**ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ОЛЕСЯ ГОНЧАРА**

Факультет прикладної математики

Кафедра обчислювальної математики та математичної кібернетики

ЗВІТ

про виконання лабораторної роботи №2

з дисципліни «Системи і методи прийняття рішень»

Виконав:

студент групи ПА-18-1

Щербак Роман

Викладач:

канд. фіз.-мат. наук, доцент

Н. Л. Козакова

Дніпро

2022

1. **Постановка задачі**
2. Опрацювати необхідний теоретичний матеріал.
3. Побудувати множини Баєса геометричним методом та методом контрольної точки для варіанту індивідуального завдання.
4. Провести аналіз якості рішення в даній інформаційній ситуації.
5. Оформити звіт про використання роботи, який повинен містити такі елементи:
   * постановку індивідуального завдання;
   * отримані множини рішень за критерієм Баєса;
   * аналіз результатів.
6. **Опис розв’язку**

Перевірку роботи програми здійснено для таких вхідних даних:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| p |  |  |  |  |
| 0.3 |  | 1 | 3 | 1 |
| 0.2 |  | 2 | 1 | 3 |
| 0.5 |  | 3 | 1 | 2 |

1. **Код програми мовою Python**

* Модуль main.py

import sympy as sp

import get\_data as gd

import bayes\_values as bv

import geometric as g

import variational as v

print()

p\_1, p\_2 = sp.symbols("p\_1, p\_2")

# file\_name = "example.txt"

file\_name = "variant.txt"

p\_var, eff = gd.get\_data\_from\_file(file\_name)

p\_geom = [p\_1, p\_2, 1-p\_1-p\_2]

print("Геометричний метод: ")

print()

Bayes\_values\_geom = bv.get\_bayes\_values(p\_geom, eff)

g.geom\_method(Bayes\_values\_geom, p\_1, p\_2, file\_name)

print()

print("Метод варіації контрольної точки:")

print()

Bayes\_values\_var = bv.get\_bayes\_values(p\_var, eff)

v.variational\_method(Bayes\_values\_var, p\_var, eff)

* Модуль geometric.py

import re as re

import matplotlib.pyplot as plt

import prettytable as pt

def geom\_method(Bayes\_val, p\_1, p\_2, data\_file):

coefficients = get\_coefs(Bayes\_val, p\_1, p\_2)

equations = get\_equations(coefficients)

fig, ax = plt.subplots()

ax.set\_xlim(0, 1)

ax.set\_ylim(0, 1)

for e in equations:

print(e)

print()

points = get\_points(equations)

line\_arr = plotting\_lines(points, ax, equations)

first\_legend = ax.legend(handles=line\_arr, loc='upper right')

ax.add\_artist(first\_legend)

Bayes\_values\_table = pt.PrettyTable()

Bayes\_values\_table.field\_names = ["Координати", "B+(p, φ\_1)", "B+(p, φ\_2)", "B+(p, φ\_3)", "Максимальне значення", "Баєсова множина"]

if data\_file == "example.txt":

x = [0, 0, 0, 0.45, 0.65, 0.9]

y = [0, 0.6, 0.9, 0, 0, 0]

elif data\_file == "variant.txt":

x = [0, 0, 0, 0.2, 0.4, 0.8, 0.6, 0.2, 0.2, 0.6, 0.4, 0.5, 0, 0]

y = [0, 0.6, 0.9, 0, 0, 0.6, 0.4, 0.6, 0.2, 0.1, 0.6, 0.8, 0.2, 0.4]

scatter\_arr = [0, 0, 0]

i = 0

while i < len(x):

row = create\_new\_row(x[i], y[i], Bayes\_val, p\_1, p\_2, points, ax, scatter\_arr)

Bayes\_values\_table.add\_row(row)

i += 1

ax.legend(handles=scatter\_arr, loc='lower right')

print(Bayes\_values\_table)

plt.title("Геометричний метод")

plt.show()

def get\_coefs(Bayes\_val, p\_1, p\_2):

coefs = []

i = 0

while i < len(Bayes\_val):

j = i+1

while j < len(Bayes\_val):

difference = Bayes\_val[i]-Bayes\_val[j]

difference = str(difference)

split\_string = re.split(r"[\*]?p\_\d", difference)

k = 0

while k < len(split\_string):

if re.search(r"\D$", split\_string[k]):

split\_string[k] += '1'

else:

temp\_string = re.split(r"\s", split\_string[k])

t = 0

while t < len(temp\_string):

if re.search(r"\D$", temp\_string[t]):

temp\_string[t] += temp\_string[t+1]

