**Computer Intelligence**

**Kunnapat Thippayapalaphonkul (590612113)**

**1 December 2019**

**รายงานผลการทดลองการบ้าน Swarm Intelligence**

ในการบ้านรายงานนี้ นั้นได้ทำการเรียกใช้ Libraries 1). numpy 2). copy 3). random 4). Matplotlib 5). pandas โดยที่ numpy นั้นเอาไว้ใช้ในการคำนวณต่างๆ เช่น dot product, mean, random matrix เป็นต้น copy ใช้ทำการ copy List, random ใช้ในการเลือก Chromosome ในขั้นตอนต่างๆ matplotlib เพื่อ Plot Graph และ pandas ใช้เพื่ออ่านไฟล์ Data set เนื่องจากเป็น excel

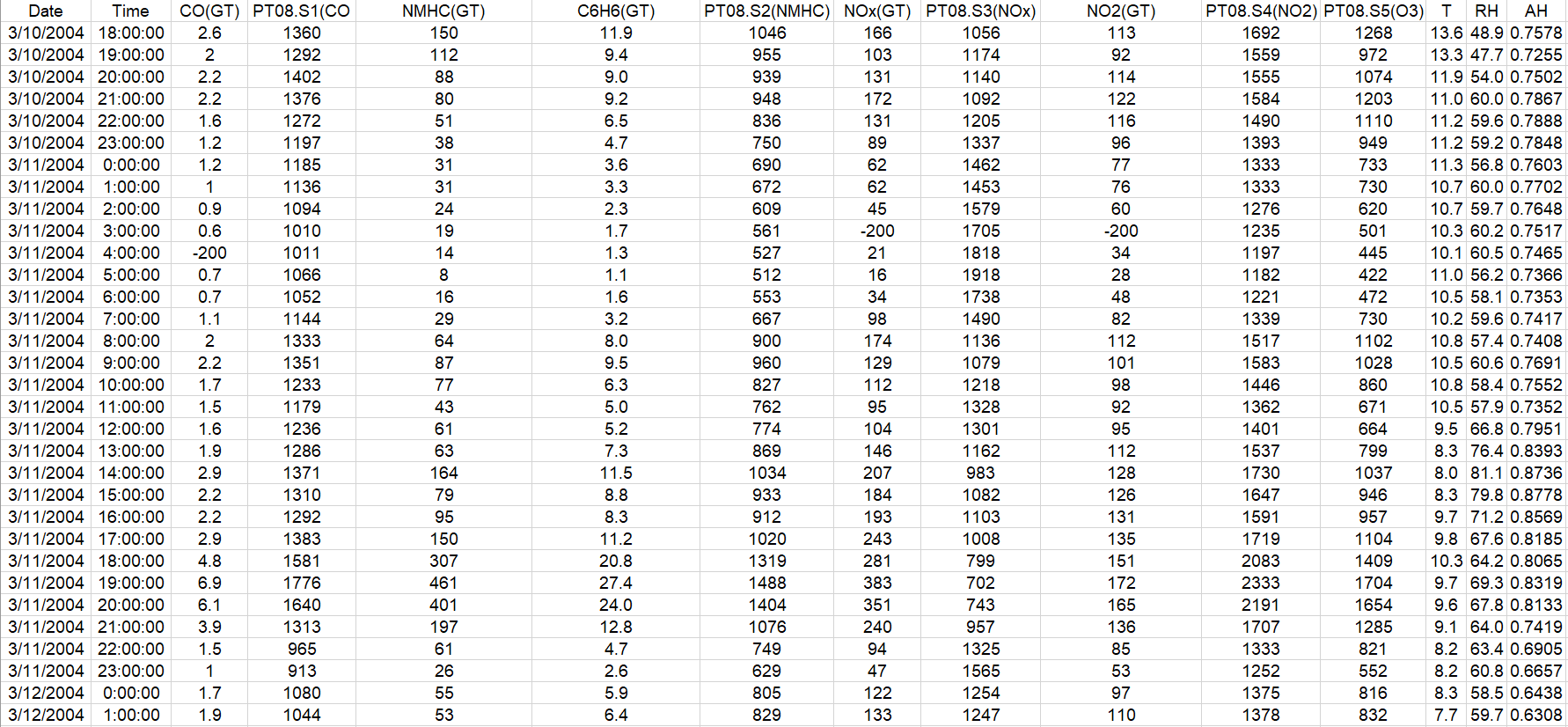
รายงานผลการทดลองทำกับ Data set [Air Quality Data set](https://archive.ics.uci.edu/ml/datasets/air+quality) โดยที่ Data set นี้มี 15 Attributes โดยที่มี 9 Features, Attributes ที่ 5 จะเปิดค่าเฉลย True hourly averaged Benzone concentration in microg/m^3 (reference analyzer) โดยที่ให้ทำการ Regression โดยที่การบ้านนี้นั้นได้ทำการใช้ Neural Network ในการคำนวณหา Error และจากขั้นตอน Backpropagation เปลี่ยนไปใช้ Swarm Intelligence แทนในการบ้านนี้ โดยมีขั้นตอนดังนี้

