论文详解：

摘要

本文通过数学和数值分析回答了一些关于COVID-19的重要问题：

1. **是否可能控制COVID-19?**
2. **如果COVID-19可以被控制，疫情的峰值何时到来，何时结束?**
3. **无症状感染者如何影响疾病的传播?**
4. **如果COVID-19无法被控制，为实现群体免疫需要感染的人口比例是多少?**
5. **社交距离措施的有效性如何?**
6. **如果COVID-19无法被控制，长期来看感染的人口比例是多少?**

为了解决这些问题，作者提出了一个时间依赖的SIR（易感-感染-康复）模型，跟踪随时间变化的传播率和康复率。使用中国国家卫生健康委员会提供的数据，预测的每日确诊病例数误差小于3%。

研究内容

1. **时间依赖的SIR模型**：
   - 传统SIR模型的传播率和康复率是时间不变的。本文提出的时间依赖SIR模型通过机器学习方法跟踪传播率和康复率，使其随时间变化，更加灵活和准确。
   - 模型预测了疫情的拐点（传播率小于康复率的那一天）为2020年2月17日，此后基本再生数R0小于1。
2. **无症状感染者的影响**：
   - 模型扩展为考虑两种类型的感染者：可检测的和不可检测的。
   - 使用一个2x2矩阵的谱半径来描述疾病是否爆发。模型显示，2020年3月2日，包括韩国、意大利和伊朗在内的几个国家处于COVID-19爆发的边缘。
3. **群体免疫**：
   - 分析显示，在至少1 - 1/R0的人口被感染和康复后可以实现群体免疫。
4. **社交距离措施**：
   - 研究了社交距离对降低R0的两种方法：减少每人的正常接触数和取消大规模聚会。
5. **长期感染比例**：
   - 在假设传染率和康复率恒定的SIR模型中，长期感染人口比例可以通过一个固定点方程计算。

数值实验

通过数值实验验证模型的有效性，并对不同的控制政策进行了讨论和建议。使用历史数据，模型预测中国的确诊病例数在控制政策维持下将达到约80,000例。

结论

本文提出的时间依赖SIR模型和扩展模型，为理解和控制COVID-19提供了一个更为动态和准确的工具。该研究展示了时间依赖参数在疫情预测和控制中的重要性，以及无症状感染者和社交距离措施对疫情传播的影响。

关键概念与符号

- **R0（基本再生数）**：衡量一个感染者在康复前平均能传染的人数。
- **β(t) 和 γ(t)**：分别是随时间变化的传播率和康复率。
- **社交距离**：通过减少接触数或取消聚会来降低传播率。

