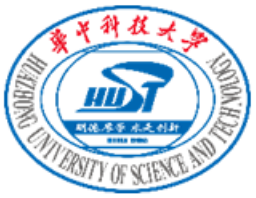


中国研究生创新实践系列大赛

中国光谷·“华为杯”第十九届中国研究生数学建模竞赛

学　　校	东南大学
参赛队号	**22102860361**
队员姓名	**1.** 朱坤傲
	2. 周宏毅
	3. 朱靖宇

中国研究生创新实践系列大赛

中国光谷·“华为杯”第十九届中国研究生数学建模竞赛

题　　目　典型温带草原土壤理化性质与放牧策略的研究

摘要

我国是第二大草原大国，草原是高原畜牧业的生产支柱，是牧民生活的物质基础，同时具有维护生物多样性、调节水土流失等重要的生态功能。放牧可以带动区域经济发展，改善牧民生活质量，但是不合理的放牧会导致草原退化，甚至造成荒漠化。本文就相关草原数据结合现实问题，对草原土壤、植被等状态进行分析，根据分析结果给出合理的放牧策略。

对于问题一，我们对问题进行适当的简化，将放牧方式和放牧强度进行量化处理，通过相乘得到载畜率。针对土壤湿度，通过分析**放牧强度对土壤状态的影响**以及**土壤状态对土壤水渗透量和实际蒸发量的影响**，结合土壤含水量计算公式，通过土壤水渗透量因子 α_a 以及实际蒸发量因子 α_d 反映放牧强度对土壤湿度的影响，结果为 $\Delta W = P + G_u + R_{\text{in}} - (\alpha_a \times Et_a + (1+\alpha_d) \times G_d + R_{\text{out}} + IC_{\text{store}})$；针对植被生物量，我们利用植物干重反映植被生物量的变化，通过**分析不同放牧强度下牧前牧后植物干重数量**，并通过**多项式拟合**得到放牧强度对植物干重的影响，结果为 $R = 5.8223 - 1.2422(PS) + 0.1893(PS)^2 - 0.0116(PS)^3$。

对于问题二，进行时间序列预测，已知 **2012 年 1 月到 2022 年 3 月的数据**，通过**滑窗**的方式生成训练集和测试集，我们在**滑窗的过程中允许存在重叠**，对数据进行了增强，最终得到 **85** 个训练样本和 **13** 个测试样本。考虑到数据集仍然较小，我们对训练集进行了 20% 的升采样并简化了模型的搭建：**单层单向 LSTM 模块 + 一层全连接层**。由于 2022 年 4 月之后的土壤蒸发量等数据缺失，故**对于 2022 年 4 月采用多变量时间序列预测**得到一份较为精确的结果，**对 2022 年 5 月之后采用单变量自回归预测**，模型在**测试集上的均方误差小于 0.2**，具体预测结果见表3。

对于问题三，已有 **5** 个历史数据点，我们使用**灰色预测法**进行土壤化学性质的预测，修改生成累加数据 $z^{(1)}(t)$ 的表达式，**通过系数 S 体现放牧强度对土壤化学性质的影响**。总共需要进行 60 次预测，**每一次预测的相对校验误差均小于 0.1，平均校验误差为 0.031**，具体预测结果见表4。

对于问题四，针对沙漠化程度指数，我们结合锡林郭勒牧区的具体情况以及现有的资料，根据评价模型 $SM = \eta \cdot \sum_{i=1}^{n} S_{Q_i} = \eta \cdot \sum_{i=1}^{n} (Q_i \cdot W_{c_i})$，使用**层次分析法**，选取七个对沙漠化程度影响最大的因子（暖气平均风速、暖期降水量、暖期平均气温、暖期植被因子、暖期地表水资源因子、暖期地下水资源因子、放牧强度），并确定其权重系数 [0.1823，0.0796，0.0693，0.1428，0.0566，0.0899，0.3792]；针对板结化程度，**模仿沙漠化程度指数预测模型，建立土壤板结化的模型**，使用**层次分析法**，根据实际情况和查阅文献，给予主要的三个因子（土壤湿度、土壤容重、有机物含量）合理的权重，**对板结化程度给出一个定量的定义：** $B = 0.4066\bar{W} + 0.3695\bar{C} + 0.2238\bar{O}$。最终能够绘制出 2012 年至 2021 年十年的沙漠化程度曲线，并对放牧策略给出合理的预测，确定放牧策略，使沙漠化程度与板结化在可接受范围内。放牧强度**使草原轻度沙漠化的阈值为 0.5 羊/天/公顷，使草原中度沙漠化的阈值为 4.7 羊/天/公顷**。

对于问题五，根据问题四中已经建立的模型，**使用荒漠化程度和板结化程度定义一个新的评价指标** $E = 0.7SM + 0.3B$，针对各个放牧小区不同的土地情况进行预测。最终得到在不同降雨量下各个放牧小区的可持续性指标随着放牧强度增加而变化的曲线，设定认为**评价指标小于 0.4 时为可接受情况**，在此基础上预测出各个放牧小区在不同降雨量下可行的最大放牧强度，并绘制出表格。

对于问题六，我们通过问题四已经得到土壤沙漠化放牧强度阈值上限为 4.7 羊/天/公顷，并结合附件 13 数据建立关于不同放牧强度土地状态的数学模型，并预测 2023 年 9 月土壤肥力、湿度和植被覆盖指数。其中土壤肥力利用附件 14 有机碳含量近似；土壤湿度和植被覆盖指数使用问题一中得到的模型带入放牧强度进行预测。具体预测结果见表13。

关键字：草原放牧策略　土壤理化性质　LSTM 时间序列预测　层次分析法　基于放牧强度的灰色预测

目录

1. 问题重述

1.1 引言

科学发展观指出：坚持以人为本，全面、协调、可持续的发展观。在处理人和自然的问题上，我们必须要认识到人类必须尊重自然、顺应自然、保护自然，人类必须遵从自然规律。草原作为我国地理环境组成的重要部分，肩负起保护生态多样性、净化空气、调节水土流失等重要作用。在过去的漫长岁月中，由于不正确的环境意识，导致部分草原地带沙化严重，严重影响到自然环境和区域气候。但是为了保护和改善生态环境并不是简单禁止放牧。我们需要指定合理的放牧与土地轮休计划，针对性地指定合理的放牧政策，同时平衡好区域经济发展和环境保护防止草原退化的关系。

1.2 问题的提出

问题 1：建立放牧方式和放牧强度对锡林郭勒草原土壤物理性质（主要是土壤强度）和植被生物量影响的数学模型。
其中自变量放牧方式可以定量分析连续放牧、轮牧、轻度放牧、生长季度轮休、禁牧五种。由于轮牧、轮休和轻度放牧均可理解为在时空范围内不完全使用牧场，放牧方式可以归一化为对牧场使用强度的不同，在本模型中使用一个**比例系数**体现；放牧强度为单位牧区内的牲畜生物量，将牛、羊、骆驼、马等生物消耗按折算系数相加，得到单位区域内牲畜**生物量**。
因变量为土壤物理性质 (土壤湿度 h) 的变化 dh/dt 和植被生物量的变化 dw/dt。分别建立自变量和因变量关系的数学模型。并和实际结果进行验证。

问题 2：根据附件相关数据建立映射关系，其中自变量为土壤蒸发数据以及降水等数据，因变量为不同深度下土壤湿度数据。建立完模型后不考虑放牧策改变的情况下即不考虑放牧政策对土壤湿度的影响进行对 2022 年以及 2023 年不同深度土壤深度湿度进行预测。

问题 3：通过自变量放牧方式和放牧强度和因变量草原土壤化学性质影响建立拟合数学模型。并通过该模型对不同自变量条件下土壤相关性质进行预测。自变量系数折算可类比于问题 1.

问题 4：利用沙漠化程度指数预测模型和相关数据建立不同放牧条件对沙漠化程度指数之间的关系，同时给出土壤板结化的定义，并结合前问中对土壤化学

性质的影响确定放牧类型，使得沙漠化指数和板结化程度最小。

问题 5：在问题 4 已经得到的最小沙漠化和板结化的条件下，改变降水量的大小，预测可持续发展情况下放牧羊的数量的最大阈值。

问题 6：在已经得到问题 1、2 放牧方式和放牧强度的条件下，对附件 13 示范放牧户和问题 4 得到的沙漠化和板结化最小放牧方案情况进行一年后土地状态的预测。

2. 模型的假设

- 不同草场土地视为相同品质土地；
- 不同种类牲畜按折算系数统一计算日食量；
- 忽视海拔，假设地面平整。

3. 符号说明

符号	意义
γ	放牧方式比例系数
S	单位面积载畜率 (羊/天/公顷)
$Infect$	放牧强度和方式影响系数
R	植被放牧前后干重比例
E	综合环境影响指数
SOC	土壤有机碳含量
SIC	土壤无机碳含量
STC	土壤全碳含量
N	土全氮含量
SM	沙漠化程度指数
Q_i	层次分析法第 i 个因子强度
W_{q_i}	层次分析法因子权重系数
V_{wind}	风速 (knots)
T	温度 (摄氏度)
$NDVI$	植被指数
W, RH	土壤湿度
C	土壤容重
O	土壤有机物含量
P	降水量
α_d	土壤水渗透量因子
α_a	实际蒸发量因子
β	灰色预测背景数据修正因子

4. 问题求解

4.1 问题一求解

4.1.1 问题分析

任务一需要收集附件 15 材料中数据并进行补全，计算每年不同放牧强度各个牧场牧前牧后植物干重数量，利用多次拟合方法对自变量放牧强度与植被生物量即植物干重的数学模型。利用土壤-植被-大气系统水平衡基本方程和土壤放牧强度、土壤水渗透量因子和实际蒸发量因子进行对方程进行修正，在缺少具体数值的情况下给出经验数值。

4.1.2 数据选择与数据预处理

本文首先对数据进行量化处理。根据题目描述和文献将放牧方式量化为比例系数。其中由于本题数据主要讨论的是选区划线轮牧方式，选择此种方式为基准比例系数 $\gamma = 1$，休牧为比例系数 $\gamma = 0$。将不同放牧强度（单位面积载畜率）量化为羊/天/公顷。其中对照组休牧为 $S = 0$ 羊/天/公顷，轻度放牧强度为 $S = 2$ 羊/天/公顷，中度放牧强度为 $S = 4$ 羊/天/公顷，重度放牧强度为 $S = 8$ 羊/天/公顷. 综合来看放牧方式和放牧强度量化为 $Infect = \gamma * S$。

本文计算干重采用计算当前牧区轮次所列举植物干重总和。在附件 15 中，由于缺少 2016 年 G20 牧场重牧方式下牧前干重数据，考虑到相同年份牲畜对牧草影响程度类似，采用回归替换法补全当前缺失数据。计算其余两个牧区 G9 和 G13 在 2016 年牧前和牧后干重均值，分别为牧前均值 430.59，牧后均值 350.31，牧后和牧前比值为 0.8147，根据该比例计算 2016 牧前 G20 牧场数据为 336(g)。最终得到 2016 年至 2020 年不同放牧强度下牧前牧后植物干重。

	2016牧前	2016牧后	2017牧前	2017牧后	2018牧前	2018牧后	2019牧前	2019牧后	2020牧前	2020牧后
无牧（G17）	297.32	925.01	142.69	596.8	59.62	760.99	78.49	579.21	373.4	1040.23
无牧（G19）	259.7	935	194.87	640.58	50.1	656.38	80	650.23	449.62	788.71
无牧（G21）	313.14	892.67	162.89	582.6	63.22	579.39	61.73	677.26	358.67	862.94
轻牧（G6）	256.65	958.32	251.31	609.15	71.87	501.56	114.49	377.63	260.15	1187.15
轻牧（G12）	420.47	748.76	207.54	514.51	56.405	373.46	103.23	600.72	429.88	902.41
轻牧（G18）	370.09	566.41	165.36	488.11	30.43	298.54	101.19	459.32	313.16	886.54
中牧（G8）	383.44	602.28	201	582.93	68.11	260.25	85.73	351.26	392.61	732.74
中牧（G11）	363.8	520.43	222.02	479.48	51.09	378.7	126.79	393.38	370.99	706.79
中牧（G16）	320.09	520.62	187.82	420.44	50.5	407.7	109.34	355.31	318.1	704.58
重牧（G9）	436.01	368.95	218.27	543.62	37.62	179.13	101.17	161.71	209.8	500.58
重牧（G13）	425.17	331.67	235.89	529.96	44.23	113.85	71.9	143.75	259.94	748.64
重牧（G20）	336	273.35	350.62	534.44	59.4	179.08	92.42	179.67	363.61	550.9