temp\_string.remove(temp\_string[t+1])

t += 1

if re.search(r"\D?\d", temp\_string[0]) and len(temp\_string) > 1:

split\_string[k] = temp\_string[0]

split\_string[k+1] = temp\_string[1]

k += 1

if len(split\_string) < 3:

if "p\_1" not in str(difference):

split\_string.insert(0, '0')

if "p\_2" not in str(difference):

split\_string.insert(1, '0')

if len(split\_string) < 3:

split\_string.insert(2, '0')

if '' in split\_string:

ind = split\_string.index('')

split\_string[ind] = '0'

coefs.append(split\_string)

j += 1

i += 1

return coefs

def get\_equations(coefs):

equations = []

i = 0

while i < len(coefs):

change\_coefs(coefs[i])

equations.append(build\_equations(coefs[i]))

i += 1

return equations

def change\_coefs(coefs):

j = 0

while j < len(coefs):

coefs[j] = coefs[j].replace(" ", "").replace("+", "")

if j == 0 or j == 2:

if re.match("\d", coefs[j]):

coefs[j] = "-" + coefs[j]

elif re.match("-\d", coefs[j]):

coefs[j] = coefs[j].replace("-", "")

if len(coefs[j]) == 0:

coefs[j] = "1"

j += 1

def build\_equations(coefs):

k = 0

while k < len(coefs):

coefs[k] = int(coefs[k])

k += 1

if coefs[1] != "1":

k = 0

coef\_p\_2 = int(coefs[1])

while k < len(coefs):

coefs[k] = round(int(coefs[k]) / coef\_p\_2, 2)

k += 1

if coefs[2] > 0:

return f"p\_2 = {coefs[0]}\*p\_1 + {coefs[2]}"

elif coefs[2] < 0:

return f"p\_2 = {coefs[0]}\*p\_1 - {abs(coefs[2])}"

else:

return f"p\_2 = {coefs[0]}\*p\_1"

def get\_points(equs):

points = []

i = 0

while i < len(equs):

point\_i = []

j = 0

try:

for\_point\_calc = equs[i].replace(r"p\_2 = ", r"").split(r"\*p\_1 ")

while j < len(for\_point\_calc):

for\_point\_calc[j] = float(for\_point\_calc[j].replace(" ", ""))

j += 1

except ValueError:

for\_point\_calc = equs[i].replace(r"p\_2 = ", r"").split(r"\*p\_1")

print(for\_point\_calc)

while j < len(for\_point\_calc):

if for\_point\_calc[j] != "":

for\_point\_calc[j] = float(for\_point\_calc[j])

j += 1

k = 0

while k < 2:

if for\_point\_calc[0] == "":

point\_i.append((k, round(for\_point\_calc[1],2)))

elif for\_point\_calc[1] == "":

point\_i.append((k, round(for\_point\_calc[0]\*k,2)))

else:

point\_i.append((k, round(for\_point\_calc[0]\*k+for\_point\_calc[1],2)))

k += 1

points.append(point\_i)

i += 1

return points

def plotting\_lines(points, ax, equs):

line\_arr = []

i = 0

while i < len(points):

x\_values = [points[i][0][0], points[i][1][0]]

y\_values = [points[i][0][1], points[i][1][1]]

ax.fill\_between([0,1], y\_values, alpha=0.4)

line = ax.plot(x\_values, y\_values, label=equs[i])[0]

line\_arr.append(line)

i += 1

return line\_arr

def get\_bayes\_val\_for\_point(x, y, Bayes\_val, p\_1, p\_2):

point\_Bayes\_val = []

for i in Bayes\_val:

i = i.replace(p\_1, x).replace(p\_2, y)

point\_Bayes\_val.append(i)

max\_value = max(point\_Bayes\_val)

max\_ind = point\_Bayes\_val.index(max\_value)

return point\_Bayes\_val, max\_value, max\_ind+1

def create\_new\_row(x, y, Bayes\_val, p\_1, p\_2, points, ax, scatter\_arr):

row = []

row.append((x, y))

point\_Bayes\_val, index, max\_index = get\_bayes\_val\_for\_point(x, y, Bayes\_val, p\_1, p\_2)