* Preprocessing Data
* Cross Validation
* Neural Network
* Fitness
* Compare Pbest
* Compare Gbest
* Update Velocity
* Update Position

**ขั้นตอน**

**Preprocessing Data**

ในขั้นตอนนี้เช็ค Data ว่าสมบูรณ์รึเปล่าในแต่ละ Attributes จากนั้นใน Data จะมี 15 Attributes ใน Data set ชุดนี้โดยจะมี Data โดยที่แต่ละ Attributes โดยเริ่มนับจาก Column แรกเริ่มเป็น 0 และที่ใช้ในการ Train ก็คือ Attributes [3, 6, 8, 10, 11, 12, 13, 14] และ Attributes ที่เป็นคำตอบในการ Regression คือ Attribute ที่ 5 ซึ่งตัวอย่าง Data set จะเป็นดัง รูปที่ 1 จะได้เห็นว่า Column 0, 1, 2, 4, 7, 9 นั้นจะไม่ใช้ในการบ้านครั้งนี้ แต่จะใช้ Column ที่เหลือทั้งหมด



รูปที่ 1 ตัวอย่าง Data set

จาก รูปที่ 1 ใน Column ที่ 5 หรือ Column ที่ชื่อ C6H6 นั้นจะเป็นคำตอบของ Data set ชุดนี้ ซึ่งข้อมูลนั้นเป็นเป็นตัวเลขทั้งหมดเลย ดังนั้นจึงทำ Regression เพื่อที่จะได้ Predict Benzone Concentration และข้อมูลในทุก Column ที่สนใจนั้นเป็นตัวเลขทั้งหมดแต่ในแต่ละ Column นั้นมีระดับที่ต่างกัน จึงทำการ Normalize ในแต่ละ Column เพื่อที่จะให้ Data นั้นอยู่ในระดับเดียวกัน [0, 1] ในแต่ละ Column จะทำการหา Max และ Min เพื่อนำมาคิด x ในแต่ละ Column ดัง [ 1 ]

[ 1 ]

**Cross Validation**

ได้ทำการแบ่ง Data set เป็น 10 Fold โดยที่แต่ละ Fold นั้นแบ่งเป็น Training set 90 % และ Testing set 10 % ดัง รูปที่ 2 ทำเป็นจำนวน 10 Fold โดยที่ Data set ชุดนี้มีขนาดจำนวน 9,357 Sample แบ่งเป็นให้ Testing set ประมาณ 10 % เท่ากับ 935 Sample ที่เหลือแบ่งใส่ Training set จำนวน 8,422 Sample โดยที่ Training set จะไม่มีข้อมูลของ Testing set เลยเพื่อเช็คความถูกต้องของ Neural Network ที่ได้ทำการ Train ว่าไม่ได้ Overfitting กับ Training set จนเกินไป