进一步研究

建议未来的研究可以进一步优化时间依赖SIR模型，并探索更多复杂网络结构下的疾病传播动态。

代码

```
import math
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import Ridge
from sklearn.model_selection import GridSearchCV
from sklearn.preprocessing import StandardScaler

# 将数据分割成训练集和测试集
def data_spilt(data, orders, start):
    x_train = np.empty((len(data) - start - orders, orders))
    y_train = data[start + orders:]

    for i in range(len(data) - start - orders):
        x_train[i] = data[i + start:start + orders + i]

    # 排除在湖北省定义确诊病例的变更日期（2020年2月12日）
    x_train = np.delete(x_train, np.s_[28 - (orders + 1) - start:28 - start], 0)
    y_train = np.delete(y_train, np.s_[28 - (orders + 1) - start:28 - start])

    return x_train, y_train

# 使用Ridge回归进行参数搜索
def ridge(x, y):
    print('\nStart searching good parameters for the task...')
    parameters = {'alpha': np.arange(0, 0.100005, 0.000005).tolist(),
                  "tol": [1e-8],
                  'fit_intercept': [True, False]}

    clf = GridSearchCV(Ridge(), parameters, n_jobs=-1, cv=5)
    clf.fit(x, y)

    print('\nResults for the parameters grid search:')
    print('Model:', clf.best_estimator_)
    print('Score:', clf.best_score_)

    return clf

# 数据
```

```
# 从https://voice.baidu.com/act/newpneumonia/newpneumonia 收集的数据
# 累计确诊病例
X_cml = np.array([41, 45, 62, 121, 199, 291, 440, 574, 835, 1279, 1985, 2761,
4535, 5997, 7736, 9720, 11821, 14411, 17238, 20471, 24363, 28060, 31211, 34598,
37251, 40235, 42708, 44730, 59882, 63932, 66576, 68584, 70635, 72528, 74279,
75101, 75993, 76392, 77041, 77262, 77779, 78190, 78630, 78959, 79389, 79968,
80174, 80302, 80422, 80565, 80710, 80813, 80859, 80904, 80924, 80955, 80980,
81003, 81201, 81048, 81077, 81116, 81151, 81235, 81300, 81416, 81498, 81600,
81747, 81846, 81960, 82078, 82213, 82341, 82447, 82545, 82631, 82724, 82802,
82875, 82930, 83005, 83071, 83157, 83249], dtype=np.float64)[:-27]
# 累计康复病例
recovered = np.array([12, 12, 16, 21, 25, 25, 28, 28, 34, 38, 49, 51, 60, 103,
124, 171, 243, 328, 475, 632, 892, 1153, 1540, 2050, 2651, 3283, 3998, 4742,
5915, 6728, 8101, 9425, 10853, 12561, 14387, 16170, 18279, 20673, 22907, 24757,
27353, 29775, 32531, 36157, 39049, 41675, 44518, 47260, 49914, 52109, 53793,
55477, 57143, 58684, 59982, 61567, 62887, 64216, 65649, 67022, 67863, 68799,
69725, 70547, 71284, 71876, 72382, 72841, 73299, 73791, 74196, 74737, 75122,
75600, 75937, 76225, 76415, 76610, 76785, 76984, 77210, 77348, 77450, 77586,
77711], dtype=np.float64)[:-27]
# 累计死亡病例
death = np.array([2, 3, 3, 3, 4, 6, 9, 18, 25, 41, 56, 80, 106, 132, 170, 213,
259, 304, 361, 425, 491, 564, 637, 723, 812, 909, 1017, 1114, 1368, 1381, 1524,
1666, 1772, 1870, 2006, 2121, 2239, 2348, 2445, 2595, 2666, 2718, 2747, 2791,
2838, 2873, 2915, 2946, 2984, 3015, 3045, 3073, 3100, 3123, 3140, 3162, 3173,
3180, 3194, 3204, 3218, 3231, 3242, 3250, 3253, 3261, 3267, 3276, 3283, 3287,
3293, 3298, 3301, 3306, 3311, 3314, 3321, 3327, 3331, 3335, 3338, 3340, 3340,
3342, 3344], dtype=np.float64)[:-27]

# 总人口数
population = 1439323776
# 数据预处理
X = X_cml - recovered - death
R = recovered + death

n = np.array([population] * len(X), dtype=np.float64)

S = n - X - R

X_diff = np.array([X[:-1], X[1:]], dtype=np.float64).T
R_diff = np.array([R[:-1], R[1:]], dtype=np.float64).T

gamma = (R[1:] - R[:-1]) / X[:-1]
beta = n[:-1] * (X[1:] - X[:-1] + R[1:] - R[:-1]) / (X[:-1] * (n[:-1] - X[:-1] -
R[:-1]))
R0 = beta / gamma

