Санкт-Петербургский политехнический университет
Петра Великого

Институт прикладной математики и механики
Высшая школа прикладной математики и вычислительной физики

Отчет по лабораторной работе № 1:

«Классификация многомерных объектов с помощью наивного Байесова классификатора»

по дисциплине: теория принятия экономических решений.

Выполнила студентка:
Заболотских Екатерина Дмитриевна
группа: 3630102/70301

Проверила:
Павлова Людмила Владимировна

Санкт-Петербург
2020 г.

Оглавление

Список таблиц

Постановка задачи

Реализовать метод дискриминантной классификации, а именно Байесовскую процедуру классификации с заменой на состоятельные оценки.

Смоделировать обучающие выборки и тестовую выборку заданных объемов из нормального трехмерного распределения. Найти оценки параметров распределения, построить дискриминантную функцию. Оценить константу C и расстояние Махаланобиса.

Произвести вероятностный анализ ошибочной классификации через построение четырехпольной таблицы сопряженности (матрицы соответствий). Оценить вероятности ошибочной классификации.

Провести анализ данных по этой же схеме из репозитория: https://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german/.

Теория

Общая задача

Пусть имеется матрица **«объект-свойство»**

$$X = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix}$$

, где каждая строка – это p-мерный вектор:

$$x_l = (x_{l1}, x_{l2}, \dots\ x_{lp}), \qquad l = \overline{1, n}$$

советующий результату наблюдения за объектом. Каждый такой вектор принадлежит χ - пространству признаков.

Совокупность объектов, относящихся к одному классу (популяции или группе), будем обозначать как D_i. Эта совокупность образует некоторое облако в χ.

Для того, чтобы процедура классификации проходила успешно, необходимо выполнение условий:

1. Облако D_i должно быть сконцентрировано в некоторой области R_i пространства χ. ($x = \overline{1, k}$)
2. В область R_i должна попасть незначительная часть облаков объектов, советующих другим классам.

Дискриминантные функции дают определение этих областей путем задания их границ в многомерном пространстве.

Если p-мерное наблюдение x попало в область R_i, но при этом вектор x является элементом области D_j, то говорят, что ситуация соответствует ошибочной и определяют вероятность такого события как:

$$P(i \mid j) = P\{x \in R_i \mid x \in D_i\}$$

Не ошибочная $x \in R_i \Rightarrow x \in D_i$.

Бинарная классификация

Параметры распределения популяции известны

Пусть объекты нашего исследования – нормально распределенные случайные многомерные величины, объединенные в две популяции D_1, D_2:

$D_1 : \mathcal{N}\left(\mu^{(1)}, \Sigma_1\right), \qquad f_1(x), \qquad \mu^{(1)}, \mu^{(2)} \in \mathbb{R}^p$

$D_2 : \mathcal{N}\left(\mu^{(2)}, \Sigma_2\right), \qquad f_2(x), \qquad \Sigma_1, \Sigma_2 \in \mathbb{R}^p \times \mathbb{R}^p$

Позволим себе упрощение: $\Sigma_1 = \Sigma_2 = \Sigma = \sigma_{mj}, \qquad m, j \in \overline{1, p}$

Построение дискриминантной функции:

Дискриминантная функция – линейная комбинация наблюдений:

$$z(x) = z = \alpha_1 x_1 + \cdots + \alpha_p x_p,$$

где $\alpha_1, \ldots, \alpha_p$ – постоянные коэффициенты.

Правило классификации:

$$\begin{cases} z \geq c \Longrightarrow x \to D_1 \\ z < c \Longrightarrow x \to D_2 \end{cases}$$

Исходя из этого правила имеем задачу: найти коэффициенты α_i и подобрать константу с.