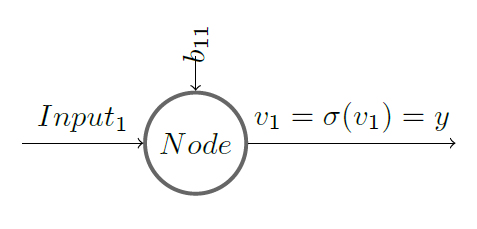
Training set 90 %

Testing set 10 %

รูปที่ 2 ตัวอย่างการทำ Cross Validation

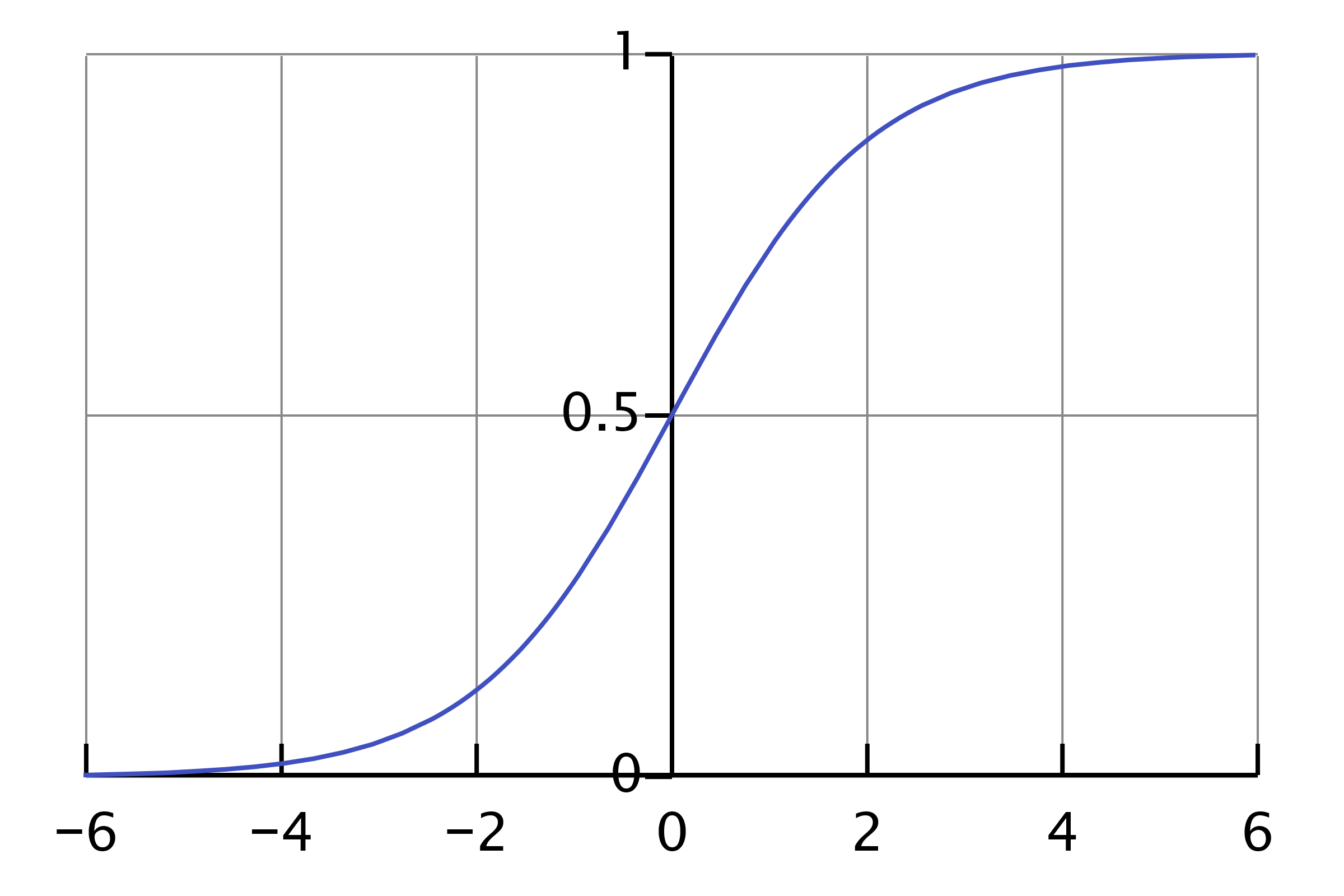
**Neural Network**

ในขั้นตอนนี้จะทำการสร้าง Neural Network โดยที่จะกำหนดค่าเริ่มต่างๆในกับ Network เพื่อที่จะ Train Network ให้ได้ผลตามที่เราต้องการ Neural Network จะมีการคำนวณดัง รูปที่ 3



