МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Ижевский государственный технический университет имени

М. Т. Калашникова»

Институт «Информатика и вычислительная техника»

Кафедра «Программное обеспечение»

Отчет по дисциплине

«Математические основы искусственного интеллекта»

на тему «Генетический алгоритм»

Выполнил:

Студент группы Б18-191-2 Гумметов Р. А.

Принял:

К.т.н., доцент Коробейников А. В.

Ижевск 2020

ЦЕЛЬ РАБОТЫ

Изучение практического примера решения задач с использованием генетического алгоритма.

ПОСТАНОВКА ЗАДАЧИ

Научить алгоритм распознавать буквы, подающиеся ему на вход.

ТЕОРИЯ

Генетический алгоритм — это эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путём последовательного подбора, комбинирования и вариации искомых параметров с использованием механизмов, напоминающих биологическую эволюцию. Является разновидностью эволюционных вычислений. Отличительной особенностью генетического алгоритма является акцент на использование оператора «скрещивания», который производит операцию рекомбинации решений-кандидатов, роль которой аналогична роли скрещивания в живой природе.

РЕАЛИЗАЦИЯ

На рис. 1-5 представлены буквы для обучения:

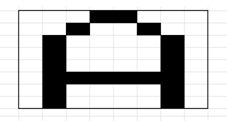


Рис. 1



Рис. 2



Рис. 3



Рис. 4

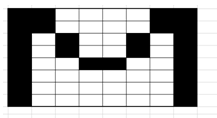


Рис. 5

На рис. 6-8 представлены буквы для проверки:

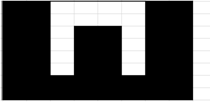


Рис. 6

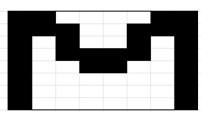


Рис. 7



Рис. 8

Код программы:

import random

class Perceptron:

def \_\_init\_\_(self, b, size):

self.b = b

self.w = []

for i in range(size):

self.w.append(random.randrange(-1 \* size, size))

defsumma(self, x):

s = 0

for i in range(len(x)):

s += self.w[i] \* x[i]

return s + self.b

defactivationF(self, x):

if self.summa(x) > 0:

return 1

return 0

def \_\_add\_\_(self, other):

v1 = random.randint(0, 63)

v2 = random.randint(0, 63)

newPerc = Perceptron(0, 64)

firstRandomValue = min(v1, v2)

secondRandomValue = max(v1, v2)

newPerc.w[:firstRandomValue] = self.w[:firstRandomValue]

newPerc.w[firstRandomValue:secondRandomValue] = other.w[firstRandomValue:secondRandomValue]

newPerc.w[secondRandomValue:] = self.w[secondRandomValue:]

return newPerc

class Learning:

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

self.perceptrons = [[Perceptron(0, 64) for i in range(100)] for j in range(letterCount)]

self.bestPerceptrons = [Perceptron(0, 64) for i in range(letterCount)]

self.bestPerceptrons2 = [Perceptron(0, 64) for i in range(letterCount)]

'''x - список значений букв, t - массив вот таких штук (0,0,0,0,1) правильных ответов'''

defexecute(self, x, t):

for i inrange(50):

# Для каждой группы персептронов(группа отвественная за 1 букву)

for letterGroupIndex in range(len(self.perceptrons)):

LetterResultsList = [[] for i in range(len(self.perceptrons[letterGroupIndex]))]

errorsCount = [0 for i in range(len(self.perceptrons[letterGroupIndex]))]

# Для всех персептронов, ответственных за распознование данной буквы

for perceptronNumber in range(len(self.perceptrons[letterGroupIndex])):

# Длякаждойбуквы

for letterValues in x:

res = self.perceptrons[letterGroupIndex][perceptronNumber].activationF(letterValues)

LetterResultsList[perceptronNumber].append(res)

# Началовыборки. Собираемвсеошибкиперсептроновбуквы

for percResultIndex in range(len(LetterResultsList)):

for valueIndex in range(len(LetterResultsList[percResultIndex])):

if LetterResultsList[percResultIndex][valueIndex] != t[letterGroupIndex][valueIndex]:

errorsCount[percResultIndex] += 1

for errori in range(len(errorsCount)):

if errorsCount[errori] == 0:

self.bestPerceptrons2=self.bestPerceptrons

self.bestPerceptrons[letterGroupIndex] = self.perceptrons[letterGroupIndex][errori]

#Получениенаилучшихобразцов

tempZip = zip(self.perceptrons[letterGroupIndex], errorsCount)

self.perceptrons[letterGroupIndex] = [perc for perc, \_ in sorted(tempZip, key=lambda tzip: tzip[1])]

newPercGroup = []

