Computer Intelligence

Kunnapat Thippayapalaphunkul 590612113

3 September 2019

รายงานผลการทดลองการบ้าน Neural Network

รายละเอียดการทดลองใช้ข้อมูล Flood data set ในการทดลองที่ 1 และได้ใช้ข้อมูล Cross.pat ในการทดลองที่ 2

- ในการทดลองที่ 1 มี 8 Input Features โดยที่มี Input จาก Station 1, Station 2 จะมี Data เป็นความสูงของระดับน้ำในเวลาที่ t-3, t-2, t-1, t-0 จากทั้งสอง Station ให้ Predict หาระดับน้ำที่สะพานนวรัตน์ในเวลา t+7 หา 1 Class Output
- ในการทดลองที่ 2 มี 2 Input Features 2 Class Output ให้หาว่าโปรแกรมจะตอบถูก Class รึเปล่าโดยทำการแสดงเป็น Confusion Matrix
- ในการทดลองแต่ละข้อแต่ละครั้งจะทำการ Random Weight, Bias ให้มีค่าไม่เท่ากัน- และกำหนด จำนวน Node ในแต่ละและจำนวน Layer, Learning Rate, Momentum Rate ในแต่ละการทดลองให้ไม่เท่ากัน
- นำข้อมูลมาจาก เว็บ https://sansanee.cpe.eng.cmu.ac.th/IntroCI/Y2019/CompHw1.htm

ขั้นตอนการ Preprocessing Data

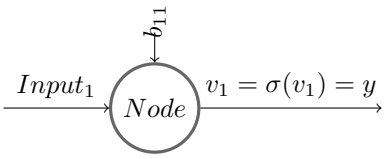
1. การเข้าไปเช็คดู Data ว่าสมบูรณ์รึเปล่า
2. ทำ Normalize Data ผมได้ทำการ Normalize Data โดย

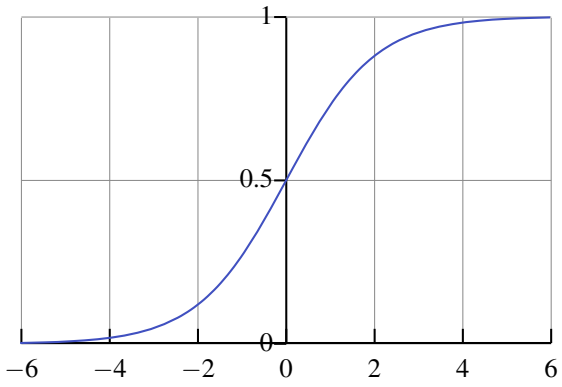
$$\frac{x - x_{min}}{x_{max} - x_{min}}$$

จะทำให้ Data อยู่ในช่วงระหว่าง [0, 1] ใน Column เดียวกันจะทำการหา x_{max}, x_{min} เพื่อเอามาคิดค่า x แต่ละตัวใน Column

Algorithm Multilayer Perceptron เป็น Algorithm ที่จะแก้ไขความถูกต้องโดยการปรับ weight ตามวิธีต่อไปนี้

1. Neural Network โดยจะเป็นเซลล์สมองตามรูปนี้



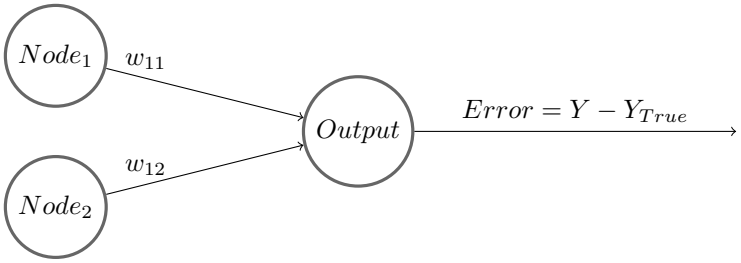


วิธีการคำนวณตามสูตร โดยที่ Input ได้ผ่านการ Normalize Data มาก่อนแล้วจึงค่อยเอามาคำนวณ ตามสูตร (Feed_Forward)