图 **1**　植物干重数量统计

4.1.3 对植被量的影响模型建立

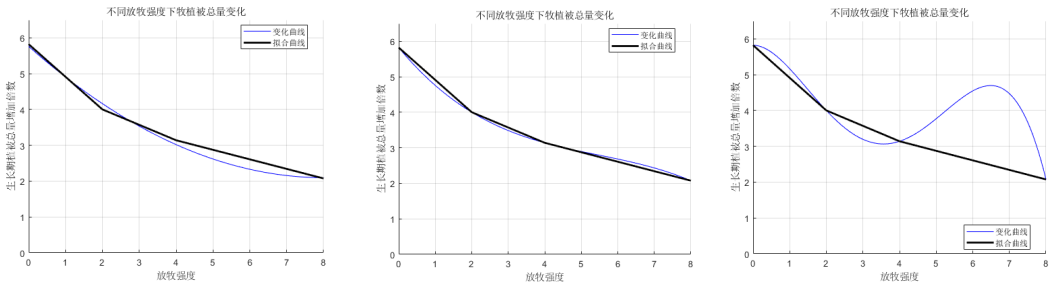
由于不同场地原有植物干重不同，本文分析放牧策略对植被生物量影响采用分析放牧前后植物干重比例 R。不同牧区相同放牧策略计算加权结果。具体方式为：设某年某牧区牧前干重为 X_i，牧区分别标号 1 到 n，以 2016 年数据为例，加权比例计算方法为

$$\bar{R}_{2016} = \frac{\sum_{i=1}^{n} X_i * R}{\sum_{i=1}^{n} X_i} \tag{1}$$

得到相同方式某年内牧前牧后干重比值后，计算 2016 年到 2020 年比值平均数。

$$R_{Average} = \frac{\sum_{i=2016}^{2020} \bar{R}}{5} \tag{2}$$

将放牧强度和方式作为自变量，得到的 $R_{Average}$ 作为因变量，做拟合计算，拟合方式采用最小二乘法。多项式拟合需要对 n 次多项式 $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \ldots \theta_n x^n$ 求其参数。最小二乘法就是找到一组 $\theta\,(\theta_0, \theta_1, \theta_2, \ldots, \theta_n)$ 使得残差平方和 $\sum_{i=1}^{n} \left(h_\theta\left(x_i\right) - y_i\right)^2$ 最小。在本文中分别对数据进行二次拟合、三次拟合、四次拟合，结果如下：



二次拟合结果　　三次拟合结果　　四次拟合结果

图 **2**　多次集合结果对比

结果表面二次拟合结果欠拟合而四次拟合结果过拟合，三次拟合效果较好。最终采用三次拟合方式。结果如下：

$$R = 5.8223 - 1.2422(\gamma S) + 0.1893(\gamma S)^2 - 0.0116(\gamma S)^3 \tag{3}$$

随着放牧强度的增大，植被放牧后相较于放目前干重比例逐步减小。越大的放牧强度对植被生长的影响越大。

4.1.4 对土壤湿度影响模型建立

根据文献 [1], 土壤含水量是衡量土壤监视程度和渗透率的重要指标 [2]，影响到植物生长、群落组成、水土涵养等方面。放牧对土壤含水量的影响在于牲畜对植被啃食和践踏的直接影响和通过对植物的影响间接影响水分吸收利用。文献 [4, 5, 6] 指出土壤含水量和放牧强度呈负相关，土壤湿度随着放牧强度的增加而减少。

土壤湿度与土壤含水量本质相同，只是表现方法上有所不同，因此可以通过土壤含水量反映土壤湿度，根据文献 [12] 可以得到土壤-植被-大气系统的水平衡基本方程，土壤含水变化量可以通过下式计算：

$$\Delta W = W_{t+1} - W_t = P + G_u + R_{\text{in}} - (Et_a + G_d + R_{\text{out}} + IC_{\text{store}}) \tag{4}$$

其中 P 为降水量，G_u 和 G_d 分别为地下水毛管上升量和土壤水渗透量，Et_a 为实际蒸发量，R_{in} 和 R_{out} 分别为入和出径流量，IC_{store} 为植被截流量。
文献 [3] 进行不同放牧强度下土地机械组成和有机质含量对比，结果如图 3。放牧强度增大，牲畜越加频繁的践踏作用破坏土壤结构，土壤压实作用越强，渗透率减小，孔隙度减小，吸收水分能力降低。土壤有机质含量减小，松散保水量大的黏粒比例减小，密度较大的沙粒含量增大。

土地利用	黏粒	粉粒	砂粒	有机质
禁牧	19. 11	32. 06	48. 43	2. 43
轻牧	16. 85	28. 96	54. 19	2. 07
中牧	17. 19	29. 93	52. 71	1. 71
重牧	13. 90	21. 96	64. 14	1. 49
耕地	14. 02	21. 40	64. 57	1. 04

图 **3**　研究区土壤机械组成和有机质含量

通过以上分析，放牧践踏草地，会使得土壤更加紧实，带来的影响是土壤密度更大，透水性更差，主要对土壤水渗透量 G_d 和实际蒸发量 Et_a 有较大的影响，

其中土壤渗透量与放牧强度呈负相关，而由于土壤变得更紧实，降水多存储于土壤表面（$0 \sim 10cm$），导致了实际蒸发量的增加，与放牧强度呈正相关。因此，我们引入土壤水渗透量因子 α_a 以及实际蒸发量因子 α_d。

因此，对式(4)进行修改：

$$\Delta W = P + G_u + R_{\text{in}} - (\alpha_a \times Et_a + (1 + \alpha_d) \times G_d + R_{\text{out}} + IC_{\text{store}}) \quad (5)$$

需要建立 α_a 和 α_d 和放牧强度的对应关系，通过最小二乘法拟合得到对应结果。由于无法搜集到相关数据进行拟合，我们通过相关文献给出一个经验数值：

表 **1**　放牧强度与土壤水渗透因子、蒸发因子经验数值

系数 \ 放牧强度	对照（NG）	轻牧（LGI）	中牧（MGI）	重牧（HGI）
$\boldsymbol{\alpha_a}$	0.0	-0.02	-0.04	-0.06
$\boldsymbol{\alpha_d}$	0.0	0.02	0.04	0.06

4.2 问题二求解

4.2.1 问题分析

根据所给的土壤湿度数据、土壤蒸发数据以及降水等数据，在保持放牧政策不变的情况下，预测未来的土壤湿度数据。已有 2012 年 1 月到 2022 年 3 月的相关数据，每个月有一个数据点，共计 111 个数据点，该数据为时间序列数据，并且用于相关数据作为协变量，因此我们考虑使用时间序列预测方法进行预测。

由于**数据量较小**，在进行模型训练之前需要对数据进行增强预处理，同时应该简化模型复杂度，降低模型训练难度。**同时考虑到土壤含水量一般受季节影响较大，我们在训练模型的时候应该考虑序列的周期性质，通过增加月份作为训练数据从而加入了序列的周期性质**。

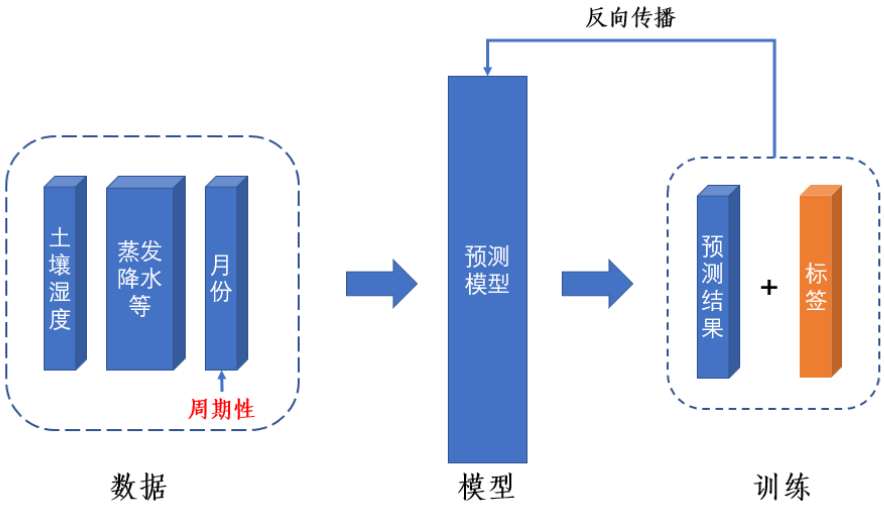


图 **4**　基于神经网络的序列预测方法

上述方法需要使用降水量、蒸发量等数据作为协变量预测未来的土壤湿度，而 2022 年 4 月分之后的降水量和蒸发量数据无法获取，如图6，**从解决实际问题角度出发，我们应更关注据当前时间较近的未来土壤湿度，而对于较远的未来数据更多的是关注其变化趋势**。

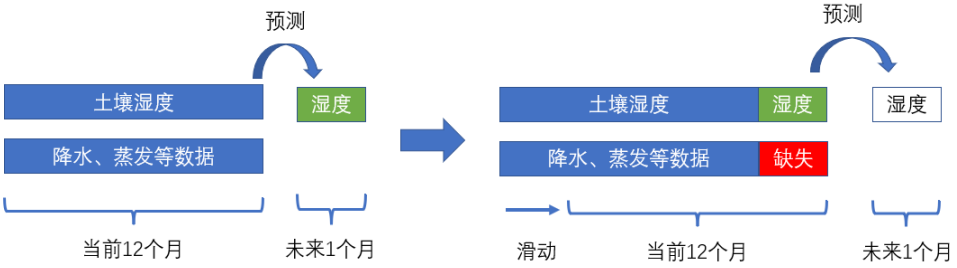


图 **5**　相关变量在预测过程中的缺失

因此，我们**对 2022 年 4 月的土壤湿度数据使用多变量的时间序列预测**，而**对 2022 年 5 月到 2023 年 12 月使用单变量自回归的方法进行预测**。

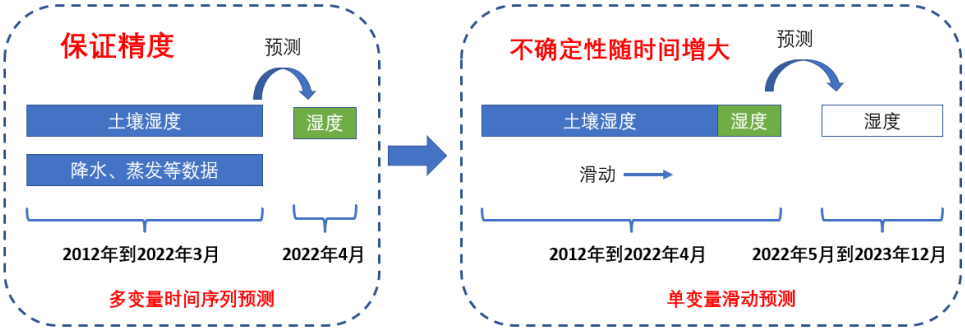


图 **6**　分段预测策略

4.2.2 数据预处理与样本生成

使用滑动窗口的方法生成样本，由于样本数量的限制，我们使用当前 12 个月的数据对第 13 个月的数据进行预测，因此在生成样本时滑窗的长度为 12。

记原始数据集 $X = \{x_1, x_2, ..., x_n\}$，$X \in R^{N \times V}$，其中 N 表示数据个数，V 表示数据的特征维度，在本文中，$N = 111$，$V = 6$。

记 $D = \{d_1, d_2, ..., d_m\}$ 为训练样本，$L = \{l_1, l_2, ..., l_m\}$ 为标签数据，则 d_k 的表达式为：

$$\begin{cases} d_k &= \{x_k, x_{k+1}, ..., x_{k+SIZE}\} \\ l_k &= \{x_{k+1}, x_{k+2}, ..., x_{k+SIZE+1}\} \end{cases} , k \in [1, N - SIZE] \tag{6}$$

其中，$SIZE$ 为窗口大小，在本文中 $SIZE = 12$。得到样本数据之后以 $8:2$

的比例进行训练集与测试集的划分，最终得到的训练样本个数为 85，测试样本为 13 个。

数据在输入到模型之前需要进行归一化处理：

$$X_{std} = \frac{X - X \cdot \min(axis = 0)}{X \cdot \max(axis = 0) - X \cdot \min(axis = 0)} \tag{7}$$

$$X_{scaled} = X_{std} * (\max - \min) + \min \tag{8}$$

选取样本最大值 X_{max} 和最小值 X_{min}，计算当前特征量和最小值之差 $X - X_{min}$ 与最大值与最小值之差 $X_{max} - X_{min}$ 的比值，归一化到 $[-1, 1]$ 区间，参与网络训练。

4.2.3 模型建立

本题针对 $10cm$ 深度湿度至 $200cm$ 深度湿度数据分别建立 4 组模型。LSTM 网络通常被用来做时间序列的预测，其对传统 RNN 进行了改进，缓解了 RNN 的梯度消失和梯度爆炸问题，我们选择使用 LSTM 网络。

