第1章 概论

课后练习

一、正误题

1. 符号主义致力于用计算机的符号操作来模拟人的认知过程其，实质就是模拟人的左脑抽象逻辑思维，通过研究人类认知系统的功能机理，用某种符号来描述人类的认知过程，并把这种符号输入到能处理符号的计算机中，从而模拟人类的认知过程，实现人工智能。（T）
2. IBM 的深蓝，主要通过博弈论算法，打败人类国际象棋冠军。（T）
3. 人工神经网络是符号主义的典型代表性技术。（F）
4. 行为主义的贡献，主要在机器人控制系统方面。（T）
5. 123Python 是合法变量（F）
6. python 与 Python 是相同的变量（F）

二、选择题

1. 目前发展势头最猛、风头最盛的（A）、深度神经网络，即属于联结主义。

A. 深度学习 B. 知识图谱 C. 专家系统

2. （B）是 Python 编程语言及其数值数学扩展包 （A）的可视化操作界面。它为利用通用的图形用户界面工具包，如 Tkinter, wxPython, Qt 或 GTK+ 向应用程序嵌入式绘图提供了应用程序接口。

A. NumPy B. Matplotlib C. Scikit-learn

4. Python 编程语言受到广泛欢迎的原因是（ABC）

A. 简单的语法和更少的编码 B. 内置 AI 项目库 C. 开源

三、编程题

1.尝试利用 python 实现电脑随机生成 1~100 之间的整数，让用户来猜，猜错时，会提示

猜的数字是大了还是小了，直到用户猜对为止，游戏结束。

```
import random
computer=random.randint(1,100)
while True:
number=int(input("请输入 100 以内的整数："))
if(number>computer):
print("大了")
elif(number<computer):
print("小了")
else:
print("恭喜你赢了")
break
```

2. 尝试利用 python 获取 100 以内的质数。

```
num=[];
i=2
for i in range(2,100):
  j=2
  for j in range(2,i):
    if(i%j==0):
      break
  else:
    num.append(i)
print(num)
```

第2章 搜索的基本策略

课后练习

一、正误题

1. 启发式搜索中，通常 OPEN 表上的节点按照他们的 f 函数值递减顺序排列（F）

2. 启发式搜索是一种利用启发式信息的搜索（T）

3. 图搜索算法中，CLOSE 表用来登记待考察的节点（F）

4. 如果搜索是以接近起始节点的程度依次扩展节点的，那么这种搜索就叫做宽度优先搜索（T）

5. Dijkstra 算法是典型的宽度优先搜索算法，该算法路径最短、效率最高（F）

6. 在进行有序搜索法中，此时 Open 表是一个按节点的启发估价函数值的大小为序排列 的一个优先队列（T）

7. 启发式搜索一定比盲目式搜索好（F）

8. 宽度优先搜索策略是一种完整的搜索策略，只要问题有解，就能找到解（T）

9. 深度优先搜索策略可能找不到最优解，也可能根本找不到解（T）

10.因为估值函数最优，所以 A*算法是最优的 A 算法（T）

二、选择题

1.下面属于盲目搜索的方法有（AB）

A.深度优先搜索 B.宽度优先搜索 C.有序搜索算法 D.启发式搜索

2.如果重排 OPEN 表是依据发 f(x)=g(x)+h(x)进行的，则称该过程为（B）

A. A*算法 B.A 算法 C.有序搜索 D.启发式搜索

3.A*算法是一种（ABD）

A. 图搜索策略 B.有序搜索策略 C.盲目搜索 D.启发式搜索

4. 应用某个算法选择 OPEN 表上具有最小 f 值的节点作为下一个要扩展的节点。这种搜索方法的算法就叫做（C）

A.盲目搜索 B.宽度优先搜索策略 C.有序搜索算法 D.极小极大分析法

5.宽度优先搜索方法能够保证在搜索树中找到一条通往目标节点（B）途径（如果有路径存在时）。

A.可行 B.最短 C.最长 D.解答

6.如果问题存在最优解，则下面几种搜索算法中，（B）必然可以得到该最优解。

A.深度优先搜索 B.宽度优先搜索 C.有序搜索算法 D.启发式搜索

7.下列哪种搜索算法中是利用问题拥有的启发信息来引导搜索，达到减少搜索范围、降低问题复杂度的目的（D）

A.深度优先搜索 B.宽度优先搜索 C.盲目搜索算法 D.启发式搜索

8.根据估价函数值，按由小到大的次序对 Open 表中的节点进行重新排序的搜索方法叫做（CD）

A.深度优先搜索 B.宽(广)度优先搜索 C.有序搜索算法 D.启发式搜索

9.Dijkstra 算法的特点有（BCD）

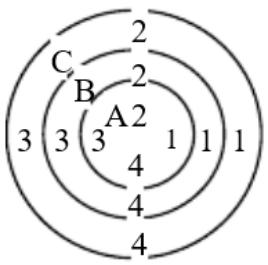
A.效率高 B.效率低 C.路径最短 D.复杂度高

10.下面属于宽度优先搜索和深度优先搜索具有的相同点是（ABD）

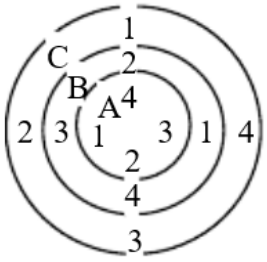
A.属于最短路径算法 B.效率低 C.搜索路线 D.复杂度高

三、编程题

1. 设有大小不等的三个圆盘 A、B、C 套在一根轴上，每个盘子上都标有数字 1、2、3、4，并且每个圆盘都可以独立的绕轴做逆时针转动，每次转动 90°，其初始状态 S_0 和目标状态 S_g 如下图所示，请用宽度优先搜索策略，求出 S_0 到 S_g 的路径。



初始状态 S0

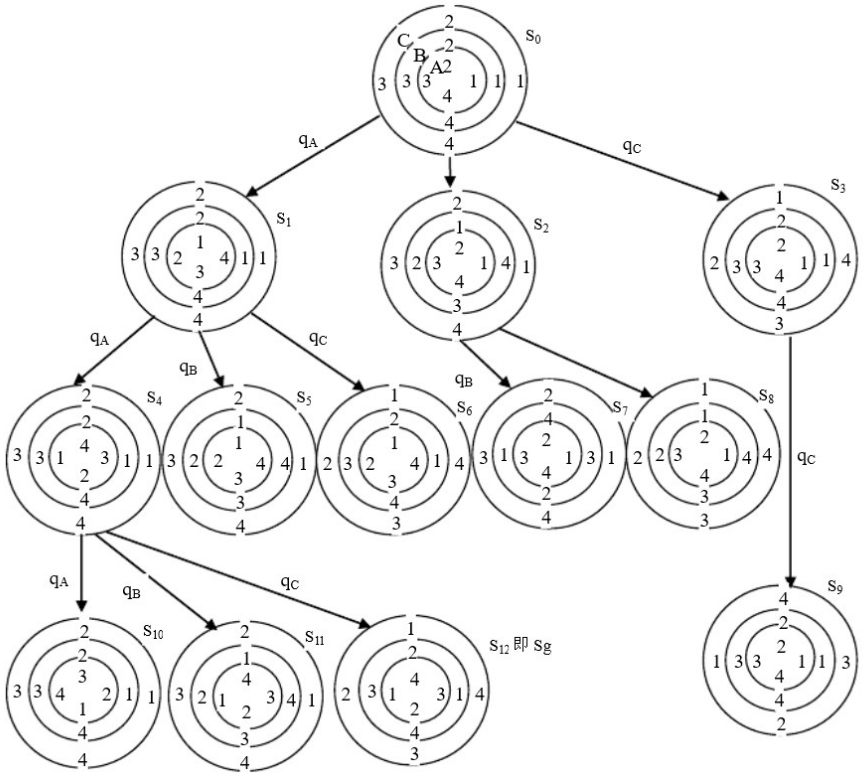


目标状态 Sg

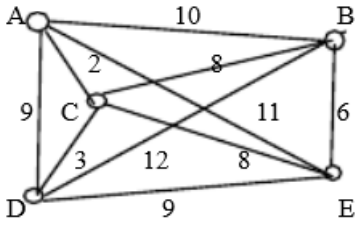
解：设用 q_A ， q_B 和 q_C 分别表示把 A 盘，B 盘和 C 盘绕轴逆时针转动 90° 这些操作（算符）的排列顺序是 q_A ， q_B ， q_C 。应用宽（广）度优先搜索，可得如下搜索树。在该搜索树中，重复出现的状态不再划出，节点旁边的标志 $S_i, i=0,1,2\cdots,$ 为按节点被扩展的顺序给出的该节点的状态标志。

由该图可以看出，从初始状态 S0 到目标状态 Sg 的路径是

S0→S1→S4→S12（Sg）

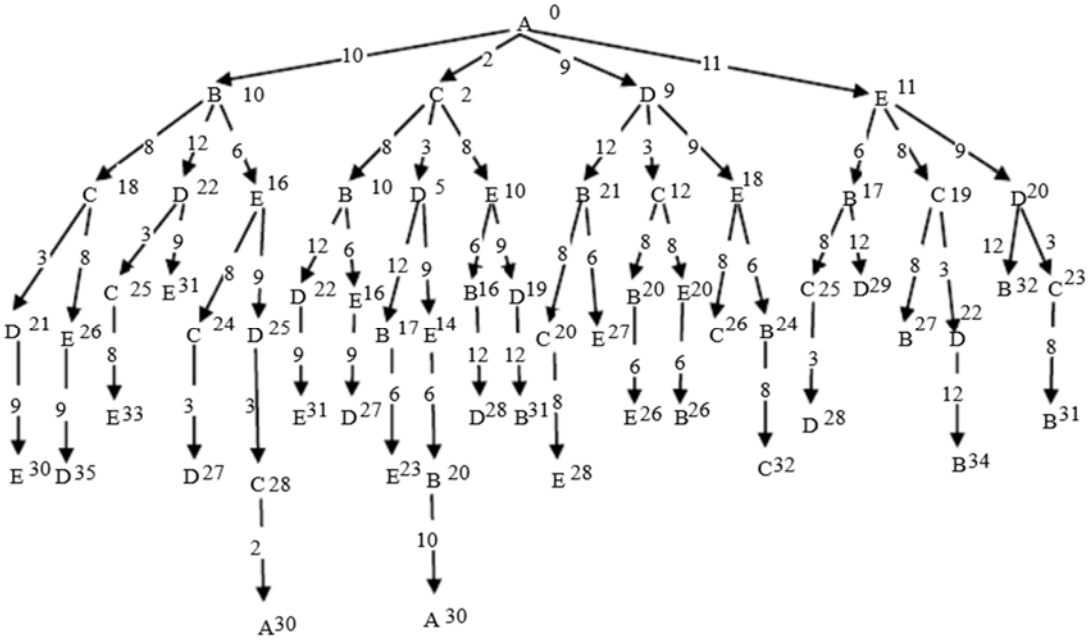


2. 下图是 5 个城市的交通图，城市之间的连线旁边的数字是城市之间路途的费用。要求从 A 城出发，经过其他各城市一次且仅一次，最后回到 A 城，请找出一条最优路线。