# Ridge回归参数
orders_beta = 3
orders_gamma = 3

start_beta = 10
start_gamma = 10

# 打印信息
print("\nThe latest transmission rate beta of SIR model:", beta[-1])
print("The latest recovering rate gamma of SIR model:", gamma[-1])
```

```
print("The latest basic reproduction number R0:", R0[-1])

# Ridge回归
x_beta, y_beta = data_spilt(beta, orders_beta, start_beta)
x_gamma, y_gamma = data_spilt(gamma, orders_gamma, start_gamma)

# 标准化数据
scaler_beta = StandardScaler()
x_beta_scaled = scaler_beta.fit_transform(x_beta)

scaler_gamma = StandardScaler()
x_gamma_scaled = scaler_gamma.fit_transform(x_gamma)

# 搜索最佳参数（注释掉，因为已知最佳参数）
# clf_beta = ridge(x_beta_scaled, y_beta)
# clf_gamma = ridge(x_gamma_scaled, y_gamma)

# 训练和测试（使用已知最佳参数）
clf_beta = Ridge(alpha=0.003765, copy_X=True, fit_intercept=False, max_iter=None,
random_state=None, solver='auto', tol=1e-08).fit(x_beta_scaled, y_beta)
clf_gamma = Ridge(alpha=0.001675, copy_X=True, fit_intercept=False,
max_iter=None, random_state=None, solver='auto', tol=1e-08).fit(x_gamma_scaled,
y_gamma)

beta_hat = clf_beta.predict(x_beta_scaled)
gamma_hat = clf_gamma.predict(x_gamma_scaled)

# 绘制训练和测试结果
plt.figure(1)
plt.plot(y_beta, label=r'$\beta (t)$')
plt.plot(beta_hat, label=r'$\hat{\beta}(t)$')
plt.legend()

plt.figure(2)
plt.plot(y_gamma, label=r'$\gamma (t)$')
plt.plot(gamma_hat, label=r'$\hat{\gamma}(t)$')
plt.legend()

# 时间依赖SIR模型
stop_X = 0 # 停止标准
stop_day = 100 # 最大迭代天数

day_count = 0
turning_point = 0

S

_predict = [S[-1]]
X_predict = [X[-1]]
R_predict = [R[-1]]

predict_beta = np.array(beta[-orders_beta:]).tolist()
predict_gamma = np.array(gamma[-orders_gamma:]).tolist()
while (X_predict[-1] >= stop_X) and (day_count <= stop_day):
    if predict_beta[-1] > predict_gamma[-1]:
        turning_point += 1
```

```
    next_beta = clf_beta.predict(np.asarray([predict_beta[-orders_beta:]]))[0]
    next_gamma = clf_gamma.predict(np.asarray([predict_gamma[-orders_gamma:]]))
[0]

    if next_beta < 0:
        next_beta = 0
    if next_gamma < 0:
        next_gamma = 0

    predict_beta.append(next_beta)
    predict_gamma.append(next_gamma)

    next_S = ((-predict_beta[-1] * S_predict[-1] *
               X_predict[-1]) / n[-1]) + S_predict[-1]
    next_X = ((predict_beta[-1] * S_predict[-1] * X_predict[-1]) /
              n[-1]) - (predict_gamma[-1] * X_predict[-1]) + X_predict[-1]
    next_R = (predict_gamma[-1] * X_predict[-1]) + R_predict[-1]

    S_predict.append(next_S)
    X_predict.append(next_X)
    R_predict.append(next_R)

    day_count += 1

# 打印信息
print('\nConfirmed cases tomorrow:', np.rint(X_predict[1] + R_predict[1]))
print('Infected persons tomorrow:', np.rint(X_predict[1]))
print('Recovered + Death persons tomorrow:', np.rint(R_predict[1]))

print('\nEnd day:', day_count)
print('Confirmed cases on the end day:', np.rint(X_predict[-2] + R_predict[-2]))

print('\nTuring point:', turning_point)

# 绘制时间依赖SIR模型的时间演变图
plt.figure(3)
plt.plot(range(len(X) - 1, len(X) - 1 + len(X_predict)), X_predict, '*-',
label=r'$\hat{X}(t)$', color='darkorange')
plt.plot(range(len(X) - 1, len(X) - 1 + len(X_predict)), R_predict, '*-',
label=r'$\hat{R}(t)$', color='limegreen')
plt.plot(range(len(X)), X, 'o--', label=r'$X(t)$', color='chocolate')
plt.plot(range(len(X)), R, 'o--', label=r'$R(t)$', color='darkgreen')
plt.xlabel('Day')
plt.ylabel('Person')
plt.title('Time evolution of the time-dependent SIR model.')

plt.legend()
plt.show()
```