Если наблюдение x поступило из D_1 популяции, то $z \sim \mathcal{N}(\xi_1, \sigma_z^2)$

$$\xi_1 = \sum_{j=1}^{p} \alpha_j \mu_j^{(1)}, \qquad \sigma_z^2 = \sum_{m=1}^{p} \sum_{j=1}^{p} \alpha_m \sigma_{mj} \alpha_j$$

Если наблюдение x поступило из D_2 популяции, то $z \sim \mathcal{N}(\xi_1, \sigma_z^2)$

$$\xi_2 = \sum_{j=1}^{p} \alpha_j \mu_j^{(2)}, \qquad \sigma_z^2 = \sum_{m=1}^{p} \sum_{j=1}^{p} \alpha_m \sigma_{mj} \alpha_j$$

Имеет смысл выбирать $\alpha_1, \ldots, \alpha_p$ таким образом, чтобы ξ_1и ξ_2 были как можно больше удалены друг от друга относительно дисперсии.

Для этого воспользуемся расстоянием Махаланобиса: $\Delta^2 = \frac{(\xi_1 - \xi_2)^2}{\sigma_z^2}$.

Получается, что задача состоит в том, чтобы $\Delta^2 \to max$.

Для этого решаем систему Роберта-Фишера:

$$\begin{cases} \alpha_1 \sigma_{11} + \alpha_2 \sigma_{12} + \cdots + \alpha_p \sigma_{1p} = \mu_1^{(1)} - \mu_1^{(2)} \\ \\ \alpha_p \sigma_{p1} + \alpha_2 \sigma_{p2} + \cdots + \alpha_p \sigma_{pp} = \mu_p^{(1)} - \mu_p^{(2)} \end{cases}$$

Запишем в матричном виде:

$$\Sigma \cdot \alpha = \mu^{(1)} - \mu^{(2)}$$

$$\alpha = \Sigma^{-1}(\mu^{(1)} - \mu^{(2)})$$

Параметры распределения популяции неизвестны

1. Найдем оценки параметров распределений:
   - Выборочное среднее: $\widehat{\mu}_j^{(k)} = \frac{1}{n_k}\sum_{i=1}^{n_k} x_{ij}^{(k)}, k = 1, 2$
   - Выборочную матрицу ковариаций:

   $$S = \frac{1}{n_1 + n_2 - 2}\left[(n_1 - 1)S^{(1)} + (n_2 - 1)S^{(2)}\right]$$

   $$S^{(k)} = \left(s_{lj}^{(k)}\right), l, j = \overline{1, p}, k = 1,2$$

   $$s_{lj}^{(k)} = \frac{1}{n_k - 1}\sum_{i=1}^{n_k}(x_{il}^{(k)} - \widehat{\mu}_l^{(k)})(x_{ij}^{(k)} - \widehat{\mu}_j^{(k)}), k = 1,2$$

   Заменим вектор параметром дискриминантной функции, $\boldsymbol{\alpha}$, оценкой **a** :

   - $\boldsymbol{\alpha} = \Sigma^{-1}(\mu^{(1)} - \mu^{(2)}) \to \widehat{\alpha} = \boldsymbol{a} = S^{-1}(\widehat{\mu}^{(1)} - \widehat{\mu}^{(2)})$

   Строим оценки ξ_1 и ξ_2:

   - $\xi_1 \to \bar{z}_1 = \frac{1}{n_1}\sum_{i=1}^{n} z_i^{(1)}$ $\quad z_i^{(1)} = a_1 x_{i1}^{(1)} + \cdots + a_p x_{ip}^{(1)}$
   - $\xi_2 \to \bar{z}_2 = \frac{1}{n_2}\sum_{i=1}^{n} z_i^{(2)}$ $\quad z_i^{(2)} = a_1 x_{i1}^{(2)} + \cdots + a_p x_{ip}^{(2)}$

   Находим константу С:

   - $\mathrm{c} = \frac{z1+z2}{2}$

   Таким образом можем произвести соотношение элементов к классам:

   - $x \cdot a < c \to x \in D_1$
   - $x \cdot a \geq c \to x \in D_2$

2. Оцениваем расстояние Махаланобиса (смешанная и несмешанная):
   - $D^2 = \frac{(\bar{z}_1 - \bar{z}_2)^2}{s_z^2}$
   - $D_H^2 = \frac{n_1 + n_2 - p - 3}{n_1 + n_2 - 2} D^2 - p\left(\frac{1}{n_1} + \frac{1}{n_2}\right)$