解：这个问题又称为旅行商问题(travelling salesman problem, TSP)或货郎担问题，是一个较有普遍性的实际应用问题。根据数学理论，对 n 个城市的旅行商问题，其封闭路径的排列总数为：(n!)/n=(n-1)! ，其计算量相当大。例如，当 n=20 时，要穷举其所有路径，即使用一个每秒一亿次的计算机来算也需要 350 年的时间。因此，对这类问题只能用搜索的方法来解决。

下图是按最小代价搜索所得到的搜索树，树中的节点为城市名称，节点边上的数字为该节点的代价。其计算公式为 $f(n_{i+1}) = g(n_i) + h(n_i, n_{i+1})$ 其中， $h(n_i, n_{i+1})$ 为节点 n_i 到 n_{i+1} 节点的边代价。



可以看出，其最短路经是 A-C-D-E-B-A 或 A-B-E-D-C-A 其实，它们是同一条路经。

第3章 搜索的高级策略

课后练习

一、正误题

1. 蚁群算法是一种应用于组合优化问题的启发式搜索算法。（T）
2. 蚁群算法是通过人工模拟蚂蚁搜索食物的过程，即通过个体之间的信息交流与相互协作最终找到从蚁穴到食物源的最短路径的。（T）
3. 蚁群算法中，蚂蚁选择路径的原理是一种负反馈机制。（F）
4. 蚂蚁系统是一种增强型学习系统。（T）
5. 动态规划阶段的顺序不同，则结果不同。（F）
6. 状态是由决策确定的。（F）
7. 用逆序法求解动态规划问题的重要基础之一是最优性原理。（T）
8. 列表法是求解某些离散变量动态规划问题的有效方法。（T）

二、选择题

1. 关于蚁群算法，说法不正确的是(A)

   A. 是最简单的进化算法　　B. 是群体智能的典型算法

   C. 模拟了蚂蚁群体解决问题的方法　D.其灵感来源于蚁群在觅食过程中，总能找到蚁穴到食物之间最短路径的现象。

2. 以下关于蚁群算法说法正确的是(ABCD)

   A. 蚁群算法是一种自组织的算法；

   B. 蚁群算法是一种本质上并行的算法；

   C. 蚁群算法是一种正反馈的算法；

   D. 蚁群算法具有较强的鲁棒性。

3. 以下关于蚁群算法的说法，错误的是(A)

   A. 蚁群算法用于预测蚁群在给定条件下的行为；

   B. 蚁群算法是从生物智能现象抽象出的算法；

   C. 蚁群算法属于群体智能；

   D. 对蚁群觅食的模拟要考虑到环境、选择路线等等多种因素。

4. 在蚁群算法中，信息素启发因子反映了蚁群在路径搜索中随机性因素的作用。其值越大，则(BD)

   A. 蚂蚁选择以前走过的路径的可能性就越大；

   B. 蚂蚁选择以前走过的路径的可能性就越小；

   C. 算法搜索的随机性越大，搜索的收敛速度会加快。

   D. 算法搜索的随机性越小，搜索结果易陷于局部最优。

5. 关于动态规划算法的特点，不正确的是(B)

   A. 使用最优化原理

   B. 完备搜索

   C. 无后效性

   D. 有重叠子问题

6. 动态规划法求解问题包括(ABCD)阶段。

   A. 划分阶段

   B. 确定状态和状态变量

   C. 确定决策并写出状态转移方程

   D. 寻找边界条件

7. 动态规划方法的解题步骤不包括(C)

   A. 分析最优解的性质，并刻画其结构特征；

   B. 递归的定义最优解；

C. 确定子过程指标函数的具体形式；

D. 根据计算最优值时得到的信息，构造问题的最优解。

三、编程题

1. 柔性作业车间调度问题：某加工系统由 6 台机床，要加工 4 个工件，每个工件有 3 道工序，如下表所示。比如工件 p_{11} 代表第一个工件的第一道工序，可由机床 1 用 2h 完成，或由机床 2 用 3h 完成，或用机床 3 用 4h 完成。使用蚁群算法，优化最大完工时间。

工序选择		加工机床及加工时间					
		1	2	3	4	5	6
J_1	p_{11}	2	3	4			
	p_{12}		3		2	4	
	p_{13}	1	4	5			
J_2	p_{21}	3		5		2	
	p_{22}	4	3		6		
	p_{23}			4		7	11
J_3	p_{31}	5	6				
	p_{32}		4		3	5	
	p_{33}			13		9	12
J_4	p_{41}	9		7	9		
	p_{42}		6		4		5
	p_{43}	1		3			3

```
"""
柔性作业车间调度问题
(Flexible Job-shop Scheduling Problem, FJSP)
"""
import numpy as np
import random
from typing import List
from matplotlib import pyplot as plt

plt.rcParams['font.family'] = ['sans-serif']
plt.rcParams['font.sans-serif'] = ['SimHei']

# 作业数，统一工序数，机器数
```

```
job_num = 4
process_num = 3
machine_num = 6

# 4 个 Job 的 3 个工序在 6 台机器上的加工时间
times = [
    [
        [2, 3, 4, None, None, None],
        [None, 3, None, 2, 4, None],
        [1, 4, 5, None, None, None]
    ],
    [
        [3, None, 5, None, 2, None],
        [4, 3, None, 6, None, None],
        [None, None, 4, None, 7, 11]
    ],
    [
        [5, 6, None, None, None, None],
        [None, 4, None, 3, 5, None],
        [None, None, 13, None, 9, 12]
    ],
    [
        [9, None, 7, 9, None, None],
        [None, 6, None, 4, None, 5],
        [1, None, 3, None, None, 3]
    ]
]

# 拓扑序的信息素浓度，初始值 100
topo_phs = [
    [100 for _ in range(job_num)]
    for _ in range(job_num * process_num)
]

def gen_topo_jobs() -> List[int]:
    """
    生成拓扑序
    Job 在时空上处理的的拓扑序(Job 索引)，这个序不能体现工序选择的机器
    :return 如[0,1,0,2,2,...]表示 p11,p21,p12,p31,p32,...
    """
    # 按照每个位置的信息素浓度加权随机给出
    # 返回的序列长，是 Job 数量*工序的数量
    len = job_num * process_num
    # 返回的序列，最后这些-1 都会被设置成 0~job_num-1 之间的索引
    ans = [-1 for _ in range(len)]
    # 记录每个 job 使用过的次数，用来防止 job 被使用超过 process_num 次
    job_use = [0 for _ in range(job_num)]
```

```
    # 记录现在还没超过 process_num 因此可用的 job_id，每次满了就将其删除
    job_free = [job_id for job_id in range(job_num)]
    # 对于序列的每个位置
    for i in range(len):
        # 把这个位置可用的 job 的信息素浓度求和
        ph_sum = np.sum(list(map(lambda j: topo_phs[i][j], job_free)))
        # 接下来要随机在 job_free 中取一个 job_id
        # 但是不能直接 random.choice，要考虑每个 job 的信息素浓度
        test_val = .0
        rand_ph = random.uniform(0, ph_sum)
        for job_id in job_free:
            test_val += topo_phs[i][job_id]
            if rand_ph <= test_val:
                # 将序列的这个位置设置为 job_id，并维护 job_use 和 job_free
                ans[i] = job_id
                job_use[job_id] += 1
                if job_use[job_id] == process_num:
                    job_free.remove(job_id)
                break
    return ans

# 每个 Job 的每个工序的信息素浓度，初始值 100
machine_phs = [
    [
        [100 for _ in range(machine_num)]
        for _ in range(process_num)
    ]
    for _ in range(job_num)
]

def gen_process2machine() -> List[List[int]]:
    """
    生成每个 Job 的每个工序对应的机器索引号矩阵
    :return: 二维 int 列表，如[0][0]=3 表示 Job1 的 p11 选择机器 m4
    """
    # 要返回的矩阵，共 job_num 行 process_num 列，取值 0~machine_num-1
    ans = [
        [-1 for _ in range(process_num)]
        for _ in range(job_num)
    ]
    # 对于每个位置，也是用信息素加权随机出一个 machine_id 即可
    for job_id in range(job_num):
        for process_id in range(process_num):
            # 获取该位置的所有可用机器号(times 里不为 None)
            machine_free = [machine_id for machine_id in range(machine_num)
                            if times[job_id][process_id][machine_id] is not None]
```

```
            # 计算该位置所有可用机器的信息素之和
            ph_sum = np.sum(list(map(lambda m: machine_phs[job_id][process_id][m], machine_free)))
            # 还是用随机数的方式选取
            test_val = .0
            rand_ph = random.uniform(0, ph_sum)
            for machine_id in machine_free:
                test_val += machine_phs[job_id][process_id][machine_id]
                if rand_ph <= test_val:
                    ans[job_id][process_id] = machine_id
                    break
    return ans

def cal_time(topo_jobs: List[int], process2machine: List[List[int]]) -> float:
    """
    给定拓扑序和机器索引号矩阵
    :return: 计算出的总时间花费
    """
    # 记录每个 job 在拓扑序中出现的次数，以确定是第几个工序
    job_use = [0 for _ in range(job_num)]
    # 循环中要不断查询和更新这两张表
    # (1)每个 machine 上一道工序的结束时间
    machine_end_times = [0 for _ in range(machine_num)]
    # (2)每个工件上一道工序的结束时间
    job_end_times = [0 for _ in range(job_num)]
    # 对拓扑序中的每个 job_id
    for job_id in topo_jobs:
        # 在 job_use 中取出工序号
        process_id = job_use[job_id]
        # 在 process2machine 中取出机器号
        machine_id = process2machine[job_id][process_id]
        # 获取 max(该 job 上一工序时间,该 machine 上一任务完成时间)
        now_start_time = max(job_end_times[job_id], machine_end_times[machine_id])
        # 计算当前结束时间，写入这两个表
        job_end_times[job_id] = machine_end_times[machine_id] = now_start_time +
times[job_id][process_id][machine_id]
        # 维护 job_use
        job_use[job_id] += 1
    return max(job_end_times)

# 迭代次数
iteration_num = 40

# 蚂蚁数量
ant_num = 30

# 绘图用
```

```
iter_list = range(iteration_num)
time_list = [0 for _ in iter_list]

best_topo_jobs = None
best_process2machine = None

# 对于每次迭代
for it in iter_list:
    # 每次迭代寻找最优的<拓扑序,机器分配>方式
    best_time = 9999999
    # 对于每只蚂蚁
    for ant_id in range(ant_num):
        # 生成拓扑序
        topo_jobs = gen_topo_jobs()
        # 生成每道工序的分配机器索引号矩阵
        process2machine = gen_process2machine()
        # 计算时间
        time = cal_time(topo_jobs, process2machine)
        # 如果时间更短，更新最优
        if time < best_time:
            best_topo_jobs = topo_jobs
            best_process2machine = process2machine
            best_time = time
    assert best_topo_jobs is not None and best_process2machine is not None
    # 更新拓扑序信息素浓度表
    for i in range(job_num * process_num):
        for j in range(job_num):
            if j == best_topo_jobs[i]:
                topo_phs[i][j] *= 1.1
            else:
                topo_phs[i][j] *= 0.9
    # 更新每个 Job 的每个工序的信息素浓度表
    for j in range(job_num):
        for p in range(process_num):
            for m in range(machine_num):
                if m == best_process2machine[j][p]:
                    machine_phs[j][p][m] *= 1.1
                else:
                    machine_phs[j][p][m] *= 0.9
    # 记录时间
    time_list[it] = best_time

# 输出解
print("\t\t[工序分配给机器的情况]")
print("\t", end='')
for machine_id in range(machine_num):
```

```
    print("\tM{}".format(machine_id + 1), end='')
print()
for job_id in range(job_num):
    for process_id in range(process_num):
        print("p{}{}\t".format(job_id + 1, process_id + 1), end='')
        for machine_id in range(machine_num):
            if machine_id == best_process2machine[job_id][process_id]:
                print("\t √ ", end='')
            else:
                print("\t-", end='')
        print("")

print("\n\t\t[工序投给机器的顺序]")
job_use = [0 for _ in range(job_num)]
for job_id in best_topo_jobs:
    print("p{}{} ".format(job_id + 1, job_use[job_id] + 1), end='')
    job_use[job_id] += 1

# 绘图
plt.plot(iter_list, time_list)
plt.xlabel("迭代轮次")
plt.ylabel("时间")
plt.title("柔性作业车间调度-蚁群算法")
plt.show()
```