$$\text{v} = w * Input$$

จะได้ v มาซึ่ง v นั้นเอามาผ่าน Activation Function ในการบ้านนี้ผมได้เลือกใช้ Activation Function คือ Sigmoid Function σ เพื่อให้ค่าไม่กระโดดคือค่าอยู่ในช่วง $[0, 1]$ แล้วพอผ่าน $\sigma(v)$ จะได้ y ที่เป็น output ของ Node นี้เพื่อเอาไปคำนวณใน Node ต่อๆไปจนถึง output Node

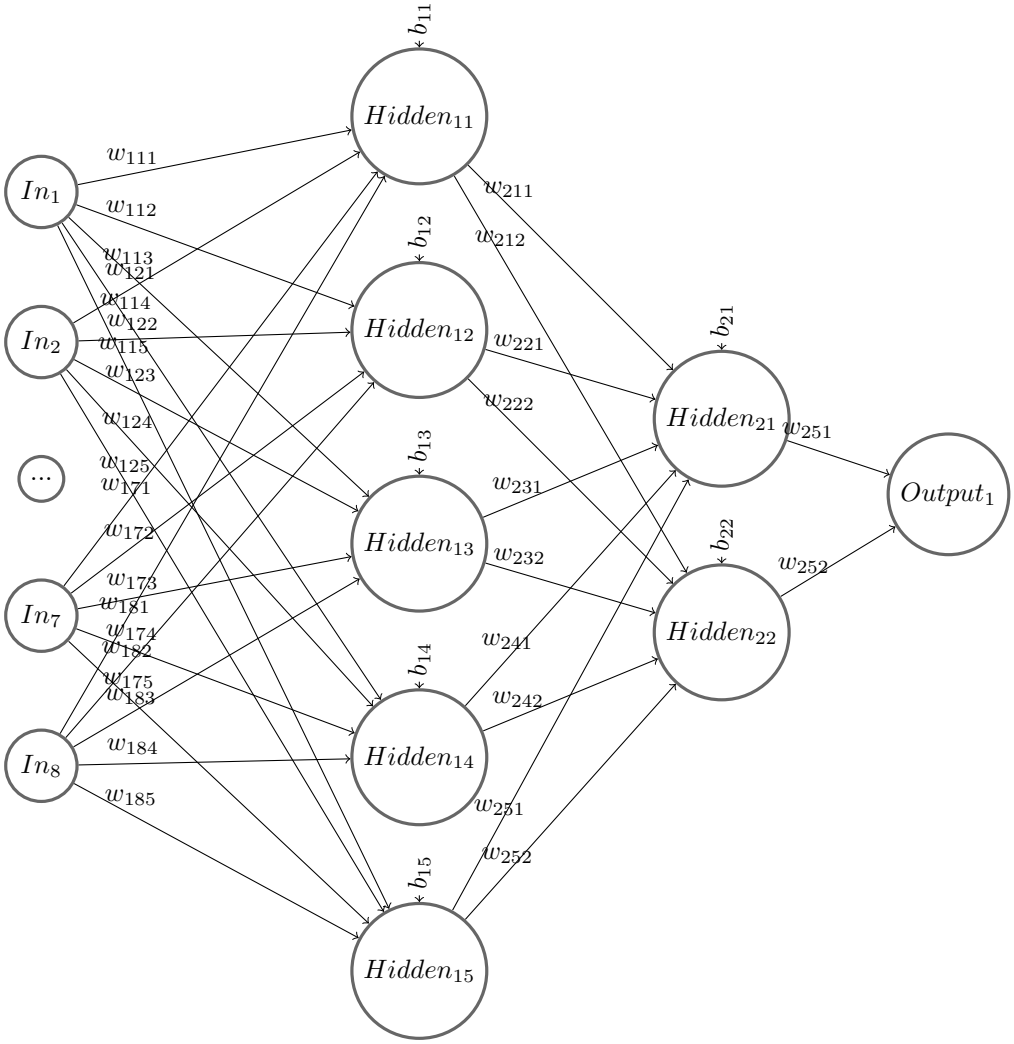
2. BackPropagation เป็นการจะปรับ weight แต่ที่คำนวณจาก error ที่คิดได้จาก Forward เพื่อให้ error น้อยลงโดยมีวิธีคิดแบบนี้