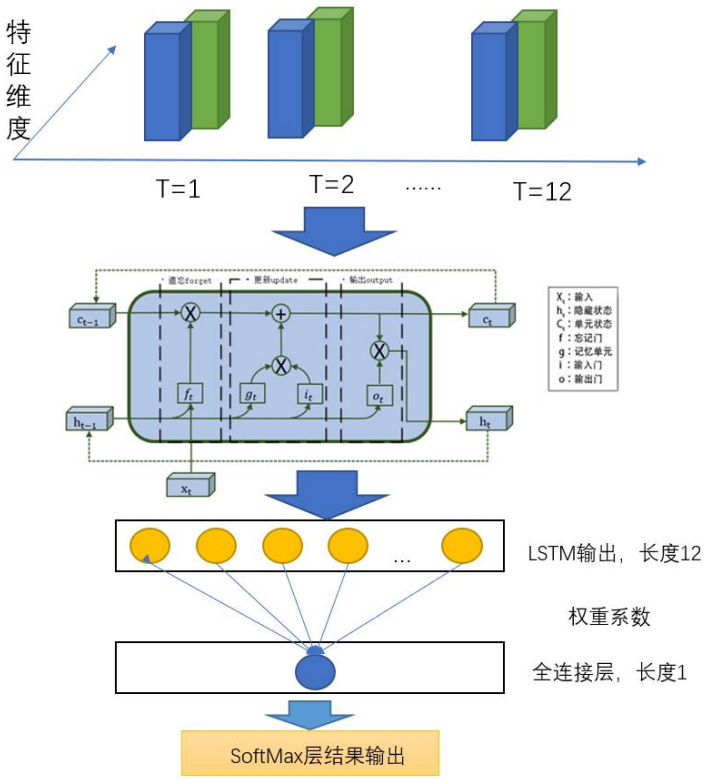


图 7　时间序列预测模型

模型的输入层首先是一个**单层单向**的 LSTM 模块，通过全连接层对每一个

时间步的状态向量加权求和，最终得到一个标量作为预测输出，通过反向传播训练网络参数。

多变量时间序列预测的输入数据特征维度为 6，分别是土壤湿度、土壤蒸发量 (W/m^2)、土壤蒸发量 (mm) 降水量、降水天数以及月份，单变量的滑动预测输入特征维度为 1。两种模型结构类似，可训练参数统计如下：

表 **2**　模型可训练参数统计

模型	可训练参数	模型总参数
多变量时间序列预测模型	333953	333953
单变量时间序列预测模型	331393	331393

算法流程图如图 6：

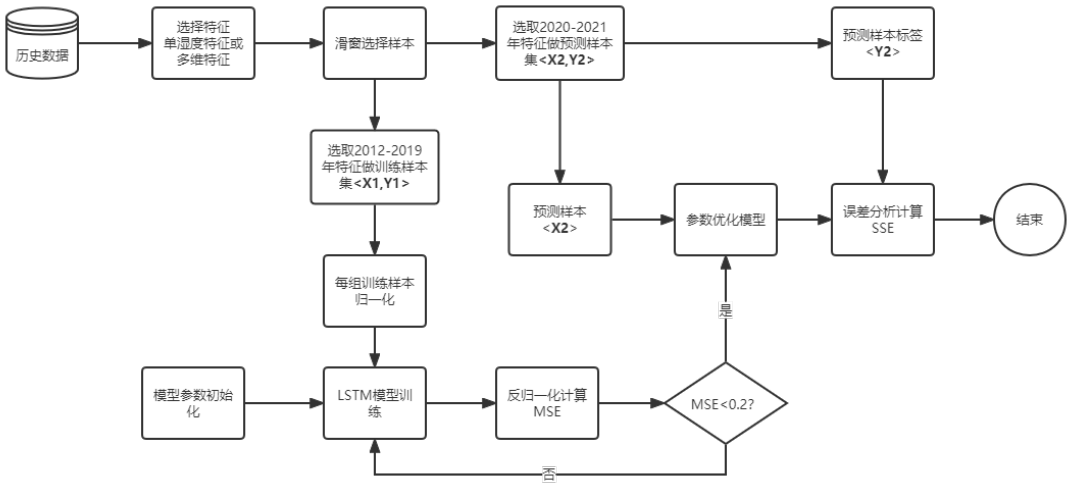


图 **8**　训练算法流程

4.2.4 结果与误差分析

各个深度土壤湿度多变量单步预测结果如图9，其中绿色的线为预测值，蓝色的线为测试集真值，绿色的线比蓝色的线长度长 1，长出的为 2022 年 4 月分土壤湿度预测结果，四种不同土壤深度对应了四个模型。图10为训练集上的损失曲线，图11为测试集上的损失曲线，从测试集模型表现来看，均方误差损失小于 0.2。

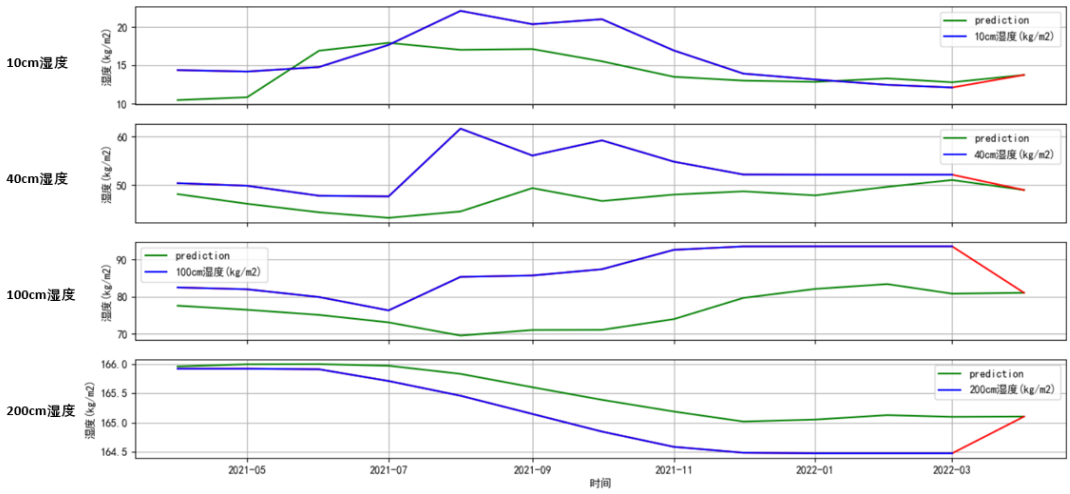


图 **9**　**2022** 年 **4** 月不同深度土壤湿度预测

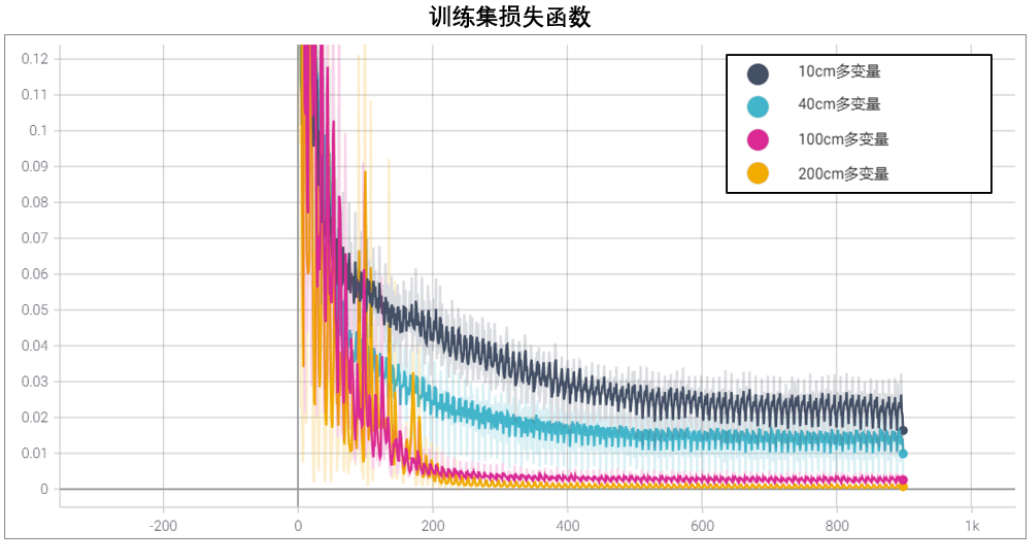


图 **10**　**2022** 年 **4** 月预测模型训练集 **loss**

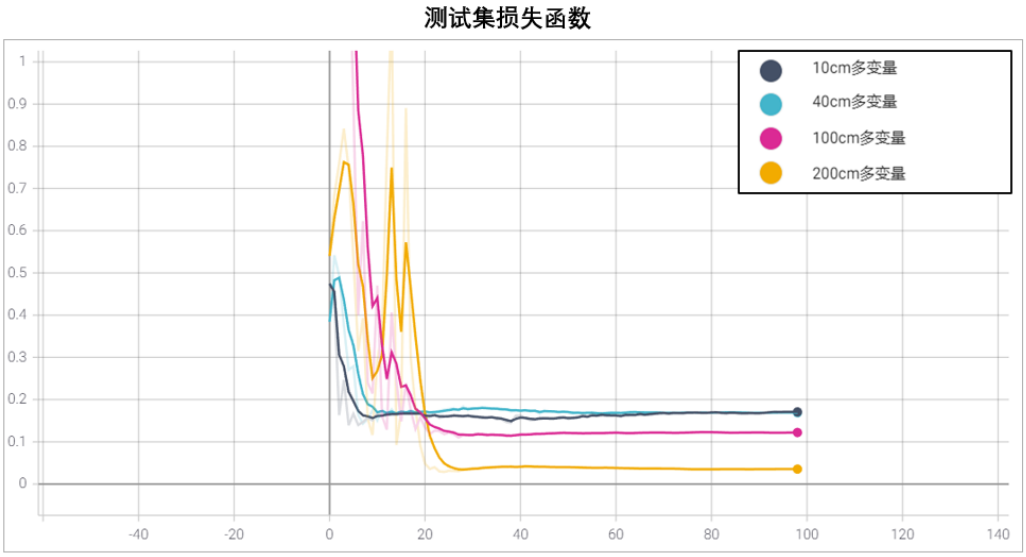


图 **11**　**2022** 年 **4** 月预测模型测试集 **loss**

对不同的土壤深度预测 2022 年 5 月到 2023 年 12 月的土壤湿度，四种不同土壤湿度对应四个模型，预测结果见图12。

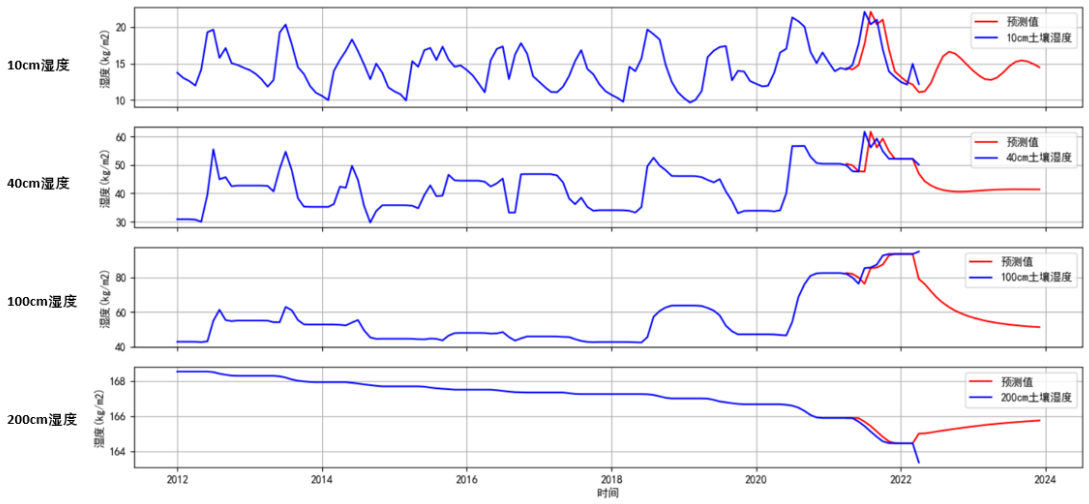


图 **12**　**2022** 年 **5** 月到 **2023** 年 **12** 月不同深度土壤湿度预测结果

通过多特征模型网络精确预测 2022 年 4 月湿度，单湿度特征网络预测此后月份湿度，具体数据见表 1。

表 **3**　**2022** 年、**2023** 年不同深度土壤湿度预测值

年份	月份	10cm 湿度 (kg/m2)	40cm 湿度 (kg/m2)	100cm 湿度 (kg/m2)	200cm 湿度 (kg/m2)
2022	4	12.16	50.05	94.81	164.78
	5	11.17439	44.22797	80.66245	165.04539
	6	12.29622	42.75946	76.52636	165.09477
	7	14.24535	41.7799	72.88776	165.15135
	8	15.97095	41.15428	69.96277	165.20571
	9	16.61589	40.78792	67.61614	165.25662
	10	16.35291	40.61488	65.71623	165.30524
	11	15.64413	40.58461	64.16106	165.35229
	12	14.80206	40.65127	62.87446	165.39793
2023	1	13.9881	40.77353	61.80048	165.44152
	2	13.3035	40.91807	60.89781	165.48281
	3	12.8516	41.06057	60.13534	165.52189
	4	12.74532	41.18561	59.48888	165.55884
	5	13.06889	41.2823	58.93784	165.59139
	6	13.77071	41.34755	58.46699	165.62248
	7	14.5972	41.38502	58.06337	165.65182
	8	15.20956	41.40078	57.71642	165.67944
	9	15.42555	41.40117	57.41744	165.70548
	10	15.28392	41.39198	57.15923	165.73005
	11	14.92176	41.37797	56.93578	165.75324
	12	14.47303	41.36268	56.74208	165.77515