2. 假设有 1 元，3 元，5 元的硬币若干（无限个），现在需要凑出 11 元，问如何组合才能使硬币的数量最少？请用动态规划方法解决这一问题。

```
# 动态规划思想 dp 方程式如下
# dp[0] = 0
# dp[i] = min{dp[i - coins[j]] + 1}, 且 其中 i >= coins[j], 0 <= j < coins.length
# 回溯法，输出可找的硬币方案
# path[i] 表示经过本次兑换后所剩下的面值，即 i - path[i] 可得到本次兑换的硬币值。

def changeCoins(coins, n):
    if n < 0: return None
    dp, path = [0] * (n + 1), [0] * (n + 1)  # 初始化
    for i in range(1, n + 1):
        minNum = i  # 初始化当前硬币最优值
        for c in coins:  # 扫描一遍硬币列表，选择一个最优值
            if i >= c and minNum > dp[i - c] + 1:
                minNum, path[i] = dp[i - c] + 1, i - c
        dp[i] = minNum  # 更新当前硬币最优值

    print('最少硬币数:', dp[-1])
    print('可找的硬币', end=': ')
```

```
    while path[n] != 0:
        print(n - path[n], end=' ')
        n = path[n]
    print(n, end=' ')

if __name__ == '__main__':
    coins, n = [1, 3, 5], 11  # 输入可换的硬币种类，总金额 n
    changeCoins(coins, n)
```

第4章 概念学习和决策树

课后练习

一、正误题

1. 概念学习可以看作是一个搜索问题的过程。（T）
2. Finds 算法缺点是无法确定是否找到了唯一合适的假设，并且容易受到噪声的影响。（T）
3. 变型空间表示了目标概念的所有合理的变型。（T）
4. 候选消除算法无法输出与训练样例一致的所有假设集合。（F）
5. 决策树分析法是通过决策树图形展示重要结局，明确思路，比较各种备选方案预期结果进行决策的方法。（T）
6. 决策树只能进行单级决策。（F）

二、选择题

1. 一个支持概念学习的算法不需要包括以下哪个部分（C）

   A. 训练数据

   B. 目标概念

   C. 划分标准

   D. 实际数据对象

2. 使用 Finds 算法，根据如下判断性别的表格，假设所有男性为正例，得到一般假设为（A）

   A. <yes,?,?,low>

   B. <yes,?,high,low>

   C. <yes,short,?,low>

   D. <yes,short,high,low>

Example	Beard	Hair	Height	Tone	M/F
1	yes	short	short	low	M
2	yes	short	high	low	M

3	no	long	short	high	F
4	yes	long	high	low	M

3. 下列关于候选消除算法，说法不正确的是（D）

A. 候选消除算法能够表示与训练样例一致的所有假设。

B. 关于假设空间 H 和训练数据 D 的一般边界 G，是在 H 中与 D 相一致的最一般成员的集合。

C. 关于假设空间 H 和训练数据 D 的特殊边界 S，是在 H 中与 D 相一致的最特殊成员的集合。

D. 候选消除算法首先将 G 集合初始化为 H 中最特殊假设，将 S 集合初始化为 H 中最一般假设

4. 决策树的构成顺序是（A）

A. 特征选择、决策树生成、决策树剪枝。

B. 决策树剪枝、特征选择、决策树生成。

C. 决策树生成、决策树剪枝、特征选择。

D. 特征选择、决策树剪枝、决策树生成。

5. 决策树法是运用（B）来分析和选择决策方案的一种系统分析方法。

A．线形图

B．树状图

C．饼图

D．柱状图

三、编程题

1. 运用候选消除算法，根据下表中的物品的大小、颜色、形状各属性，预测是否需要购买。

Example	Size	Color	Shape	Purchase
1	big	red	circle	no
2	small	red	triangle	no
3	small	red	circle	yes

4	big	blue	circle	no
5	small	blue	circle	no

```
import numpy as np
import csv

class CSVReader:
    @staticmethod
    def get_csv_as_2d_array(filename):
        datafile = open(filename, 'r')
        datareader = csv.reader(datafile, delimiter=',')
        data = []
        for row in datareader:
            data.append(row)
        return data

class CandidateElimination:
    def __init__(self, table):
        self.table = table
        self.n = len(table)
        self.m = len(table[0])
        self.s = []
        self.g = [[]]
        self.g0 = []
        for i in range(self.m - 1):
            self.s.append("0")
            self.g0.append("?")
        self.g[0] = self.g0

    def start_calculate(self):
        print(self.table)
        self.print_s_and_g(0)
        for i in range(self.n):
            if self.is_positive(i):
                self.do_action_for_positive(i)
            else:
                self.do_action_for_negative(i)
            self.print_s_and_g(i + 1)

    def print_s_and_g(self, i):
        print("S ", i, " =", self.s)
        print("G ", i, " =", self.g)
        print()

    def is_positive(self, raw):
        if self.table[raw][len(self.table[0]) - 1] == "yes":
            return True
        else:
```

```
        return False

    def do_action_for_positive(self, raw):
        self.generate_s(raw)
        self.clean_g()

    def clean_g(self):
        i = 0
        while i < len(self.g):
            for j in range(self.m - 1):
                if self.s[j] != self.g[i][j] and self.g[i][j] != "?":
                    self.g.pop(i)
                    i -= 1
                    break
            i += 1

    def generate_s(self, raw):
        for j in range(self.m - 1):
            if self.s[j] == "0":
                self.s[j] = self.table[raw][j]
            elif self.s[j] != self.table[raw][j]:
                self.s[j] = "?"

    def do_action_for_negative(self, raw):
        self.generate_g(raw)

    def generate_g(self, raw):
        change = False
        g = []
        for i in range(len(self.g)):
            for j in range(self.m - 1):
                if self.table[raw][j] != self.s[j] and self.s[j] != "?":
                    if self.is_empty(self.g[i]):
                        new_g = self.create_new_g(j, raw)
                        g.append(new_g)
                        change = True
                    elif self.table[raw][j] == self.g[i][j]:
                        for k in range(0, self.m - 1):
                            if k != j:
                                new_g = self.create_new_g(k, raw)
                                new_g[j] = self.g[i][j]
                                g.append(new_g)
                                change = True
                        break
                elif self.table[raw][j] == self.s[j] and self.table[raw][j] == self.g[i][j]:
                    break
                elif self.not_contradiction(self.g[i], self.table[raw]) and self.not_dont_care(self.g[i]):
                    g.append(self.g[i])
```

```
                change = True
                break

        if change:
            self.g = g

    def create_new_g(self, j, raw):
        new_g = []
        for i in range(j):
            new_g.append("?")
        except_arr = self.get_except(raw, j)
        if self.s[j] == "?" or self.s[j] == "0":
            while len(except_arr) > 0:
                new_g.append(except_arr.pop())
        else:
            new_g.append(self.s[j])
        for i in range(j + 1, self.m - 1):
            new_g.append("?")
        return new_g

    def get_except(self, raw, j):
        set_of_choice = self.get_set_of_column_choice(j)
        set_of_choice.remove(self.table[raw][j])
        return set_of_choice

    def get_set_of_column_choice(self, j):
        set_of_column = set()
        for i in range(self.n):
            set_of_column.add(self.table[i][j])
        return set_of_column

    def not_contradiction(self, g, table):
        for i in range(len(g)):
            if g[i] != "?" and g[i] == table[i]:
                return False
        return True

    def is_empty(self, g):
        for i in range(len(g)):
            if g[i] != "?":
                return False
        return True

    def not_dont_care(self, g):
        for i in range(len(g)):
            if g[i] != "?":
                return True
        return False
```

```
def main():
    table = [['big','red','circle','no'],
             ['small','red','triangle','no'],
             ['small','red','circle','yes'],
             ['big','blue','circle','no'],
             ['small','blue','circle','yes']]
    candidate_elimination_obj = CandidateElimination(table)
    candidate_elimination_obj.start_calculate()

main()
```

