Documentação do Código para Estimação de Parâmetros no Modelo SIR

Autor: Thaisse Dias Paes

30 de março de 2025

1 Introdução

Este código tem como objetivo estimar os parâmetros do modelo SIR, que descreve a dinâmica de propagação de uma doença em uma população. As duas abordagens utilizadas para a estimação dos parâmetros são:

- **Algoritmo Genético (AG):** Implementado com a biblioteca DEAP, que explora de forma estocástica o espaço de parâmetros.
- **Mínimos Quadrados Não Lineares (MQNL):** Utilizando a função `curve_fit` da biblioteca SciPy, que ajusta o modelo aos dados simulados.

No modelo SIR, os parâmetros a serem estimados são:

- β: Taxa de transmissão da doença.
- γ: Taxa de recuperação.

2 Estrutura do Código

O código está organizado nas seguintes seções:

2.1 Geração dos Dados Reais (Simulados)

- Os valores reais dos parâmetros são definidos como $\beta_{\text{real}} = 0{,}30$ e $\gamma_{\text{real}} = 0{,}10$.
- São definidas a população total N e os estados iniciais para os grupos Suscetível (S_0), Infectado (I_0) e Recuperado (R_0).
- Um vetor de tempo `t_data` é criado para simular a evolução do sistema ao longo de 100 dias.

A função `modelo_sir` implementa as equações diferenciais do modelo SIR:

```
def modelo_sir(t, y, beta, gamma):
    S, I, R = y
    dSdt = -beta * S * I / N
    dIdt = beta * S * I / N - gamma * I
    dRdt = gamma * I
    return [dSdt, dIdt, dRdt]
```

Em seguida, o solver `solve_ivp` é utilizado para simular o comportamento do sistema e os dados reais de infectados são gerados com a adição de ruído.

2.2 Configuração do Algoritmo Genético

- São definidas as classes `FitnessMin` e `Individual` utilizando a biblioteca DEAP para formular o problema de minimização.
- Os atributos individuais, que representam os parâmetros β e γ, são gerados dentro de intervalos plausíveis (por exemplo, $\beta \in [0.1, 0.5]$ e $\gamma \in [0.01, 0.3]$).
- São configurados os operadores genéticos: cruzamento (utilizando `cxBlend`), mutação (com `mutGaussian`) e seleção (através de torneio com `selTournament`).

A função de fitness avalia um indivíduo calculando o erro médio quadrático entre os dados simulados (utilizando os parâmetros do indivíduo) e os dados reais de infectados. Penalizações são aplicadas para indivíduos com parâmetros fora dos limites especificados:

```
def fitness(individual):
    beta, gamma = individual
    penalty = 0
    if not (0.1 <= beta <= 0.5): penalty += 10 * abs(beta - 0.3)
    if not (0.01 <= gamma <= 0.3): penalty += 10 * abs(gamma - 0.1)

    try:
        sol = solve_ivp(modelo_sir, [0, 100], [S0, I0, R0], args=(beta,
    gamma),
                        t_eval=t_data, method='RK45', dense_output=True)
        I_pred = sol.sol(t_data)[1]
        return np.mean((I_pred - I_real)**2) + penalty,
    except:
        return 1e6,
```

O algoritmo evolutivo é executado com uma população de 100 indivíduos por 150 gerações e o melhor indivíduo é selecionado.

2.3 Ajuste via Mínimos Quadrados

Para comparação, o método dos Mínimos Quadrados é aplicado:

- A função `modelo_para_ajuste` adapta o modelo SIR para a função `curve_fit`.
- Os parâmetros são ajustados a partir de palpites iniciais e limites pré-definidos.

O código para o ajuste é o seguinte:

```
def modelo_para_ajuste(t, beta, gamma):
    sol = solve_ivp(modelo_sir, [0, 100], [S0, I0, R0], args=(beta,
   gamma),
                    t_eval=t, method='RK45', dense_output=True)
    return sol.sol(t)[1]

popt, _ = curve_fit(modelo_para_ajuste, t_data, I_real, p0=[0.3, 0.1],
                    bounds=([0.1, 0.01], [0.5, 0.3]))
```

2.4 Visualização dos Resultados

Os resultados são visualizados por meio de um gráfico:

- Um gráfico de dispersão (scatter) exibe os dados reais dos infectados.
- São plotadas duas curvas: uma obtida com o Algoritmo Genético (AG) e outra com o método dos Mínimos Quadrados.
- A legenda indica os valores estimados dos parâmetros β e γ para cada método.

O trecho de código responsável pela plotagem é:

```
plt.figure(figsize=(12, 6))
plt.scatter(t_data, I_real, color='k', s=20, label="Dados Reais", alpha
    =0.6)
plt.plot(t_data, solve_ivp(modelo_sir, [0, 100], [S0, I0, R0],
                           args=tuple(best_ag), t_eval=t_data).y[1],
         'r--', label=f"AG:   ={best_ag[0]:.3f},   ={best_ag[1]:.3f}")
plt.plot(t_data, solve_ivp(modelo_sir, [0, 100], [S0, I0, R0],
                           args=tuple(popt), t_eval=t_data).y[1],
         'b:', label=f"M nimos Quadrados:   ={popt[0]:.3f},   ={popt
    [1]:.3f}")
plt.xlabel("Tempo (dias)")
plt.ylabel("Infectados (I)")
plt.legend()
plt.grid(True, alpha=0.3)
plt.show()
```

Além disso, os resultados finais são impressos no terminal em formato tabular.

3 Conclusão

Este código integra técnicas de simulação e otimização para a estimação dos parâmetros do modelo SIR. A comparação entre o Algoritmo Genético e o método dos Mínimos Quadrados permite identificar as vantagens e limitações de cada abordagem. Em particular, enquanto o AG demonstra maior robustez frente a ruído e à exploração do espaço de parâmetros, o método dos Mínimos Quadrados oferece uma solução computacionalmente mais eficiente quando se dispõe de boas condições iniciais.

Esta documentação serve como guia para a compreensão da estrutura e funcionamento do código, facilitando a sua manutenção, a replicação dos experimentos e a eventual extensão para outros modelos dinâmicos.