#Скрещивание

for sortedPercIndex in range(len(self.perceptrons[letterGroupIndex]) // 2 + 1):

firstPerc = self.perceptrons[letterGroupIndex][sortedPercIndex]

secondPerc = self.perceptrons[letterGroupIndex][sortedPercIndex+1]

childPerc = firstPerc + secondPerc

# Мутация

for g in range(len(self.perceptrons[letterGroupIndex][sortedPercIndex].w)):

if random.randint(1, 5) % 4 == 1:

childPerc.w[g] \*= random.randint(65, 140) / 100

newPercGroup.append(childPerc)

newPercGroup += self.perceptrons[letterGroupIndex][:50]

self.perceptrons[letterGroupIndex] = newPercGroup

for i in range(len(self.perceptrons)):

self.bestPerceptrons[i]=self.bestPerceptrons[i]+self.bestPerceptrons2[i]

defshowResult(self, letter, dic):

result = [per.activationF(letter) for per in self.bestPerceptrons]

print(dic.get(tuple(result)), result)

m = (1,1,0,0,0,0,1,1,

1,1,0,0,0,0,1,1,

1,0,1,0,0,1,0,1,

1,0,1,0,0,1,0,1,

1,0,0,1,1,0,0,1,

1,0,0,0,0,0,0,1,

1,0,0,0,0,0,0,1,

1,0,0,0,0,0,0,1)

a = (0,0,0,1,1,0,0,0,

0,0,1,0,0,1,0,0,

0,1,0,0,0,0,1,0,

0,1,0,0,0,0,1,0,

0,1,0,0,0,0,1,0,

0,1,1,1,1,1,1,0,

0,1,0,0,0,0,1,0,

0,1,0,0,0,0,1,0)

sh = (1,1,0,1,1,0,1,1,

1,1,0,1,1,0,1,1,

1,1,0,1,1,0,1,1,

1,1,0,1,1,0,1,1,

1,1,0,1,1,0,1,1,

1,1,0,1,1,0,1,1,

1,1,1,1,1,1,1,1,

1,1,1,1,1,1,1,1)

e = (1,1,1,1,1,1,1,1,

1,1,1,1,1,1,1,1,

1,1,0,0,0,0,0,0,

1,1,1,1,1,1,0,0,

1,1,1,1,1,1,0,0,

1,1,0,0,0,0,0,0,

1,1,1,1,1,1,1,1,

1,1,1,1,1,1,1,1)

v = (1,1,1,1,1,1,1,0,

1,1,0,0,0,0,0,1,

1,1,0,0,0,0,0,1,

1,1,1,1,1,1,1,0,

1,1,1,1,1,1,1,0,

1,1,0,0,0,0,0,1,

1,1,0,0,0,0,0,1,

1,1,1,1,1,1,1,1)

blursedSH = (1,1,0,0,0,0,1,1,

1,1,0,0,0,0,1,1,

1,1,0,1,1,0,1,1,

1,1,0,1,1,0,1,1,

1,1,0,1,1,0,1,1,

1,1,0,1,1,0,1,1,

1,1,1,1,1,1,1,1,

1,1,1,1,1,1,1,1)

blursedE = (1,1,1,1,1,1,1,1,

1,1,1,1,1,1,1,1,

1,1,0,0,0,0,0,0,

1,1,1,1,1,1,1,1,

1,1,1,1,1,1,1,1,

1,1,0,0,0,0,0,0,

1,1,1,1,1,1,1,1,

1,1,1,1,1,1,1,1)

blursedM = (1,1,0,0,0,0,1,1,

1,1,1,0,0,1,1,1,

1,0,1,0,0,1,0,1,

1,0,1,1,1,1,0,1,

1,0,0,1,1,0,0,1,

1,0,0,0,0,0,0,1,

1,0,0,0,0,0,0,1,

1,0,0,0,0,0,0,1)

dic = {(1,0,0,0,0): 'М', (0,1,0,0,0): 'А', (0,0,1,0,0): 'Ш', (0,0,0,1,0): 'Е', (0,0,0,0,1): 'В'}

tuples = [(1,0,0,0,0), (0,1,0,0,0), (0,0,1,0,0), (0,0,0,1,0), (0,0,0,0,1)]

learning = Learning(5)

learning.execute([m, a, sh, e, v], tuples)

learning.showResult(m, dic)

learning.showResult(a, dic)

learning.showResult(sh, dic)

learning.showResult(e, dic)

learning.showResult(v, dic)

learning.showResult(blursedSH, dic)

learning.showResult(blursedE, dic)

learning.showResult(blursedM, dic)

Самый частый вариант:

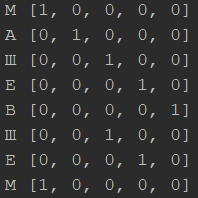


Рис. 9

Иногда персептроны не различают одну из букв:

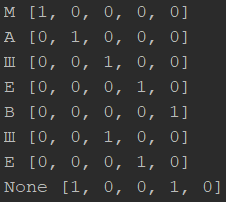


Рис. 10

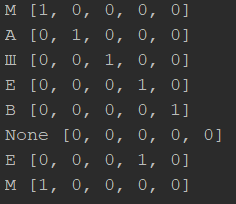


Рис. 11

Генетический алгоритм работает на достаточно высоком уровне, что позволяет распознать ему 2, а иногда и 3 буквы.