2. 运用决策树算法，根据下列动物数据表，判断动物是否为鱼类。其中 no surfacing 指不浮出水面能否生存，flipper 指是否有脚。

Example	No surfacing	Flipper	Fish
1	yes	no	yes
2	yes	no	yes
3	yes	yes	no
4	no	no	no
5	no	no	no

```
from math import log
import operator

#生成样本数据集
def createDataSet():
    dataSet = [[1,1,'yes'],
               [1,1,'yes'],
               [1,0,'no'],
               [0,1,'no'],
               [0,1,'no']]
    labels = ['no surfacing','flipper']
    return dataSet,labels

#  no surfacing 指的是 不浮出水面能否生存 1 标识 是 0 指的是否
# flipper 指的是是否有脚
#  yes  no 指的是否是鱼类
def calcShannonEnt(dataSet):
    numEntries = len(dataSet) # 用上面的 createDataSet dataSet 这个值就是 5
    #定义标签字典
    labelCounts = {}

    # 为所有可能的分类创建字典
    for featVec in dataSet:
```

```
        currentLabel = featVec[-1]  #这个-1 指的是去取最后一个维度 对应数据 dataSet 这里取的是 yes 和 no
        if currentLabel not in labelCounts.keys():
            # 如果当前分类标签不在 标签字典中
            labelCounts[currentLabel] = 0
        # 其他情况 分类标签分类加 1
        labelCounts[currentLabel] += 1
    #定义香农熵 以 2 为底数求对数
    shannonEnt = 0.0
    for key in labelCounts:
        #计算 yes 或者 No 出现的概率
        pro = float(labelCounts[key])/numEntries
        # 计算香农熵
        shannonEnt -= pro*log(pro,2)
    return shannonEnt

#dataSet 是待划分的数据集， 划分数据集的特征 axis 特征的返回值 value
#最后是创建了一个新的列表对象
def splitDataSet(dataSet, axis , value):
    # 创建新 list 对象
    retDataSet = []
    for featVec in  dataSet:
        if featVec[axis] == value:
            reducedFeatVec = featVec[:axis]
            reducedFeatVec.extend(featVec[axis+1:])
            retDataSet.append(reducedFeatVec)
    return retDataSet

# 选择最好的特征值进行数据集划分
def chooseBestFeatureToSplit(dataSet):
    # len(dataSet[0]）是计算这一行有多少列，即有多少个特征值
    numFeatures = len(dataSet[0])-1 # -1 是最后一个特征值就不要记录在内了，算 baseEntrop 的时候已经算了
最后一个特征值 yes no
    baseEntropy = calcShannonEnt(dataSet)
    bestInfoGain = 0.0
    bestFeature = -1
    for i in range(numFeatures):
        #创建唯一的分类标签列表 也就是说提取 dataSet 每一行第 i 个值 就提取 dat
        featList = [example[i] for example in dataSet]
        # 取出有几种特征值
        uniqueVals = set(featList)
        newEntropy = 0.0
        for value in uniqueVals:
            #创建特征值的子数据集
            subDataSet = splitDataSet(dataSet,i, value)
            #计算该特征值数据对总数在数据对总数出现的概率
            pro = len(subDataSet)/float(len(dataSet))
```

```
            #计算分割出来的子集香农熵
            newEntropy += pro*calcShannonEnt(subDataSet)
        #计算信息增益 得到最好的特征值 这个理论是这样的 g(D,A) = H(D)-H(D/A）
        infoGain = baseEntropy-newEntropy
        #取出最大的信息增益，此时特征值最大
        if(infoGain >bestInfoGain):
            bestInfoGain = infoGain
            bestFeature = i
    return bestFeature

'''
#构建决策树是根据特征值的消耗来计算的，如果后面的特征值已经全部用完了
但是还没有分出结果，这个时候就需要使用多数表决方式计算节点分类
最后返回最大的分类
'''
def majorityCnt(classList):
    # 分类的字典
    classCount = {}
    for vote in range(classList):
        #如果不在 分类字典中
        if vote not in classCount.keys(): classCount[vote] = 0
        classCount[vote] += 1
        # 根据出现的次数大到小排序
        sortedClassCount = sorted(classCount.items(),key=operator.itemgetter(1),reverse=True)
    return sortedClassCount[0][0]

#创建决策树
def createTree(dataSet, labels):
    # 获取数据样本每组最后一组的特征值 这里是 yes,no
    classList = [example[-1] for example in dataSet]
    # 如果说这个 classList 全部都是 yes 或者全部是 no 那肯定子返回 yes 或者 no
    if(classList.count(classList[0]) == len(classList)):
        return classList[0]
    #如果遍历完所有的特征返回出现次数最多的
    #是用消耗特征值的方式进行构造决策树的，每次会消掉一个特征值
    if len(dataSet[0]) == 1:
        return majorityCnt(classList)
    #选择最好的特征值
    bestFeat = chooseBestFeatureToSplit(dataSet)
    bestFeatLabel = labels[bestFeat]
    myTree = {bestFeatLabel:{}}
    # 删除 labels 中的一特征值
    del(labels[bestFeat])
    #找到特征值那一列
    featValues = [example[bestFeat] for example in dataSet]
    uniqueVals = set(featValues)
```

```
    for value in uniqueVals:
        # labels 列表的赋值
        subLabels = labels[:]
        myTree[bestFeatLabel][value]=createTree(splitDataSet(dataSet,bestFeat,value),subLabels)
    return myTree

dataSet,lables = createDataSet()
shannonEnt= calcShannonEnt(dataSet)
my = createTree(dataSet,lables)
print(my)
```

第5章 线性回归和分类

课后练习

一、正误题

1. 线性回归方法可以适应弥散高的数据集。（F）

2. 逻辑回归实际上是一种基于概率来判断样本属于哪一类的分类方法。（T）

3. 使用 ElasticNet 方式，所得到的模型将像纯粹的 Lasso 回归一样稀疏，但同时具有与岭回归提供的正则化能力。（T）

4.回归问题和分类问题都有可能发生过拟合。（T）

5.给定 n 个数据点，如果其中一半用于训练，另一半用于测试，则训练误差和测试误差之间的差别会随着 n 的增加而减小。（T）

二、选择题

1. Lasso 回归加入 *w* 的(B)范数作为惩罚项，以确定系数中的数目较多的无效项。

A. L1 B. L2 C.L3

2. 向量 x=[1,2,3,4,-9,0]的 L1 范数是多少？(B)

A. 1 B.19 C.6 D.sqrt(111)

3.关于 L1 正则和 L2 正则，下面说法正确的是(BD)

A.L2 范数可以防止过拟合，提升模型的泛化能力，但 L1 做不到这点

B.L2 正则化标识各个参数的平方的和的开方值

C. L2 正则化又叫“Lasso regularization”

D.L1 范数会使权值稀疏

4.使用带有 L1 正则化的 logistic 回归做二分类，其中 C 是正则化参数，w1 和 w2 是 x1 和 x2 的系数。当你把 C 值从 0 增加至非常大的值时，下面哪个选项是正确的？(B)

A. 第一个 w2 成了 0，接着 w1 也成了 0

B. 第一个 w1 成了 0，接着 w2 也成了 0

C. w1 和 w2 同时成了 0

D. 即使在 C 成为大值之后，w1 和 w2 都不能成 0

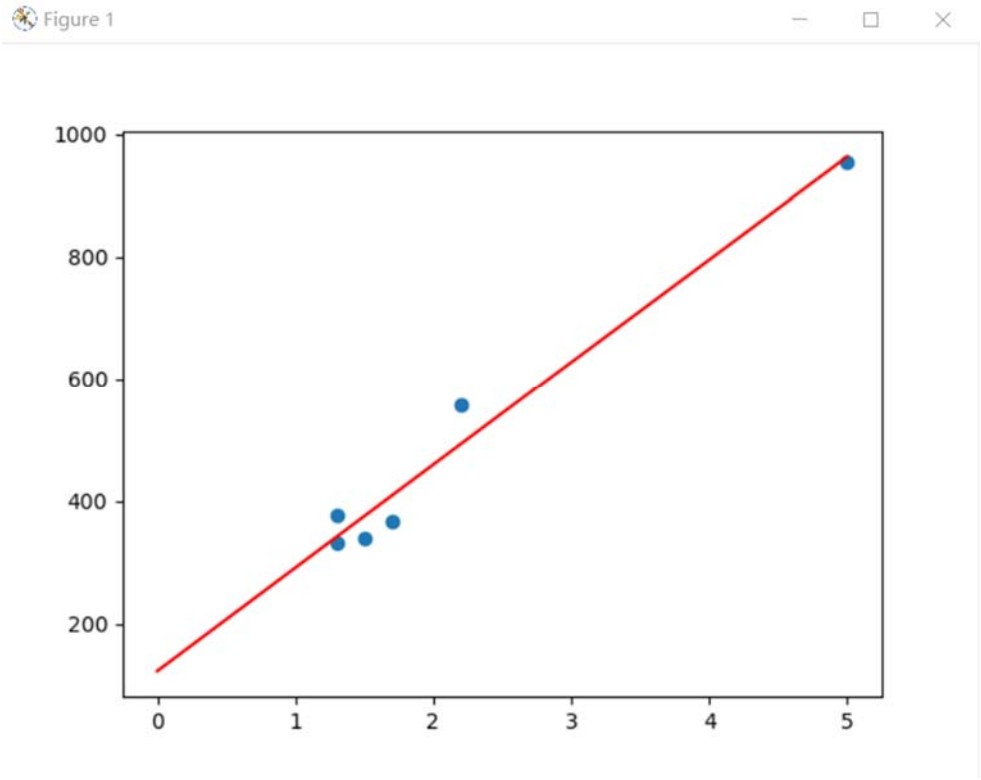
三、编程题

1. 根据在线广告投入费用（见下表）预测每月电子商务销售量

在线广告费用	每月电子商务销售量
1.7	368
1.5	340
1.3	376
5	954
1.3	331
2.2	556
2.8	?