จากนั้นหา Gradient ของแต่ละ Layer เพื่อที่จะปรับ weight ในแต่ละเส้นโดยการคิด Error ออกมาแล้วทำการ Chain rules กลับไปทีละเส้นเพื่อที่จะกลับไปปรับ weight ในแต่ละเส้นให้มีความถูกต้องเพื่อให้ output ที่คิดออกมาจากการปรับ weight ให้ตรงกับคำตอบจริงๆมากขึ้น

การทดลองที่ 1 เป็นการหาค่า MSE (Mean Square Error)
วิธีการทดลอง 1 ครั้งที่ 1

1. การทดลองที่ 1 ทดลองข้อ 1 โดยโครงสร้าง มี 3 Layer โดยที่ Layer ชั้นแรกมี 5 Node, ชั้นถัดไปมี 2 Node, 1 Output Node, และมี bias ทุกๆ Node

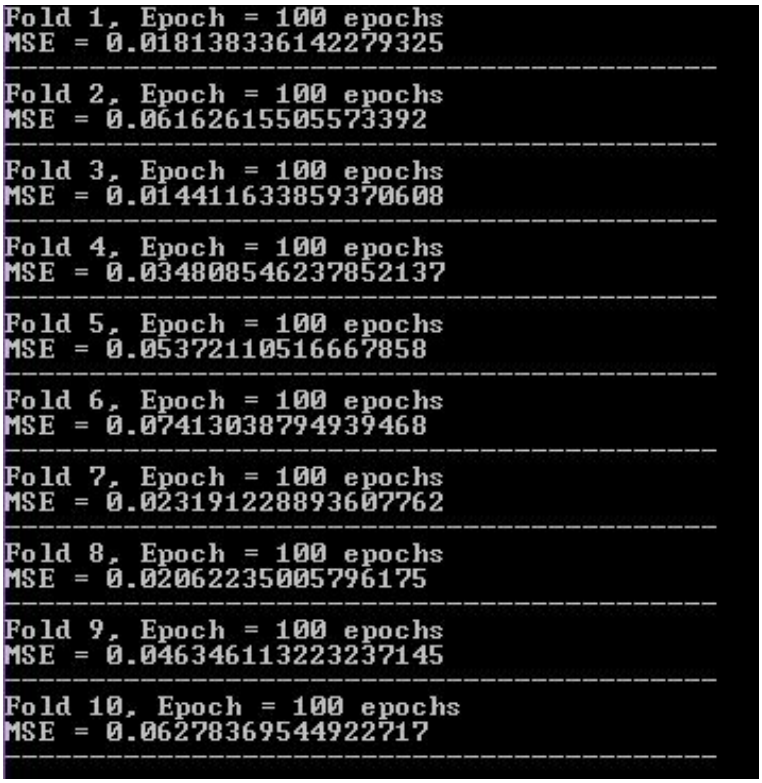


โดยที่ Input Node มีจำนวน Node เท่ากับจำนวน Features ของ Data จะ Predict หาค่า MSE (Mean Square Error) เพื่อหาความถูกต้องของ Data

2. โดยที่ Model นี้มี Learning Rate = 0.1, Momentum Rate = 0.5 กำหนดจำนวน Epoch = 100 epochs