4.3 问题三求解

4.3.1 问题分析

由于本题所求化学性质含量为小样本不确定性系统，且需要利用已有数据预测未来值，使用传统机器学习网络由于样本量稀少不能保证训练结果质量，本题使用 GM(1,1) 灰色预测模型对已知数据建模并进行对 2022 年同期土壤化学数据的预测。

由于题目要求预测模型需要反映放牧强度对土壤化学性质的的影响，我们对 GM(1,1) 进行了修改：在计算累加数据时，根据不同的放牧强度给数据进行加权，通过查阅相关文献资料以及实验确定具体的权重值，以此增加放牧强度对土壤化学物质的影响。

4.3.2 模型建立

GM(1,1) 灰色预测首先对数据序列进行累加 $x^{(1)}(k)=\sum_{i=1}^{k}x^{(0)}(i)$ 用来弱化数据随机性，更明显显示规律性。假设序列 $x^{(1)}(k)$ 满足微分方程 $\frac{dx^{(1)}}{dt}+ax^{(1)}=u$。由于此微分方程的数据是离散的，所以 $\frac{dx^{(1)}}{dt}$ 写成 $\frac{\Delta x^{(1)}}{\Delta t}$。由于系统中步长为 1，$\Delta t=t-(t-1)=1$，$\Delta x^{(1)}=x^{(1)}(t)-x^{(1)}(t-1)=x^{(0)}(t)$。可以得到方程 $x^{(0)}(t)+ax^{(1)}=u$，即

$$x^{(0)}(t)=-ax^{(1)}+u \tag{9}$$

由于 $x^{(0)}(t)$ 以及 $x^{(1)}(t)$ 为实际数据，可以使用最小二乘法求解参数 a 和 u。进一步为了消除数据随机性，将 $x^{(1)}(k)$ 修正为 $z^{(1)}(k)$。其中 $z^{(1)}(k)$ 取前后两个时刻均值更为合理。文献 [7] 提出重度放牧会增加土壤有机碳含量。文献 [8] 指出牧区土壤氮含量随放牧强度影响程度为中牧 > 重牧 > 轻牧 > 对照组，高强度放牧条件下家畜排泄粪尿能够逐步增加土壤全氮含量并且随着放牧轮次增加而影响逐步增大，但是牧区土壤全氮含量始终不如对照封闭区全氮含量高。应用文献 [9] 提出的改进的灰色预测模型，在本题中，将放牧强度作为影响因子参与灰色预测，具体公式为：

$$z^{(1)}(t)=(0.5+\beta)x^{(1)}(t)+0.5x^{(1)}(t-1),\mathrm{t}=2,\ldots\mathrm{n} \tag{10}$$

其中 β 为和放牧强度有关的修正系数，其范围为 $\beta\in[-1,1]$。根据文献 [7, 8] 和实际数据，计算 SOC 土壤有机碳时 β 修正为 $\{'LGI':0.02,'MGI':0.04,'HGI':0.08,'NG':-0.06\}$，计算全 N 含含量时修正为 $\{'LGI':0.01,'MGI':0.05,'HGI':$

$-0.09,' NG' : 0.0\}$。由于缺少相关研究背景，计算 SIC 土壤无机碳时不考虑放牧强度影响，只做简单灰度预测。将修正系数带入 $z^{(1)}(t)$ 后方程改为

$$x^{(0)}(t) = -az^{(1)}(t) + u \tag{11}$$

其中 $x^{(0)}(t)$ 与 $z^{(1)}(t)$ 均为已知数据，剩下的未知参数 a 和 u，使用最小二乘法计算求解得到 $\widehat{a}$ 与 $\widehat{u}$。带入原微分方程求出 $\widehat{x}^{(1)}(k)$ 拟合值。则预测值为：$\widehat{x}^{(0)}(k) = \widehat{x}^{(1)}(k) - \widehat{x}^{(1)}(k-1)$。改进的灰色预测算法流程如图：

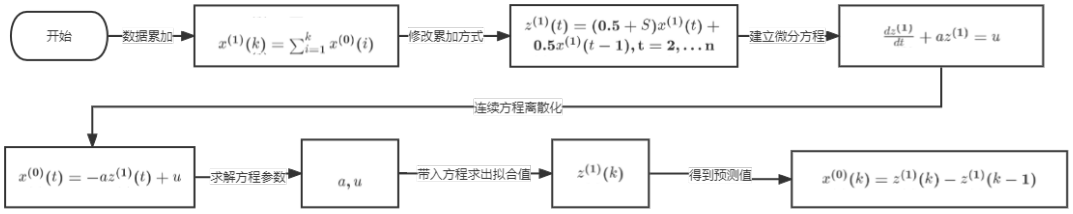


图 **13**　改进的灰色预测算法流程

4.3.3 结果与分析

使用附件 14 中数据，不同放牧强度和放牧小区中不同化学性质按照年份建立本文所提的改进灰色预测模型，将放牧强度作为影响因子引入模型。同一年有多个数据的使用平均数据作为当年数据。建立改进灰度模型并对 2022 年数据进行预测。模型预测得到 SOC 土壤有机碳含量、SIC 土壤无机碳含量和土壤全 N 含量，$STC = SOC + SIC$。土壤 C/N 比为 STC/N。计算结果如下：

对四种放牧强度分别选取其中一个小区，可视化预测结果如图14~17，其中绿线为历史数据。

表 **4**　**2022** 年土壤化学成分预测值

放牧强度	Plot 放牧小区	SOC 土壤有机碳	SIC 土壤无机碳	STC 土壤全碳	全 N	土壤 C/N 比
NG	G17	18.1142	4.9064	22.2812	2.3358	9.7069
	G19	18.1956	3.7306	21.2857	2.4239	8.9303
	G21	23.0461	3.1443	24.6886	2.8121	8.8787
LGI	G6	14.942	2.1045	15.5606	2.2521	6.8174
	G12	16.7033	2.4806	17.8945	2.0588	8.7411
	G18	22.6584	5.2947	26.6213	2.5875	10.4621
MGI	G8	14.2765	1.7331	14.9243	2.0082	7.4058
	G11	14.8084	2.5571	16.8519	2.0303	8.2656
	G16	15.0881	9.5614	24.6619	1.8139	13.7742
HGI	G9	18.1461	2.8346	19.8592	2.4251	7.7893
	G13	17.2798	3.3587	20.2327	2.1715	9.3324
	G20	15.7886	3.7866	18.9832	2.1131	9.0003

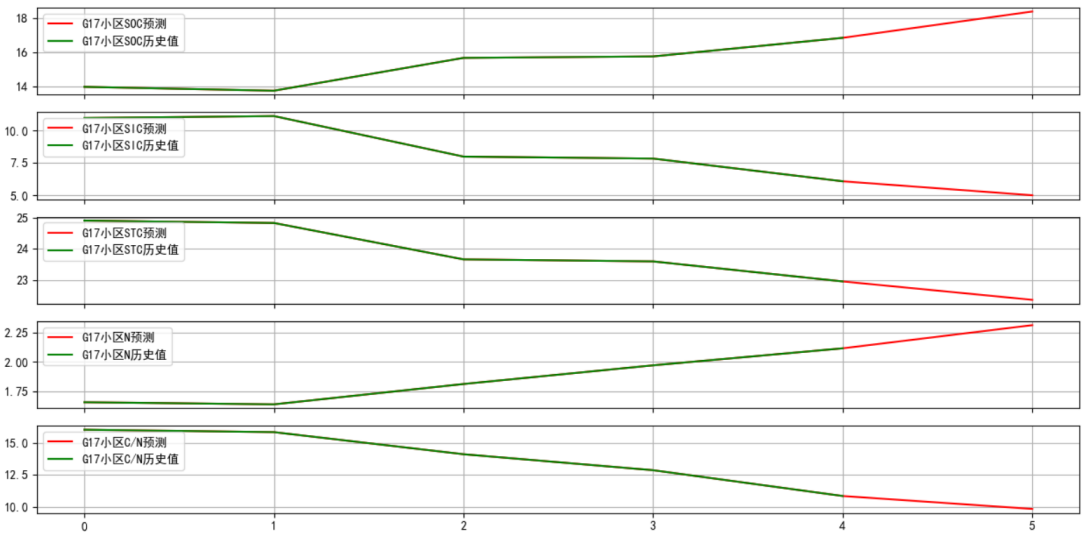


图 **14**　对照组 **G17** 小区预测结果

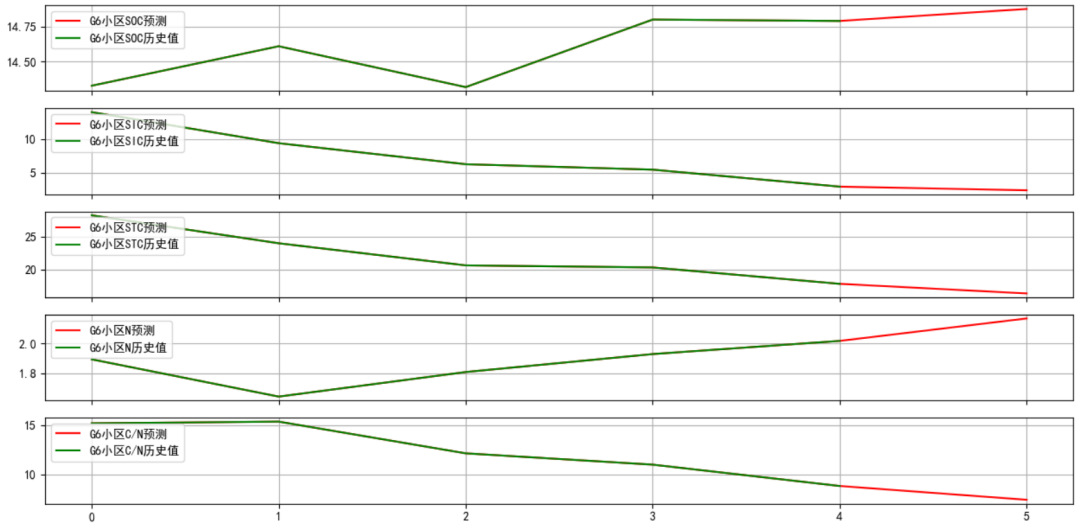


图 **15**　轻牧组 **G6** 小区预测结果

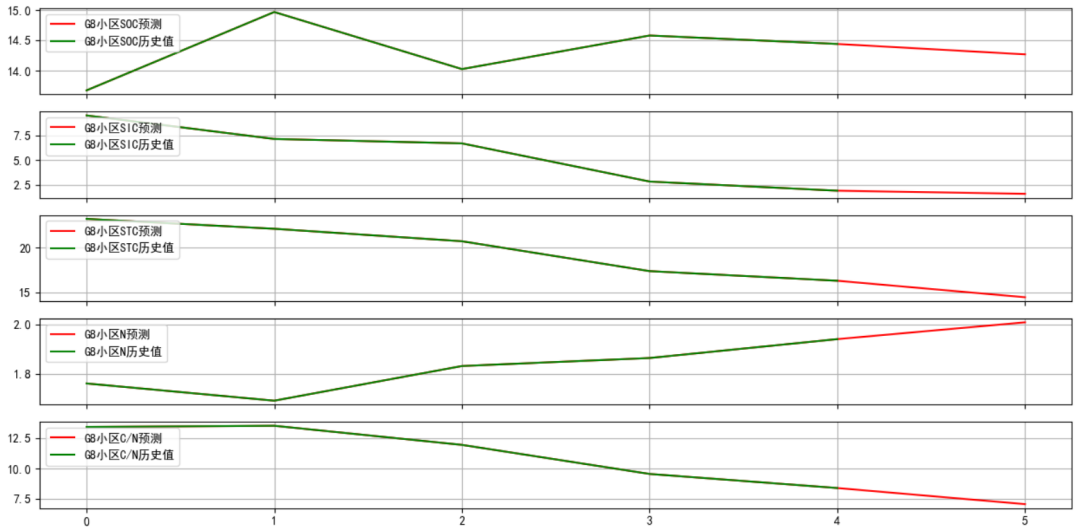


图 **16**　中牧组 **G8** 小区预测结果

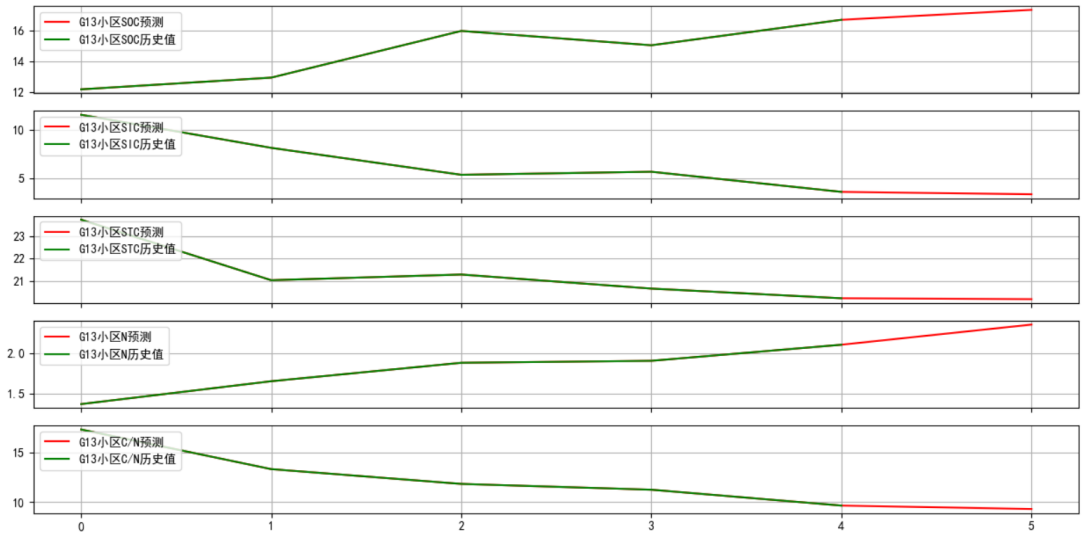


图 **17**　重牧组 **G13** 小区预测结果

12 个放牧小区的预测相对校验平均误差为 0.031，各小区不同土壤化学性质的预测相对校验误差均小于 0.2，处于可接受范围，其中土壤无机碳 (SIC) 的预测相对校验误差如表5。

