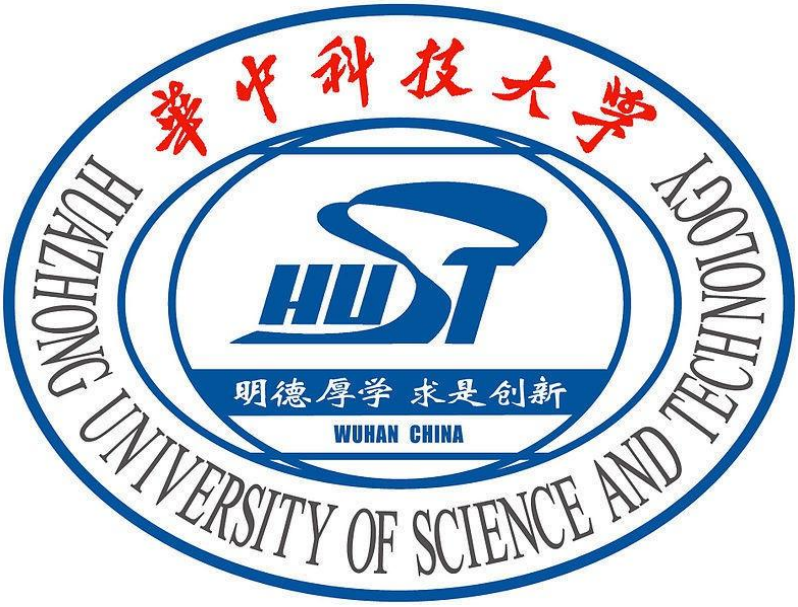
2022 年华中科技大学电气与电子工程学院

证券投资训练营

预习报告



院　系　：电气与电子工程学院

班　级　：电气 2005 班

姓　名　：鲍林奕

学　号　：U202012349

任课教师：郑玮

目录

一、 预习目标（对应课程目标 1）

1.1 对这门课的期望

- 希望了解人工智能的相关知识，开拓视野；
- 希望学习一些简单的神经网络搭建，例如 CNN、RNN、LSTM 等；
- 希望学到更细致的神经网络内容，例如 PINN、基于注意力机制的 LSTM 等；
- 希望学会如何在实际问题中应用、修改神经网络模型。

1.2 对人工智能的理解

- 可以替代人工劳作的计算机智能。

1.3 预习计划

- 学习一些人工智能知识；
- 自己运行并修改一些神经网络程序。

二、 基础知识预习（对应课程目标 1）

2.1 工具安装和环境配置

- Python（pycharm/vscode）+Anconada
- GitHub

2.2 基础知识预习

- 通过《数学与建模仿真》课程学习掌握的 python 语言，能够实现简单的机器学习；
- 通过 Github、CSDN 学习一些简单的神经网络实例。

三、 预习项目设计(对应课程目标 2，3)

3.1 项目选题

2022 电工杯赛题《高比例风电电力系统储能运行及配置分析》

3.2 程序设计

用粒子群算法得到不同情况下 P1、P2、W 的最优解：

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import csv
data = pd.read_csv(r'C:\Users\BLY13\Desktop\负荷功率.csv', encoding='GBK')
P_all = np.expand_dims(data[['负荷总功率']], axis=1)[:, 0]
lenth = len(P_all)
p1 = np.random.rand(96, 1)
```

```
p2 = np.random.rand(96, 1)
p3 = np.random.rand(96, 1)
W = np.random.rand(96, 1)
# 参数—碳捕集单位成本 h，负荷功率 P0，碳排放量 H， 耗煤系数 A、B、C
H1, H2, H3 = 0.72, 0.75, 0.79
A1, A2, A3 = 0.226, 0.588, 0.785
B1, B2, B3 = 30.42, 65.12, 139.6
C1, C2, C3 = 786.80, 451.32, 1049.50

for j in range(lenth):
    h = 0
    P0 = P_all[j]
    # 加速常数
    c1 = 2
    c2 = 2

    S = [1, 1]
    left = [180, 90]
    right = [600, 300]

    xmin = 180
    xmax = 300

    NDim = 2

    max_iter = 600  # 迭代次数
    num_particle = 100  # 粒子数目

    # 惯性权重随迭代次数增加而逐渐减小，有利于算法收敛
    start_weight = 0.9
    end_weight = 0.4
    weight_step = (start_weight - end_weight) / max_iter