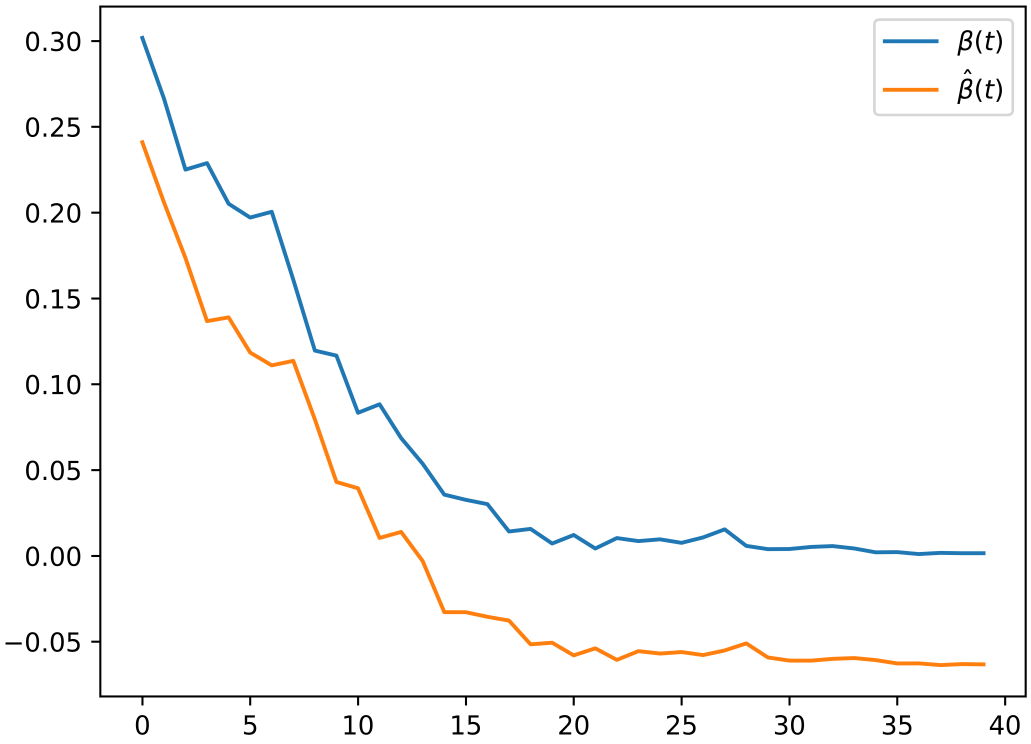
output

1. 传染率（β）、康复率（γ）和基本再生数（R0）
2. 明日预测
3. 最终预测结果

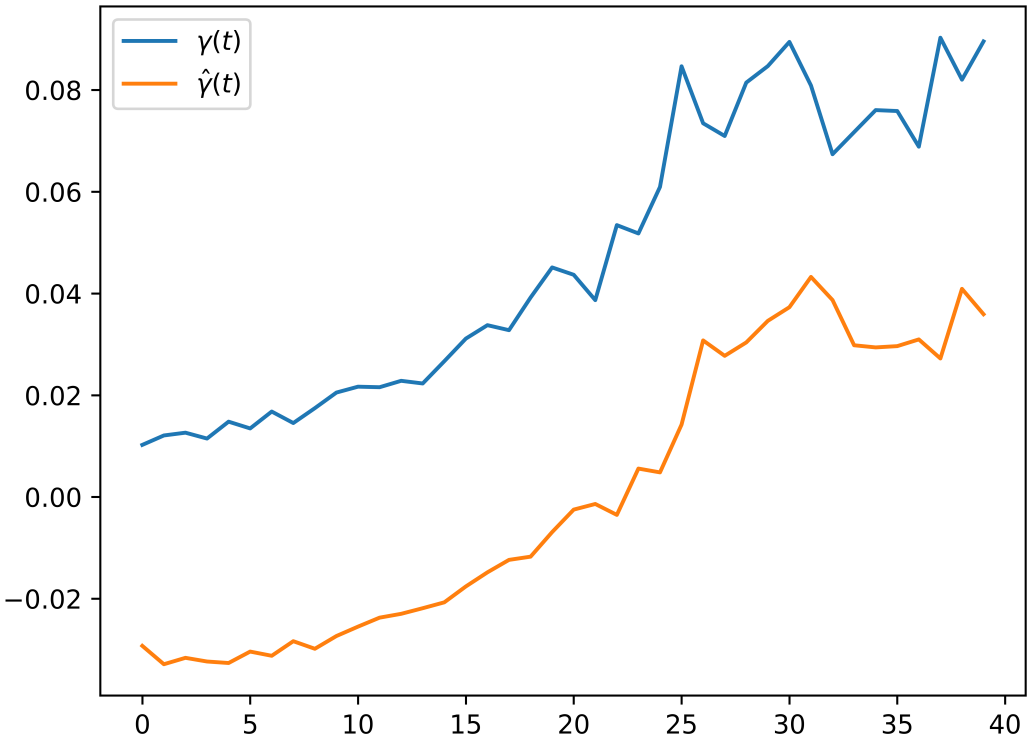
```
#最新的传染率 β:
The latest transmission rate beta of SIR model: 0.001541641696450796
#最新的康复率 γ
The latest recovering rate gamma of SIR model: 0.08954423592493298
#最新的基本再生数 R0
The latest basic reproduction number R0: 0.01721653750826787

#明日预测
#明日预测的确诊病例总数:
Confirmed cases tomorrow: 81005.0
#明日预测的感染者人数:
Infected persons tomorrow: 13576.0
#明日预测的康复和死亡总数
Recovered + Death persons tomorrow: 67429.0

#最终预测结果
#疫情结束日(End day)
End day: 101

#疫情结束日的确诊病例总数:
Confirmed cases on the end day: 81005.0

#转折点（Turning point）
Turing point: 0
```