รูปที่ 3 Node ของ Neural Network

หลังจากคำนวณแต่ละ Node เสร็จ จะทำการนำค่าที่คำนวณได้ไปทำการผ่าน Activation Function โดยที่ Activation Function ที่ผมได้เลือกใช้ในการบ้านนี้คือ Sigmoid Function โดยผลที่นำไปผ่าน Sigmoid Function จะได้ผลดัง รูปที่ 4



รูปที่ 4 ค่า Sigmoid Function

ผลลัพธ์ที่ได้จาก Sigmoid Function นั้นจะอยู่ระหว่าง [0, 1] ซึ่งทำการคำนวณแต่ละ Node ไปเรื่อย ๆ จนถึง Node สุดท้ายเพื่อที่จะหาค่า Error ของ Neural Network โดยการกำหนดค่าเริ่มต้นต่างๆให้กับ Neural Network มีดังนี้

* Input, Output ของ Neural Network สำหรับ Training set
* กำหนดจำนวน Node ในแต่ละ Layer
* จำนวน Population
* จำนวน Group\_population

**Fitness**

ขั้นตอนนี้หลังการคำนวณ Loss หรือ Error จาก Network โดยปกติเราจะอยากได้ค่า Error ของ Network ให้มีค่าน้อยที่สุดเท่าที่จะทำได้ โดยที่ในการบ้านนี้จะใช้การคำนวณ Error ทั้งหมดคือ Mean Absolute Error (MAE) และค่า Fitness จะเป็นค่าผลลัพธ์ของแต่ละ Group\_population ก็จะใช้ค่า MAE เช่นเดียวกับ Error เลยดัง [ 2 ]

[ 2 ]

**Compare Pbest**

ในขั้นตอนนี้นั้นจะทำการเปรียบเทียบเพื่อหาค่า Fitness ที่ด๊ที่สุดในตัวเองที่สามารถทำได้ในแต่ละ iteration ซึ่งถ้าเจอค่าที่ดีที่สุดแล้วจะทำการอัพเดทค่าที่ดีที่สุดและตำแหน่งที่ดีที่สุด ให้กับตัว Pbest

**Compare Gbest**

ในขั้นตอนนี้นั้นจะเหมือนกับขั้นตอน Pbest เลยก็คือทำการเทียบค่า Fitness ของทุกคนเพื่อหาค่าของ Fitness ที่ดีที่สุดในสังคมและในทุก iteration เพื่อที่จะได้ค่า Fitness ที่ดีที่สุดและตำแหน่งที่ดีที่สุดด้วย

**Update Velocity**

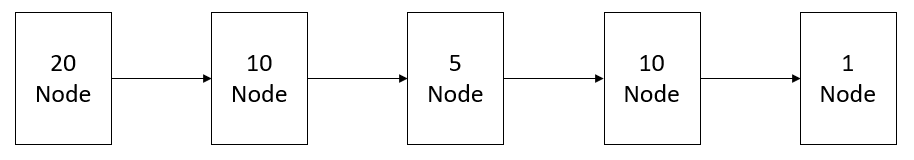
ในขั้นตอนนี้นั้นจะทำการอัพเดทค่าความเร็วของแต่ละกลุ่มและภายในกลุ่มคือแต่ละคนด้วย

**ผลการทดลอง**

การทดลองครั้งนี้นั้นจะทดลองโดยมี 20 Chromosome และมี 35 Generation

**การทดลองครั้งที่ 1**

ซึ่งจะทำการทดลองโดยสร้าง Neural Network ที่มี Layer และ Node ดัง รูปที่ 5



รูปที่ 5 จำนวน Layer และ Node ในแต่ละ Layer

ตาราง 1 ค่า Fitness ของแต่ละ Fold ของการทดลองครั้งที่ 1

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

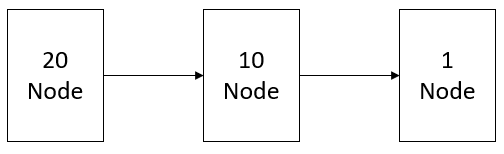
จาก ตาราง 1 จะเห็นได้ว่าแต่ละ Fold จะได้ค่า Fitness ออกมาหลังจากการ Train ด้วย Genetic Algorithms แล้วค่าอาจจะยังไม่สูงมากและผลของการ Test แบ่ง Fold ดัง *ตาราง 2* และ Fold ที่มีค่า Fitness มากที่สุดคือ Fold ที่ 8

