CPE 261456 Introduction to Computational Intelligence

Computer Assignment 4 (Swarm Intelligence)

จัดทำโดย

นาย ณัชพล เพทายเทียมทอง

รหัสนักศึกษา 580610633

เสนอ

รศ.ดร. ศันสนีย์ เอื้อพันธ์วิริยะกุล

ภาควิชา วิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเชียงใหม่ ปีการศึกษา 2561

Swarm Intelligence

ขั้นตอนการทำงานของโปรแกรม

ทำการทดลองกับ Dataset Air Quality Data Set จาก UCI Machine learning Repository

โดยจะใช้ Attribute ที่ 3, 6, 8, 10, 11, 12, 13, 14 เป็น features และ Attribute ที่ 5 เป็น Desire Output ซึ่งตอนแรกนั้นต้องทำการนำ Desire output ของ 5 วันถัดไปมาแทนที่ Desire output ดั้งเดิม ในกรณีที่ต้องทำการ predict 5 วันถัดไป แต่ถ้าต้องทำการ predict 10 วันก็ต้องนำ desire output ของ 10 วันถัดไปก็ต้องนำ desire out มาแทนที่ ถัดมาจะทำการตัด missing value ที่เป็นค่า -200 จะถัดแถวที่มีค่า -200 ทิ้งไปเลย จากนั้นก็มาตัด Attribute ที่ไม่ได้ใช้ออก ซึ่งขั้นตอนนี้จะทำใน excel

หลังจากได้ข้อมูลมาแล้วจะทำการ normalize ข้อมูลโดยจะทำการหา min และ max ของแต่ละ column จากนั้นก็นำค่าในแต่ละ column – min / max – min

หลังจากนั้นจะทำการ initial individual โดย individual พวกนี้จะทำหน้าที่เป็น weight ของ Neural Network โดยในการ initial individual นั้นจะกำหนดชุดของ individual และสร้างตามจำนวนโครงสร้างของ multilayer perceptron โดยจะ initial individual ให้มีค่าอยู่ระหว่าง -1 ถึง 1

จากนั้นก็กำหนดค่า pbest และ velocity ให้กับ individual นั้น ๆ โดยการทดลอง swarm intelligence นี้จะใช้อัลกอริทึม individual best ซึ่งจะทำการหาค่าประสิทธิภาพที่คำนวณได้เทียบกับ pbest ของตัวเองเท่านั้น ซึ่งถ้าค่าประสิทธิภาพใหม่ดีกว่า pbest ก็จะอัพเดตค่าและตำแหน่งของตัวเอง จากนั้นก็กลับค่า velocity ซึ่งค่า p ของ velocity นั้นจะสุ่มให้มีค่าอยู่ในขอบเขตของ [0.1,3] โดยโปรแกรมนี้จะทำงานตามจำนวนรอบที่กำหนดไว้ถึงจะหยุด

การทดลอง

การ predict 5 วันล่วงหน้า

การทดลองที่ 1 กำหนดให้โครงสร้างของ Neural Network นั้นเป็น 8-3-1 และจำนวน individual เป็น 100 และ จำนวนรอบเป็น 100

----------Round 1----------

[0.17743038491978866]

----------Round 2----------

[0.4088688338897424]

----------Round 3----------

[0.5090990607380081]

----------Round 4----------

[0.1658508614653804]

----------Round 5----------

[0.20976804183827158]

----------Round 6----------

[0.2637477970950559]

----------Round 7----------

[0.9219022916385956]

----------Round 8----------

[0.5083369289816538]

----------Round 9----------

[0.13300604616582687]

----------Round 10----------

[0.4814401959559954]

Average MAE: 0.37794504426883185

การทดลองที่ 2 กำหนดให้โครงสร้างของ Neural Network นั้นเป็น 8-5-1 และ จำนวนรอบเป็น 100

----------Round 1----------

[0.33643488089089835]

----------Round 2----------

[0.18697714311940244]

----------Round 3----------

[0.738321733759234]

----------Round 4----------

[0.7797763238390102]

----------Round 5----------

[0.6913990426499714]

----------Round 6----------

[0.12350109150567146]

----------Round 7----------

[0.6460755793449647]

----------Round 8----------

[0.5144606715509502]

----------Round 9----------

[0.4418492373017821]

