МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И.УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЁТ

по практической работе №5
по дисциплине «Машинное обучение»
Тема: Кластеризация

Студент гр. 6304 _____ Корытов П.В.
Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2020

1. Задание 1

$$D = \{2, 3, 4, 10, 11, 12, 20, 25, 30\}. \tag{1.1}$$

Начальные средние значения — $\mu_1 = 2, \mu_2 = 4, \mu_3 = 6$.

$$C_1 = \{2, 3\}, C_2 = \{4\}, C_3 = \{10, 11, 12, 20, 25, 30\}. \tag{1.2}$$

Новые средние значения:

$$\mu_1 = 2.5, \mu_2 = 4, \mu_3 = 18. \tag{1.3}$$

Кластеры после 1-й итерации:

$$C_1 = \{2, 3\}, C_2 = \{4, 10, 11\}, C_3 = \{12, 20, 25, 30\}. \tag{1.4}$$

Средние значения для 2-й итерации:

$$\mu_1 = 2.5, \mu_2 = 8.33, \mu_3 = 21.75. \tag{1.5}$$

2. Задание 2

2.1. Оценка максимального правдоподобия

$$\mu_i = \frac{\sum_{j=1}^{n} w_{ij} \cdot x_j}{\sum_{j=1}^{n} w_{ij}}. \tag{2.1}$$

$$\mu_1 = \frac{0.9 \cdot 2 + 0.8 \cdot 3 + 0.3 \cdot 7 + 0.1 \cdot 9 + 0.9 \cdot 2 + 0.8 \cdot 1}{0.9 + 0.8 + 0.3 + 0.1 + 0.9 + 0.8} \approx 2.58 \tag{2.2}$$

$$\mu_2 = \frac{0.1 \cdot 2 + 0.1 \cdot 3 + 0.7 \cdot 7 + 0.9 \cdot 9 + 0.1 \cdot 2 + 0.2 \cdot 1}{0.1 + 0.1 + 0.7 + 0.9 + 0.1 + 0.2} \approx 6.62 \tag{2.3}$$

2.2. Вероятности принадлежности точки к кластерам

Для точки $x = 5$

$$\mu_1 = 2, \mu_2 = 7; \sigma_1 = \sigma_2 = 1; P(C_1) = P(C_2) = 0.5; P(x = 5) = 0.029.$$

$$f(x_1|\mu_1, \sigma_1^2) = \frac{1}{\sqrt{2\pi}\sigma_1} \exp -\frac{(x-\mu_1)^2}{2\sigma_1^2} \approx 0.0044. \tag{2.4}$$

$$f(x_2|\mu_2, \sigma_2^2) \approx 0.0539. \tag{2.5}$$

$$\begin{aligned} P(C_1|x) &= \frac{f(x|\mu_1, \sigma_1^2) \cdot P(C_1)}{f(x|\mu_1, \sigma_1^2) \cdot P(C_1) + f(x|\mu_2, \sigma_2^2) \cdot P(C_2)} \\ &= \frac{0.0044 \cdot 0.5}{(0.0044 \cdot 0.5 + 0.0539 \cdot 0.5)} \\ &\approx 0.0758. \end{aligned} \tag{2.6}$$

$$P(C_2|x) = \frac{0.0539 \cdot 0.5}{(0.0044 \cdot 0.5 + 0.0539 \cdot 0.5)} \approx 0.9241. \tag{2.7}$$

3. Задание 3

Для выполнения задания написана программа на Python. Код приведен в приложении А.

Результаты:

1. RC, single link:
   1.1. [[0], [1], [2], [3], [4], [5]]
   1.2. [[0], [1], [2], [3, 4], [5]]
   1.3. [[0, 3, 4], [1], [2], [5]]
   1.4. [[0, 1, 3, 4], [2], [5]]
   1.5. [[0, 1, 2, 3, 4], [5]]
   1.6. [[0, 1, 2, 3, 4, 5]]
2. SMC, complete link:
   2.1. [[0], [1], [2], [3], [4], [5]]
   2.2. [[0], [1], [2], [3, 4], [5]]
   2.3. [[0, 5], [1], [2], [3, 4]]
   2.4. [[0, 5], [1, 2], [3, 4]]
   2.5. [[0, 3, 4, 5], [1, 2]]
   2.6. [[0, 1, 2, 3, 4, 5]]
3. JS, group average:
   3.1. [[0], [1], [2], [3], [4], [5]]

3.2. [[0], [1], [2], [3, 4], [5]]