3. สร้าง Cross Validation ขึ้นมา 10 Fold เพื่อทดสอบ Model ที่สร้างขึ้นมา

ผลการทดลองข้อที่ 1 ทดลองครั้งที่ 1



รูปที่ 1: Fold 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Fold 1 MSE $\approx$ 0.018
Fold 2 MSE $\approx$ 0.062
Fold 3 MSE $\approx$ 0.014
Fold 4 MSE $\approx$ 0.035
Fold 5 MSE $\approx$ 0.054
Fold 6 MSE $\approx$ 0.074
Fold 7 MSE $\approx$ 0.023
Fold 8 MSE $\approx$ 0.021
Fold 9 MSE $\approx$ 0.046
Fold 10 MSE $\approx$ 0.063

ลองเปลี่ยน Learning Rate เป็น 0.001, Momemtum Rate เป็น 0.5 ทดลองกับ Model เดิม

รูปที่ 2: Fold 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Fold 1 MSE $\approx$ 0.085
Fold 2 MSE $\approx$ 0.085
Fold 3 MSE $\approx$ 0.054
Fold 4 MSE $\approx$ 0.049
Fold 5 MSE $\approx$ 0.050
Fold 6 MSE $\approx$ 0.089
Fold 7 MSE $\approx$ 0.060
Fold 8 MSE $\approx$ 0.077
Fold 9 MSE $\approx$ 0.053
Fold 10 MSE $\approx$ 0.055

ลองเปลี่ยน Learning Rate เป็น 0.00357, Momemtum Rate เป็น 0.5 ทดลองกับ Model เด

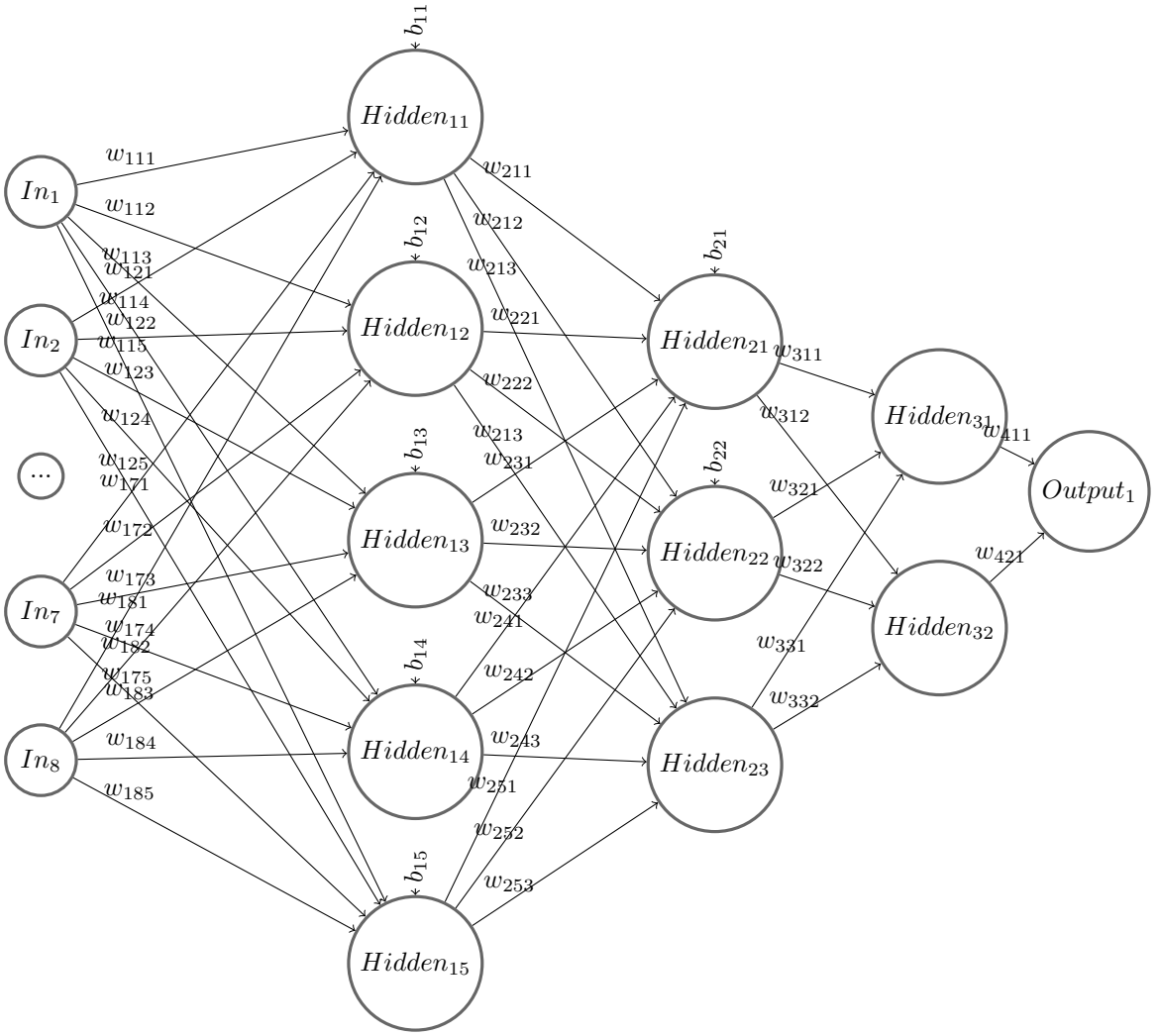
```
Fold 1, Epoch = 100 epochs
MSE = 0.08481439440178083
----------------------------------------
Fold 2, Epoch = 100 epochs
MSE = 0.08427705191919847
----------------------------------------
Fold 3, Epoch = 100 epochs
MSE = 0.04471282818659102
----------------------------------------
Fold 4, Epoch = 100 epochs
MSE = 0.05129464657105179
----------------------------------------
Fold 5, Epoch = 100 epochs
MSE = 0.052375303792745
----------------------------------------
Fold 6, Epoch = 100 epochs
MSE = 0.09061949256959502
----------------------------------------
Fold 7, Epoch = 100 epochs
MSE = 0.05632560345738148
----------------------------------------
Fold 8, Epoch = 100 epochs
MSE = 0.0810268286049898
----------------------------------------
Fold 9, Epoch = 100 epochs
MSE = 0.05394980506215717
----------------------------------------
Fold 10, Epoch = 100 epochs
MSE = 0.056356030497791136
----------------------------------------
```

