Лабораторная работа 4
Метод bootstrap

Гунько Анастасия Николаевна
Б9123-01.03.02сп

14.04.2025

Bootstrap

— это статистический метод, который позволяет оценивать характеристики распределений и параметры на основе имеющихся данных.

Такой подход позволяет имитировать получение выборки из генеральной совокупности
Из исходной выборки данных многократно извлекаются выборки того же размера с возвращением, эти выборки будут использоваться для оценки интересующих статистик.

Преимущество метода в необязательности знания типа распределения данных

Алгоритм

1. Исходные данные:
   - Имеем выборку $X = \{X_1, ..., X_n\}$ объёма n
   - Определяем оцениваемый параметр θ (среднее, дисперсию и т.д.)
2. Генерация выборок:

- Задаём количество повторов B
- Для каждого $b = \overline{1, B}$:
  - Формируем выборку X_b^* случайным извлечением n элементов с возвращением из X
  - Вычисляем оценку(статистику) $\hat{\theta}_b^*$ по выборке X_b^*

3. Обработка результатов:

   - Доверительный интервал по процентилям:
     - Сортируем (ранжируем) $\{\hat{\theta}^*_{(1)}, ..., \hat{\theta}^*_{(B)}\}$
     - 95% доверительный интервал: $[\hat{\theta}^*_{(L)}, \hat{\theta}^*_{(U)}]$, где $L = 0.025B$, $U = 0.975B$

Оценки параметров распределений

Равномерное распределение U(a,b):

- Параметр a (нижняя граница) оценивается как минимум выборки:

$$\hat{a} = \min\{x_1, \ldots, x_n\}$$

- Параметр b (верхняя граница) оценивается через размах:

$$\hat{b} = \max\{x_1, \ldots, x_n\} - \min\{x_1, \ldots, x_n\}$$

Распределение Бернулли Bernoulli(p):

- Параметр p оценивается как выборочное среднее:

$$\hat{p} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$$

Биномиальное распределение Bin(k,p):

- Параметр k оценивается методом моментов:

$$\hat{k} = \left[\frac{\overline{X}^2}{\overline{X} - (S^2 - \overline{X}^2)/\overline{X}}\right]$$

где $\overline{X}$ - выборочное среднее, S^2 - выборочная дисперсия

- Параметр p оценивается как:

$$\hat{p} = \frac{\overline{X}}{\hat{k}}$$

Нормальное распределение N(μ,σ):

- Параметр μ оценивается выборочным средним:

$$\hat{\mu} = \frac{1}{n} \cdot \sum_{i=1}^{n} x_i$$

- Параметр σ оценивается исправленной выборочной дисперсией:

$$\hat{\sigma} = \sqrt{\frac{1}{n-1} \sum_{i=1}^{n} \left(x_i - \overline{X} \right)}$$

