Громыко Артём Артурович
Б9122-02.03.01сцт

Постановка задачи. Описание моделей

Модель конкуренции двух видов — способ математически описать динамику взаимодействующих биологических популяций, которые конкурируют за общий ресурс, такой как пища или территория.
Модель представляет собой систему двух обыкновенных дифференциальных уравнений первого порядка, которые задают скорость изменения численности каждой популяции. Уравнения выглядят так:

$$\begin{cases} \dot{x} = x\left(r_1 - \beta_1 x - \alpha_2 y\right) \\ \dot{y} = y\left(r_2 - \beta_2 y - \alpha_1 x\right) \end{cases}$$

Здесь x и y являются *фазовыми переменными*. Они отражают численность двух популяций, в заданный момент времени.

Так же, в модели присутствуют следующие параметры:

- r_1, r_2 — *мальтузианские параметры* популяций. Это разница между их собственной рождаемостью и смертностью, определяющая скорость роста в отсутствие внутривидовых и межвидовых взаимодействий;
- β_1, β_2 — параметры, характеризующие силу внутривидовых взаимодействий каждой из групп *(например, конкуренция за ресурсы)*;
- α_1, α_2 — параметры, характеризующие силу взаимодействия групп x, y между собой.

Для определенности будем считать, что все параметры положительны.

Входными данными для модели будем считать начальную численность обоих групп в нулевой момент времени.
Таким образом, *решением* будем называть пару функций $x(t), y(t)$, удовлетворяющую следующей начальной задаче:

$$\begin{cases} \dot{x} = x\left(r_1 - \beta_1 x - \alpha_2 y\right) \\ \dot{y} = y\left(r_2 - \beta_2 y - \alpha_1 x\right) \quad (1) \\ x(0) = x_0 \quad y(0) = y_0 \end{cases}$$

Аналитическое исследование системы

Поиск равновесий

Найдем равновесия модели

$$\begin{cases} x\left(r_1 - \beta_1 x - \alpha_2 y\right) = 0 \\ y\left(r_2 - \beta_2 y - \alpha_1 x\right) = 0 \end{cases}$$

Решив систему, получаем следующие равновесия:
Тривиальное

$$x_t = y_t = 0$$

Нетривиальные

$$x_1 = 0, y_1 = \frac{r_2}{\beta_2}$$
$$x_2 = \frac{r_1}{\beta_1}, y_2 = 0$$
$$x_3 = \frac{\alpha_1\beta_2 - \alpha_2 r_2}{\beta_1\beta_2 - \alpha_1\alpha_2}, y_3 = \frac{\beta_1 r_2 - \alpha_1 r_1}{\beta_1\beta_2 - \alpha_1\alpha_2}$$

Исследование на устойчивость

Для проверки устойчивости равновесий найдем собственные значения матрицы Якоби системы в данных точках.

$$J - \lambda E = \begin{pmatrix} r_1 - \alpha_2 y - 2\beta_1 x - \lambda & -\alpha_2 x \\ -\alpha_1 y & r_2 - \alpha_1 x - 2\beta_2 y - \lambda \end{pmatrix}$$

Тривиальное равновесие

$$\det(J - \lambda E)|_{x=x_t, y=y_t} = (r_1 - \lambda)(r_2 - \lambda) = 0 \implies \begin{cases} \lambda_1 = r_1 \\ \lambda_2 = r_2 \end{cases}$$

Так как $\operatorname{Im}\lambda_{1,2} = 0, \operatorname{Re}\lambda_1 > 0, \operatorname{Im}\lambda_2 > 0$, можем заключить, что тривиальное равновесие является *неустойчивым узлом*.