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

y_train = np.array([368, 340, 376, 954,331, 556])  #输入数据集
X_train = np.array([1.7, 1.5, 1.3, 5, 1.3, 2.2])  #输出数据集
plt.scatter(X_train, y_train, label= 'Train Samples')  #描绘数据

X_train = X_train.reshape(-1, 1)  #改变输入数据维度
lr = LinearRegression()  #学习预测
lr.fit(X_train, y_train)  #训练数据

a=range(6)   #利用预测出的斜率和截距绘制直线
b=[lr.intercept_+lr.coef_[0]*i for i in a]

plt.plot(a,b, 'r')

print(lr.intercept_)#打印出截距
print(lr.coef_)  #打印出斜率

plt.show()
```

程序运行效果：



预测当? = 2.8 时的销量：? = 123.94 + 167.80 ∗ 2.8 = 593.78

2. 使用 Logistic 回归来预测患有疝病的马的存活问题。

病马的训练数据已经给出来了，如下形式存储在文本文件中:

2.000000 1.000000 38.500000 66.000000 28.000000 3.000000 3.000000 0.000000 2.000000 5.000000 4.000000 4.000000 0.000000 0.000000 0.000000 3.000000 5.000000 45.000000 8.400000 0.000000 0.000000 0.000000

1.000000 1.000000 39.200000 88.000000 20.000000 0.000000 0.000000 4.000000 1.000000 3.000000 4.000000 2.000000 0.000000 0.000000 0.000000 4.000000 2.000000 50.000000 85.000000 2.000000 2.000000 0.000000

2.000000 1.000000 38.300000 40.000000 24.000000 1.000000 1.000000 3.000000 1.000000 3.000000 3.000000 1.000000 0.000000 0.000000 0.000000 1.000000 1.000000 33.000000 6.700000 0.000000 0.000000 1.000000

1.000000 9.000000 39.100000 164.000000 84.000000 4.000000 1.000000 6.000000 2.000000 2.000000 4.000000 4.000000 1.000000 2.000000 5.000000 3.000000 0.000000 48.000000 7.200000 3.000000 5.300000 0.000000

2.000000 1.000000 37.300000 104.000000 35.000000 0.000000 0.000000 6.000000 2.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 74.000000 7.400000 0.000000 0.000000 0.000000

2.000000 1.000000 0.000000 0.000000 0.000000 2.000000 1.000000 3.000000 1.000000 2.000000 3.000000 2.000000 2.000000 1.000000 0.000000 3.000000 3.000000 0.000000 0.000000 0.000000 0.000000 1.000000

1.000000 1.000000 37.900000 48.000000 16.000000 1.000000 1.000000 1.000000 1.000000 3.000000 3.000000 3.000000 1.000000 1.000000 0.000000 3.000000 5.000000 37.000000 7.000000 0.000000 0.000000 1.000000

完整代码：

```
import numpy as np
import random

# sigmoid 阶跃函数
def sigmoid(inX):
    # return 1.0 / (1 + exp(-inX))
    return 1.0/(1+np.exp(-inX))


# 随机梯度上升算法（随机化）
def stocGradAscent1(dataMatIn, classLabels, numIter=150):
    m,n = np.shape(dataMatIn)
    # 创建与列数相同的矩阵的系数矩阵，1 行 3 列
    weights = np.ones(n)
    # 随机梯度, 循环 150,观察是否收敛
    for j in range(numIter):
        # [0, 1, 2 .. m-1]
        dataIndex = list(range(m))
        for i in range(m):
            # i 和 j 的不断增大，导致 alpha 的值不断减少，但是不为 0
            alpha = 4/(1.0+j+i)+0.01    # alpha 会随着迭代不断减小，但永远不会减小到 0，因为后边还有一个常数项 0.0001
            # 随机产生一个 0～len()之间的一个值
            # random.uniform(x, y) 方法将随机生成下一个实数，它在[x,y]范围内,x 是这个范围内的最小值，y 是这个范围内的最大值。
            randIndex = int(random.uniform(0,len(dataIndex)))
            # sum(dataMatrix[i]*weights)为了求 f(x)的值， f(x)=a1*x1+b2*x2+..+nn*xn
            h = sigmoid(sum(dataMatIn[dataIndex[randIndex]]*weights))
            error = classLabels[dataIndex[randIndex]] - h
            weights = weights + alpha * error * dataMatIn[dataIndex[randIndex]]
            del(dataIndex[randIndex])
    return weights


# 分类函数，根据回归系数和特征向量来计算 Sigmoid 的值
def classifyVector(inX, weights):
    '''
    Desc:
        最终的分类函数，根据回归系数和特征向量来计算 Sigmoid 的值，大于 0.5 函数返回 1，否则返回 0
    Args:
```

```
        inX -- 特征向量，features
        weights -- 根据梯度下降/随机梯度下降 计算得到的回归系数
    Returns:
        如果 prob 计算大于 0.5 函数返回 1
        否则返回 0
    '''
    prob = sigmoid(sum(inX*weights))
    if prob > 0.5: return 1.0
    else: return 0.0

# 打开测试集和训练集,并对数据进行格式化处理
def colicTest():
    '''
    Desc:
        打开测试集和训练集，并对数据进行格式化处理
    Args:
        None
    Returns:
        errorRate -- 分类错误率
    '''
    frTrain = open('horseColicTraining.txt')
    frTest = open('horseColicTest.txt')
    trainingSet = []
    trainingLabels = []
    # 解析训练数据集中的数据特征和 Labels
    # trainingSet 中存储训练数据集的特征，trainingLabels 存储训练数据集的样本对应的分类标签
    for line in frTrain.readlines():
        currLine = line.strip().split('\t')
        lineArr = []
        for i in range(21):
            lineArr.append(float(currLine[i]))
        trainingSet.append(lineArr)
        trainingLabels.append(float(currLine[21]))
    # 使用 改进后的 随机梯度下降算法 求得在此数据集上的最佳回归系数 trainWeights
    trainWeights = stocGradAscent1(np.array(trainingSet), trainingLabels, 500)
    errorCount = 0
    numTestVec = 0.0
    # 读取 测试数据集 进行测试，计算分类错误的样本条数和最终的错误率
    for line in frTest.readlines():
        numTestVec += 1.0
        currLine = line.strip().split('\t')
        lineArr = []
        for i in range(21):
            lineArr.append(float(currLine[i]))
        if int(classifyVector(np.array(lineArr), trainWeights)) != int(currLine[21]):
            errorCount += 1
```

```
    errorRate = (float(errorCount) / numTestVec)
    print ("the error rate of this test is: %f" % errorRate)
    return errorRate

# 调用 colicTest() 10 次并求结果的平均值
def multiTest():
    numTests = 10
    errorSum = 0.0
    for k in range(numTests):
        errorSum += colicTest()
    print ("after %d iterations the average error rate is: %f" % (numTests, errorSum/float(numTests)))

multiTest()
```

输出结果：

```
the error rate of this test is: 0.388060
the error rate of this test is: 0.582090
the error rate of this test is: 0.298507
the error rate of this test is: 0.253731
the error rate of this test is: 0.283582
the error rate of this test is: 0.313433
the error rate of this test is: 0.522388
the error rate of this test is: 0.283582
the error rate of this test is: 0.313433
the error rate of this test is: 0.283582
after 10 iterations the average error rate is: 0.352239
```

第6章 统计学习方法

课后练习[1]

一、问答题

1. 请写出贝叶斯公式，并描述朴素贝叶斯分类方法的原理和步骤。

答：贝叶斯公式：$P(A|B)=\frac{P(B|A)\cdot P(A)}{P(B)}$，朴素贝叶斯分类是一种十分简单的分类算法，它的思想基础是这样的：对于给出的待分类项，求解在此项出现的条件下各个类别出现的概率，哪个最大，就认为此待分类项属于哪个类别。

2. 支持向量机的基本思想是什么？

答：基本思想是拟合类别之间可能的、最宽的“街道”。它的目的是使决策边界之间最大间隔化，从而分隔出两个类别的训练实例。SVM 在执行软间隔分类时，实际上是在完美分类和拟合最宽街道之间进行妥协（允许少量的实例最终落在街道上）。在训练非线性数据集时，记得使用核函数。

3. 什么是支持向量？

答： SVM 训练完成后，位于边界上的实例被称为支持向量。决策边界完全由支持向量决定。非支持向量地实例则对决策边界没有任何影响。计算预测结果只涉及支持向量，而不涉及整个训练集。

4. 使用 SVM 时，对输入值进行缩放为什么重要？

答：SVM 拟合类别之间可能的、最宽的“街道”，所以如果训练集不经缩放，SVM 将趋于忽略值较小的特征。

5. SVM 分类器在对实例进行分类时，会输出信心分数么？概率呢？

答： SVM 分类时输出的是测试实例与决策边界的距离，也可以将其用作信心分数。但这个分数不能直接转换为类别概率的估算。如果创建 SVM 时，在 Scikit-Learn 中设置 probability=True，那么训练完成后，算法将使用逻辑回归对 SVM 分数进行校准（对训练数据额外进行 5-折交叉验证的训练），从而得到概率值。这会给 SVM 添加 predict_proba

（）和 predict_log_proba（）两种方法

6. 如果训练集有上千万个实例和几百个特征，你应该使用 SVM 原始问题还是对偶问题来训练模型？

答：因为核 SVM 只能使用对偶问题，所以此问题只适用于线性支持向量机。原始问题的计算复杂度与训练实例的数量成正比，而其对偶形式的计算复杂度与某个介于 m2 和 m3 的数量成正比。所以，应使用原始问题。

7. 假设你用 RBF 核训练了一个 SVM 分类器，看起来似乎对训练集拟合不足，你应该提升还是降低 γ （gamma）？ C 呢？

答： 可能是由于过度正则化，提升 gamma 和 C。

二、编程题

1. MNIST 手写数字数据集是一个开源数据集，用该数据集训练一个 SVM 模型，实现手写数字的识别（分类）。

程序参考：