Вероятность ошибочной классификации

Параметры распределения популяции известны

$$\hat{P}(2\,|1) = \Phi\left(\frac{K - {}^{1}\!/_{2}\,\Delta^{2}}{\Delta}\right), \qquad \hat{P}(1\,|2) = \Phi\left(\frac{-K - {}^{1}\!/_{2}\,\Delta^{2}}{\Delta}\right)$$

Параметры распределения популяции неизвестны

$$\hat{P}(2\,|1) = \Phi\left(\frac{K - {}^{1}\!/_{2}\,D_H^2}{D_H}\right), \qquad \hat{P}(1\,|2) = \Phi\left(\frac{-K - {}^{1}\!/_{2}\,D_H^2}{D_H}\right)$$

$$K = ln\left(\frac{q_1}{q_2}\right)$$

$\Phi(y) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{y} e^{-t^2/2}\,dt$ – функция распределения Лапласа.

Решение

Работа с модельными данными

Модельные данные составляем из трехмерных нормально распределенных случайных векторов:

$$x_k = (x_{k1}, x_{k2}, x_{k3}), \qquad k = \overline{1, n}$$

где n – мощность выборки.

Строим две обучающие выборки – матрицы размером (110, 3) и (90, 3). Данные подчиняются нормальному распределению с математическим ожиданием:

- $\mu_1 = (5, 5, 5)$ $\mu_2 = (-5, 5, 5)$ для «хорошо» разделенных данных;
- $\mu_1 = (3, 3, 3)$ $\mu_2 = (-3, 3, 3)$ для «средне» разделенных данных;
- $\mu_1 = (1, 2, 1)$ $\mu_2 = (-1, 2, 1)$ для «плохо» разделенных данных;

соответственно, и матрицей ковариаций: $\Sigma = \begin{pmatrix} 7 & 3 & 2 \\ 3 & 10 & 5 \\ 2 & 5 & 8 \end{pmatrix}$

Строим тестовые данные – матрицы размером (510, 3) и (500, 3), подчиняющиеся таким же правилам распределения.

«Обучаем» классификатор и получаем оценки параметров распределения и дискриминантную функцию. Находим расстояние Махаланобиса и вероятности ошибочной классификации.

Строим четырехпольную таблицу сопряженности.

Работа с данными из репозитория German

Рассмотрим группы признаков: {1, 3, 10, 12, 21} и {2, 4, 10, 20} из матрицы «объект-свойство» из репозитория German.

Составим обучающие выборки (100, 5) и (90, 5) для обеих групп признаков.

Строим классификатор и получаем оценки параметров распределения и дискриминантную функцию. Находим расстояние Махаланобиса и вероятности ошибочной классификации.

Теперь берем тестовые выборки с теми же признаками (300, 5) (400, 5) советующие вышеприведённым группам признаков, и классифицируем данные с уже «обученным» классификатором.

Строим четырехпольную таблицу сопряженности.

Сравниваем вероятности ошибочной классификации.

Результаты

Работа с модельными данными

«Хорошо» разделенные данные

Характеристики классификатора:

(ОВ: $n_1 = 110$ $n_2 = 90$)

- Вероятности ошибочной классификации:

  $\hat{P}(2 \mid 1) = 0.02411$

  $\hat{P}(1 \mid 2) = 0.01912$

- Смещенное и несмещенное расстояния Махаланобиса:

  $D^2 = 17.18925$

  $D_H^2 = 16.78139$

Четырехпольная таблица сопряженности для обучающих выборок:

Pred→ True↓	1	2
1	107	3
2	3	87

Таблица 1: ОВ "хорошо" разделенные данные

Вероятность ошибочной классификации по четырехпольной таблице сопряженности:

$\tilde{P}(2) = 0.0(3)$

$\tilde{P}(1) = 0.0(27)$

Четырехпольная таблица сопряженности для тестовых выборок:

(ТВ: $n_1 = 510$ $n_2 = 500$)

Pred→ True↓	1	2
1	496	14
2	15	485

Таблица 2: ТВ "хорошо" разделенные данные

Вероятность ошибочной классификации по четырехпольной таблице сопряженности:

$\tilde{P}(2) = 0.03$

$\tilde{P}(1) = 0.02745$

«Средне» разделенные данные

Характеристики классификатора:

(ОВ: $n_1 = 110\ n_2 = 90$)

- Вероятности ошибочной классификации:

  $\hat{P}(2 \mid 1) = \ 0.146503$

  $\hat{P}(1 \mid 2) = \ 0.110754$

- Смещенное и несмещенное расстояния Махаланобиса:

  $D^2 = 5.50992$

  $D_H^2 = 5.338$

Четырехпольная таблица сопряженности для обучающих выборок:

Pred→ True↓	1	2
1	96	13
2	11	79

Таблица 3: ОВ "средне" разделенные данные

Вероятность ошибочной классификации по четырехпольной таблице сопряженности:

$\tilde{P}(2) = \ 0.1(2)$

$\tilde{P}(1) = \ 0.1(18)$

Четырехпольная таблица сопряженности для тестовых выборок:

(ТВ: $n_1 = 510\ n_2 = 500$)

Pred→ True↓	1	2
1	468	42
2	45	455

Таблица 4: ТВ "средне" разделенные данные

Вероятность ошибочной классификации по четырехпольной таблице сопряженности:

$\tilde{P}(2) = \ 0.09$

$\tilde{P}(1) = \ 0.08235$

«Плохо» разделенные данные

Характеристики классификатора:

(ОВ: $n_1 = 110\ n_2 = 90$)

- Вероятности ошибочной классификации:

  $\hat{P}(2 \mid 1) = \ 0.46394$

  $\hat{P}(1 \mid 2) = \ 0.2737$

- Смещенное и несмещенное расстояния Махаланобиса:

  $D^2 = 0.616534$

  $D_H^2 = 0.543472$

Четырехпольная таблица сопряженности для обучающих выборок:

Pred→ True ↓	1	2
1	68	42
2	35	55

Таблица 5: ОВ "плохо" разделенные данные

Вероятность ошибочной классификации по четырехпольной таблице сопряженности:

$\tilde{P}(2) = \ 0.3(8)$

$\tilde{P}(1) = \ 0.3(81)$

Четырехпольная таблица сопряженности для тестовых выборок:

(ТВ: $n_1 = 510\ n_2 = 500$)

Pred→ True ↓	1	2
1	346	164
2	197	303

Таблица 6: ТВ "плохо" разделенные данные

Вероятность ошибочной классификации по четырехпольной таблице сопряженности:

$\tilde{P}(2) = \ 0.394$

$\tilde{P}(1) = \ 0.3215686$

Работа с данными из репозитория German

Группа признаков: {1, 3, 10, 12, 21}

Характеристики классификатора:

(ОВ: $n_1 = 100\ n_2 = 90$)

- Вероятности ошибочной классификации:

  $\hat{P}(2 \mid 1) = 0.3203516$

  $\hat{P}(1 \mid 2) = 0.425734$

- Смещенное и несмещенное расстояния Махаланобиса:

  $D^2 = 0.56851$

  $D_H^2 = 0.493086$

Четырехпольная таблица сопряженности для обучающих выборок:

Pred→ True ↓	1	2
1	73	27
2	39	51

Таблица 7: German ОВ 1

Вероятность ошибочной классификации по четырехпольной таблице сопряженности:

$\tilde{P}(2) = 0.4(3)$

$\tilde{P}(1) = 0.27$

Четырехпольная таблица сопряженности для тестовых выборок:

(ТВ: $n_1 = 400\ n_2 = 300$)

Pred→ True ↓	1	2
1	326	74
2	110	190

Таблица 8: German ТВ 1

Вероятность ошибочной классификации по четырехпольной таблице сопряженности:

$\tilde{P}(2) = 0.3(6)$

$\tilde{P}(1) = 0.185$

Группа признаков: {2, 4, 10, 20}

Характеристики классификатора:
(ОВ: $n_1 = 100\ n_2 = 90$)

- Вероятности ошибочной классификации:

  $\hat{P}(2 \mid 1) = 0.33286$

  $\hat{P}(1 \mid 2) = 0.4566$

- Смещенное и несмещенное расстояния Махаланобиса:

  $D^2 = 0.424882$

  $D_H^2 = 0.352509$

Четырехпольная таблица сопряженности для обучающих выборок:

Pred→ True ↓	1	2
1	54	46
2	26	64

Таблица 9: German OB 2

Вероятность ошибочной классификации по четырехпольной таблице сопряженности:

$\tilde{P}(2) = 0.2(8)$

$\tilde{P}(1) = 0.46$

Четырехпольная таблица сопряженности для тестовых выборок:

(ТВ: $n_1 = 400\ n_2 = 300$)

Pred→ True ↓	1	2
1	137	263
2	85	215

Таблица 10: German TB 2

Вероятность ошибочной классификации по четырехпольной таблице сопряженности:

$\tilde{P}(2) = 0.28(3)$

$\tilde{P}(1) = 0.6575$

Выводы

Работа с модельными данными

1. Классификатор лучше справляется с «хорошо» разделенными данными. Причем чем более отдалены облака данных друг от друга, тем меньше вероятность ошибочной классификации. Чем хуже отделенность данных, тем больше классификация похожа на случайный выбор.

2. Объем обучающей выборки влияет на эффективность классификатора: чем больше объем обучающей выборки, тем точнее будет классификация.

Работа с данными из репозитория German

1. Данные из репозитория «плохо» разделены, из-за чего мы получаем большую вероятность ошибочной классификации. Причем она практически не меняется в зависимости от размера обучающих выборок, значит данные однородны. С такими данными работа классификатора практически бесполезна.

Таким образом можно сделать вывод, что байесовская процедура классификации эффективна для данных, которые можно отделить друг от друга, т.е. зависит от типа данных и их распределения.