    # 目标函数，x0,x1,x2
    def objective(x0, x1, x2):
        return h / 4 * (H1 * x0 + H2 * x1 + H3 * (P0 - x0 - x1)) + 21/80 * (A1 * x0**2 + A2 * x1**2 + A3 * (P0 - x0 - x1)**2 + B1 * x0 + B2 * x1 + B3 * (P0 - x0 - x1) + C1 + C2 + C3)


    # 初始化粒子群，每个粒子在定义域内随机选择一个初始位置
    particle = xmin + (xmax - xmin) * np.random.rand(num_particle, NDim)
    # 初始速度
    V = 0.5 * (xmax - xmin) * (np.random.rand(num_particle, NDim) - 0.5)
    # 粒子适应度
```

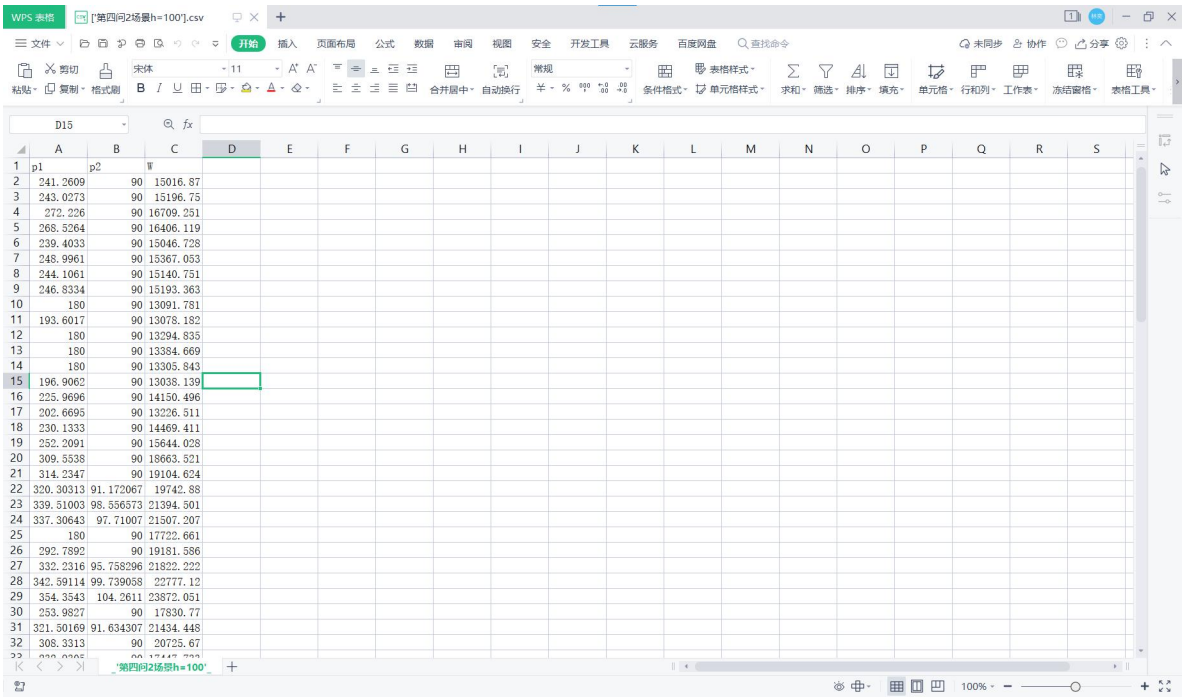
```
    fitness = np.zeros((num_particle, 1))
    # 粒子最佳位置
    pbest = np.zeros((num_particle, NDim))

    # 计算初始适应度
    for i in range(num_particle):
        pg1 = particle[i, :]
        pg2 = P0 - pg1[0] - pg1[1]
        if (45 <= pg2).all() and (pg2 <= 150).all():
            tmp_array = pg1
            if (tmp_array >= left).all() and (tmp_array <= right).all():
                fitness[i] = objective(pg1[0], pg1[1])
            else:
                # 若 pg1 和 pg2 不满足潮流上下限约束则适应度为无穷
                fitness[i] = np.inf
        else:
            # 若 pg2 不满足给定出力上下限约束则适应度为无穷
            fitness[i] = np.inf
    for i in range(num_particle):
        pbest[i, :] = particle[i, :]

    # 每个粒子的当前最佳适应度
    pbest_value = fitness
    # 当前的最佳适应度
    gbest_value = np.min(pbest_value)
    # 当前最佳适应度粒子的位置
    gbest = particle[np.argmin(pbest_value)]  # np.argim()返回最小值索引位置