3.3. [[0, 5], [1], [2], [3, 4]]

3.4. [[0, 5], [1, 2], [3, 4]]

3.5. [[0, 3, 4, 5], [1, 2]]

3.6. [[0, 1, 2, 3, 4, 5]]

Дендрограммы:

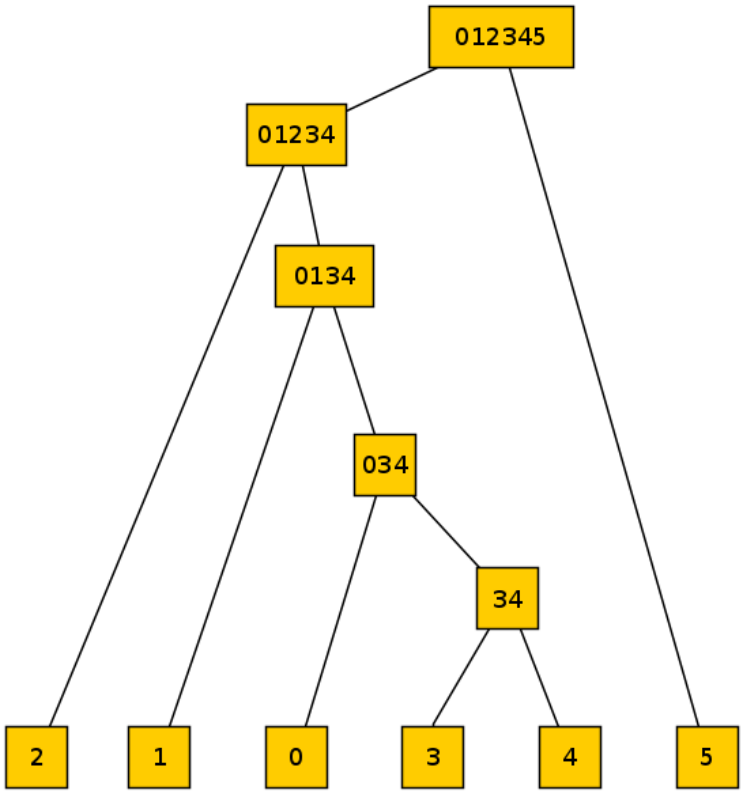

Рисунок 1 – RC, single link

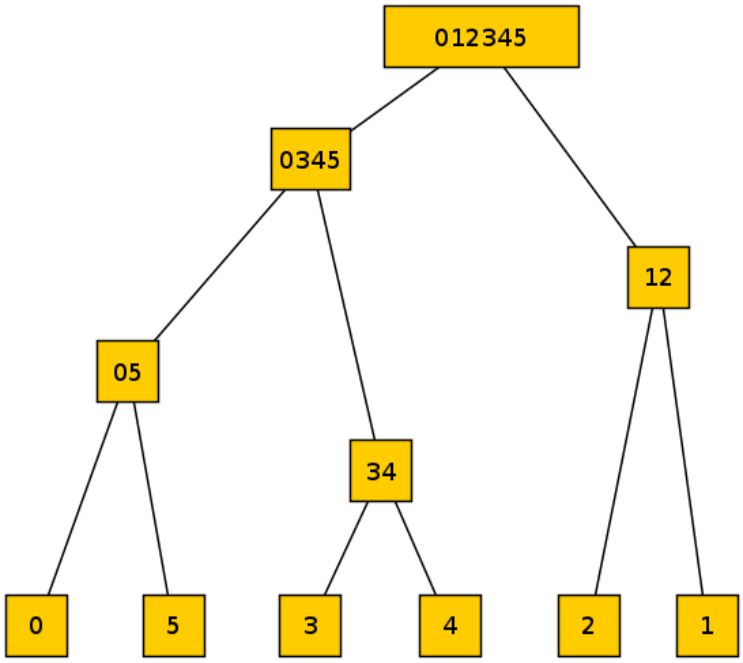

Рисунок 2 – SMC, complete link

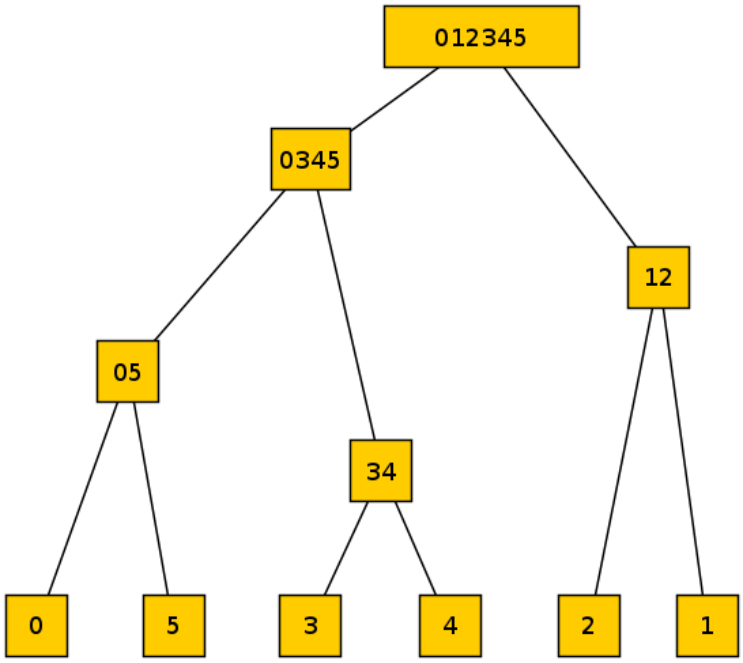

Рисунок 3 – JC, group average

ПРИЛОЖЕНИЕ А

Реализация кластеризации

```
import numpy as np
import tabulate

tabulate._table_formats['booktabs_raw'] = tabulate._table_formats['latex_booktabs']._replace(**{
    'headerrow': tabulate._table_formats['latex_raw'].headerrow,
    'datarow': tabulate._table_formats['latex_raw'].datarow,
})


X = [
    [1, 0, 1, 1, 0],
    [1, 1, 0, 1, 0],
    [0, 0, 1, 1, 0],
    [0, 1, 0, 1, 0],
    [1, 0, 1, 0, 1],
    [0, 1, 1, 0, 0]
]

def get_n(x_1, x_2):
    res = [[0, 0], [0, 0]]
    for a, b in zip(x_1, x_2):
        res[a][b] += 1
    return res

def jc(x_1, x_2):
    n = get_n(x_1, x_2)
    return n[1][1] / (n[1][1] + n[1][0] + n[0][1])

def smc(x_1, x_2):
    n = get_n(x_1, x_2)
    return (n[1][1] + n[0][0]) / np.sum(n)

def rc(x_1, x_2):
    n = get_n(x_1, x_2)
    return n[1][1] / np.sum(n)

def single_link(D, C_1, C_2):
    min_ = None
    for c_1 in C_1:
        for c_2 in C_2:
            d = D[c_1][c_2]
            if min_ is None or d < min_:
                min_ = d
    return min_
```

```
45
46  def complete_link(D, C_1, C_2):
47      max_ = None
48      for c_1 in C_1:
49          for c_2 in C_2:
50              d = D[c_1][c_2]