Приложение

Код работы: https://github.com/KateZabolotskih/EconomicDecisionMaking

```
from math import sqrt, log, exp, fabs, erf
import numpy as np
from matplotlib import pyplot as plt

# functions for counting covariation matrix made of consistent estimates
def s_k_l_j(m, l, j, n_k, means):
    s = 0
    for i in range(n_k):
        s += (m[l][i] - means[l]) * (m[j][i] - means[j])
    return s / (n_k - 1)

def s_k(samples, p, n_k, means):
    s_k = np.zeros((p, p))
    for y in range(p):
        for x in range(p):
            s_k[x, y] = s_k_l_j(samples, y, x, n_k, means)
    return s_k

def covariation_matrix(s1, s2, mean1, mean2):
    p = len(s1)
    n_1 = len(s1[0])
    n_2 = len(s2[0])
    s1 = s_k(s1, p, n_1, mean1)
    s2 = s_k(s2, p, n_2, mean2)
    cov = ((n_1 - 1) * s1 + (n_2 - 1) * s2) / (n_1 + n_2 - 2)
    return cov

# the main class of the task of the discriminant analysis
class BayesianClassification:
    # initialization of consistent estimates:
    # mean, covariation matrix, z1, z2, q1, q2
    # also counting mahalanobis distance and the probability of erroneous
classification
    def __init__(self, sample1, sample2, mean1 = 0, mean2 = 0, cov = 0):
        self.sample1 = sample1
        self.sample2 = sample2
        self.n1 = len(sample1[0])
        self.n2 = len(sample2[0])

        if (mean1 == 0):
            self.mean1 = np.array([np.mean(row) for row in self.sample1])
            self.mean2 = np.array([np.mean(row) for row in self.sample2])
            self.cov_matrix = covariation_matrix(self.sample1, self.sample2,
self.mean1, self.mean2)
        else:
            self.mean1 = mean1
            self.mean2 = mean2
            self.cov_matrix = cov

        self.a = np.linalg.inv(self.cov_matrix).dot(self.mean1 - self.mean2)
        print(self.cov_matrix)
        self.z1, self.z2 = 0, 0
        for i in range(self.n1):
            self.z1 += self.a.dot(self.sample1[:, i])
        for i in range(self.n2):
```

```
            self.z2 += self.a.dot(self.sample2[:, i])
        self.z1 /= self.n1
        self.z2 /= self.n2

        self.r1 = self.n1 / (self.n1 + self.n2)
        self.r2 = self.n2 / (self.n1 + self.n2)

        self.c = (self.z1 + self.z2) / 2 + log(self.r1 / self.r2)

        self.D = 0
        self.DH = 0
        self._culc_mahalanobis_distance()
        self.P_12 = 0
        self.P_21 = 0
        self._culc_prob()

    def _classify(self, array, is_q_set=True):
        compare = lambda arr: arr.dot(self.a) < self.c + log(self.r2 / self.r1)
        if not is_q_set:
            compare = lambda arr: arr.dot(self.a) < self.c
        if compare(array):
            return 0
        else:
            return 1

    def plot_data_3D(self, sample_1, sample_2):
        x_max = max(max(sample_1[0]), max(sample_2[0])) + 1
        x_min = min(min(sample_1[0]), min(sample_2[0])) - 1
        y_max = max(max(sample_1[1]), max(sample_2[1])) + 1
        y_min = min(min(sample_1[1]), min(sample_2[1])) - 1
        n_1 = len(sample_1[0])
        n_2 = len(sample_2[0])

        x = np.linspace(x_min, x_max, 4)
        y = np.linspace(y_min, y_max, 4)
        X, Y = np.meshgrid(x, y)
        f = lambda _x, _y: ((- self.a[0] * _x - self.a[1] * _y) +
np.full(_x.shape, self.c)) / self.a[2]
        Z = f(X, Y)

        fig = plt.figure()
        ax = fig.add_subplot(projection='3d')

        for i in range(n_1):
            x = sample_1[:, i]
            if self._classify(x) == 1:
                ax.scatter(x[0], x[1], x[2], c='teal', marker='+')
            else:
                ax.scatter(x[0], x[1], x[2], c='teal', marker='o')

        for i in range(n_2):
            x = sample_2[:, i]
            if self._classify(x) == 0:
                ax.scatter(x[0], x[1], x[2], c='mediumpurple', marker='*')
            else:
                ax.scatter(x[0], x[1], x[2], c='mediumpurple', marker='^')

        ax.set_xlabel('X')
        ax.set_ylabel('Y')
        ax.set_zlabel('Z')
        ax.plot_wireframe(X, Y, Z, color='grey')
```

```
        plt.show()
        return

    def classify(self, sample_1, sample_2):
        q1 = 0
        q2 = 0
        n1 = len(sample_1[0])
        n2 = len(sample_2[0])

        for i in range(n1):
            x = sample_1[:, i]
            if self._classify(x) == 1:
                q1 += 1

        for i in range(n2):
            x = sample_2[:, i]
            if self._classify(x) == 0:
                q2 += 1
        print('q1 = ' + str(q1) + ' q2 = ' + str(q2))

    def _culc_prob(self):
        F = lambda x: 1 / 2 * (1 + erf(x / sqrt(2)))
        K = log(self.r2 / self.r1)

        self.P_12 = F((-K - self.DH / 2) / sqrt(self.D))

        self.P_21 = F((K - self.DH / 2) / sqrt(self.D))

    def _culc_mahalanobis_distance(self, p=3):
        _sum = (self.a.dot(self.cov_matrix)).dot(self.a)

        self.D = (self.z1 - self.z2) * (self.z1 - self.z2) / _sum
        self.DH = fabs((self.n1 + self.n2 - p - 3) / (self.n1 + self.n2 - 2) *
self.D - p * (1 / self.n1 + 1 / self.n2))
```