ตาราง 2 แสดงค่า Fitness ของ Test set แต่ละ Fold ของการทดลองครั้งที่ 1

|  |  |
| --- | --- |
| **Fold** | **Fitness** |
| Fold 1 | 8.875551589865664 |
| Fold 2 | 9.217609779417426 |
| Fold 3 | 11.289165075825347 |
| Fold 4 | 9.049694714214484 |
| Fold 5 | 10.265479046571183 |
| Fold 6 | 12.301882481038685 |
| Fold 7 | 13.31800491142547 |
| Fold 8 | 14.391515902334096 |
| Fold 9 | 12.754501890627129 |
| Fold 10 | 11.870991459054318 |

**การทดลองครั้งที่ 2**

การทดลองครั้งที่ 2 จะทำการลดจำนวน Layer ลงมาดัง รูปที่ 6



รูปที่ 6 จำนวน Layer และ Node ในแต่ละ Layer

ตาราง 3 ค่า Fitness ของแต่ละ Fold ของการทดลองครั้งที่ 2

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

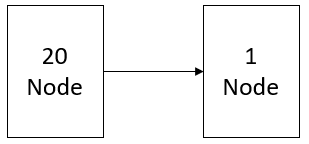
จาก ตาราง 3 จะเห็นได้ว่าหลายๆ Fold นั้นมีแนวโน้มที่จะมีค่า Fitness เพิ่มขึ้นใน Generation หลังๆ และค่า Fitness จาก Test ผลของการ Test แบ่ง Fold ดัง ตาราง 4 จะเห็นได้ว่าค่า Fitness นั้นเพิ่มขึ้นจากการทดลองครั้งที่ 1 และ Fold ที่มี ค่า Fitness มากที่สุดคือ Fold ที่ 8

ตาราง 4 แสดงค่า Fitness ของ Test set แต่ละ Fold ของการทดลองครั้งที่ 2

|  |  |
| --- | --- |
| **Fold** | **Fitness** |
| Fold 1 | 12.350715175399525 |
| Fold 2 | 10.4216173231093 |
| Fold 3 | 11.272331913555087 |
| Fold 4 | 9.237426522058652 |
| Fold 5 | 11.073170562399588 |
| Fold 6 | 15.49166242488285 |
| Fold 7 | 10.560876767961634 |
| Fold 8 | 17.4445331597968 |
| Fold 9 | 15.291112823649474 |
| Fold 10 | 15.059421505896173 |

**การทดลองครั้งที่ 3**

การทดลองครั้งที่ 3 จะทำการลดจำนวน Layer ลงมาดัง รูปที่ 7



รูปที่ 7 จำนวน Layer และ Node ในแต่ละ Layer

ตาราง 5 ค่า Fitness ของแต่ละ Fold ของการทดลองครั้งที่ 3

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

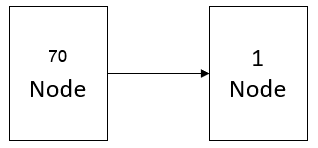
จาก ตาราง 5 จะเห็นจะว่าค่า Fitness ของแต่ละ Fold นั้นเพิ่มขึ้นบาง Fold มีแนวโน้มว่าค่า Fitness จะเพิ่มขึ้นถ้าเพิ่ม Generation เข้าไปและผลของการ Test แบ่ง Fold ดัง ตาราง 6 จะเห็นได้ว่าค่า Fitness นั้นเพิ่มขึ้นจากการทดลองครั้งที่ 2 และ Fold ที่มี ค่า Fitness มากที่สุดคือ Fold ที่ 10

ตาราง 6 แสดงค่า Fitness ของ Test set แต่ละ Fold ของการทดลองครั้งที่ 3

|  |  |
| --- | --- |
| **Fold** | **Fitness** |
| Fold 1 | 10.768842655422105 |
| Fold 2 | 11.325696362554527 |
| Fold 3 | 13.581821190911247 |
| Fold 4 | 9.673482611004273 |
| Fold 5 | 10.728741427814445 |
| Fold 6 | 19.051465188987898 |
| Fold 7 | 16.958914791306785 |
| Fold 8 | 17.995275736363272 |
| Fold 9 | 12.342098779449191 |
| Fold 10 | 20.313412354667513 |