表 **5**　各放牧小区土壤无机碳 **(SIC)** 预测相对校验误差

放牧小区	放牧强度	相对校验误差	放牧小区	放牧强度	相对校验误差
G17	NG	0.0413	G8	MGI	0.1545
G19	NG	0.0284	G11	MGI	0.0247
G21	NG	0.0415	G16	MGI	0.0849
G6	LGI	0.0690	G9	HGI	0.1011
G12	LGI	0.1116	G13	HGI	0.0928
G18	LGI	0.0909	G20	HGI	0.0513

4.4 问题四求解

4.4.1 问题分析

由于本题背景给出了沙漠化程度指数 SM 的公式，如式12，在多因子模型中需要决定各个因子的相对权重。使用层次分析法计计算各个因子权重大小并建立最终模型。类似于沙漠化程度指数模型，土壤板结化指数同样建立相关多因子模型。

4.4.2 模型因子 Q_i 选择

由文献 [10] 得到沙漠化程度指数预测模型表达式为：

$$SM = \eta \cdot \sum_{i=1}^{n} S_{Q_i} = \eta \cdot \sum_{i=1}^{n} (Q_i \cdot W_{c_i}) \tag{12}$$

其中 SM 表示沙漠化程度指数，η 为调节系数，模型有 n 个因子，Q_i 表示指标中第 i 个因子强度。W_{c_i} 表示因子权重系数。文献 [10] 中给出了塔里木地区相关因子权重如表6。

表 **6**　文献 **[10]** 因子以及权重

权重系数 WA	影响因素 Bi	权重系数 Wbi	指标因子 Qi	权重系数 Wci
1	气象因素 B1	0.3275	暖期平均风速 Q1	0.1802
			暖期降水量 Q2	0.0787
			暖期平均气温 Q3	0.0685
	地表因素 B2	0.4126	暖期植被因子 Q4	0.2036
			暖期地表水资源因子 Q5	0.0808
			暖期地下水资源因子 Q6	0.1282
	人文因素 B3	0.2599	人口密度 Q7	0.0509
			牲畜密度 Q8	0.1282
			人均家庭经营纯收入 Q9	0.0808

由于本文背景差异，我们重新给定指标因子和权重系数。文献 [10, 11] 指数沙漠化因子分别考虑气象因素、地表因素、人文因素。本文选择因子为风速因

子、降水量因子、温度因子、植被盖度因子、地表水资源因子、地下水资源因子、放牧强度因子。由于 5-9 月为植物生长月，只考虑植物生长暖季的相关因子。

4.4.3 气象因素

土地沙漠化的气象因子通常考虑风速因子、降水量因子、温度因子。对于风速因子，风速 V_{wind} 和沙漠化程度成正比，平均风速大于 $8(knots)$ 强度归一化为 1，平均风速小于 $4(knots)$ 强度归一化为 0，两者之间归一化为 $\frac{V_{wind}-4(knots)}{8(knots)-4(knots)}$ 。

降水量和沙漠化呈反比，5-9 月暖季降水量 P 大于 $600mm$ 强度取 1，小于 $100mm$ 强度取 0，介于两者归一化 $\frac{P-100mm}{600mm-100mm}$。

对于温度因子 T，由于植物生长最适宜温度为 25 °C 至 30 °C 处于此温度范围因子设为 0，抑制植物生长的最低气温为 5 °C，最高气温为 35 °C，高于 35 °C 或低于 5 °C 因子设为 1，对于 5 °C 至 25 °C 之间的温度归一化为 $\frac{25\,^\circ\mathrm{C}-T}{25\,^\circ\mathrm{C}-5\,^\circ\mathrm{C}}$，对于 30 °C 至 35 °C 之间的温度归一化为 $\frac{35\,^\circ\mathrm{C}-T}{35\,^\circ\mathrm{C}-30\,^\circ\mathrm{C}}$。

4.4.4 地表因素

地表因素考虑植被盖度因子、地表水资源因子、地下水资源因子。植被指数 (NDVI) 与沙漠化程度呈反比。植被指数为 1 时归一化强度为 1；植被指数为-1 时归一化强度为 0；植被系数介于 (-1,1) 时，归一化为 $\frac{NDVI-(-1)}{1-(-1)}$ 。

由于地表水资源没有直接数据表示，使用土壤 $10cm$ 深度湿度 RH_{10cm} 表示地表水资源系数。地表水资源越多沙漠化程度越低。RH_{10cm} 大于 $20kg/m^2$ 归一化强度为 1，土壤 $10cm$ 深度湿度小于 $10kg/m^2$ 归一化强度为 0，介于两者的数据归一化为 $\frac{RH_{10cm}-10kg/m^2}{20kg/m^2-10kg/m^2}$

与之类似的地下水资源没有直接数据，使用壤 $100cm$ 深度湿度 RH_{100cm} 表示地下水资源系数。地下水资源越多沙漠化程度越低。RH_{100cm} 大于 $80kg/m^2$ 归一化强度为 1，土壤 $100cm$ 深度湿度小于 $40kg/m^2$ 归一化强度为 0，介于两者的数据归一化为 $\frac{RH_{100cm}-40kg/m^2}{80kg/m^2-40kg/m^2}$

4.4.5 人文因素

放牧强度因子考虑牲畜密度，牲畜密度越高对土地沙漠化影响越大。对照组归一化为 0，轻度放牧归一化为 0.25，中度放牧归一化为 0.5，重度放牧归一化

为 1。

4.4.6 因子权重 W_{c_i} 计算

本文采取七个影响因子参与计算，采用层次分析法 (AHP, Analytic Hierarchy Process) 计算因子权重。具体步骤如下

1. 构造判断矩阵
   根据文献 [11] 经验，采用 1 9 标度方法构建影响沙漠化各指标因子之间重要性比较判断矩阵。
2. 计算权重
   计算判断矩阵最大特征值 $\lambda_{\max}$ 和对应特征向量 W，W 标准化后即为同层次元素相对于上一层次因素权重
3. 一致性检验
   一致性指标是用来衡量矩阵不一致程度的数量指标，用来判断矩阵可信程度。本文使用 CR 修正 CI 后得到的 CR 代替 CI 作为一致化检验的指标，当 $CR <= 0.1$ 时认为判断矩阵具有可接受的一致性。其中

$$\begin{aligned} CI &= (\lambda_{\max} - n)/(n-1) \\ CR &= CI/RI \end{aligned} \tag{13}$$

n	1	2	3	4	5	6	7	8	9	10	11
RI	0	0	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49	1.51

图 **18**　一致性指标 RI

判断矩阵如下表：

表 **7**　**A-B** 层判断矩阵、权重以及一致性检验

A	B1	B2	B3	权重 W	一致性检验
B1	1	1/5	2/3	0.33131	λmax=3.0735
B2	2/3	1	1	0.28943	CI=0.036757
B3	1.5	1	1	0.37926	CR=0.063374≤0.1

表 **8**　B_1 与 $C_1 - C_3$ 层判断矩阵、权重以及一致性检验

B1	C1	C2	C3	权重 W	一致性检验
C1	1	2	3	0.5503	λmax=3.02
C2	1/2	1	1	0.2404	CI=0.01
C3	1/3	1	1	0.2093	CR=0.0172≤0.1

表 **9**　B_2 与 $C_4 - C_6$ 层判断矩阵、权重以及一致性检验

B2	C4	C5	C6	权重 W	一致性检验
C4	1	2	2	0.4934	λmax=3.05
C5	1/2	1	1/2	0.1958	CI=0.025
C6	1/2	2	1	0.3108	CR=0.0431≤0.1

4. 因子权重合成

表 **10**　整体权重

	权重系数 WA	影响因素 Bi	权重系数 Wbi	指标因子 Qi	权重系数 Wci
				暖期平均风速	0.1823
		气象因素 B1	0.3313	暖期降水量	0.0796
				暖期平均气温	0.0693
沙漠化预警指标体系	1			暖期植被因子	0.1428
		地表因素 B2	0.2894	暖期地表水资源因子	0.0566
				暖期地下水资源因子	0.0899
		人文因素 B3	0.3792	放牧强度	0.3792

4.4.7 沙漠化程度指数值计算

得到沙漠化程度指数影响因素和相关权重系数后，使用公式 $SM = \eta \cdot \sum_{i=1}^{n} S_{Q_i} = \eta \cdot \sum_{i=1}^{n} (Q_i \cdot W_{c_i})$ 验证所得模型。由于附件所给数量为全锡林郭勒盟家畜头数，全盟牧场面积无法得知，使用牲畜数量代替放牧强度定性分析近

10 年锡林郭勒沙漠化指数。将家畜头数大于 14000000 归一化为 1，小于 10000000 归一化为 0，之间的数量归一化为 $(Count - 10000000)/(14000000 - 10000000)$。定性计算近 10 年锡林郭勒盟沙漠化程度指数如图：

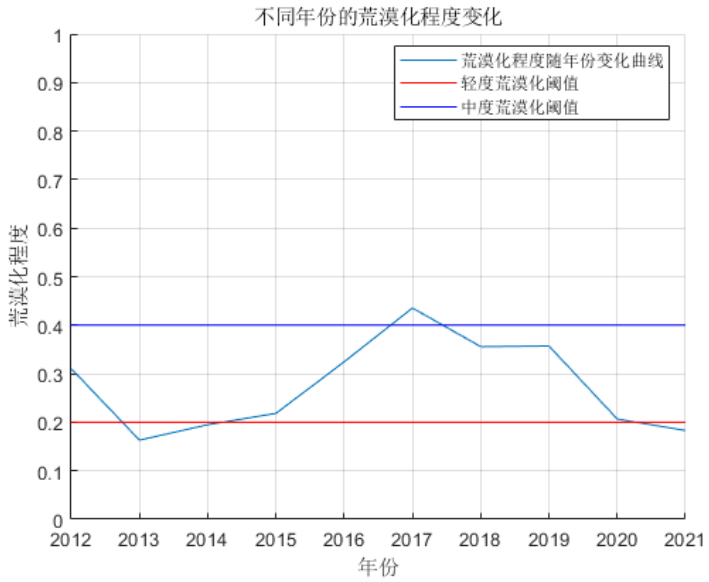


图 **19**　近十年土壤沙漠化趋势计算

4.4.8 土壤板结化公式推导

土壤板结化程度定性描述为:

$$B = f(W, C, O) \tag{14}$$

其中 W 为土壤湿度，C 为土壤容重，O 为有机物含量。计算土壤板结化指数采用层次分析法计算因子权重。

由于土壤板结化主要表现在近地面图层。土壤湿度使用 10cm 深度湿度进行计算。附件 14 所给土壤湿度 W_{10cm} 单位为 kg/m^2，高度为 $0.1m$，换算为 $W_{10cm}/0.1m = 10W_{10cm}kg/m^3 = \frac{W_{10cm}}{100}g/cm^3$。湿度因子归一化为湿度小于 5% 为 1，大于 30% 归一化为 0。介于二者之间的数据归一化为 $\frac{30\%-W_{10cm}}{30\%-5\%}$。

土壤容重 pb 由公式 $\frac{g*100}{V*(100+w)}$ 计算而得。其中 g 为土壤重量，V 为土壤体积，w 为土壤湿度。容重因子归一化为小于 1.1 为 0，大于 1.4 为 1，介于二者之间数据归一化为 $\frac{pb-1.1}{1.4-1.1}$。

土壤有机物含量使用土壤有机碳含量 SOC。有机碳含量小于 12 时归一化为 1，大于 22 时归一化为 0，介于二者时间数据归一化为 $\frac{SOC-12}{22-12}$ 。

构建判断因子计算权重并通过一致性检测。

表 **11**　土壤板结化程度判断矩阵、权重以及一致性检验

A	B1	B2	B3	权重 W	一致性检验
B1	1	1	2	0.4066	λmax=3.0092
B2	1	1	1.5	0.3695	CI=0.0046014
B3	0.5	2/3	1	0.2238	CR=0.0079334≤0.1