```
from __future__ import print_function


import numpy as np
import multiprocessing


from sklearn.datasets import load_digits
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC


# For reproducibility
np.random.seed(1000)
```

```
if __name__ == '__main__':
    # Load dataset
    digits = load_digits()

    # Define a param grid
    param_grid = [
        {
            'kernel': ['linear', 'rbf', 'poly', 'sigmoid'],
            'C': [0.1, 0.2, 0.4, 0.5, 1.0, 1.5, 1.8, 2.0, 2.5, 3.0]
        }
    ]

    # Create a train grid search on SVM classifier
    gs = GridSearchCV(estimator=SVC(), param_grid=param_grid,
                      scoring='accuracy', cv=10, n_jobs=multiprocessing.cpu_count())
    gs.fit(digits.data, digits.target)

    print(gs.best_estimator_)
    print('Kernel SVM score: %.3f' % gs.best_score_)
```

2. 在加州住房数据集上训练一个 SVM 回归模型。

程序参考：

```
from __future__ import print_function

import numpy as np
import matplotlib.pyplot as plt
```

```
from sklearn.svm import SVR
from sklearn.model_selection import cross_val_score

# For reproducibility
np.random.seed(1000)

def show_dataset(Y, Y_pred=None):
    fig, ax = plt.subplots(1, 1, figsize=(30, 25))
    ax.grid()
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    if Y_pred is not None:
        ax.plot(Y_rel,label='rel')
        ax.plot(Y_pred,label='predict')
    plt.show()

if __name__ == '__main__':
    # Create dataset
    boston = load_boston()
    X = boston.data
    y = boston.target
    train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.25, random_state=33)
    ss_x = StandardScaler()
    x_train = ss_x.fit_transform(train_X)
    x_test = ss_x.transform(test_X)
    ss_y = StandardScaler()
```

```
    y_train = ss_y.fit_transform(train_y.reshape(-1, 1))
    y_test = ss_y.transform(test_y.reshape(-1, 1))


    # Create and train a Support Vector regressor
    svr = SVR(kernel='poly', degree=2, C=1.5, epsilon=0.5)
    svr_scores = cross_val_score(svr, x_train, y_train, scoring='neg_mean_squared_error',
cv=10)
    print('SVR CV average negative squared error: %.3f' % svr_scores.mean())
    svr.fit(x_train, y_train)
    Y_pred = svr.predict(x_test)
    Y_Pred = ss_y.inverse_transform(Y_pred)
    show_dataset(test_y, Y_Pred)
```

第7章 人工神经网络和深度学习

课后练习

一、正误题

1. 在神经网络的训练中， 我们一般将参数全部初始化为 0。（F）
2. ReLU 函数的输出是非零中心化的， 给后一层的神经网络引入偏置偏移，会影响梯度下降的效率。（T）
3. sigmoid 函数不是关于原点中心对称的，这会导致之后的网络层的输出也不是零中心的，进而影响梯度下降运作。tanh 激活函数解决了这个不足。（T）
4. 一般来说 batch Size 越大，其确定的下降方向越不准，引起训练 loss 震荡越大。（F）
5. RNN 的短期记忆问题是由其梯度消失问题造成的。（T）
6. LSTM 网络具有三个门，遗忘门，输入门，输出门。（T）

二、选择题

1. 神经网络中常见的超参数有：（ABCD）

   A. 梯度下降法迭代的步数

   B. 学习率

   C. 隐藏层数目

   D. 正则化参数

2. 假设你建立一个神经网络。 你决定将权重和偏差初始化为零。 以下哪项陈述是正确的？（D）

   A. 即使在第一次迭代中，第一个隐藏层的神经元也会执行不同的计算， 他们的参数将以各自方式进行更新。

   B. 第一个隐藏层中的每一个神经元都会计算出相同的结果，但是不同层的神经元会计算不同的结果。

C. 第一个隐藏层中的每个神经元将在第一次迭代中执行相同的计算。 但经过一次梯度下降迭代后，他们将会计算出不同的结果。

D. 第一个隐藏层中的每个神经元节点将执行相同的计算。 所以即使经过多次梯度下降迭代后，层中的每个神经元节点都会计算出与其他神经元节点相同的结果。

3. 以下属于机器学习中用来防止过拟合的方法的是：（BC）

A. Xiaver 初始化

B. Dropout

C. 增加训练数据，比如数据增强、对抗网络生成数据

D. 增加神经网络层数

4. 你正在构建一个识别足球（y = 1）与篮球（y = 0）的二元分类器。 你会使用哪一种激活函数用于输出层？（B）

A. tanh

B. sigmoid

C. Leaky ReLU

D. ReLU

5. 池化层在卷积神经网络中扮演了重要的角色，下列关于池化层的论述正确的有（ABD）

A. 池化操作可以扩大感受野

B. 池化操作可以实现数据的降维

C. 池化操作是一种线性变换

D. 池化操作具有平移不变性

6. 假设有一个三分类问题，某个样本的标签为（1，0，0），模型的预测结果为（0.5，0.4，0.1），则交叉熵损失值（取自然对数结果）约等于（A）

A. 0.7

B. 0.8

C. 0.6

D. 0.5

7. 在网络训练时，loss 在最初几个 epoch 没有下降，可能原因是（D）

A. 学习率过低

B. 正则参数过高

C. 陷入局部最小值

D. 以上都有可能

三、编程题

1. 根据如下数据表，利用它训练神经网络，从而能够预测正确的输出值。

	Input			Output
Training data 1	0	0	1	0
Training data 2	1	1	1	1
Training data 3	1	0	1	1
Training data 4	0	1	1	0
New Situation	1	0	0	?

```
import numpy as np

class NeuralNetwork():
    def __init__(self):
        # seeding for random number generation
        np.random.seed(1)
        #converting weights to a 3 by 1 matrix with values from -1 to 1 and mean of 0
        self.synaptic_weights = 2 * np.random.random((3, 1)) - 1

    def sigmoid(self, x):
        #applying the sigmoid function
        return 1 / (1 + np.exp(-x))

    def sigmoid_derivative(self, x):
        #computing derivative to the Sigmoid function
        return x * (1 - x)

    def train(self, training_inputs, training_outputs, training_iterations):
        #training the model to make accurate predictions while adjusting weights continually
        for iteration in range(training_iterations):
```

```
            #siphon the training data via  the neuron
            output = self.think(training_inputs)
            #computing error rate for back-propagation
            error = training_outputs - output
            #performing weight adjustments
            adjustments = np.dot(training_inputs.T, error * self.sigmoid_derivative(output))
            self.synaptic_weights += adjustments

    def think(self, inputs):
        #passing the inputs via the neuron to get output
        #converting values to floats
        inputs = inputs.astype(float)
        output = self.sigmoid(np.dot(inputs, self.synaptic_weights))
        return output

if __name__ == "__main__":
    #initializing the neuron class
    neural_network = NeuralNetwork()
    print("Beginning Randomly Generated Weights: ")
    print(neural_network.synaptic_weights)
    #training data consisting of 4 examples--3 input values and 1 output
    training_inputs = np.array([[0,0,1],
                                [1,1,1],
                                [1,0,1],
                                [0,1,1]])
    training_outputs = np.array([[0,1,1,0]]).T
    #training taking place
    neural_network.train(training_inputs, training_outputs, 15000)
    print("Ending Weights After Training: ")
    print(neural_network.synaptic_weights)
    user_input_one = str(input("User Input One: "))
    user_input_two = str(input("User Input Two: "))
    user_input_three = str(input("User Input Three: "))
    print("Considering New Situation: ", user_input_one, user_input_two, user_input_three)
    print("New Output data: ")
    print(neural_network.think(np.array([user_input_one, user_input_two, user_input_three])))
```

2. 波士顿房价预测问题：下载该数据集并将其直接使用文件名 housing.csv 保存到当前文件中。该数据集描述了波士顿郊区房屋的 13 个数值属性，并涉及对这些郊区房屋价格的建模（数千美元）。输入属性包括犯罪率，非零售营业面积比例，化学物质浓度等。因为所有输入和输出属性都是数字属性，并且有 506 个实例可以使用。使用 Keras 和 python 的 scikit-learn 库来实现了对房价的回归预测。

（此处下载数据集：https://raw.githubusercontent.com/jbrownlee/Datasets/master/housing.data）

```
import numpy
```

```
import pandas
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline

# load dataset
dataframe = pandas.read_csv("housing.csv", delim_whitespace=True, header=None)
dataset = dataframe.values
# split into input (X) and output (Y) variables
X = dataset[:, 0:13]
Y = dataset[:, 13]
# define base mode
def baseline_model():
    # create model
    model = Sequential()
    model.add(Dense(13, input_dim=13, init='normal', activation='relu'))
    model.add(Dense(6, init='normal', activation='relu'))
    model.add(Dense(1, init='normal'))
    # Compile model
    model.compile(loss='mean_squared_error', optimizer='adam')
    return model

