­­МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

­­­



Лабораторна робота №2

з курсу «Методи та системи штучного інтелекту»

для студентів базового напрямку 6.08.04 "Комп’ютерні науки"

(заочна форма навчання)

Варіант 10

Виконав студент гр. КНз-3

Чалий Михайло

­­

Львів 2015

## Мета роботи

Вивчити принципи роботи максимінного алгоритму розпізнавання образів. Написати програму реалізації алгоритму з графічним інтерфейсом користувача.

## Теоретичні відомості

Алгоритм використовує евклідову відстань. Алгоритм, у принципі, аналогічній схемi евристичного алгоритму порогової величини, за виключенням тiєї обставини, що, в першу чергу, він виявляє найвіддаленіші кластери. Один з об’єктів (X1) довільним чином назначається центром першого кластера. Потім відшукується образ, розміщений вiд образа X1 найдалi, який призначається центром кластера Z2. На третьому кроцi алгоритму здiйснюється обчислення вiдстаней мiж всiма iншими образами вибiрки i центрами кластерiв Z1 i Z2. В кожнiй парi цих вибірок вибирається мiнiмальне. Пiсля цього видiляється максимальне з цих мiнiмальних вiдстаней. Якщо останнє складає значну частину вiдстанi мiж кластерами Z1 i Z2 (половина цiєї вiдстанi), вiдповiдний образ призначається центром кластера Z3. Iнакше - виконання алгоритму припиняється.

В загальному випадку описана процедура повторюється до тих пiр, поки на якомусь кроцi не буде отримане максимальне значення вiдстанi, для якої умова, що викликає видiлення кластера, не виконується.

При віднесенні інших образів до одного з отриманих кластерів використовується критерій, що передбачає введення цього образу в той кластер, центр якого для нього найближчий.

## Завдання

10. X1(-5,-3) X2(-4,-3) X3(-3,-4) X4(-2,-3) X5(4,-2) X6(5,-2) X7(5,-3) X8(6,-3) X9(7,-4) X10(-2,0) X11(-2,1) X12(-1,2) X13(0,2)

## Рішення

import sys, math, random

from point import \*

from pylab import \*

class MinMax(object):

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

self.\_points = [ Point(p[0], p[1]) for p in points ]

self.\_centers = []

def cluster(self):

self.\_centers = [self.\_points[0]]

max\_distances = []

while True:

other\_points = filter(lambda x: x not in self.\_centers, self.\_points)

def row\_min\_distance(other\_point):

min\_distance = 10000 # Should be max int

min\_point = None

for z\_point in self.\_centers:

item\_distance = other\_point.distanceTo(z\_point)

if item\_distance < min\_distance:

min\_distance = item\_distance

min\_point = other\_point

return (min\_point, min\_distance)

def col\_max\_distance(other\_points):

max\_distance = 0

max\_point = None

for other\_point in other\_points:

row\_distance = row\_min\_distance(other\_point)

item\_distance = row\_distance[1]

if item\_distance > max\_distance:

max\_distance = item\_distance

max\_point = row\_distance[0]

return (max\_point, max\_distance)

result = col\_max\_distance(other\_points)

max\_distance = result[1]

max\_point = result[0]

if len(max\_distances) == 0 or (max\_distance > min(max\_distances) / 2):

max\_distances.append(max\_distance)

self.\_centers.append(max\_point)

else:

return

s = MinMax([

(-5,-3),

(-4,-3),

(-3,-4),

(-2,-3),

(4,-2),

(5,-2),

(5,-3),

(6,-3),

(7,-4),

(-2,0),

(-2,1),

(-1,2),

(0,2) ])

s.dump\_points()

s.cluster()

s.dump\_cluster()

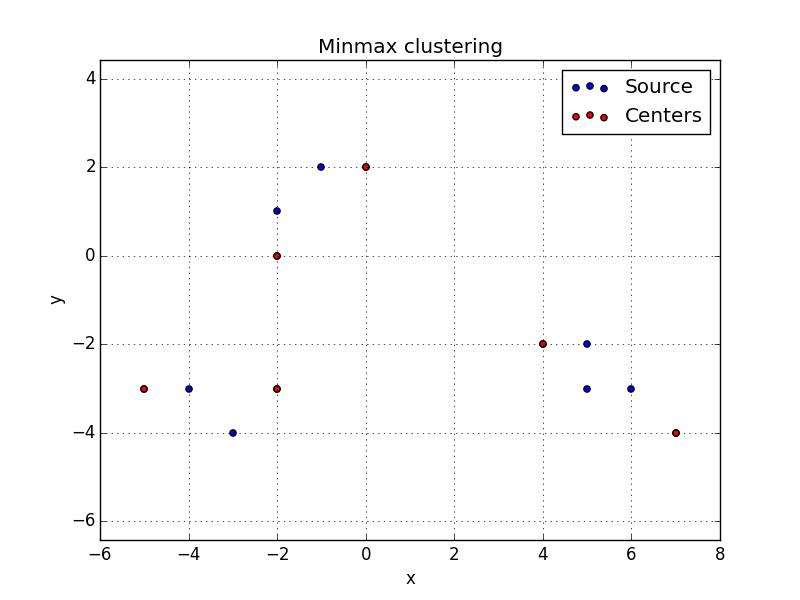
s.plot\_cluster('k1\_2\_14');

## Результат

[(-5; -3), (-4; -3), (-3; -4), (-2; -3), (4; -2), (5; -2), (5; -3), (6; -3), (7; -4), (-2; 0), (-2; 1), (-1; 2), (0; 2)]

Cluster centers with 0.5 thresold:

[(-5; -3), (7; -4), (0; 2), (4; -2), (-2; -3), (-2; 0)]



## Висновки

Ознайомився з технологією програмування із використанням пакета SDK та ECLIPSE, примітивними типами даних і операціями над ними.