JM.py

import numpy as np  
from scipy.optimize import basinhopping  
import matplotlib.pyplot as plt  
  
# 设置matplotlib支持中文  
plt.rcParams['font.sans-serif'] = ['SimHei']  # 使用黑体  
plt.rcParams['axes.unicode\_minus'] = False  # 正确显示负号  
  
# 数据集  
NTDSDataset = np.array([  
    9, 12, 11, 4, 7, 2, 5, 8, 5, 7, 1, 6, 1, 9, 4, 1, 3, 3, 6, 1, 11, 33, 7, 91, 2, 1, 87, 47, 12, 9, 135, 258, 16, 35  
])  
NTDSDataset\_old = np.array([  
    9, 12, 11, 4, 7, 2, 5, 8, 5, 7, 1, 6, 1, 9, 4, 1, 3, 8, 6, 1, 1, 33, 7, 91, 2, 1, 87, 47, 12, 9, 135, 258, 16, 35  
])      # 用这个数据集的前26个数据估计参数，得到的结果和书上的一致  
  
  
# 定义方程组  
def equations(p, x):  
    N0, phi = p  
    n = len(x)  
    f1 = n / (N0 \* np.sum(x) - np.sum((np.arange(n)) \* x)) - phi  
    f2 = np.sum(1 / (N0 - np.arange(n))) - n / (N0 - (1 / np.sum(x)) \* np.sum((np.arange(n)) \* x))  
    return np.array([f1, f2])  
  
# 目标函数：方程组的平方和  
def objective(p, x):  
    return np.sum(equations(p, x) \*\* 2)  # 这里可以安全地做平方运算  
  
# 参数估计  
def estimate\_parameters(x):  
    n = len(x)  
    initial\_guess = [32, 0.01]  # 初始猜测值  
    result = basinhopping(objective, initial\_guess, niter\_success=50,  
                          minimizer\_kwargs={"method": "BFGS", "args": (x,)})  
    return result.x  
  
# 计算累计故障时间  
def cumulative\_failure\_time(data):  
    return np.cumsum(data)  
  
# 计算预测的累计故障时间  
def predict\_cumulative\_failure\_time(N0, phi, data\_length):  
    failure\_times = []  
    current\_time = 0  
    for i in range(1, data\_length + 1):  
        current\_time += 1 / (phi \* (N0 - i + 1))  
        failure\_times.append(current\_time)  
    return failure\_times  
  
# 主函数  
def main():  
    # 估计参数  
    x = NTDSDataset  
    # x = NTDSDataset\_old[:26]  
    N0, phi = estimate\_parameters(x)  
    print(f"估计的N0: {N0}, 估计的phi: {phi}")  
  
    # 计算实际的累计故障时间  
    actual\_cumulative\_time = cumulative\_failure\_time(NTDSDataset)  
  
    # 计算预测的累计故障时间，绘制到N0为止  
    N = int(N0) + 1  
    predicted\_cumulative\_time = predict\_cumulative\_failure\_time(N0, phi, N)  
  
    # 绘制实际和预测的累计故障时间  
    plt.figure(figsize=(10, 6))  
    plt.plot(range(1, len(NTDSDataset) + 1), actual\_cumulative\_time, label='实际累计故障时间', marker='o')  
    plt.plot(range(1, N + 1), predicted\_cumulative\_time[:N], label='预测累计故障时间', marker='x')  
    plt.xlabel('故障次数')  
    plt.ylabel('累计故障时间')  
    plt.title('实际与预测累计故障时间比较')  
    plt.legend()  
    plt.show()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
    main()

GO.py

import numpy as np  
import matplotlib.pyplot as plt  
import math  
  
# 设置matplotlib支持中文  
plt.rcParams['font.sans-serif'] = ['SimHei']  # 使用黑体  
plt.rcParams['axes.unicode\_minus'] = False  # 正确显示负号  
  
# 故障时间数据  
t = [9, 21, 32, 36, 43, 45, 50, 58, 63, 70, 71, 77, 78, 87, 91, 92, 95, 98, 104, 105,  
     116, 149, 156, 247, 249, 250, 337, 384, 396, 405, 540, 798, 814, 849]  
  