# fix random seed for reproducibility
seed = 7
numpy.random.seed(seed)
# evaluate model with standardized dataset
estimators = []
estimators.append(('standardize', StandardScaler()))
estimators.append(('mlp', KerasRegressor(build_fn=baseline_model, nb_epoch=50, batch_size=5, verbose=0)))
pipeline = Pipeline(estimators)
# use 10-fold cross validation to evaluate this baseline model
kfold = KFold(n_splits=10, random_state=seed)
results = cross_val_score(pipeline, X, Y, cv=kfold)
print("Standardized: %.2f (%.2f) MSE" % (results.mean(), results.std()))
```

第8章 聚类

课后练习

一、正误题

1. K-mean 算法会存在陷入局部极值的情况，可以使用不同的初始化值，多次实验来解决该问题。（T）

2. K-means 算法不能够保证收敛。（F）

3. DBSCAN 算法是一种著名的密度聚类算法，该方法基于一组“邻域”参数来刻画样本的紧密程度。（T）

4. DBSCAN 对参数不敏感。（F）

5. 聚类算法中的谱聚类算法是一种分层算法。（F）

二、选择题

1. 有关聚类分析说法错误的是：（D）

A. 无须有标记的样本

B. 可以用于提取一些基本特征

C. 可以解释观察数据的一些内部结构和规律

D. 聚类分析一个簇中的数据之间具有高差异性

2. 使用 K-means 算法得到了三个聚类中心，分别是[1,2]，[-3,0]，[4,2]，现输入数据 X=[3,1]，则 X 属于第几类。（C）

A. 1

B. 不能确定

C. 3

D. 2

3. 对一组无标签的数据 X，使用不同的初始化值运行 K-means 算法 50 次，如何评测这 50 次

聚类的结果哪个最优。（D）

A. 暂无方法

B. 需要获取到数据的标签才能评测

C. 最后一次运行结果最优

D. 优化目标函数值最小的一组最优

4. 关于 K-means 说法不正确的是：（D）

A. 算法可能终止于局部最优解

B. 簇的数目 k 必须事先给定

C. 对噪声和离群点数据敏感

D. 适合发现非凸形状的簇

5. 闵可夫斯基距离表示为曼哈顿距离时 p 为：（A）

A. 1

B. 2

C. 3

D. 4

三、编程问题

1. 使用 sklearn 自带的 wine 数据集，构建聚类模型 K-Means，对该模型用轮廓分数进行评分并画出折线图。

```
from sklearn.datasets import load_wine#wine 数据集
from sklearn.cluster import KMeans#K-Means 聚类模型
from sklearn.model_selection import train_test_split#数据集划分
from sklearn.preprocessing import StandardScaler#标准差标准化
from sklearn.decomposition import PCA#pca 降维
from sklearn.metrics import silhouette_score    #聚类评分标准
import matplotlib.pyplot as plt#数据可视化

wine = load_wine()
data = wine['data']
target = wine['target']
#数据集划分为训练集，测试集
```