Нетривиальные равновесия

x_1, y_1

$$\det(J - \lambda E)|_{x=x_1, y=y_1} = \left(r_1 - \alpha_2 \frac{r_2}{\beta_2} - \lambda\right)\left(r_2 - 2\beta_2 \frac{r_2}{\beta_2} - \lambda\right) \implies \begin{cases} \lambda_1 = r_1 - \dfrac{\alpha_2 r_2}{\beta_2} \\ \lambda_2 = -r_2 \end{cases}$$

- $r_1\beta_2 > \alpha_2 r_2$: седло (неустойчиво)
- $r_1\beta_2 < \alpha_2 r_2$: устойчивый узел

x_2, y_2

$$\det(J - \lambda E)|_{x=x_2, y=y_2} = (-r_1 - \lambda)\left(r_2 - \frac{\alpha_1 r_1}{\beta_1} - \lambda\right) \implies \begin{cases} \lambda_1 = r_2 - \dfrac{\alpha_1 r_1}{\beta_1} \\ \lambda_2 = -r_1 \end{cases}$$

- $r_2\beta_1 > \alpha_1 r_1$: седло (неустойчиво)
- $r_2\beta_1 < \alpha_1 r_1$: устойчивый узел

x_3, y_3

$$\det(J - \lambda E)|_{x=x_3, y=y_3} = (r_1 - \alpha_2 y_3 - 2\beta_1 x_3 - \lambda)\,(r_2 - \alpha_1 x_3 - 2\beta_2 y_3 - \lambda) - \alpha_1\alpha_2 x_3 y_3 =$$
$$= (-\beta_1 x_3 - \lambda)(-\beta_2 y_3 - \lambda) - \alpha_1\alpha_2 x_3 y_3 = 0$$
$$\lambda^2 + (\beta_2 y_3 + \beta_1 x_3)\lambda + (\beta_1\beta_2 - \alpha_1\alpha_2)x_3 y_3 = 0$$
$$p = \beta_1 x_3 + \beta_2 y_3 > 0$$
$$q = (\beta_1\beta_2 - \alpha_1\alpha_2)x_3 y_3 = \frac{(\alpha_1\beta_2 - \alpha_2 r_2)(\beta_1 r_2 - \alpha_1 r_1)}{\beta_1\beta_2 - \alpha_1\alpha_2}$$
$$D = p^2 - 4q$$

По теореме Виета:

$$\lambda_1 + \lambda_2 = -p < 0$$
$$\lambda_1 \cdot \lambda_2 = q$$

Рассмотрим несколько случаев:

- $q < 0$

$$\begin{cases} \lambda_1 \cdot \lambda_2 < 0 \\ \lambda_1 + \lambda_2 < 0 \\ D = p^2 - 4q > 0 \end{cases} \implies \begin{matrix} \lambda_1, \lambda_2 \in R \\ \lambda_1 \cdot \lambda_2 < 0 \end{matrix}$$

  Седло

- $q > 0, D \geq 0$

$$\begin{cases} \lambda_1 \cdot \lambda_2 > 0 \\ \lambda_1 + \lambda_2 < 0 \\ D = p^2 - 4q > 0 \end{cases} \implies \begin{matrix} \lambda_1, \lambda_2 \in R \\ \lambda_1 < 0, \lambda_2 < 0 \end{matrix}$$

  Устойчивый узел

- $q > 0, D < 0$

$$\lambda_{1,2} = \frac{-p \pm \sqrt{p^2 - 4q} \cdot i}{2}$$
$$\operatorname{Re}\lambda_{1,2} < 0, \operatorname{Im}\lambda_{1,2} \neq 0$$

  Устойчивый фокус

Численное решение модели

Для проведения численных экспериментов были выбраны данные по конкуренции двух видов микробов: стандартные пивные дрожжи Saccharomyces cerevisiae и более медленно растущие дрожжи Schizosaccharomyces kephir.

Для численного решения задачи Коши, описывающей нашу модель, был выбран метод семейства Кунге-Кутта 4 порядка, в силу его достаточной точности и простоты реализации. Для подбора параметров использовался нелинейный метод наименьших квадратов, предоставляемый пакетом scipy языка программирования Python.

В итоге были получены оптимальные параметры и численно найдены решения системы ДУ. Так же был построен тренд на 25 шагов по времени.
Сравнение динамики и фазовых траекторий модели с реальными данными представлена на рисунках 1-2.