รูปที่ 3: Fold 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Fold 1 MSE $\approx$ 0.084
Fold 2 MSE $\approx$ 0.084
Fold 3 MSE $\approx$ 0.044
Fold 4 MSE $\approx$ 0.051
Fold 5 MSE $\approx$ 0.052
Fold 6 MSE $\approx$ 0.090
Fold 7 MSE $\approx$ 0.056
Fold 8 MSE $\approx$ 0.081
Fold 9 MSE $\approx$ 0.054
Fold 10 MSE $\approx$ 0.056

วิธีการทดลอง 1 ครั้งที่ 2

1. การทดลองที่ 2 ทดลองข้อ 1 โดยโครงสร้าง มี 4 Layer โดยที่ Layer ชั้นแรกมี 5 Node, ชั้นถัดไปมี 3 Nodes, ชั้นถัดไปมี 2 Nodes, 1 Output Node, และมี bias ทุกๆ Node



โดยที่ Input Node มีจำนวน Node เท่ากับจำนวน Features ของ Data จะ Predict หาค่า MSE (Mean Square Error) เพื่อหาความถูกต้องของ Data เหมือนเดิม

2. โดยที่ Model นี้มี Learning Rate = 0.1, Momentum Rate = 0.5 กำหนดจำนวน Epoch = 20 epoch