则土壤板结化指数公式定义为

$$B = 0.4066\overline{W} + 0.3695\overline{C} + 0.2238\overline{O} \tag{15}$$

其中 $\overline{W}, \overline{C}, \overline{O}$ 为归一化后因子。

4.4.9 计算合理放牧策略

利用本题沙漠化程度指数公式和土壤板结化指数公式计算放牧策略模型。为了保持草地资源可持续发展，需要使牧场沙漠化程度在可承受范围内。其他因子不变的情况下改变放牧强度，计算沙漠化程度指数。自然因子和地表因子取 2016 年至 2021 年数据平均值参与计算。得到放牧强度和沙漠化指数的关系如图：

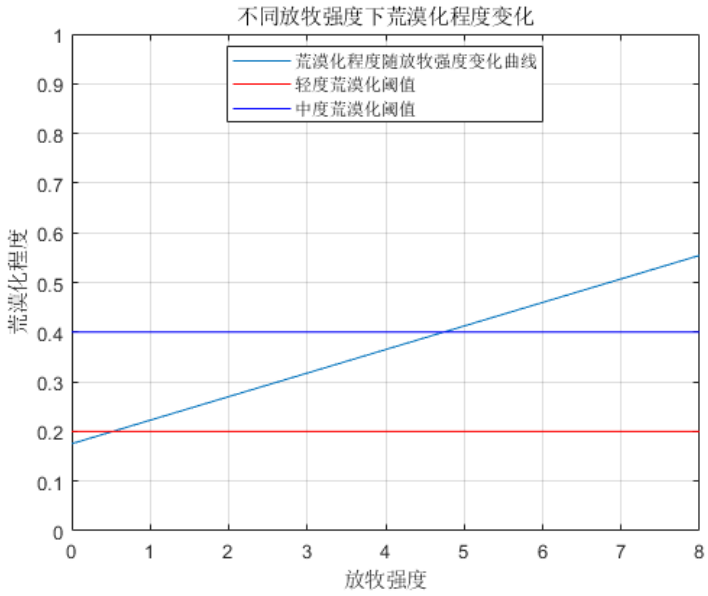


图 **20**　放牧强度和荒漠化之间关系

由图 12 可得，放牧强度 0.5 羊/天/公顷时达到非沙漠化指数最大阈值，放牧强度 4.7 羊/天/公顷时达到轻度沙漠化程度上限阈值。由于放牧强度和有机碳含量为负相关，有机碳含量和土壤板结化程度呈负相关，我们可以得出结论：放牧强度和土壤板结化程度呈正相关。降低放牧强度有助于抑制土壤板结化。我们可以认为为了避免沙漠化，结合沙漠化程度指数和板结化程度，放牧强度应该小于 0.5 羊/天/公顷。

4.5 问题五求解

4.5.1 问题分析

利用问题四所得土壤沙漠化程度指数和土壤板结化程度计算公式，改变降水量因子条件，在结果在可持续发展范围内反推放牧强度。

4.5.2 预测求解

设综合环境影响指数 S, 定义：

$$S = 0.7SM + 0.3B \tag{16}$$

令土壤沙漠化程度指数 $SM < 0.4$, 综合环境影响指数 $S < 0.5$。自然因素和地表因素除降水量外取锡林郭勒 2016 年至 2020 年暖期平均数据。改变年降水量 P_{year} 为指定 $300mm$ 到 $1200mm$, 结合近 10 年平均暖期降水量占全年降水量比例为 $n = 0.614$。暖期降水量 P_{warm} 因素取 $P_{warm} = 0.614P_{year}$，求满足条件的最大放牧强度单位为：羊/公顷/天。

如图，以 G18 牧区为例，在不同降水量条件下，画出放牧强队与土壤沙漠化和综合环境影响指数的关系，求得在轻度沙漠化条件下放牧强度最大阈值。

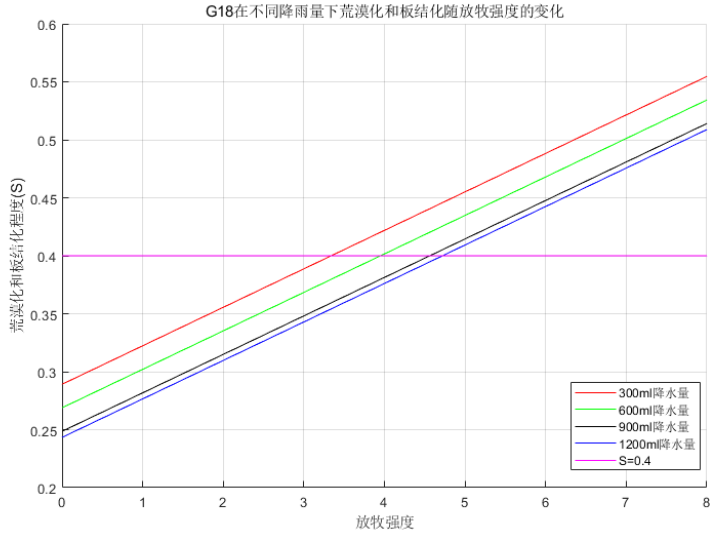


图 **21** **G18** 牧区不同年降水量下沙漠化和板结化随放牧强度变换关系

结合附件 14 和附件 15 数据，求得不同放牧小区在年降水量分别为 $300mm$，$600mm$，$900mm$，$1200mm$ 条件下满足轻度沙漠化范围内最大放牧阈值。结果如表

表 **12**　各放牧小区不同降水量下可持续发展条件 **(**轻度沙漠化**)** 下最大放牧强度

放牧小区	300mm	600mm	900mm	1200mm
G17	3	3.6	4.2	4.4
G19	3.1	3.7	4.3	4.4
G21	2.9	3.5	4.1	4.3
G6	2.9	3.5	4.1	4.3
G12	2.9	3.5	4.1	4.3
G18	3.3	3.9	4.5	4.7
G8	2.9	3.5	4.1	4.3
G11	2.8	3.5	4.1	4.2
G16	2.9	3.5	4.1	4.2
G9	2.8	3.5	4.1	4.2
G13	2.8	3.4	4.1	4.2
G20	2.8	3.4	4	4.2

4.6 问题六求解

4.6.1 问题分析

问题 4 求出了轻度沙漠化条件下最大放牧强度阈值为 4.7 羊/天/公顷，附件 13 中示范牧户放牧强度分别为：根据问题 1 中所得放牧策略对草原土壤湿度和植被生物量影响的模型，我们可以根据已知的放牧强度条件下可以预测湿度和植被生物量的变化情况。结合问题 3 中土壤化学性质变化建立模型计算放牧强度对土壤肥力变化的数学关系。

4.6.2 模型建立

根据文献 [14] 土壤肥力可以由土壤有机碳含量表示。根据附件 14 数据，首先求得 2012 年到 2020 年不同放牧强度下各个牧区有机碳平均值，并得到不同放牧强度下土壤肥力和时间的变化关系。并使用灰度预测模型预测 2023 年不同放牧强度下土壤肥力如图。

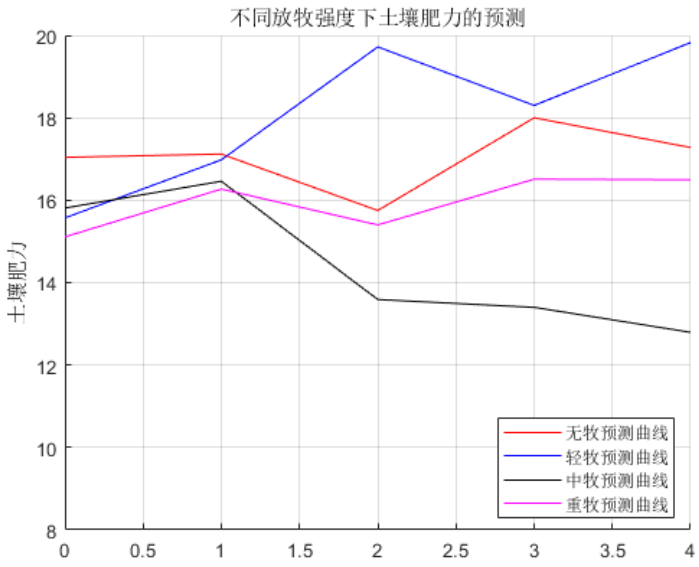


图 **22** **2023** 年四种放牧强度土壤肥力预测值

根据预测 2023 年土壤肥力和放牧强度数据建立放牧强度为自变量，土壤肥力为因变量的拟合方程，带入所需放牧强度求得对应土壤肥力，结果如表。

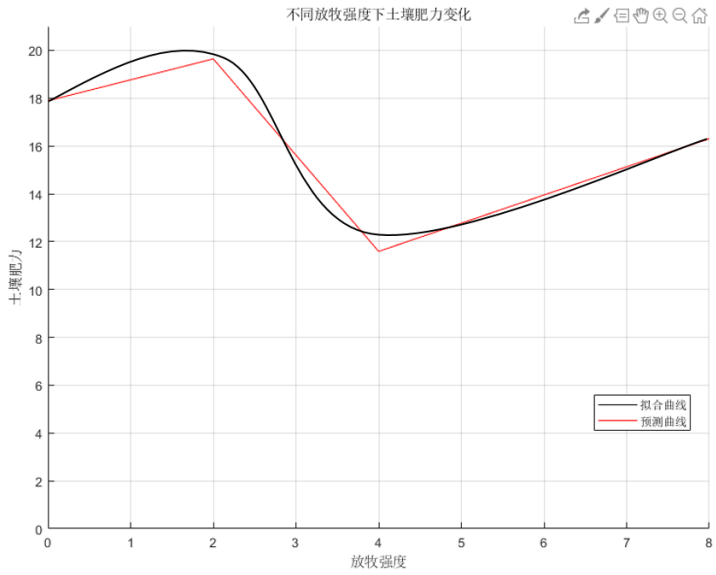


图 **23**　**2023** 年土壤肥力随放牧强度变化关系

土壤植被影响程度使用植被干重牧前与牧后比值 R 代替。使用问题 1 求得的放牧强度和土壤植被量拟合函数 $R = 5.8223 - 1.2422(\gamma S) + 0.1893(\gamma S)^2 - 0.0116(\gamma S)^3$ 带入放牧强度计算土壤植被牧前牧后比例。

土壤湿度使用问题 1 求得的放牧强度和 10cm 深度土壤湿度关系求出对应放牧强度下土壤湿度。

土壤肥力、土壤植被牧前牧后比值、土壤湿度和不同牧户放牧策略关系如表

表 **13**　不同放牧强度下 **2023** 年土壤肥力、土壤湿度、土壤植被含量预测值

牧户	放牧强度	土壤肥力 (SOC 含量)	土壤放牧前后植物干重比值	10cm 深度土壤湿度
牧户 1	2.68	17.3	3.63	15.23
牧户 2	2.13	19.5	3.92	15.31
牧户 3	5.38	13.2	2.81	14.28
牧户 4	2.49	18.4	3.72	15.28
无沙漠化对照组	0.5	18.4	5.25	15.76
轻度沙漠化对照组	4.7	12.6	3.72	15.38

参考文献

[1] 许宏斌. 不同放牧强度对呼伦贝尔草甸草原群落特征及群落生物量分布的影响 [D]. 内蒙古大学,2018.

[2] Mulqueen, John et al. “Evaluation of penetrometers for measuring soil strength.” Journal of Terramechanics 14 (1977): 137-151.

[3] 王悦骅. 模拟降水对不同载畜率放牧荒漠草原植物多样性的影响 [D]. 内蒙古农业大学,2019.DOI:10.27229/d.cnki.gnmnu.2019.000392.

[4] Abril A, Bucher E H. The effects of overgrazing on soil microbial community and fertility in the Chaco dry savannas of Argentina.[J]. Applied Soil Ecology, 1999, 12(2):159-167.

[5] 张伟华, 关世英, 李跃进. 不同牧压强度对草原土壤水分、养分及其地上生物量的影响 [J]. 干旱区资源与环境, 2000, 14(4):61-64.

[6] 戎郁萍, 韩建国, 王培, 等. 放牧强度对草地土壤理化性质的影响 [J]. 中国草地学报, 2001, 23(4):41-47.

[7] 赵美曼. 荒漠草原不同放牧强度土壤理化特性及可蚀性的空间分布 [D]. 山东农业大学,2022.DOI:10.27277/d.cnki.gsdnu.2022.000743.

[8] 戎郁萍, 韩建国, 王培, 毛培胜. 放牧强度对草地土壤理化性质的影响 [J]. 中国草地,2001(04):42-48.

[9] 武彧睿. 多变量灰色预测模型 MGM（1，m，N）的改进及其应用研究 [D]. 南京信息工程大学,2022.DOI:10.27248/d.cnki.gnjqc.2022.001081.

[10] 刘敦利. 基于栅格尺度的土地沙漠化预警模式研究 [D]. 乌鲁木齐：新疆大学,2010.

[11] 花婷, 王训明. 东亚干旱半干旱区沙漠化与气候变化相互影响研究进展 [J]. 地理科学进展,2014,33(06):841-852.

[12] 侯琼, 王英舜, 杨泽龙, 等. 基于水分平衡原理的内蒙古典型草原土壤水动态模型研究 [J]. 干旱地区农业研究,2011,29(05):197-203.

[13] 王华静, 徐留兴, 葛成冉, 万建英. 放牧强度对草地土壤性状影响的研究进展 [J]. 安徽农业科学,2008,36(34):15074-15075.DOI:10.13989/j.cnki.0517-6611.2008.34.139.