**การทดลองครั้งที่ 4**

จากการทดลองครั้งที่ 3 จะทำการลดจำนวน Layer ลงมาดัง รูปที่ 8



รูปที่ 8 จำนวน Layer และ Node ในแต่ละ Layer

ตาราง 7 ค่า Fitness ของแต่ละ Fold ของการทดลองครั้งที่ 4

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

จาก ตาราง 7 จะเห็นจะว่าค่า Fitness ของแต่ละ Fold นั้นเพิ่มขึ้นบาง Fold มีแนวโน้มว่าค่า Fitness จะเพิ่มขึ้นถ้าเพิ่ม Generation เข้าไป แต่ผลของการ Test แบ่ง Fold ดัง ตาราง 8 จะเห็นได้ว่าค่า Fitness นั้นไม่ได้เพิ่มขึ้นจากการทดลองครั้งที่ 3 และ Fold ที่มี ค่า Fitness มากที่สุดคือ Fold ที่ 10 โดยรวมแล้วการทดลองครั้งที่ 3 ได้ค่า Fitness ดีกว่า

ตาราง 8 แสดงค่า Fitness ของ Test set แต่ละ Fold ของการทดลองครั้งที่ 4

|  |  |
| --- | --- |
| **Fold** | **Fitness** |
| Fold 1 | 7.9504293760685885 |
| Fold 2 | 12.265499747775097 |
| Fold 3 | 10.676106383737464 |
| Fold 4 | 8.163241400163932 |
| Fold 5 | 10.912484201019566 |
| Fold 6 | 18.502282011175303 |
| Fold 7 | 13.409408648323211 |
| Fold 8 | 16.82669916071412 |
| Fold 9 | 11.699634616430426 |
| Fold 10 | 20.281525437178466 |

**สรุปผลการทดลอง**

จากผลการทดลองทั้ง 4 ครั้งนั้นจะเห็นได้ว่าจากกราฟของแต่ละ Fold ทั้ง 10 Fold นั้นพอลดจำนวน Hidden Layer ลงนั้นทำให้ Generation หลัง ๆ หลาย ๆ Fold นั้นจะได้ค่า Fitness ที่เยอะขึ้น ๆ เรื่อย ๆ กับ Training set เนื่องจากมาได้ทำการที่ Chromosome ที่มีค่า Fitness มากสุดนั้นได้ ไปต่อใน Generation ต่อไปเรื่อย ๆ เกิดการ Crossover ระหว่าง Chromosome ทำให้ Generation ถัด ๆ ไปได้ค่า Fitness ที่ดีขึ้นเรื่อยๆกับ Network และหลังจากการทำ Crossover ก็จะมี Mutate ต่อในขั้นต่อถัดไปซึ่งการ Mutate นี้อาจจะให้ Chromosome ที่มีค่า Fitness ใน Generation ปัจจุบันไม่ค่อยดีนั้นได้ทำการ Mutate ให้มีค่า Fitness ที่ดีขึ้นได้ทำให้ผลดีขึ้น แต่ก็อาจจะทำให้ Chromosome ที่มีค่า Fitness ใน Generation ปัจจุบันที่ดีอยู่แล้วเกิดการ Mutate ทำให้ค่า Fitness นั้นต่ำลงก็เป็นไปได้เหมือนกัน ผลของการทดลองทั้ง 4 ครั้งกับ Test set Fold ที่มีค่า Fitness ดีที่สุดมีดัง ตาราง 9 จะเห็นได้ว่าลดจำนวน Hidden Layer ลงทำให้ได้ค่า Fitness ที่มากขึ้นเนื่องจากอาจจะมี Hidden Layer มากเกินไปทำให้อาจจะ Overfitting มากเกินไปหรืออาจจะเพราะ Train ด้วย Generation ที่น้อยจนเกินไป

ตาราง 9 สรุปของการทดลอง Fold ไหนที่มีค่า Fitness มากที่สุด

|  |  |  |
| --- | --- | --- |
| **การทดลอง** | **Fold** | **Fitness** |
| 1 | 8 | 14.391515902334096 |
| 2 | 8 | 17.4445331597968 |
| 3 | 10 | 20.313412354667513 |
| 4 | 10 | 20.281525437178466 |

**ภาคผนวก**