51              if max_ is None or d > max_:
52                  max_ = d
53      return max_
54
55  def group_average(D, C_1, C_2):
56      avg = 0
57      for c_1 in C_1:
58          for c_2 in C_2:
59              avg += D[c_1][c_2]
60      avg /= len(C_1) * len(C_2)
61      return avg
62
63  import copy
64
65  def do_cluster(X, p_dist, c_dist):
66      clusters = [[i] for i in range(len(X))]
67      levels = [copy.deepcopy(clusters)]
68      D = [[p_dist(x_1, x_2) for x_1 in X] for x_2 in X]
69      while len(clusters) > 1:
70          min_i, min_j, min_d = None, None, None
71          for i in range(len(clusters)):
72              for j in range(i + 1, len(clusters)):
73                  d = c_dist(D, clusters[i], clusters[j])
74                  if min_d is None or d < min_d:
75                      min_i, min_j, min_d = i, j, d
76          clusters[min_i].extend(clusters.pop(min_j))
77          clusters = [sorted(c) for c in clusters]
78          levels.append(copy.deepcopy(clusters))
79      return levels
80
81  def save_levels(levels, name):
82      with open(name, 'w') as f:
83          f.write('\\begin{enumerate}\n')
84          for level in levels:
85              f.write(f'\\item \\ {level}\n')
86          f.write('\\end{enumerate}')
87
88  l_1 = do_cluster(X, rc, single_link)
89  save_levels(l_1, 'l_1.tex')
90  l_1
91
92  l_2 = do_cluster(X, smc, complete_link)
```

```
93  save_levels(l_2, 'l_2.tex')
94  l_2
95
96  l_3 = do_cluster(X, jc, group_average)
97  save_levels(l_3, 'l_3.tex')
98  l_3
```