[14] 刘占锋, 傅伯杰, 刘国华, 朱永官. 土壤质量与土壤质量指标及其评价 [J]. 生态学报,2006(03):901-913.

附录 A python 源代码

第二问训练样本生成及数据集构建：

```
#dataset.py
def generate_samples(raw_data: np.ndarray, target: np.ndarray,
    raw_tgt_in: np.ndarray, is_test=False, raw_date=None):
    """利用滑窗生成样本"""
    res_data = []
    res_label = []
    res_tgt_in = []
    # res_date = []
    length = raw_data.shape[0]
    seen = Constant.INPUT_WINDOW
    blind = Constant.OUTPUT_WINDOW
    for i in range(length - seen - blind):
        data = raw_data[i:i + seen]
        label = target[i + blind:i + seen + blind]
        tgt_in = raw_tgt_in[i:i + seen]
        # date = raw_date[i + blind:i + seen + blind]
        # res_date.append(date)
        res_data.append(data)
        res_label.append(label)
        res_tgt_in.append(tgt_in)
    if is_test:
        res_data.append(raw_data[length - seen:length])
        res_tgt_in.append(raw_tgt_in[length - seen:length])
        label = target[length - seen + blind:length]
        label = np.append(label, np.array([0]))
        label = label.reshape(-1, 1)
        res_label.append(label)
        # date = raw_date[length - seen + blind:length]
        # date = np.append(date, date[-1] +
            relativedelta(months=+1))
        # res_date.append(date)
    return np.array(res_data, dtype=float), np.array(res_label,
        dtype=float), np.array(res_tgt_in, dtype=float)
```

```
def scalar(data):
    """归一化处理"""
    mms = MinMaxScaler(feature_range=(-1, 1))
    res = mms.fit_transform(data)
    return res, mms


def generate_dataset():
    """土壤湿度预测数据集生成"""
    humidity_depth = Constant.DEPTH
    humidity_col = ['year', humidity_depth]
    # humidity_col = ['year', '10', '40', '100', '200']
    humidity_raw = pd.read_excel('data/humidity.xls',
        names=['year', 'lon', 'lat', '10', '40', '100', '200'])
    humidity_raw = humidity_raw[humidity_col]
    steam_col = ['year', 'mon', 'wm2', 'steam_mm']
    steam_raw = pd.read_excel('data/steam.xls', names=['mon',
        'year', 'lon', 'lat', 'wm2', 'steam_mm'])
    weather_use_col = ['降水量(mm)', '降水天数']
    weather_col = ['raw_mm', 'rain_day']

    weather = pd.DataFrame(columns=weather_col)
    steam = pd.DataFrame(columns=steam_col)
    humidity = pd.DataFrame(columns=humidity_col)
    years = [2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019,
        2020, 2021, 2022]
    for year in years:
        humidity = humidity.append(humidity_raw[humidity_raw.year
            == year], ignore_index=True)
        steam = steam.append(steam_raw[steam_raw.year == year],
            ignore_index=True)
        weather_raw = pd.read_excel(f'data/weather/{year}年.xls',
            usecols=weather_use_col)
        weather_raw.columns = weather_col
        weather = weather.append(weather_raw, ignore_index=True)
    # 数据整合
    for col in steam_col:
        humidity[col] = steam[col]
    for col in weather_col:
```

```
    humidity[col] = weather[col]
# humidity.to_csv('data/merge_data.csv', index=False)
years = humidity[['year', 'mon']].values
date = [datetime.date(ym[0], ym[1], 1) for ym in years]
humidity = humidity.drop(['year'], axis=1)

data = humidity.to_numpy().copy()
target = humidity[humidity_depth].to_numpy().copy()
target = target.reshape(target.shape + (1,))
humidity[humidity_depth] = humidity[humidity_depth].shift(1,
    fill_value=0) # 生成tgt，即后移一位
tgt_in = humidity.to_numpy()
# 归一化处理
data, _ = scalar(data)
target, mms = scalar(target)
tgt_in, _ = scalar(tgt_in)
# 划分训练集和测试
div = int(data.shape[0] * 0.8)
train_data = data[:div]
test_data = data[div:]
train_tgt = target[:div]
test_tgt = target[div:]
train_tgt_in = tgt_in[:div]
test_tgt_in = tgt_in[div:]

# 滑窗生成数据
train_data, train_label, train_tgt_in =
    generate_samples(train_data, train_tgt, train_tgt_in)
test_data, test_label, test_tgt_in =
    generate_samples(test_data, test_tgt, test_tgt_in,
    is_test=True)
print(f'train_data shape:{train_data.shape}, train_label
    shape:{train_label.shape}, tgt:{train_tgt_in.shape}')

print(f'共有训练集：{train_data.shape[0]}条，\
测试集{test_data.shape[0]}条')
return (train_data, train_label, train_tgt_in), (test_data,
    test_label, test_tgt_in), mms
```

```
class MyDataset(Dataset):
    """数据集类"""
    def __init__(self, ds, has_tgt=True):
        self.data = ds[0]
        self.label = ds[1]
        self.has_tgt = has_tgt
        if has_tgt:
            self.tgt_in = ds[2]

    def __getitem__(self, item):
        if self.has_tgt:
            return self.data[item], self.label[item],
                self.tgt_in[item]
        else:
            return self.data[item], self.label[item]

    def __len__(self):
        return self.data.shape[0]
```

LSTM 预测模型搭建：

```
# rnn_model.py
class LSTMModel(nn.Module):
    def __init__(self, n_input, n_hidden=128, n_layers=3,
        drop_prob=0.1):
        """
        LSTM初始化函数
        :param n_input: 输入数据的特征维度
        :param n_hidden: 隐藏层特征维度
        :param n_layers: LSTM块的个数
        :param drop_prob: dropout的概率
        """
        super(LSTMModel, self).__init__()

        self.n_layers = n_layers
        self.n_hidden = n_hidden

        self.lstm = nn.LSTM(n_input, n_hidden, n_layers,
```

```
            dropout=drop_prob)
        self.fc = nn.Linear(n_hidden, 1)
        self.dropout = nn.Dropout(drop_prob)

    def forward(self, x):
        _batch_size = x.size(0)
        _hidden = self.init_hidden(_batch_size, x.device)
        x = x.permute(1, 0, 2)
        x, hidden1 = self.lstm(x, _hidden)
        # x = self.dropout(x)
        out = x.contiguous().view(-1, _batch_size, self.n_hidden)
        out = out.permute(1, 0, 2)
        # out = out[:, -1, :] # 取最后一个点的数据作为预测特征
        out = self.fc(out)
        return out

    def init_hidden(self, batch_size, device):
        weight = next(self.parameters()).data
        hidden = (weight.new(self.n_layers, batch_size,
            self.n_hidden).zero_().to(device),
                weight.new(self.n_layers, batch_size,
                    self.n_hidden).zero_().to(device))

        return hidden


def lstm_init_weights(m):
    if type(m) == nn.LSTM:
        for name, param in m.named_parameters():
            if 'weight_ih' in name:
                torch.nn.init.orthogonal_(param.data)
            elif 'weight_hh' in name:
                torch.nn.init.orthogonal_(param.data)
            elif 'bias' in name:
                param.data.fill_(0)
    elif type(m) == nn.Linear:
        torch.nn.init.orthogonal_(m.weight)
        m.bias.data.fill_(0)
```

```
def get_parameter_number(net):
    total_num = sum(p.numel() for p in net.parameters())
    trainable_num = sum(p.numel() for p in net.parameters() if
        p.requires_grad)
    return {'Total': total_num, 'Trainable': trainable_num}
```

多变量单步时间序列预测训练及预测：

```
# rnn_train.py
def train(args, model, data_loader, criterion, optimizer, epoch,
    writer):
    model.train()
    batch_len = len(data_loader)

    process_bar = tqdm(total=batch_len)
    for idx, (data, label, tgt_in) in enumerate(data_loader):
        process_bar.update(1)
        data, label = data.to(args.device).float(),
            label.to(args.device).float()
        # label = label.reshape((label.shape[0], label.shape[1],
            -1))
        # label = label.permute((1, 0, 2))
        optimizer.zero_grad()
        if args.have_decoder:
            tgt_in = tgt_in.to(args.device).float()
            output = model(data, tgt_in)
        else:
            output = model(data)
        loss = criterion(output, label)
        loss.backward()
        torch.nn.utils.clip_grad_norm_(model.parameters(), 0.7)
        optimizer.step()

        writer.add_scalar("train_loss", loss.item(), epoch *
            batch_len + idx)

        if idx % 10 == 0:
            tqdm.write(f'[epoch:{epoch:>3d}] loss:{loss.item():.7f}
```

```
            .')


def test(args, model, data_loader, criterion):
    model.eval()
    total_loss = 0.
    predictions = torch.Tensor(0)
    truth = torch.Tensor(0)
    with torch.no_grad():
        for data, label, tgt_in in data_loader:
            data, label = data.to(args.device).float(),
                label.to(args.device).float()
            # label = label.reshape((label.shape[0],
                label.shape[1], -1))
            # label = label.permute((1, 0, 2))
            if args.have_decoder:
                tgt_in = tgt_in.to(args.device).float()
                output = model(data, tgt_in)
            else:
                output = model(data)
            total_loss += criterion(output, label).item()
            predictions = torch.cat((predictions, output[:,
                -Constant.OUTPUT_WINDOW].view(-1).cpu()), 0)
            truth = torch.cat((truth, label[:,
                -Constant.OUTPUT_WINDOW].view(-1).cpu()), 0)

        loss = total_loss / len(data_loader)
        tqdm.write(f'valid loss : {loss:.5f}')
        tqdm.write("-" * 100)
    return truth, predictions, loss


def main(args):
    # 获取模型
    model = LSTMModel(n_input=Constant.N_INPUT).to(args.device).\
            apply(lstm_init_weights)

    train_ds, test_ds, min_max = generate_dataset()
    train_dataset = MyDataset(train_ds)
```

```
    train_dataset_loader = DataLoader(train_dataset,
        batch_size=args.batch_size, shuffle=args.shuffle)
    test_dataset = MyDataset(test_ds)
    test_dataset_loader = DataLoader(test_dataset, batch_size=1,
        shuffle=False)

    criterion = nn.MSELoss()
    optimizer = torch.optim.AdamW(model.parameters(),
        lr=args.learn_rate)
    scheduler = torch.optim.lr_scheduler.StepLR(optimizer,
        step_size=1, gamma=0.95)

    writer = SummaryWriter('tensorboard')

    for epoch in range(args.epochs):
        # train
        train(args, model, train_dataset_loader, criterion,
            optimizer, epoch, writer)
        # validation
        truth, predictions, valid_loss = test(args, model,
            test_dataset_loader, criterion)

        truth = min_max.inverse_transform(truth.reshape((-1, 1)))
        predictions =
            min_max.inverse_transform(predictions.reshape((-1, 1)))
        PlotUtils.plot(truth, predictions, epoch)
        # print('truth:', truth.flatten())
        print('truth:', ",".join(str(i) for i in truth.flatten()))
        print('prediction:', ",".join(str(i) for i in
            predictions.flatten()))
        # scheduler
        scheduler.step()

        writer.add_scalar("valid_loss", valid_loss, epoch)


if __name__ == "__main__":
    import argparse
```

```
    parser = argparse.ArgumentParser()
    parser.add_argument("--epochs", type=int, default=100)
    parser.add_argument("--learn_rate", "-lr", type=float,
        default=0.005)
    parser.add_argument("--device", type=str)
    parser.add_argument("--shuffle", type=bool, default=False)
    parser.add_argument("--batch_size", type=int, default=10)
    parser.add_argument("--have_decoder", type=bool,
        default=False)
    parser.add_argument("--model", type=str)

    args = parser.parse_args()
    args.device = "cuda" if torch.cuda.is_available() else "cpu"
    print(f"You are using {args.device}......")

    main(args)
```