----------Round 10----------

[0.11763334642264225]

Average MAE: 0.4576429050384528

การทดลองที่ 3 กำหนดให้โครงสร้างของ Neural Network นั้นเป็น 8-5-1 และ จำนวนรอบเป็น 10

----------Round 1----------

[0.34245202609343095]

----------Round 2----------

[0.3645108817868779]

----------Round 3----------

[0.18349415501504124]

----------Round 4----------

[0.6097478377659891]

----------Round 5----------

[0.4301064379346776]

----------Round 6----------

[0.2416678148927275]

----------Round 7----------

[0.3450930976903122]

----------Round 8----------

[0.5332231284038191]

----------Round 9----------

[0.13259862401016315]

----------Round 10----------

[0.4037097205129659]

Average MAE: 0.35866037241060045

การ predict 10 วันล่วงหน้า

การทดลองที่ 1 กำหนดให้โครงสร้างของ Neural Network นั้นเป็น 8-3-1 และ จำนวนรอบเป็น 10

----------Round 1----------

[0.08135090814173282]

----------Round 2----------

[0.12680247207628215]

----------Round 3----------

[0.1430762471403716]

----------Round 4----------

[0.16011325509470342]

----------Round 5----------

[0.15074358923245465]

----------Round 6----------

[0.32176517063246807]

----------Round 7----------

[0.21185513941500375]

----------Round 8----------

[0.1505017939594424]

----------Round 9----------

[0.6080135796364424]

----------Round 10----------

[0.289488614471942]

Average MAE: 0.22437107698008435

การทดลองที่ 2 กำหนดให้โครงสร้างของ Neural Network นั้นเป็น 8-5-1 และ จำนวนรอบเป็น 10

----------Round 1----------

[0.43599351152151994]

----------Round 2----------

[0.7500770303232621]

----------Round 3----------

[0.13672895893394998]

----------Round 4----------

[0.14291644771162307]

----------Round 5----------

[0.47075385457152075]

----------Round 6----------

[0.3372339807388399]

----------Round 7----------

[0.20935522394689052]

----------Round 8----------

[0.09977511667839474]

----------Round 9----------

[0.4503862768735398]

----------Round 10----------

[0.22855343581711637]

Average MAE: 0.32617738371166566

การทดลองที่ 3 กำหนดให้โครงสร้างของ Neural Network นั้นเป็น 8-10-1 และ จำนวนรอบเป็น 10

----------Round 1----------

[0.3412047181326345]

----------Round 2----------

[0.38134528957199115]

----------Round 3----------

[1.0780264361787755]

----------Round 4----------

[0.8695039393003953]

----------Round 5----------

[0.7938814359971321]

----------Round 6----------

[0.3286865207211991]

----------Round 7----------

[0.4839761033182562]

----------Round 8----------

[0.13715789778268989]

----------Round 9----------

[0.4547495568957185]

----------Round 10----------

[0.26145566342873916]

Average MAE: 0.5129987561327531

วิเคราะห์ผลการทดลอง

จากการทำ predict 5 วันถัดมานั้นจะเห็นได้ว่า การทดลองที่ 1 และ 2 นั้นให้ MAE มากกว่าการทดลองที่ 3 เนื่องจาก over train เพราะว่าจำนวนรอบในการทำงานของโปรแกรมนั้น การทดลองที่ 1 และ 2 มากกว่าการทดลองที่ 3 ซึ่งหลังจากนั้นก็ทำการทดลองกับ การทำ predict 10 วันถัดมานั้นจะกำหนดให้ จำนวนรอบในการทำงานเป็น 10 เท่ากันทุกการทดลองแต่ถึงอย่างนั้นการทดลองที่ 2 และ 3 ของการำ predict 10 วันมานั้นมีค่า MAE มากกว่า การทดลองที่ 1 ทำให้เห็นได้ว่าโครงสร้างของ Neural Network ที่มีมากกว่าอาจจะไม่ได้ให้ผลลัพธ์ที่ดีเสมอไปเนื่องจากเกิด over train