Bayes\_set = f"φ\_{max\_index}"

if max\_index == 1:

if scatter\_arr[0] == 0:

scatter\_arr[0] = ax.scatter(x, y, c = "red", label = Bayes\_set, marker='o')

else:

ax.scatter(x, y, c = "red")

elif max\_index == 2:

if scatter\_arr[1] == 0:

scatter\_arr[1] = ax.scatter(x, y, c = "blue", label = Bayes\_set, marker='o')

else:

ax.scatter(x, y, c = "blue")

elif max\_index == 3:

if scatter\_arr[2] == 0:

scatter\_arr[2] = ax.scatter(x, y, c = "green", label = Bayes\_set, marker='o')

else:

ax.scatter(x, y, c = "green")

for i in point\_Bayes\_val:

row.append(i)

row.append(index)

row.append(Bayes\_set)

return row

* Модуль variational.py

import sympy as sp

import numpy as np

import re as re

def variational\_method(Bayes\_val, p\_arr, eff\_arr):

diff = []

i = 0

while i < len(Bayes\_val):

j = i+1

while j < len(Bayes\_val):

if Bayes\_val[i] > Bayes\_val[j]:

diff.append(Bayes\_val[i]-Bayes\_val[j])

else:

diff.append(Bayes\_val[j]-Bayes\_val[i])

j += 1

i += 1

q\_1, q\_2, q\_3 = sp.symbols("q\_1, q\_2, q\_3")

variations\_arr = [q\_1, q\_2, q\_3]

build\_border\_between\_sets(0, 1, eff\_arr, p\_arr, diff[0], variations\_arr)

print()

build\_border\_between\_sets(0, 2, eff\_arr, p\_arr, diff[1], variations\_arr)

print()

build\_border\_between\_sets(1, 2, eff\_arr, p\_arr, diff[2], variations\_arr)

def build\_border\_between\_sets(set\_1, set\_2, eff\_arr, p\_arr, delta, var\_arr):

d = []

i = 0

while i < len(eff\_arr):

d.append(eff\_arr[i][set\_1] - eff\_arr[i][set\_2])

i += 1

dot\_prod = (np.dot(d, var\_arr))

points = get\_points(dot\_prod, delta, var\_arr, p\_arr)

get\_border\_equation(points)

def get\_points(dot, delta, var\_arr, p\_arr):

points = []

i = 0

while i < 2:

res = sp.solve([sp.Eq(dot, delta), sp.Eq(var\_arr[i], 0), sp.Eq(sum(var\_arr), 0)])

res\_vals = list(res.values())

j = 0

point = []

while j < len(p\_arr):

point.append(p\_arr[j]+res\_vals[j])

j += 1

points.append(point)

i += 1

return points

def get\_border\_equation(points):

p\_1, p\_2 = sp.symbols("p\_1, p\_2")

border\_equation = [(points[1][1]-points[0][1])\*(p\_1-points[0][0]), ((p\_2-points[0][1])\*(points[1][0]-points[0][0]))]

i = 0

while i < len(border\_equation):

border\_equation[i] = str(border\_equation[i])

if type(re.search(r"\Ap\_\d", border\_equation[i])) != None:

border\_equation[i] = re.sub(r"\Ap\_", "1\*p\_", border\_equation[i])

border\_equation[i] = re.sub(r"p\_\d", r"", border\_equation[i])

border\_equation[i] = re.split(r" ", border\_equation[i])

j = 0

while j < len(border\_equation[i]):

if border\_equation[i][j] == "+" or border\_equation[i][j] == "-":

border\_equation[i][j] += border\_equation[i][j+1]

border\_equation[i].remove(border\_equation[i][j+1])

break

j += 1

k = 0

while k < len(border\_equation[i]):

if type(re.search(r"\\*", border\_equation[i][k])) == re.Match:

border\_equation[i].insert(0, re.sub(r"\\*", "", border\_equation[i][k]))

border\_equation[i][k+1] = ""

border\_equation[i].remove(border\_equation[i][k+1])

break

k += 1

i += 1

i = 0

while i < len(border\_equation):

j = 0

while j < len(border\_equation[i]):

border\_equation[i][j] = float(border\_equation[i][j])

j += 1

i += 1

b = round((-border\_equation[1][1] + border\_equation[0][1]) / (border\_equation[1][0]), 2)

k = round((border\_equation[0][0])/(border\_equation[1][0]),2)

if b > 0:

print(f"p\_2 = {k}\*p\_1 + {b}")

elif b == 0:

print(f"p\_2 = {k}\*p\_1 + {b}")

else:

print(f"p\_2 = {k}\*p\_1{b}")

* Модуль bayes\_values.py

def get\_bayes\_values(p\_arr, eff\_arr):

