + "'s length must be " + str(n) +  
 ". Received: " + str(value))  
 for single\_value in value\_list:  
 try:  
 dtype(single\_value)  
 except (ValueError, TypeError):  
 raise ValueError(  
 "The " + name + "'s type must be a list or tuple of " + str(  
 n) + " " + str(dtype) + " . Received: " + str(  
 value) + " "  
 "including element " + str(single\_value) + " of type" + " "  
 + str(type(single\_value)))  
 return value\_list  
  
 def forward(self, inputs):  
 conv\_out = F.conv2d(inputs,  
 self.weight\_conv,  
 padding=self.\_padding,  
 stride=self.\_stride,  
 dilation=self.\_dilation,  
 groups=self.\_in\_channels,  
 data\_format=self.\_data\_format)  
  
 out = F.conv2d(conv\_out,  
 self.weight\_pointwise,  
 bias=self.bias\_pointwise,  
 padding=0,  
 stride=1,  
 dilation=1,  
 groups=1,  
 data\_format=self.\_data\_format)  
  
 return out  
# 定义encoder结构  
class Encoder(paddle.nn.Layer):  
 def \_\_init\_\_(self, in\_channels, out\_channels):  
 super(Encoder, self).\_\_init\_\_()  
  
 self.relus = paddle.nn.LayerList(  
 [paddle.nn.ReLU() for i in range(2)])  
 self.separable\_conv\_01 = SeparableConv2D(in\_channels,  
 out\_channels,  
 kernel\_size=3,  
 padding='same')  
 self.bns = paddle.nn.LayerList(  
 [paddle.nn.BatchNorm2D(out\_channels) for i in range(2)])  
  
 self.separable\_conv\_02 = SeparableConv2D(out\_channels,  
 out\_channels,  
 kernel\_size=3,  
 padding='same')  
 self.pool = paddle.nn.MaxPool2D(kernel\_size=3, stride=2, padding=1)  
 self.residual\_conv = paddle.nn.Conv2D(in\_channels,  
 out\_channels,  
 kernel\_size=1,  
 stride=2,  
 padding='same')  
  
 def forward(self, inputs):  
 previous\_block\_activation = inputs  
  
 y = self.relus[0](inputs)  
 y = self.separable\_conv\_01(y)  
 y = self.bns[0](y)  
 y = self.relus[1](y)  
 y = self.separable\_conv\_02(y)  
 y = self.bns[1](y)  
 y = self.pool(y)  
  
 residual = self.residual\_conv(previous\_block\_activation)  
 y = paddle.add(y, residual)  
  
 return y  
# 定义decoder结构  
class Decoder(paddle.nn.Layer):  
 def \_\_init\_\_(self, in\_channels, out\_channels):  
 super(Decoder, self).\_\_init\_\_()  
  
 self.relus = paddle.nn.LayerList(  
 [paddle.nn.ReLU() for i in range(2)])  
 self.conv\_transpose\_01 = paddle.nn.Conv2DTranspose(in\_channels,  
 out\_channels,  
 kernel\_size=3,  
 padding=1)  
 self.conv\_transpose\_02 = paddle.nn.Conv2DTranspose(out\_channels,  
 out\_channels,  
 kernel\_size=3,  
 padding=1)  
 self.bns = paddle.nn.LayerList(  
 [paddle.nn.BatchNorm2D(out\_channels) for i in range(2)]  
 )  
 self.upsamples = paddle.nn.LayerList(  
 [paddle.nn.Upsample(scale\_factor=2.0) for i in range(2)]  
 )  
 self.residual\_conv = paddle.nn.Conv2D(in\_channels,  
 out\_channels,  
 kernel\_size=1,  
 padding='same')  
  
 def forward(self, inputs):  
 previous\_block\_activation = inputs  
 y = self.relus[0](inputs)  
 y = self.conv\_transpose\_01(y)  
 y = self.bns[0](y)  
 y = self.relus[1](y)  
 y = self.conv\_transpose\_02(y)  
 y = self.bns[1](y)  
 y = self.upsamples[0](y)  
 residual = self.upsamples[1](previous\_block\_activation)  
 residual = self.residual\_conv(residual)  
 y = paddle.add(y, residual)  
 return y  
# 定义分割模型结构  
class PetNet(paddle.nn.Layer):  
 def \_\_init\_\_(self, num\_classes):  
 super(PetNet, self).\_\_init\_\_()  
  