单变量滑窗时间序列预测：

```
# iterate_prediction.py
def generate_samples(raw_data: np.ndarray):
    res_data = []
    res_label = []
    length = raw_data.shape[0]
    seen = Constant.INPUT_WINDOW
    blind = Constant.OUTPUT_WINDOW
    for i in range(length - seen - blind):
        data = raw_data[i:i + seen]
        label = raw_data[i + blind:i + seen + blind]
        res_data.append(data)
        res_label.append(label)
    return np.array(res_data, dtype=float), np.array(res_label,
        dtype=float)


def scalar(data):
    """归一化处理"""
    mms = MinMaxScaler(feature_range=(-1, 1))
    res = mms.fit_transform(data)
```

```
    return res, mms


def generate_dataset():
    humidity_depth = Constant.DEPTH
    df = pd.read_csv('data/merge_data.csv',
        usecols=[humidity_depth])
    data_raw = df[humidity_depth].values
    data_raw = data_raw.reshape(-1, 1)
    data, min_max = scalar(data_raw)

    data, label = generate_samples(data)
    div = int(data.shape[0] * 0.8)
    train_data = data[:div]
    train_label = label[:div]
    test_data = data[div:]
    test_label = label[div:]

    return (train_data, train_label), (test_data, test_label),
        min_max


def train(model, data_loader, criterion, optimizer, epoch):
    model.train()
    batch_len = len(data_loader)

    process_bar = tqdm(total=batch_len)
    for idx, (data, label) in enumerate(data_loader):
        process_bar.update(1)
        data, label = data.to(device).float(),
            label.to(device).float()
        optimizer.zero_grad()
        output = model(data)
        loss = criterion(output, label)
        loss.backward()
        torch.nn.utils.clip_grad_norm_(model.parameters(), 0.7)
        optimizer.step()

        if idx % 10 == 0:
```

```
            tqdm.write(f'[epoch:{epoch:>3d}] loss:{loss.item():.7f}
                .')


def test(model, data_loader, criterion):
    model.eval()
    total_loss = 0.
    predictions = torch.Tensor(0)
    truth = torch.Tensor(0)
    with torch.no_grad():
        for data, label in data_loader:
            data, label = data.to(device).float(),
                label.to(device).float()
            output = model(data)
            total_loss += criterion(output, label).item()
            predictions = torch.cat((predictions, output[:,
                -Constant.OUTPUT_WINDOW].view(-1).cpu()), 0)
            truth = torch.cat((truth, label[:,
                -Constant.OUTPUT_WINDOW].view(-1).cpu()), 0)

        loss = total_loss / len(data_loader)
        tqdm.write(f'valid loss : {loss:.5f}')
        tqdm.write("-" * 100)
    return truth, predictions, loss


def main():
    model =
        LSTMModel(n_input=1).to(device).apply(lstm_init_weights)
    train_ds, test_ds, min_max = generate_dataset()

    train_dataset = MyDataset(train_ds, has_tgt=False)
    train_dataset_loader = DataLoader(train_dataset,
        batch_size=10, shuffle=False)
    test_dataset = MyDataset(test_ds, has_tgt=False)
    test_dataset_loader = DataLoader(test_dataset, batch_size=1,
        shuffle=False)

    criterion = torch.nn.MSELoss()
```

```
    optimizer = torch.optim.AdamW(model.parameters(), lr=0.001)
    scheduler = torch.optim.lr_scheduler.StepLR(optimizer,
        step_size=1, gamma=0.95)

    best = None

    for epoch in range(100):
        # train
        train(model, train_dataset_loader, criterion, optimizer,
            epoch)
        # validation
        truth, predictions, valid_loss = test(model,
            test_dataset_loader, criterion)

        if best is None:
            best = valid_loss
        if valid_loss <= best:
            best = valid_loss
            model_save_path = os.path.join('backup', 'lstm_best.pt')
            torch.save(model.state_dict(), model_save_path)
        truth = min_max.inverse_transform(truth.reshape((-1, 1)))
        predictions =
            min_max.inverse_transform(predictions.reshape((-1, 1)))
        PlotUtils.plot(truth, predictions, epoch)
        # scheduler
        scheduler.step()
    goahead(model)


def goahead(model=None, depth=Constant.DEPTH):
    if model is None:
        model =
            LSTMModel(n_input=1).to(device).apply(lstm_init_weights)
        model.load_state_dict(torch.load(
         f'backup/lstm_best_{depth}.pt'))
        model.eval()
    humidity_depth = depth
    df = pd.read_csv('data/merge_data.csv',
        usecols=[humidity_depth])
```

```
data_raw = df[humidity_depth].values
data_raw = data_raw.reshape(-1, 1)
data_raw, min_max = scalar(data_raw)
data, _ = generate_samples(data_raw)

window = data[-1]
window = window.reshape((1,) + window.shape)
window = torch.from_numpy(window)
window = window.to(device).float()
cache = data[-1].flatten().tolist()
for i in range(21): # 总共需要预测21个月
    output = model(window)
    for j in range(12): # 窗口长度为12
        if j < 11:
            window[0, j, 0] = window[0, j + 1, 0]
        else:
            window[0, j, 0] = output[0, -1, 0]
            cache.append(float(output[0, -1,
                0].cpu().detach().numpy()))
print(cache)
cache = np.array(cache)
cache = cache.reshape((-1, 1))
cache = min_max.inverse_transform(cache)
cache = cache.flatten()
prediction = pd.DataFrame(cache[12:])
prediction.to_csv(f'backup/{Constant.DEPTH}.csv')
cache_x = pd.date_range(end='20231201',
    periods=cache.shape[0], freq='MS')
fig = plt.figure(figsize=(15.36, 7.49))
# plt.plot(cache_x, cache, color='red', label='prediction')
data_raw = min_max.inverse_transform(data_raw)
data_x = pd.date_range(start='20120101',
    periods=data_raw.shape[0], freq='MS')
# print(cache_x[0])
# print(data_x[-1])
# plt.plot(data_x, data_raw.flatten(), color='blue',
    label='humidity')
plt.legend()
plt.grid()
```

```
    # fig.savefig(f'graph/{Constant.DEPTH}cm.png')
    # plt.show()
    return cache_x, cache, data_x, data_raw.flatten()


if __name__ == "__main__":
    # main()
    goahead()
    dd = ['10', '40', '100', '200']
    fig, ax = plt.subplots(4, 1, sharex=True)
    ax = ax.flatten()
    for i, d in enumerate(dd):
        x1, y1, x2, y2 = goahead(depth=d)
        ax[i].plot(x1, y1, color='red', label='预测值')
        ax[i].plot(x2, y2, color='blue', label=f'{d}cm土壤湿度')
        ax[i].legend()
        ax[i].grid()
        ax[i].set_ylabel('湿度(kg/m2)')
        if i==3:
            ax[i].set_xlabel('时间')
    plt.show()
```

改进的灰色预测模型

```
# gery_model.py
class GM11(object):
    def __init__(self, sys_data: pd.DataFrame, predstep: int = 2,
        S: float = 0.):
        self.S = S # 放牧强度对土壤化学性质影响系数
        self.data = np.array(sys_data.iloc[:, 0].values)
        self.data_shape = self.data.shape[0]
        self.data = self.data.reshape((self.data_shape, 1))
        self.coff = []
        self.sim_values = np.zeros((self.data_shape, 1))
        self.predstep = predstep
        self.pred_values = np.zeros((self.predstep, 1))
        self.error = []
        self.rel_errors = []
```

```
def __lsm(self):
    data_cum = np.cumsum(self.data) # 累加
    Y = self.data[1:]
    # 计算背景值
    background_values = np.zeros((self.data_shape - 1,
        1)).reshape((self.data_shape - 1, 1))
    for i in range(1, self.data_shape):
        background_values[i - 1][0] = (0.5 + self.S) *
            data_cum[i] + (1 - 0.5) * data_cum[i - 1]
    one_array = np.ones((self.data_shape - 1,
        1)).reshape((self.data_shape - 1, 1))
    background_values = -background_values
    B = np.hstack((background_values, one_array))
    # 用最小二乘求解参数
    self.coff = np.matmul(np.linalg.inv(np.matmul(B.T, B)),
        np.matmul(B.T, Y))

def fit(self) -> np.array:
    self.__lsm()
    a = self.coff[0][0]
    b = self.coff[1][0]
    # 求解拟合值
    self.sim_values[0] = self.data[0][0]
    for i in range(2, self.data_shape + 1):
        self.sim_values[i - 1] = (1 - np.exp(a)) *
            (self.data[0][0] - b / a) * np.exp(-a * (i - 1))
    self.sim_values = self.sim_values.reshape((1,
        self.data_shape))[0]
    return self.sim_values

def predict(self) -> np.array:
    self.__lsm()
    a = self.coff[0][0]
    b = self.coff[1][0]
    # 求解预测值
    for i in range(self.data_shape + 1, self.data_shape + 1 +
        self.predstep):
        self.pred_values[i - 1 - self.data_shape][0] = (1 -
            np.exp(a)) * (self.data[0][0] - b / a) * np.exp(
```

```
            -a * (i - 1))
        self.pred_values = self.pred_values.reshape((1,
            self.predstep))[0]
        return self.pred_values

    def loss(self) -> list:
        for i in range(self.data_shape):
            self.error.append(abs(self.sim_values[i] -
                self.data[i][0]) / self.data[i][0])
        return sum(self.error) / len(self.error)

    def errors(self):
        for i in range(self.data_shape):
            self.rel_errors.append(self.data[i][0] -
                self.sim_values[i])
        return self.rel_errors
```

第三问灰色预测

```
cols = ['year', 'area', 'intensity', 'SOC', 'SIC', 'STC', 'N',
    'C/N']
features = ['SOC', 'SIC', 'STC', 'N', 'C/N']
area = ['G17', 'G19', 'G21', 'G6', 'G12', 'G18', 'G8', 'G11',
    'G16', 'G9', 'G13', 'G20']
coff_S = {
    'SOC': {'LGI': 0.02, 'MGI': 0.04, "HGI": 0.08, "NG": -0.06},
    'N': {'LGI': 0.01, 'MGI': 0.05, "HGI": -0.09, "NG": 0.0},
    'SIC': {'LGI': 0.0, 'MGI': 0.0, "HGI": 0.08, "NG": 0.0},
    'C/N': {'LGI': 0.0, 'MGI': 0.0, "HGI": 0.08, "NG": 0.0},
    'STC': {'LGI': 0.0, 'MGI': 0.0, "HGI": 0.08, "NG": 0.0},
}
def grey_predict():
    df = pd.read_excel('data/attach14.xlsx', names=cols)
    result = pd.DataFrame(columns=['area', 'SOC', 'SIC', 'STC',
        'N', 'C/N'])
    count = 0
    loss = []
    for area_name in area:
        df_area = df[df['area'] == area_name]
```

```
        s = df_area['intensity'].iloc[0]
        df_area = df_area.groupby('year').mean()
        tmp = []

        fig, ax = plt.subplots(len(features), 1, sharex=True)
        ax = ax.flatten()
        for i, col in enumerate(features):
            data = pd.DataFrame(data=df_area[col])
            a = GM11(data, predstep=1, S=coff_S[col][s])
            a.fit()
            # print('GM(1,1)的拟合值是:  ', a.fit())
            prediction = a.predict()
            print('{:^5}{:^8}的预测值是: {:>8.4f}, 相对误差校验:
                {:<.4f}'.format(area_name, col, prediction[0],
                a.loss()))
            # print('GM(1,1)的预测误差是:  ', a.loss())
            # print('errors:', a.errors())
            tmp.extend(prediction)
            count += 1
            loss.append(a.loss())
            raw_data = df_area[col].values.tolist()
            predictions = raw_data + [prediction[0]]
            ax[i].plot(predictions, color='red',
                label=f'{area_name}小区{col}预测')
            ax[i].plot(raw_data, color='green',
                label=f'{area_name}小区{col}历史值')
            ax[i].legend(loc=2)
            ax[i].grid()
        plt.show()
        result =
            result.append(pd.DataFrame(columns=result.columns,
            data=[[area_name] + tmp]), ignore_index=True)
    result.to_csv('data/q3_result.csv')
    print('count={}, average of loss={}'.format(count,
        np.mean(loss)))


def main():
    grey_predict()
```

```
    # for a in area:
    #    visualization(a)


if __name__ == "__main__":
    # visualization()
    # add()
    main()
```

附录 B Matlb 源程序

```
% 问题一 计算拟合曲线
clc,clear
load data;
data=table2array(data);
x=[0,2,4,8];
p=zeros(12,5);
 for i=1:12
    for j=1:5
       p(i,j)=data(i,2*j)/data(i,2*j-1);
    end
 end
 p1=zeros(4,5); %存储同一放牧方式每年的加权增加比例
 p2=zeros(4,1);%存储同一放牧方式五年的平均增加比例
 for i=1:4
    for j=1:5
      p1(i,j)=(p(3*i,j)*data(3*i,j*2-1)+p(3*i-1,j)...
      *data(3*i-1,j*2-1)+p(3*i-2,j)*data(3*i-1,j*2-1))...
      /(data(3*i,j*2-1)+data(3*i-1,j*2-1)+data(3*i-2,j*2-1));
    end
 end

 for i=1:4
   p2(i)=(p1(i,1)+p1(i,2)+p1(i,3)+p1(i,4)+p1(i,5))/5;
 end

 P = polyfit(x,p2',3);
xi = 0:0.01:8;
yi = polyval(P,xi); %多项式求值
set(0,'defaultfigurecolor','w');
grid();
 hold on;
plot(xi,yi,'b');   %作拟合曲线
plot(x,p2,'k','LineWidth',2);
axis([0,8,0,6.5]) ;
legend('变化曲线','拟合曲线','Location','Best');
ylabel('生长期植被总量增加倍数');
xlabel('放牧强度');
```

```
title('不同放牧强度下牧植被总量变化');

%%---------------------------------%%
%问题四：一致性检验和权向量计算
A=[1 1 2;1 1 1.5;1/2 2/3 1];
[n,n]=size(A);
[v,d]=eig(A);
r=d(1,1);
CI=(r-n)/(n-1);
RI=[0 0 0.58 0.90 1.12 1.24 1.32 1.41 1.45 1.49 1.52 1.54 1.56
   1.58 1.59];
CR=CI/RI(n);
if(CR<0.10)
   CR_RESULT='通过';
else
   CR_RESULT='不通过';
end
% %权向量计算
w=v(:,1)/sum(v(:,1));
w=w';
```