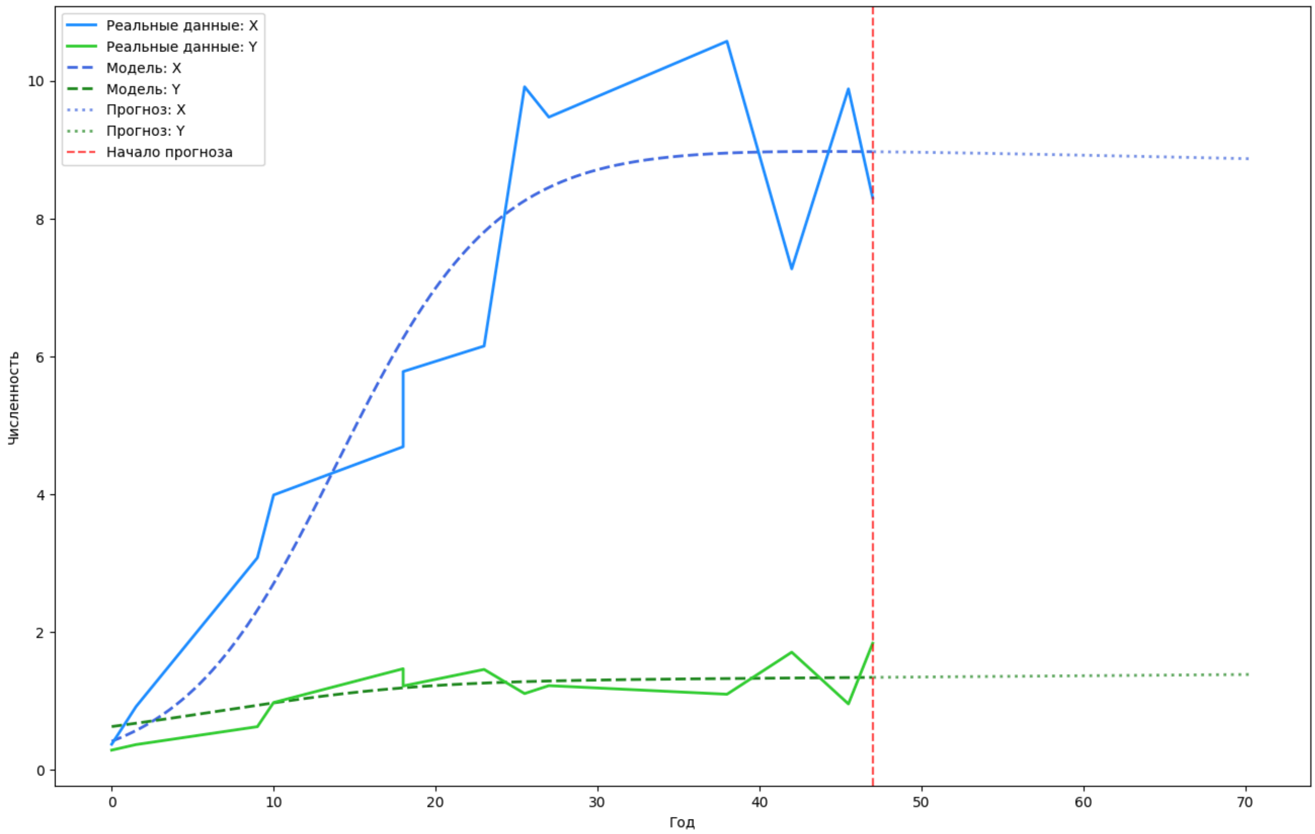

Рис. 1. Сравнение динамического поведения модели с реальными данными, прогноз

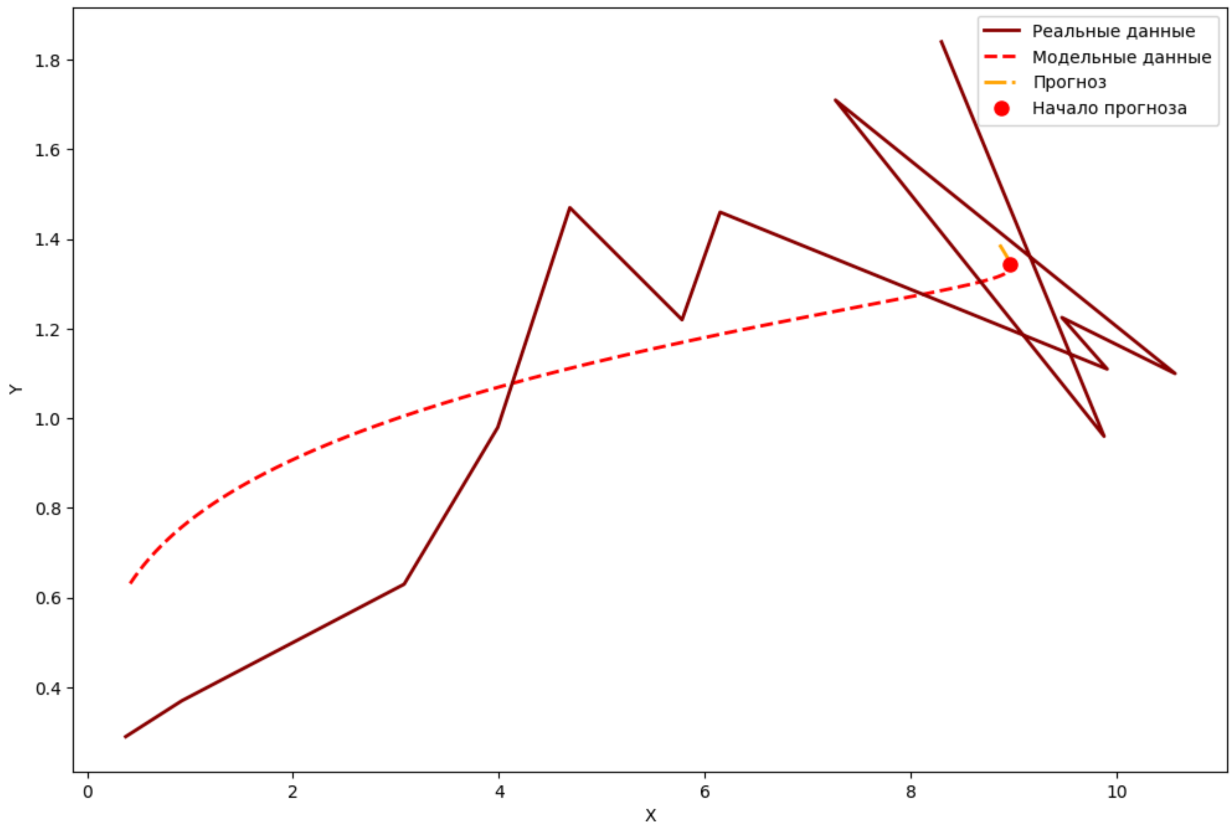

Рис. 2. Сравнение фазового портрета с реальными данными

Заключение

В результате проведенного исследования была успешно применена модель конкуренции двух популяций для анализа конкуренции двух видов дрожжей. Аналитическое исследование системы позволило идентифицировать состояния равновесия, найдены условия устойчивости этих равновесий.
Практическим итогом работы стала численная реализация модели. С помощью нелинейного метода наименьших квадратов были подобраны оптимальные параметры системы, что позволило добиться качественного соответствия модельных траекторий реальным данным.

Листинг

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

from scipy.optimize import minimize
from scipy.integrate import solve_ivp


df = pd.read_csv('data/mmc2.csv')
x_name, y_name, t_name = 'X', 'Y', 'Time'

t_data = df[t_name].values
H_data = df[x_name].values
L_data = df[y_name].values
y0 = [0.4184, 0.6315]

def objective(params):
    y_pred = solve_model(params, t_data, y0)

    error_H = np.mean((y_pred[0] - H_data)**2)
    error_L = np.mean((y_pred[1] - L_data)**2)

    return error_H + error_L

def solve_model(params, t, y0):
    r1, r2, a1, a2, b1, b2 = params
    def lotka_volterra(t, y):
        H, L = y
        dHdt = r1 * H - a2 * H * L - b1 * H * H
        dLdt = r2 * L - a1 * H * L - b2 * L * L
        return [dHdt, dLdt]

    solution = solve_ivp(lotka_volterra, [t[0], t[-1]], y0, t_eval=t, method='RK45')
    return solution.y

initial_guess = np.random.random(size=6)
print("Начальное приближение:", initial_guess)

bounds = [(0.001, 5), (0.001, 5), (0.001, 2), (0.001, 2), (0.0001, 2), (0.0001, 2)]
result = minimize(objective, initial_guess, method='L-BFGS-B')

