****

**Московский авиационный институт**

**(Национальный исследовательский университет)**

**Институт № 3**

**Кафедра 311**

**Интеллектуальные системы**

**Лабораторная работа № 4**

**«Метод потенциальных функций»**

**Выполнил студент  
Веденеев Максим Кириллович**

**Группа М3З-501-БК**

**Дата 06.10.2023 г.**

**Принял преподаватель  
Кос Оксана Игоревна**

Оглавление

[Цель лабораторной работы 3](#__RefHeading___1)

[Глава 1. Метод потенциальных функций 4](#__RefHeading___2)

[Глава 2. Реализация 5](#__RefHeading___3)

[Глава 3. Листинг программы 6](#__RefHeading___4)

[Итоги лабораторной работы 11](#__RefHeading___5)

[Список литературы 12](#__RefHeading___6)

# Цель лабораторной работы

Изучить:

1. Метод потенциальных функций.
2. Реализовать его.

# Глава 1. Метод потенциальных функций

Метод потенциальных функций - метрический классификатор, частный случай метода ближайших соседей. Позволяет с помощью простого алгоритма оценивать вес («важность») объектов обучающей выборки при решении задачи классификации.

Общая идея метода иллюстрируется на примере электростатического взаимодействия элементарных частиц. Известно, что потенциал («мера воздействия») электрического поля элементарной заряженной частицы в некоторой точке пространства пропорционален отношению заряда частицы (Q) к расстоянию до частицы (r):



Метод потенциальных функций реализует полную аналогию указанного выше примера. При классификации объект проверяется на близость к объектам из обучающей выборки. Считается, что объекты из обучающей выборки «заряжены» своим классом, а мера «важности» каждого из них при классификации зависит от его «заряда» и расстояния до классифицируемого объекта.

Метод потенциальных функций заключается в выборе в качестве  весовой функции следующего вида:

, где

*  - функция, убывающая с ростом аргумента. Константа a нужна чтобы избежать проблем с делением на ноль. Для простоты обычно полагают a=1.
*  - расстояние от объекта u до i-того ближайшего к u объекта — 
*  - параметр, задающий «ширину потенциала» объекта , 
*  - параметр, задающий «заряд», то есть степень «важности» объекта ,  при классификации

# Глава 2. Реализация

Обучим сеть распознавать следующие образы:

@-----@  
@---@@-  
@--@---  
@@@----  
@--@---  
@---@@-  
@-----@

-@@@@@-  
-@---@-  
-@---@-  
-@---@-  
-@---@-  
-@---@-  
-@---@-

@@@@@@@  
---@---  
---@---  
---@---  
---@---  
---@---  
---@---

На вход будем подавать искаженный образ буквы К:

@@--@--  
@---@@-  
@--@---  
@-@----  
@--@---  
@-@-@@-  
@@----@

Для входного изображения вычисляем потенциалы, а затем по наибольшему определяем результат.

Potentials:

[27027.027027027027, 1477.1048744460857, 2061.855670103093]

Result:

@-----@  
@---@@-  
@--@---  
@@@----  
@--@---  
@---@@-  
@-----@

# Глава 3. Листинг программы

#!/usr/bin/python

# -\*- coding: utf-8 -\*-

from \_\_future\_\_ import with\_statement

import numpy as np

import os

import string

from os.path import isdir, isfile

MAX\_SIZE = 512

SHAPE\_SIDE = 7

dictionary = {'-': -1, '@': 1}

back\_ictionary = {-1: '-',1: '@'}

SHAPES\_PATH = "known\_shapes"

INPUT\_PATH = "modified\_shapes"

def charfor(x):

return dictionary[x]

def printshape(obraz, size):

i = 0

out\_str = ""

for o in obraz:

out\_str += str(back\_ictionary[o])

i += 1

if i % size == 0:

print(out\_str)

out\_str = ""

#

# ��������� ������ �� ������ � ����������

# ������ �� �� ���������

#

def parse(dir):

shapes\_files = []

for filename in os.listdir(dir):

path = os.path.join(dir, filename)

if isfile(path):

shapes\_files.append(path)

shapes = []

for path in shapes\_files:

shape = parse\_shape(path)

shapes.append(shape)

return shapes

#

# ��������� ���� � ������� � �����������

# � ������

#

def parse\_shape(path):

with open(path) as f:

contents = f.read(MAX\_SIZE)

contents = contents.replace("\n", "")

contents = contents.replace("\r", "")

shape = []

for c in contents:

shape.append(dictionary[c])

if len(shape) != pow(SHAPE\_SIDE, 2):

raise Exception("Shape size must be %gx%g" % (SHAPE\_SIDE, SHAPE\_SIDE))

return shape

class PotentialMethod:

def \_\_init\_\_(self):

self.shapes = parse(SHAPES\_PATH)

self.input = parse(INPUT\_PATH)[0]

self.Ps = [0.0] \* len(self.shapes)

def get\_result(self):

print("Input:")

printshape(self.input, SHAPE\_SIDE)

# ����� ���������� ���������

max\_p = -1

idx\_max\_p = -1

for i, p in enumerate(self.Ps):

if p > max\_p:

max\_p = p

idx\_max\_p = i

return self.shapes[idx\_max\_p]

def potential(self):

for i, shape in enumerate(self.shapes):

r = self.compare(self.input, shape)

self.Ps[i] += 1\_000\_000 / (1 + r \*\* 2)

print("Potentials:")

print(self.Ps)

def compare(self, b1, b2):

count = 0

assert len(b1) == len(b2) # ������� ������������ ����� �����

b1\_pixels = self.to\_pixels(b1)

b2\_pixels = self.to\_pixels(b2)

for i in range(7):

for j in range(7):

if b1\_pixels[i][j] != b2\_pixels[i][j]:

count += 1

return count

def to\_pixels(self, shape):

n = int(np.sqrt(len(shape)))

result = np.zeros((n, n), dtype=int)

k = 0

for i in range(n):

line\_int = shape[k:k + n]

result[i] = line\_int

k += n

return result

if \_\_name\_\_ == "\_\_main\_\_":

potential\_method = PotentialMethod()

potential\_method.potential()

result = potential\_method.get\_result()

print("Result:")

printshape(result, SHAPE\_SIDE)

# Итоги лабораторной работы

Мы освоили метод потенциальных функций.

# Список литературы

1. [[interface.ru](http://www.interface.ru/home.asp?artId=20184)] – Распознавание образов методом потенциальных функций (аналогично)