ภาคผนวก

Code ของส่วน main.py

```
import pandas as pd
import copy
import random
import xlrd
import numpy as np


class MLPPSO:
    def __init__(self, architect, individuals):
        self.architect = architect
        self.individuals = individuals
        self.initWeight = []

        for i in range(1, len(architect) - 1):
            perceptron_curr = architect[i]
            perceptron_prev = architect[i - 1]

        perceptron_currO = architect[i + 1]
        perceptron_prevO = architect[i]
        for ind in self.individuals:
            weight_prev = []
            weight_curr = []
            for c in range(perceptron_curr):
                weight_prev.append([])
                for p in range(perceptron_prev):
                    weight_prev[-1].append(ind[p + (c * perceptron_prev)])
            for p in range(perceptron_currO):
                weight_curr.append([])
                for c in range(perceptron_prevO):
                    weight_curr[-1].append(ind[(perceptron_prev *
perceptron_curr) + (p + (p * perceptron_curr))])

        self.initWeight_Cur = weight_curr
        self.initWeight_Prev = weight_prev

    def train(self, X, Y):
        errors = []
        self.weight_curr = copy.deepcopy(self.initWeight_Cur)
        self.weight_prev = copy.deepcopy(self.initWeight_Prev)
        error = []
        for i in range(int(X.shape[0])):
            Alltest = [X[i]]
            yh = []
            y = []
            o = []
            out = []
```

```
            for j in range(len(self.weight_prev)):
                y.append(np.dot(Alltest, self.weight_prev[j]))
                activate_y = np.tanh(y)
                yh.append(activate_y)

            for k in range(len(self.weight_curr)):
                o = (np.asarray(yh) * np.asarray(self.weight_curr[k]))
                activate_o = np.tanh(sum(o[-1]))
                out.append(activate_o)

            error.append(out)
        error = np.array(error).reshape(len(error))
        errors.append(findMAE(error, Y))
        return errors


def findMAE(error, Y):
    return sum(abs(Y - error)) / len(Y)


def main():
    fitavg = 0.0
    avgerror = 0.0
    num_individual = 100
    # data_set = pd.read_excel("AQPredict5hr.xlsx").values
    data_set = pd.read_excel("AQPredict10hr.xlsx").values
    trainX = data_set[:, :-1].tolist()
    trainY = data_set[:, 8].tolist()

    p = random.uniform(0.1, 3)

    findMinX = np.min(trainX, 0)
    findMaxX = np.max(trainX, 0)
    findMinY = np.min(trainY, 0)
    findMaxY = np.max(trainY, 0)
    normX = np.divide(np.subtract(trainX, findMinX), np.subtract(findMaxX,
findMinX)).tolist()
    normY = np.divide(np.subtract(trainY, findMinY), np.subtract(findMaxY,
findMinY)).tolist()

    for c in range(0, 10):
        individuals = []
        pbest = []
        architect = [8, 10, 1]
        print("--------------Round {}-------------".format(c + 1))
        part = int(round(len(normX) / 10.0, 0))
        last = (c * part + part) - 1

        if c == 9:
            last = len(normX) - 1

        tmpX = []
```

```
        tmpY = []
        strainX = copy.deepcopy(normX)
        strainY = copy.deepcopy(normY)
        for i in range(last, c * part - 1, -1):
            tmpX.append(strainX[i])
            tmpY.append(strainY[i])
            strainX.pop(i)
            strainY.pop(i)
        rTestX = np.array(tmpX)
        rTestY = np.array(tmpY)
        rTrainX = np.array(strainX)
        rTrainY = np.array(strainY)

        for i in range(num_individual):
            individuals.append([])
            for j in range(architect[1] + 1):
                for k in range(architect[0]):
                    individuals[-1].append(random.uniform(-1, 1))

            pbest.append([num_individual, individuals[i]])
        individuals = np.asarray(individuals)
        v = np.asarray(individuals)
        mlp = MLPPSO(architect, individuals)
        for iter in range(10):
            fitness = mlp.train(rTrainX, rTrainY)
            for i in range(len(fitness)):
                if fitness[i] < pbest[i][0]:
                    pbest[i] = [fitness[i].copy(), individuals[i].copy()]
            for j in range(1, len(v)):
                v[j] = v[j - 1] + p * (pbest[j][1] - individuals[j])
                individuals[i] += v[j]

        fitness = mlp.train(rTestX, rTestY)
        print(fitness)
        fitavg += fitness[0]
    print("Average MAE: ", fitavg / 10)


if __name__ == '__main__':
    main()
```