```
data_train,data_test,target_train,target_test = train_test_split(data,target,test_size=0.2,random_state=125)
#标准差标准化（规则）
stdScaler = StandardScaler().fit(data_train)
data_std_train = stdScaler.transform(data_train)
data_std_test = stdScaler.transform(data_test)
#pca 降维
pca_model = PCA(n_components=10).fit(data_std_train)#规则
data_pca_train = pca_model.transform(data_std_train)
data_pca_test = pca_model.transform(data_std_test)
print(data_pca_test)
#聚类模型
kmeans = KMeans(n_clusters=3,random_state=42).fit(data)
print('聚类模型为：',kmeans)
#聚类评分
#轮廓系数评分：
sil_score = []
for j in range(2,15):
    kmeans2 = KMeans(n_clusters=j, random_state=42).fit(data)
    score2=silhouette_score(data,kmeans2.labels_)
    sil_score.append(score2)
plt.rcParams['font.sans-serif'] = 'simhei'
plt.rcParams['axes.unicode_minus'] = False
plt.figure(figsize=(10,6))
plt.title('轮廓系数评分折线图')
plt.plot(range(2,15),sil_score,linewidth=1.5,linestyle='-',c='red')
plt.xticks(range(2,15,1))
plt.show()
```

第9章 知识表示方法

课后练习

一、正误题

1. 知识是人的大脑通过思维重新组合和、系统化的信息集合。(T)

2. 产生式表示，又称为 IF-THEN 表示， IF 后面部分描述了规则的结论，而 THEN 后面部分描述了规则的先决条件。（F）

3. 语义网络是知识的图解表示（T）

4. 知识表示的主体包括 3 类：表示方法的设计者、表示方法的使用者以及知识本身。（F）

二、选择题

1. 下列关于知识的说法正确的是（ABC）

A．知识是经过削减、塑造、解释和转换的信息

B．知识是经过加工的信息

C．知识是事实、信念和启发式规则

D．知识是凭空想象的

2. 下列哪些不属于谓词逻辑的基本组成部分？（D）

A．谓词符号

B．变量符号

C．函数符号

D．操作符

3. 语义网络表示法一般以下哪种继承是不存在的？（D）

A．值继承

B．“如果需要”继承

C．“默认”继承

D．左右继承

4. 语义网络中的推理过程主要有 (CD)

A.假元推理

B.合一

C.继承

D.匹配

5. 在框架表示法中，为了描述更复杂更广泛的事件，可把框架发展为(B)

A.专家系统 B.框架系统 C.槽 D.语义网络

三、编程题

1. 设有如下语句，请用相应的谓词公式分别把他们表示出来：

(1) 有的人喜欢梅花，有的人喜欢菊花，有的人既喜欢梅花又喜欢菊花 。

解：定义谓词

P(x)：x 是人

L(x,y)：x 喜欢 y

其中，y 的个体域是{梅花，菊花}。

将知识用谓词表示为：

$(\exists x)(P(x)\rightarrow L(x, 梅花)\vee L(x, 菊花)\vee L(x, 梅花)\wedge L(x, 菊花))$

(2) 有人每天下午都去打篮球。

解：定义谓词

P(x)：x 是人

B(x)：x 打篮球

A(y)：y 是下午

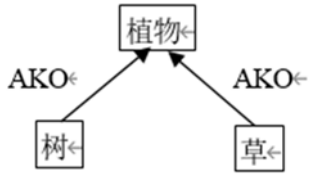
将知识用谓词表示为：

$(\exists x)(\forall y)(A(y)\rightarrow B(x)\wedge P(x))$

2. 请把下列命题用一个语义网络表示出来：

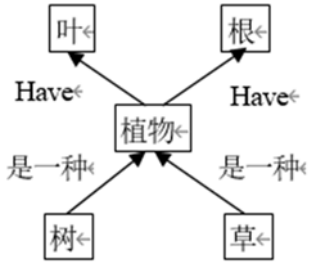
(1) 树和草都是植物；

解：



(2) 树和草都有叶和根；

解：



3. 假设有以下一段天气预报：“北京地区今天白天晴，偏北风 3 级，最高气温 12º，最低气温 -2º，降水概率 15%。”请用框架表示这一知识。

解：

Frame<天气预报>

地域：北京

时段：今天白天

天气：晴

风向：偏北

风力：3 级

气温：最高：12 度

最低：-2 度

降水概率：15%

第10章 经典逻辑推理

课后练习

一、正误题

1. 证明是根据已知为真的命题，来确定某一命题真实性的思维形式（T）

2. 当 P→Q 为真时，我们可以通过肯定后件 Q 为真来推出前件 P 为真（F）

3. 由 P→¬Q 为假，可知 P 为真，Q 为真（T）

4. 在不同谓词公式中，出现的多个谓词的谓词名相同但个体不同的情况，此时推理过程可以直接进行匹配（F）

5. 由¬P∨Q 为真，可推出 P→Q 为真（T）

二、选择题

1. 假设 P 为真，Q 为假，下列公式为真的是 （A）

A． P∨Q

B． P∧Q

C． P=>Q

D． ～P

2. 谓词演算的基本积木块是（C）

A.谓词符号

B.合适公式

C.原子公式

D.量词

3. 当 P 为真，¬ Q 也为真时，下列为真的公式是（B）

A. P∧Q　　B.P∨Q　　C.P→Q　　D.P↔Q

4. 经典逻辑推理的方法不包括那个（D）

A.自然演绎推理　　B.归结演绎推理　　C.于或形演绎推理　D.假设推理

5. 下列说法不正确的是（C）

A. 永真性：如果谓词公式 P 对个体域 D 上的任何一个解释都取得真值 T，则称 P 在 D 上是永真的

B. 可满足性：对于谓词公式 P，如果至少存在一个解释使得公式 P 在此解释下的真值为 T，则称公式 P 是可满足的

C. 永真性：如果谓词公式 P 对个体域 D 上，存在一个解释都取得真值 T，则称 P 在 D 上是永真的

D. 不可满足性：如果谓词公式 P 对个体域 D 上的任何一个解释都取得真值 F，则称 P 在 D 上是永久假的，如果 P 在每个非空个体域上均永假，则称 P 永假

6. 下列哪个公式是正确的（B）

A. $(P\rightarrow Q)\wedge(Q\rightarrow P)\Leftrightarrow P\wedge(Q\wedge R)$

B. $(P\wedge Q)\wedge R\Leftrightarrow P\wedge(Q\wedge R)$

C. $P\vee(Q\wedge R)\Leftrightarrow(P\wedge Q)\vee(P\wedge R)$

D. $\neg(\exists x)P(x)\Leftrightarrow(\exists x)(\neg P(x))$

三、**编程题**

1. 任何兄弟都有同一个父亲，John 和 Peter 是兄弟，且 John 的父亲是 David，问 Peter 的父亲是谁？

解：第一步：将已知条件用谓词公式表示出来，并化成子句集。那么，要先定义谓词。

（1） 定义谓词：

设 Father(x,y)表示 x 是 y 的父亲。设 Brother(x,y)表示 x 和 y 是兄弟。

（2） 将已知事实用谓词公式表示出来：F1: 任何兄弟都有同一个父亲。

$(x)(y)(z)(Brother(x,y)\wedge Father(z,x)\rightarrow Father(z,y))$

F2: John 和 Peter 是兄弟。

Brother(John, Peter)

F3: John 的父亲是 David。

Father(David, John)

（3） 将它们化成子句集，得

S1={¬Brother(x,y)∨ ¬Father(z,x)∨Father(z,y), Brother(John, Peter), Father(David, John)}

第二步：把问题用谓词公式表示出来，并将其否定与谓词 ANSWER 做析取。

设 Peter 的父亲是 u，则有：Father(u, Peter)

将其否定与 ANSWER 做析取，得

G: ¬Father(u, Peter) ∨ ANSWER(u)

第三步：将上述公式 G 化为子句集 S2，并将 S1 和 S2 合并到 S。

S2={¬Father(u, Peter) ∨ ANSWER(u)}

S=S1∪S2

将 S 中各子句列出如下：

（1）¬Brother(x,y)∨ ¬Father(z,x)∨Father(z,y)

（2）Brother(John, Peter)

（3）Father(David, John)

（4）¬Father(u, Peter) ∨ ANSWER(u)

第四步：应用归结原理进行归结。

（5）¬Brother(John,y) ∨ Father(David,y)

（1）与（3）归结，σ={ David/z, John/x}

（6）¬Brother(John, Peter) ∨ ANSWER(David)

（4）与（5）归结，σ={ David/u, Peter/y}

（7）ANSWER(David) （2）与（6）归结

第五步：得到了归结式 ANSWER(David)，答案即在其中，所以 u=David,即 Peter 的父亲是

David。

2. 张某被盗，公安局派出五个侦察员去调查。研究案情时，侦察员 A 说“赵与钱中至少有一人作案”；侦察员 B 说“钱与孙中至少有一人作案”；侦察员 C 说“孙与李中至少有一人作案”；侦察员 D 说“赵与孙中至少有一人与此案无关”；侦察员 E 说“钱与李中至少有一人与此案无关”。如果这五个侦察员的话都是可信的，试用归结演绎推理求出谁是盗窃犯。

解：第一步：将 5 位侦察员的话表示成谓词公式，为此先定义谓词。

设谓词 P(x)表示是作案者，所以根据题意：

A: P(zhao) ∨ P(qian)　　B: P(qian) ∨ P(sun)

C: P(sun) ∨ P(li)　　D: ¬P(zhao) ∨ ¬P(sun)

E: ¬P(qian) ∨ ¬P(li)

以上每个侦察员的话都是一个子句。

第二步：将待求解的问题表示成谓词。设 y 是盗窃犯，则问题的谓词公式为 P(y)，将其否定并与 ANSWER(y)做析取：¬P(y) ∨ ANSWER(y)

第三步：求前提条件及¬P(y) ∨ ANSWER(y)的子句集，并将各子句列表如下：

（1） P(zhao) ∨ P(qian)

（2） P(qian) ∨ P(sun)

（3） P(sun) ∨ P(li)

（4） ¬P(zhao) ∨ ¬P(sun)

（5） ¬P(qian) ∨ ¬P(li)

（6） ¬P(y) ∨ ANSWER(y)

第四步：应用归结原理进行推理。

（7） P(qian) ∨ ¬P(sun)　　(1)与(4)归结

（8） P(zhao) ∨ ¬P(li)　　(1)与(5)归结

（9） P(qian) ∨ ¬P(zhao)　　(2)与(4)归结

（10） P(sun) ∨ ¬P(li) (2)与(5)归结

（11） ¬P(zhao) ∨ P(li) (3)与(4)归结

（12） P(sun) ∨ ¬P(qian) (3)与(5)归结

（13） P(qian) (2)与(7)归结

（14） P(sun) (2)与(12)归结

（15） ANSWER(qian) (6)与(13)归结，σ={qian/y}

（16） ANSWER(sun) (6)与(14)归结, σ={sun/y}

所以，本题的盗窃犯是两个人：钱和孙。

第11章 专家系统

课后练习

一、选择题

1．能根据学生的特点、弱点和基础知识，以最适当的教案和教学方法对学生进行教学和辅导的专家系统是：（D）

A．解释专家系统 B．调试专家系统 C．监视专家系统 D．教学专家系统

2．用于寻找出某个能够达到给定目标的动作序列或步骤的专家系统是：（D）

A．设计专家系统 B．诊断专家系统 C．预测专家系统 D．规划专家系统

3．能对发生故障的对象（系统或设备）进行处理，使其恢复正常工作的专家系统是：（A）

A．修理专家系。。统 B．诊断专家系统 C．调试专家系统 D．规划专家系统

4．能通过对过去和现在已知状况的分析，推断未来可能发生的情况的专家系统是：（B）

A．修理专家系统 B．预测专家系统 C．调试专家系统 D．规划专家系统

二、简答题

1　专家系统的定义？

答：专家系统是一个含有大量的某个领域专家水平的知识与经验的智能计算机程序系统，能够利用人类专家的知识和解决问题的方法来处理该领域问题。

2　专家系统程序与常规的应用程序之间有何不同?

答：专家系统程序与常规的应用程序之间的不同主要体现在：

（1）与一般应用程序把问题求解的知识隐含地编入程序不同，专家系统把其应用领域的问题求解知识单独组成一个实体，即为知识库。

（2）专家系统的透明性是专家系统与传统程序的重要区别

透明性即专家系统能够对自己的行为作出解释，向用户解释它的行为动机和得出答案的

推理过程，便于发现系统的错误，并进行调试和维护。而常规程序仅仅以程序员了解的方式表现行为，用户无法了解传统程序系统处理问题的过程和答案的推理过程。

（3）专家系统与常规应用程序模拟的对象不同

专家系统模拟的是人类专家在问题领域上的推理，仿效人类专家的问题求解能力，而不是模拟问题领域本身。传统应用程序则是通过建立数学模型来模拟问题领域本身。

三、编程题

实现 11.4.2 中专家系统的规则提取。

答案见书附的本章程序。

第12章 人脸识别

课后练习

一、问答题

请举例说明人脸识别的应用场景。

参考答案：

人脸识别是基于人的脸部特征信息进行身份识别的一种生物识别技术。针对输入的人脸图像或者视频流，首先判断其是否存在人脸，如果存在人脸，则进一步的给出每个脸的位置、大小和各个主要面部器官的位置信息。然后依据这些信息，进一步提取每个人脸中所蕴涵的身份特征数据，并将其与已知的人脸进行对比，从而识别每个人脸的身份。

人脸识别主要用于身份识别，可以广泛应用于公安、金融、机场、地铁、边防口岸等多个对人员身份进行自然比对识别的重要领域。典型的人脸识别应用场景包括：

- 刷脸登录。常规登录模式就是网页请求登录，也可以使用刷脸登录；手机上有指纹识别，也有人脸识别开机；小区前台，通过刷脸登记出入客人信息。
- 刷脸门禁。通过工卡刷门禁，或通过指纹、刷脸开启门禁。
- 刷脸签到、会议签到、活动管理。民政方面，忘带身份证，可以通过刷脸确认身份。
- 抓拍捕捉人脸发现可疑人物，实现嫌疑人定位。
- 智能相册分类。
- 商场流量统计。通过摄像头捕捉人脸，调用后台判断是否是真人头像，然后捕获人流量，根据底库人脸，还能计算反客量等。
- 景区出入园人脸检票。在第一次入园时录入人脸，然后就可以“刷脸”游览景区内各个景点，同时出入景区也将更加便捷。

二、编程题

编程实现一个简单的基于人脸识别的考勤打卡系统。

程序参考：

第13章 自然语言处理

课后练习

一、问答题

请举例说明自然语言处理的应用场景。

参考答案：

自然语言处理是计算机科学领域及人工智能领域的一个重要研究方向，其目的是让计算机能够处理、理解以及运用人类语言，以实现人和计算机之间的有效通讯。

自然语言处理的典型应用场景包括：

- 信息提取：从指定文本范围中提取出重要信息，例如时间、地点、人物、事件等，可以帮人们节省大量时间成本，且效率更高。比如文摘生成利用计算机自动从原始文献中摘取文字，成果能够完整准确反映出文献的中心内容。
- 文本生成：根据限定条件或输入内容的不同，进行数据到文本或文本到文本的生成。
- 智能问答：对一个自然语言表达的问题进行某种程度的分析（例如实体链接、关系式、形成逻辑表达式等），分析完毕后在知识库中查找可能的候选答案，通过排序机智找出最佳的答案进行回复。比如电商行业中广泛应用的自动回复客服，通过回复许多基本而重复的问题，从而过滤掉大量重复问题，使得人工客服能够更好地服务客户。
- 机器翻译：通过把输入的源语言文本通过自动翻译获得另一种语言的文本，是自然语言处理中最为人所熟知的场景，比如百度翻译、Google 翻译等。
- 文本挖掘：包括文本聚类、分类、情感分析以及对挖掘的信息和知识通过可视化、交互式界面进行表达。
- 舆论分析：通过收集和处理海量信息，对网络舆情进行自动化的分析，帮助分析哪些话题是目前的热点，同时对热点的传播路径及发展趋势进行分析判断，以实现及时应对网络舆情。

- 知识图谱：又称科学知识图谱，在图书情报界称为知识域可视化或知识领域映射地图，是显示知识发展进程与结构关系的一系列各种不同的图形。以可视化技术为载体来描述知识资源及其载体，挖掘、分析、构建、绘制和显示知识及它们之间的相互联系。

三、编程题

请为某企业提供的

参考文献

[1] 朱福喜编著. 人工智能(第 3 版). 清华大学出版社, 2016.

[2] [意] 朱塞佩·博纳科尔索(著). 罗娜、汪文发(译). 机器学习算法(第 2 版). 机械工业出版社, 2020.

[3] 蔡自兴. 人工智能及其应用(第 5 版). 清华大学出版社, 2016

[4] 王永庆. 人工智能原理与方法. 西安: 西安交通大学出版社

[5] 尹朝庆. 人工智能方法与应用. 武汉: 华中科技大学出版社, 2007.

[6] Tom Michell. 机器学习:一种人工智能方法. 清华大学出版社,

[7] 机器学习实战

[8] 史忠植. 人工智能. 机械工业出版社, 2016.

[9] 李航. 统计学习方法.

[10] 周志华. 机器学习.

[11] 吕韶义,刘复岩. 基于决策树的规则获取[C]. 管理学报杂志社编辑部.第七届计算机模拟与信息技术学术会议论文集,1999:149-150.

[12] 王东, 利节, 许莎. 人工智能. 清华大学出版社, 2019.

致谢

在本书的写作过程中，除了参考同类的大量书籍、文献外，还参考了众多网络资源，如CSDN 博客、知乎、Hitbub 等。在此，特对参考的网络资源表示感谢，如有侵权，请随时联系作者。

主要参考的网络资源包括但不限于：

[1] https://blog.csdn.net/weixin_45458665/article/details/102687272