****

**电气与电气工程学院**

**2021-2022学年**

**招商证券人工智能**

**工程训练营预习报告**

|  |  |  |
| --- | --- | --- |
| 班级 | 学号 | 姓名 |
| 电气2012班 | U202011582 | 段一杰 |
|  |  |  |
|  |  |  |

华中科技大学电气与电子工程学院

1. 年8月
2. 预习目标
   1. 对这门课的期望

我选择这门课是因为我对编程特别感兴趣，用着手机电脑却不知道它的内部结构和代码的实现，我很想了解生活中的代码，比如人脸识别、自动驾驶等，这些技术在以前似乎是不可能的，而现在就出现在我们眼前，让我很好奇他们是如何运作的，代码是如何编写的。

希望从这门课学到人工智能的定义和思想，基本会使用python，熟练掌握调包和调参，完成几个有意思的作业。

* 1. 对人工智能的理解

我认为人工智能是人机对话、机器人下棋、自动驾驶等让机器表现出人的智慧的技术，这些技术可应用于各个行业，取代人工提高效率。但是技术的不成熟不稳定也会带来风险，比如自动驾驶出现车祸。

我对人工智能的期望是未来可以实现机器人自动学习知识，像人类一样思考，拥有人的感情和逻辑，就是把人的思想注入到机器中。

* 1. 预习计划
     1. 学习python基本语法
     2. 写一个有实际功能的代码，并可以运行
     3. 学会使用第三方库，完成线性回归
     4. 调研什么是人工智能，什么是机器学习

1. 基础知识预习

语言：python的基本数据类型有整数、浮点数、复数，控制结构有分支结构、循环结构，学会一些命令，比如pip，基本掌握。

工具：我使用的是vscode，学习了如何打开文件夹，如何运行代码，如何选择编译器，熟练掌握。

框架：数据科学计算库有Numpy、Scipy、pandas、matplotlib等，大致了解功能，可以调包调参使用。

人工智能包含机器学习，机器学习包含深度学习，深度学习包含神经网络。

人工智能：利用计算机和机器模仿人类思维的问题解决和决策制定能力。

机器学习：通过使用统计方法，对算法进行训练，以进行分类或预测，揭示数据挖掘项目中的关键洞察。然后，这些洞察可推动应用和业务中的决策，有效影响关键增长指标

深度学习：深度学习由“万能近似定理”支撑。这个定理就是神经网络可以拟合任何函数，不管这个函数的表达是多么的复杂。深度学习的强大也带来了对应的问题——黑箱化。我们比较难知道网络具体在做些什么，很难解释神经网络在解决问题的时候，为什么要这么做，为什么有效果。

神经网络：神经网络，也称为人工神经网络 (ANN) 或模拟神经网络 (SNN)，是机器学习的子集，并且是深度学习算法的核心。其名称和结构是受人类大脑的启发，模仿了生物神经元信号相互传递的方式。人工神经网络 (ANN) 由节点层组成，包含一个输入层、一个或多个隐藏层和一个输出层。 每个节点也称为一个人工神经元，它们连接到另一个节点，具有相关的权重和阈值。 如果任何单个节点的输出高于指定的阈值，那么该节点将被激活，并将数据发送到网络的下一层。否则，不会将数据传递到网络的下一层。