 self.conv\_1 = paddle.nn.Conv2D(3, 32,  
 kernel\_size=3,  
 stride=2,  
 padding='same')  
 self.bn = paddle.nn.BatchNorm2D(32)  
 self.relu = paddle.nn.ReLU()  
 in\_channels = 32  
 self.encoders = []  
 self.encoder\_list = [64, 128, 256]  
 self.decoder\_list = [256, 128, 64, 32]  
  
 # 根据下采样个数和配置循环定义子Layer，避免重复写一样的程序  
 for out\_channels in self.encoder\_list:  
 block = self.add\_sublayer('encoder\_{}'.format(out\_channels),  
 Encoder(in\_channels, out\_channels))  
 self.encoders.append(block)  
 in\_channels = out\_channels  
 self.decoders = []  
  
 # 根据上采样个数和配置循环定义子Layer，避免重复写一样的程序  
 for out\_channels in self.decoder\_list:  
 block = self.add\_sublayer('decoder\_{}'.format(out\_channels),  
 Decoder(in\_channels, out\_channels))  
 self.decoders.append(block)  
 in\_channels = out\_channels  
  
 self.output\_conv = paddle.nn.Conv2D(in\_channels,  
 num\_classes,  
 kernel\_size=3,  
 padding='same')  
 def forward(self, inputs):  
 y = self.conv\_1(inputs)  
 y = self.bn(y)  
 y = self.relu(y)  
  
 for encoder in self.encoders:  
 y = encoder(y)  
  
 for decoder in self.decoders:  
 y = decoder(y)  
 y = self.output\_conv(y)  
 return y  
  
num\_classes = 4  
network = PetNet(num\_classes)  
model = paddle.Model(network)  
model.summary((-1, 3,) + IMAGE\_SIZE)  
  
train\_dataset = PetDataset(mode='train') # 训练数据集  
val\_dataset = PetDataset(mode='test') # 验证数据集  
  
optim = paddle.optimizer.RMSProp(learning\_rate=0.001,  
 rho=0.9,  
 momentum=0.0,  
 epsilon=1e-07,  
 centered=False,  
 parameters=model.parameters())  
model.prepare(optim, paddle.nn.CrossEntropyLoss(axis=1))  
model.fit(train\_dataset,  
 epochs=15,  
 batch\_size=32,  
 verbose=1)  
  
predict\_dataset = PetDataset(mode='predict')  
predict\_results = model.predict(predict\_dataset)  
  
plt.figure(figsize=(10, 10))  
  
i = 0  
mask\_idx = 0  
  
with open('./datasets/Oxford-IIIT Pet/predict.txt', 'r') as f:  
 for line in f.readlines():  
 image\_path, label\_path = line.strip().split('\t')  
 resize\_t = T.Compose([  
 T.Resize(IMAGE\_SIZE)  
 ])  
 image = resize\_t(PilImage.open(image\_path))  
 label = resize\_t(PilImage.open(label\_path))  
  
 image = np.array(image).astype('uint8')  
 label = np.array(label).astype('uint8')  
  
 if i > 8:  
 break  
 plt.subplot(3, 3, i + 1)  
 plt.imshow(image)  
 plt.title('Input Image')  
 plt.axis("off")  
  
 plt.subplot(3, 3, i + 2)  
 plt.imshow(label, cmap='gray')  
 plt.title('Label')  
 plt.axis("off")  
 # 模型只有一个输出，所以通过predict\_results[0]来取出预测的结果  
 # 映射原始图片的index来取出预测结果，提取mask进行展示  
 data = predict\_results[0][mask\_idx][0].transpose((1, 2, 0))  
 mask = np.argmax(data, axis=-1)  
  
 plt.subplot(3, 3, i + 3)  
 plt.imshow(mask.astype('uint8'), cmap='gray')  
 plt.title('Predict')  
 plt.axis("off")  
 i += 3  
 mask\_idx += 1  
plt.show()