print("Найденные параметры:", result.x)
r1, r2, a1, a2, b1, b2 = result.x

def lotka_volterra_model(t, y):
    H, L = y
    dHdt = r1 * H - a1 * H * L - b1 * H * H
    dLdt = r2 * L - a2 * H * L - b2 * L * L
    return [dHdt, dLdt]

def runge_kutta_4th_order(f, y0, t_span, n_steps):
    t_start, t_end = t_span
    t = np.linspace(t_start, t_end, n_steps)
    h = (t_end - t_start) / (n_steps - 1)

    y = np.zeros((len(y0), n_steps))
    y[:, 0] = y0

    for i in range(n_steps - 1):
        k1 = h * np.array(f(t[i], y[:, i]))
        k2 = h * np.array(f(t[i] + h/2, y[:, i] + k1/2))
        k3 = h * np.array(f(t[i] + h/2, y[:, i] + k2/2))
        k4 = h * np.array(f(t[i] + h, y[:, i] + k3))

        y[:, i+1] = y[:, i] + (k1 + 2*k2 + 2*k3 + k4) / 6

    return t, y
```

```
H0 = df[x_name].iloc[0]
L0 = df[y_name].iloc[0]

n_steps = 1000
y0 = [L0[0], H0[0]]

forecast_period = 1.5
t_max_original = df[t_name].iloc[-1]
t_min_original = df[t_name].iloc[0]
t_range_original = t_max_original - t_min_original

# Временной диапазон для прогноза
t_span_forecast = (t_min_original, t_min_original + t_range_original * forecast_period)
n_steps_forecast = int(n_steps * forecast_period)

# Вычисляем прогноз
t_forecast, y_forecast = runge_kutta_4th_order(lotka_volterra_model, y0, t_span_forecast, n_steps_forecast)

# Разделяем данные на исторические и прогнозные
historical_mask = t_forecast <= t_max_original
forecast_mask = t_forecast > t_max_original

t_historical = t_forecast[historical_mask]
t_forecast_only = t_forecast[forecast_mask]

H_historical = y_forecast[0][historical_mask]
H_forecast = y_forecast[0][forecast_mask]

L_historical = y_forecast[1][historical_mask]
L_forecast = y_forecast[1][forecast_mask]

# Визуализация с прогнозом
plt.figure(figsize=(16, 10))

# Исторические данные
plt.plot(df[t_name], df[x_name], 'dodgerblue', label='Реальные данные: X', zorder=2, linewidth=2)
plt.plot(df[t_name], df[y_name], 'limegreen', label='Реальные данные: Y', zorder=2, linewidth=2)

# Модель на историческом периоде
plt.plot(t_historical, H_historical, 'royalblue', linestyle='--', label='Модель: X', zorder=1, linewidth=2)
plt.plot(t_historical, L_historical, 'forestgreen', linestyle='--', label='Модель: Y', zorder=1, linewidth=2)

# Прогноз
plt.plot(t_forecast_only, H_forecast, 'royalblue', linestyle=':', alpha=0.7, label='Прогноз: X', zorder=1, linewidth=2)
plt.plot(t_forecast_only, L_forecast, 'forestgreen', linestyle=':', alpha=0.7, label='Прогноз: Y', zorder=1, linewidth=2)

# Вертикальная линия разделяющая историю и прогноз
plt.axvline(x=t_max_original, color='red', linestyle='--', alpha=0.7, label='Начало прогноза')

plt.legend()
plt.xlabel('Год')
plt.ylabel('Численность')
plt.show()

# Фазовый портрет с прогнозом
plt.figure(figsize=(12, 8))

plt.plot(df[x_name], df[y_name], 'darkred', label='Реальные данные', zorder=2, linewidth=2)
plt.plot(H_historical, L_historical, 'red', linestyle='--', label='Модельные данные', zorder=1, linewidth=2)
plt.plot(H_forecast, L_forecast, 'orange', linestyle='-.', label='Прогноз', zorder=1, linewidth=2)

# Точка начала прогноза
start_forecast_idx = np.where(t_forecast >= t_max_original)[0][0]
plt.plot(H_historical[-1], L_historical[-1], 'ro', markersize=8, label='Начало прогноза')

plt.legend()
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```