    # 记录每次迭代的最佳目标函数值
    obj_fun_val = np.zeros((max_iter, 1))  # 初值为 0

    for iter in range(max_iter):
        # 惯性权重
        w = start_weight - (iter + 1) * weight_step
        # 速度
        V = w * V + c1 * np.random.rand() * (pbest - particle) + c2 * \
            np.random.rand() * (gbest - particle)  # 更新速度
        # 更新粒子位置
        particle += V
        # 将粒子位置约束在 xmin 和 xmax 之间
        particle[:, 0] = np.clip(particle[:, 0], 180, 600)
        particle[:, 1] = np.clip(particle[:, 1], 90, 300)

        # 计算适应度
```

```
        for i in range(num_particle):
            pg1 = particle[i]
            pg2 = P0 - pg1[0] - pg1[1]
            if (45 <= pg2).all() and (pg2 <= 150).all():
                tmp_array = pg1
                if (tmp_array >= left).all() and (tmp_array <= right).all():
                    # 满足约束
                    fitness[i] = objective(pg1[0], pg1[1])
                else:
                    # 若 pg1 和 pg2 不满足潮流上下限约束则适应度为无穷
                    fitness[i] = np.inf
            else:
                # 若 pg2 不满足给定出力上下限约束则适应度为无穷
                fitness[i] = np.inf

        # 更新粒子位置和最佳适应度
        pbest_value = np.minimum(pbest_value, fitness)

        # 更新每个粒子取得最佳适应度的位置
        for i in range(num_particle):
            if pbest_value[i] == fitness[i]:
                pbest[i] = particle[i]

        # 本次迭代的最佳适应度
        current_min_value = np.min(pbest_value)
        # 总体最佳适应度
        gbest_value = np.minimum(current_min_value, gbest_value)
        if gbest_value == current_min_value:
            # 总体粒子最佳位置
            gbest = pbest[np.argmin(pbest_value)]
        obj_fun_val[iter] = gbest_value
    p1[j] = particle[-1, 0]
    p2[j] = particle[-1, 1]
    p3[j] = P0 - p1[j] - p2[j]
    W[j] = obj_fun_val[-1]
    print(p1[j], p2[j], p3[j], W[j])  # [-1]获取最后一个元素
w = np.hstack((p1, p2, p3, W))
title = ["p1", "p2", "p3", "W"]
Name = ['机组 4.3']
data = w.reshape(96, 4)
i_str = str(Name)
path = i_str + '.csv'
try:
    with open(path, 'w', newline='') as t:  # newline 是来控制空的行数的
```

```
        writer = csv.writer(t)  # 这一步是创建一个 csv 的写入器
        writer.writerow(title)  # 写入标签
        writer.writerows(data)  # 写入样本数据
except:
        pass
print('ok')
```

3.3 运行结果

成功得到 P1、P2、W 的最优解并写入 csv 文件：



	p1	p2	W
1	241.2609	90	15016.87
2	243.0273	90	15196.75
3	272.226	90	16709.251
4	268.5264	90	16406.119
5	239.4033	90	15046.728
6	248.9961	90	15367.053
7	244.1061	90	15140.751
8	246.8334	90	15193.363
9	180	90	13091.781
10	193.6017	90	13078.182
11	180	90	13294.835
12	180	90	13384.669
13	180	90	13305.843
14	196.9062	90	13038.139
15	225.9696	90	14150.496
16	202.6695	90	13226.511
17	230.1333	90	14469.411
18	252.2091	90	15644.028
19	309.5538	90	18663.521
20	314.2347	90	19104.624
21	320.30313	91.172067	19742.88
22	339.51003	98.556573	21394.501
23	337.30643	97.71007	21507.207
24	180	90	17722.661
25	292.7892	90	19181.586
26	332.2316	95.758296	21822.222
27	342.59114	99.739058	22777.12
28	354.3543	104.2611	23872.051
29	253.9827	90	17830.77
30	321.50169	91.634307	21434.448
31	308.3313	90	20725.67

四、 预习总结（对应课程目标 4，5，7）

- 成功掌握了基本的数据学习；
- 成功实现了简单的机器学习；
- 成功学会了使用 Python 处理、保存数据，提升了数据处理能力；
- 了解了人工智能的一部分内容，对神经网络有初步了解。