IMG

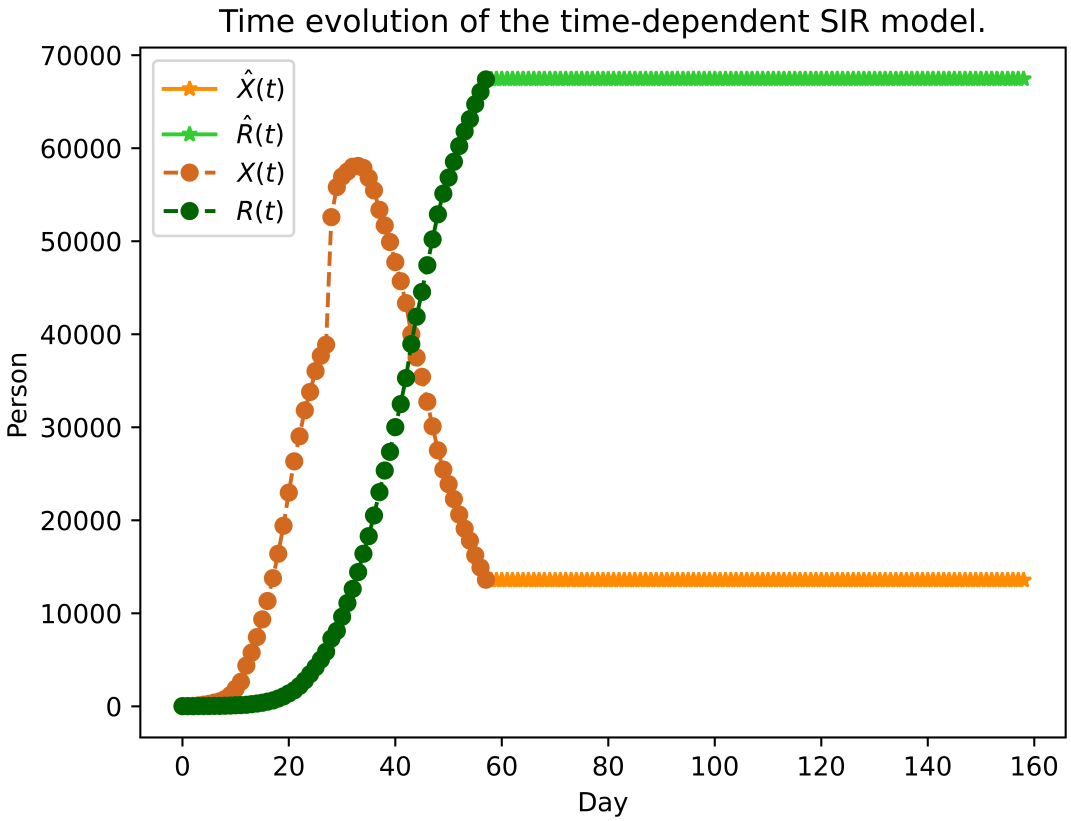
传染率 beta 的对比图



康复率 gamma 的对比图



时间依赖 SIR 模型的时间演变图:


Time evolution of the time-dependent SIR model.
70000
60000
50000
40000
30000
20000
10000
0
Person
0
20
40
60
80
100
120
140
160
Day
$\hat{X}(t)$
$\hat{R}(t)$
$X(t)$
$R(t)$