1. 预习项目设计
   1. 项目选题

选择八皇后问题，八个皇后放在棋盘上，不在同一行、同一列、同一斜线，是一个引人深思的问题，人力思考比较困难，所以使用回溯算法得出所有可能的布局。

* 1. 程序设计

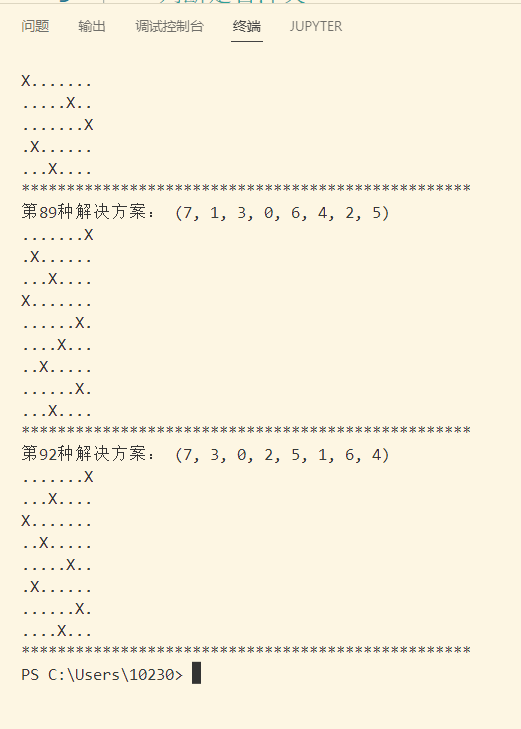


* 1. 代码实现

见附录

* 1. 运行结果

希望得到每一行皇后的坐标以及形象的图片，实际得到的：



1. 预习总结

我学习到了python的基本语法、环境和框架，能编写出一个能正常运行的代码，学会了调用第三方库的方法，实现了线性回归。了解了人工智能、机器学习、深度学习、神经网络等概念，将来希望进一步学习相关内容。

**附录**

**def** conflict(state, nextColumn):

    """

    判断是否冲突

    因为坐标是从0开始的，所以state的长度代表了下一行的行坐标

    :param state:（7，4，6，0，2） 标记每行皇后所在的位置 （0，7）一行八列 （2，4） （3，6） （4，0） （5，2）

    :param nextColumn:下一行的列坐标

    :return:

    """

    nextRow = rows = len(state) *# 5*

    for row in range(rows):   *#  0,1,2,3,4*

*# 获取当前行的列*

        column = state[row]

        """

        如何判断是否冲突：

            1. 如果列的差值为0，说明两皇后在同一列

            2. 如果列的差值等于行的差值，说明两皇后在对角线上

        """

        if abs(column - nextColumn) in [0, nextRow - row]:

            return True

    return False

*# 采用生成器的方式来产生每一个皇后的位置，并用递归来实现下一个皇后的位置*

**def** queens(num, state=()):

    """

    基于递归采用回溯算法，算出每一种结果

    :param num: 皇后的数量  8

    :param state: 列坐标。初始为空。参数为元组不为列表，因为参数只能为不可变数据类型

    :return:

    """

*# 每一行的列坐标都是从0:7的*

*# 0,1,2,3,4,5,6,7*

    for pos in range(num):

*# 默认state为空。长度为0，但是是不冲突的*

*# 判断是否冲突，state为空时不冲突*

        if not conflict(state, pos): *# 回溯法的体现*

*# 如果state的长度为7，即到达了倒数第二行，也就是前7行皇后都已经找到了位置，最后一行又没有冲突，返回最后一行的列坐标*

            if len(state) == num - 1:

*# 最后一行的（pos,）=最后一行的result，然后再递归回去求倒数第二行的result*

                yield (pos,)

            else:

                for result in queens(num, state + (pos,)):

                    """

                    递归实现求state：

                        1. 向下递归

                        第一次（行）： pos=0，刚开始不会进入if len(state) == num - 1，进入执行else，会执行queens(num, state + (pos, ))，

                        第二次（行）： 进入else，再调用queens(num, state + (pos, )),递归执行queens(num, state + (pos,) + (pos,))

                        第三次（行）： 进入else，再调用queens(num, state + (pos,) + (pos,),递归执行queens(num, state + (pos,) + (pos,) + (pos,))

                        ...

                        第七次（行）： 执行和上面的一样，不过此时state的长度为7

                        第八次（行）： 执行f len(state) == num - 1:求出最后一行的列坐标(pos,)

                        2.向上递归

                        求出第八行的列坐标，就可以求出第七行的（pos,），返回的是第七行和第八行的列坐标（（pos，） + result）

                        根据下一行的结果依次求出上一行的结果；

                        ....

                        最后求出第一行的列坐标，返回整体结果

                    """

                    yield (pos,) + result

**def** prettyprint(solution):

    """

    进行友好展示：为了至关表现棋盘，用X表示皇后的位置

    :param solution:

    :return:

    """

**def** line(pos, length=len(solution)):

        return '.' \* (pos) + 'X' + '.' \* (length - pos -1)

    for pos in solution:

        print(line(pos))

if \_\_name\_\_ == '\_\_main\_\_':

    solutions = queens(8)

    for index, solution in enumerate(solutions):

        print('第%d种解决方案：' %(index + 1), solution )

        prettyprint(solution)

        print('\*' \* 50)