# 训练数据和验证数据  
train\_t = t[:26]  
test\_t = t[26:]  
  
# 估计参数a和b  
def GO\_model(train\_t):  
    epsilon = 0.000\_001  
    N = len(train\_t)  
    D = sum(train\_t) / (N \* max(train\_t))  
    xl = xm = xr = 0  
    if 0 < D < 1 / 2:  
        xl = (1 - 2 \* D) / 1  
        xr = 1 / D  
    else:  
        raise ValueError("无解！")  
    while 1:  
        xm = (xl + xr) / 2  
        if abs(xl - xr) < 1e-10:  
            xm = (xl-xr)/2  
            break  
        f = (1 - D \* xm) \* math.e\*\*xm + (D - 1) \* xm - 1  
        if f > epsilon:  
            xl = xm  
        elif f < -epsilon:  
            xr = xm  
        else:  
            break  
  
    b = xm / max(train\_t)  
    a = N / (1 - math.e\*\*(-b \* max(train\_t)))  
    print("a =", a, "b =", b)  
  
    return a, b  
  
a, b = GO\_model(train\_t)  
  
# 计算累计故障数量  
def m(t, a, b):  
    return a \* (1 - np.exp(-b \* np.array(t)))  
  
# 反解GO模型的累计故障时间  
def inverse\_m(m\_values, a, b):  
    return -1 / b \* np.log(1 - np.array(m\_values) / a)  
  
  
# 训练数据的预测值  
train\_m = m(train\_t, a, b)  
  
# 验证数据的预测值  
test\_m = m(test\_t, a, b)  
  
# 绘制图表  
plt.figure(figsize=(10, 6))  
plt.plot(train\_t, range(1, len(train\_t) + 1), 'bo-', label='训练数据')  
plt.plot(test\_t, range(len(train\_t) + 1, len(t) + 1), 'ro-', label='验证数据')  
plt.plot(train\_t + test\_t, range(1, len(t) + 1), 'b-')  
plt.plot(t, m(t, a, b), 'g--', label='模型预测')  
plt.scatter(t, m(t, a, b), color='green')  # 添加实心点  
plt.scatter(train\_t, range(1, len(train\_t) + 1), color='blue')  
plt.scatter(test\_t, range(len(train\_t) + 1, len(t) + 1), color='red')  
plt.plot(test\_t, range(len(train\_t) + 1, len(t) + 1), 'r-')  # 修改为红色线条  
plt.xlabel('故障时间')  
plt.ylabel('累计故障数量')  
plt.title('GO模型验证与可视化')  
plt.legend()  
plt.grid(True)  
plt.show()

Compare.py

import numpy as np  
import matplotlib.pyplot as plt  
from JM import estimate\_parameters, cumulative\_failure\_time, predict\_cumulative\_failure\_time  
from GO import m, inverse\_m, GO\_model  
  
# 设置matplotlib支持中文  
plt.rcParams['font.sans-serif'] = ['SimHei']  # 使用黑体  
plt.rcParams['axes.unicode\_minus'] = False  # 正确显示负号  
  
# 使用相同的数据集  
dataset = np.array([  
    9, 12, 11, 4, 7, 2, 5, 8, 5, 7, 1, 6, 1, 9, 4, 1, 3, 3, 6, 1, 11, 33, 7, 91, 2, 1, 87, 47, 12, 9, 135, 258, 16, 35  
])  
  
# JM模型预测  
N0, phi = estimate\_parameters(dataset)  
actual\_cumulative\_time = cumulative\_failure\_time(dataset)  
predicted\_cumulative\_time\_jm = predict\_cumulative\_failure\_time(N0, phi, len(dataset))  
  
# GO模型预测  
train\_t = np.cumsum(dataset)  
a, b = GO\_model(train\_t)  
predicted\_cumulative\_time\_go = inverse\_m(range(1, len(dataset) + 1), a, b)  
  
# 绘制比较图  
plt.figure(figsize=(12, 8))  
plt.plot(range(1, len(dataset) + 1), actual\_cumulative\_time, label='实际累计故障时间', marker='o')  
plt.plot(range(1, len(dataset) + 1), predicted\_cumulative\_time\_jm, label='JM模型预测', marker='x')  
plt.plot(range(1, len(dataset) + 1), predicted\_cumulative\_time\_go, label='GO模型预测', marker='s')  
plt.xlabel('故障次数')  
plt.ylabel('累计故障时间')  
plt.title('JM模型与GO模型的比较')  
plt.legend()  
plt.grid(True)  
plt.show()