Bayes\_val = []

i = 0

while i < len(eff\_arr[0]):

res = 0

j = 0

while j < len(eff\_arr):

res += eff\_arr[j][i] \* p\_arr[j]

if type(res) == float:

res = round(res, 2)

j += 1

Bayes\_val.append(res)

i += 1

return Bayes\_val

* Модуль get\_data.py

def get\_data\_from\_file(file\_name):

p = []

eff = []

f = open(str(file\_name))

for s in f:

s = s.strip().split(" ")

arr = []

i = 0

while i < len(s):

if i == 0:

p.append(float(s[i]))

else:

arr.append(int(s[i]))

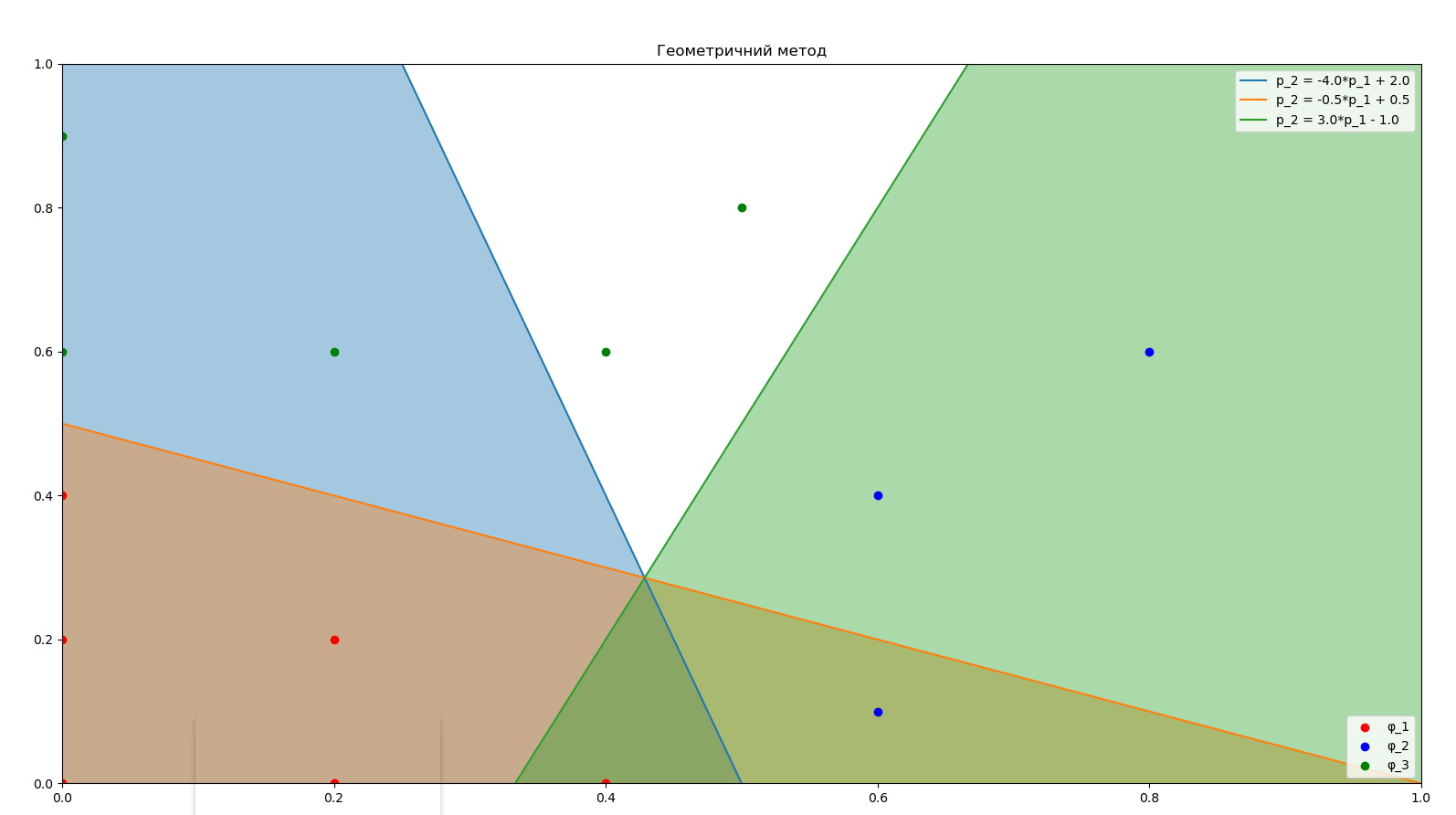
i += 1

eff.append(arr)

return p, eff

1. **Скриншоти роботи програми**

****

****

1. **Висновок**

Внаслідок роботи геометричного методу отримано три множини Баєса:

Внаслідок роботи методу варіації точки отримано три множини Баєса:

Помітно, що всі множини попарно подібні одна до одної, незалежно від методу