Код реализации

```
import numpy as np
import pandas as pd
import scipy.stats as stats

params = {
    'U': {'loc': 15, 'scale': 42},
    'Bernoulli': {'p': 0.36},
    'Binom': {'n': 87, 'p': 0.29},
    'Norm': {'loc': 78, 'scale': 20}
}

random_state = 9

samples = {
    'U_100': stats.uniform.rvs(**params['U'], size=100,
        random_state=random_state),
    'U_1000': stats.uniform.rvs(**params['U'], size=1000,
        random_state=random_state),
```

```
    'Bernoulli_100': stats.bernoulli.rvs(**params['Bernoulli'], size=100,
        random_state=random_state),
    'Bernoulli_1000': stats.bernoulli.rvs(**params['Bernoulli'], size=1000,
        random_state=random_state),
    'Binom_100': stats.binom.rvs(**params['Binom'], size=100,
        random_state=random_state),
    'Binom_1000': stats.binom.rvs(**params['Binom'], size=1000,
        random_state=random_state),
    'Norm_100': stats.norm.rvs(**params['Norm'], size=100,
        random_state=random_state),
    'Norm_1000': stats.norm.rvs(**params['Norm'], size=1000,
        random_state=random_state),
}

def my_bootstrap(data, statistic, confidence_level=0.95, n_iterations=1000):
    stats = []
    for _ in range(n_iterations):
        sample = np.random.choice(data, size=len(data), replace=True)
        stats.append(statistic_func(sample))

    alpha = (1 - confidence_level) / 2
    lower = np.percentile(stats, alpha * 100)
    upper = np.percentile(stats, (1 - alpha) * 100)
    return (lower, upper)


def estimate_uniform_loc(data):
    return np.min(data)

def estimate_uniform_scale(data):
    return np.max(data) - np.min(data)

def estimate_bernoulli_p(data):
    return np.mean(data)

def estimate_binom_n(data):
    m1 = np.mean(data)
    m2 = np.mean(np.square(data))
```

```
    n_est = m1 / (1 - (m2 - m1**2) / m1) if m1 != 0 else np.max(data)
    return max(np.ceil(n_est), np.max(data))

def estimate_binom_p(data):
    n_est = estimate_binom_n(data)
    p_est = np.mean(data) / n_est if n_est != 0 else 0
    return np.clip(p_est, 0, 1)

def estimate_normal_mean(data):
    return np.mean(data)

def estimate_normal_std(data):
    return np.std(data)

results = []

for name, data in samples.items():
    if 'U_' in name:
        loc_low, loc_up = my_bootstrap(data, estimate_uniform_loc)
        scale_low, scale_up = my_bootstrap(data, estimate_uniform_scale)

        res_loc = stats.bootstrap((data,), estimate_bernoulli_p,
                  method='percentile')
        res_scale = stats.bootstrap((data,), estimate_bernoulli_p,
                    method='percentile')

        results.extend([
            {'Распределение': name, 'Параметр': 'loc',
             'Истинное значение':  params['U']['loc'],
             'Нижняя граница ДИ': loc_low, 'Верхняя граница ДИ': loc_up,
             'Scipy Stats ДИ': res_loc.confidence_interval},
            {'Распределение': name, 'Параметр': 'scale',
             'Истинное значение':  params['U']['scale'],
             'Нижняя граница ДИ': scale_low, 'Верхняя граница ДИ': scale_up,
             'Scipy Stats ДИ': res_scale.confidence_interval}
        ])

    elif 'Bernoulli_' in name:
```

```
    p_low, p_up = my_bootstrap(data, estimate_bernoulli_p)

    res_p = stats.bootstrap((data,), estimate_bernoulli_p,
            method='percentile')

    results.append({
        'Распределение': name, 'Параметр': 'p',
        'Истинное значение': params['Bernoulli']['p'],
        'Нижняя граница ДИ': p_low, 'Верхняя граница ДИ': p_up,
        'Scipy Stats ДИ': res_p.confidence_interval
    })

elif 'Binom_' in name:
    k_low, k_up = my_bootstrap(data, estimate_binom_n)
    p_low, p_up = my_bootstrap(data, estimate_binom_p)

    res_k = stats.bootstrap((data,), estimate_binom_n,
            method='percentile')
    res_p = stats.bootstrap((data,), estimate_binom_p,
            method='percentile')

    results.extend([
        {'Распределение': name, 'Параметр': 'k',
         'Истинное значение': params['Binom']['n'],
         'Нижняя граница ДИ': k_low, 'Верхняя граница ДИ': k_up,
         'Scipy Stats ДИ': res_k.confidence_interval},
        {'Распределение': name, 'Параметр': 'p',
         'Истинное значение': params['Binom']['p'],
         'Нижняя граница ДИ': p_low, 'Верхняя граница ДИ': p_up,
         'Scipy Stats ДИ': res_p.confidence_interval}
    ])

elif 'Norm_' in name:
    mean_low, mean_up = my_bootstrap(data, estimate_normal_mean)
    std_low, std_up = my_bootstrap(data, estimate_normal_std)

    res_mean = stats.bootstrap((data,), estimate_normal_mean,
               method='percentile')
```

```
        res_std = stats.bootstrap((data,), estimate_normal_std,
                   method='percentile')

        results.extend([
            {'Распределение': name, 'Параметр': 'mean',
             'Истинное значение': params['Norm']['loc'],
             'Нижняя граница ДИ': mean_low,
             'Верхняя граница ДИ': mean_up,
             'Scipy Stats ДИ': res_mean.confidence_interval},
            {'Распределение': name, 'Параметр': 'std',
             'Истинное значение': params['Norm']['scale'],
             'Нижняя граница ДИ': std_low,
             'Верхняя граница ДИ': std_up,
             'Scipy Stats ДИ': res_std.confidence_interval}
        ])


results_df = pd.DataFrame(results)
print('\n')
print("Оценка параметров распределений:\n")
print(results_df[['Распределение', 'Параметр', 'Истинное значение',
    'Нижняя граница ДИ', 'Верхняя граница ДИ', 'Scipy Stats ДИ']]
    .to_string(index=False))
print('\n')
```

Итоговая таблица

Оценка параметров распределений:

Распределение	Параметр	Истинное значение	Нижняя граница ДИ	Верхняя граница ДИ	Scipy Stats ДИ
U_100	loc	15.00	15.435714	16.619979	(32.82730735767097, 37.450532914317044)
U_100	scale	42.00	39.155554	40.865374	(32.81617731904289, 37.43263469361845)
U_1000	loc	15.00	15.002584	15.256017	(35.38787405901569, 36.86010496494446)
U_1000	scale	42.00	41.501736	41.884508	(35.404550318644034, 36.86345801529476)
Bernoulli_100	p	0.36	0.230000	0.410000	(0.23, 0.41)
Bernoulli_1000	p	0.36	0.325975	0.386025	(0.328, 0.387)
Binom_100	k	87.00	48.000000	109.000000	(47.0, 108.0)
Binom_100	p	0.29	0.228170	0.536306	(0.2289867389491243, 0.5257491134751773)
Binom_1000	k	87.00	60.000000	80.000000	(60.0, 81.0)
Binom_1000	p	0.29	0.309672	0.419349	(0.312674375, 0.4217833333333333)
Norm_100	mean	78.00	73.610587	81.997099	(73.60941636516633, 81.84617848628477)
Norm_100	std	20.00	17.872856	23.977477	(17.576244509407353, 23.89014719262372)
Norm_1000	mean	78.00	77.322360	79.872364	(77.34205920047171, 79.87195729468618)
Norm_1000	std	20.00	19.489091	21.229071	(19.521139590066344, 21.2383523603176)