3. สร้าง Cross Validation ขึ้นมา 10 Fold เพื่อทดสอบ Model ที่สร้างขึ้นมา

ผลการทดลองข้อที่ 1 ทดลองครั้งที่ 2

รูปที่ 4: Fold 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Fold 1 MSE $\approx$ 0.060
Fold 2 MSE $\approx$ 0.135
Fold 3 MSE $\approx$ 0.011
Fold 4 MSE $\approx$ 0.051
Fold 5 MSE $\approx$ 0.054
Fold 6 MSE $\approx$ 0.089
Fold 7 MSE $\approx$ 0.029
Fold 8 MSE $\approx$ 0.075
Fold 9 MSE $\approx$ 0.043
Fold 10 MSE $\approx$ 0.096

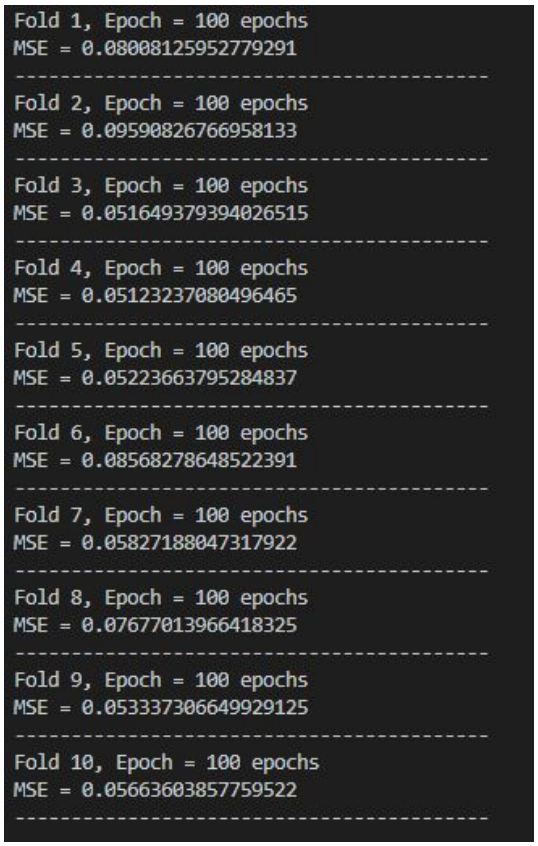
ลองเปลี่ยน Learning Rate เป็น 0.001, Momemtum Rate เป็น 0.5 ทดลองกับ Model เดิม

```
Fold 1, Epoch = 100 epochs
MSE = 0.08189648021350748
----------------------------------------
Fold 2, Epoch = 100 epochs
MSE = 0.08367635647134689
----------------------------------------
Fold 3, Epoch = 100 epochs
MSE = 0.04693798089613812
----------------------------------------
Fold 4, Epoch = 100 epochs
MSE = 0.05252692628392334
----------------------------------------
Fold 5, Epoch = 100 epochs
MSE = 0.05821584889876033
----------------------------------------
Fold 6, Epoch = 100 epochs
MSE = 0.0774239001643712
----------------------------------------
Fold 7, Epoch = 100 epochs
MSE = 0.059359190528195445
----------------------------------------
Fold 8, Epoch = 100 epochs
MSE = 0.08773980395266665
----------------------------------------
Fold 9, Epoch = 100 epochs
MSE = 0.05304548589268908
----------------------------------------
Fold 10, Epoch = 100 epochs
MSE = 0.05951697664862768
----------------------------------------
```

รูปที่ 5: Fold 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Fold 1 MSE $\approx$ 0.082
Fold 2 MSE $\approx$ 0.083
Fold 3 MSE $\approx$ 0.047
Fold 4 MSE $\approx$ 0.052
Fold 5 MSE $\approx$ 0.058
Fold 6 MSE $\approx$ 0.059
Fold 7 MSE $\approx$ 0.077
Fold 8 MSE $\approx$ 0.087
Fold 9 MSE $\approx$ 0.053
Fold 10 MSE $\approx$ 0.059

ลองเปลี่ยน Learning Rate เป็น 0.00357, Momemtum Rate เป็น 0.5 ทดลองกับ Model เดิม

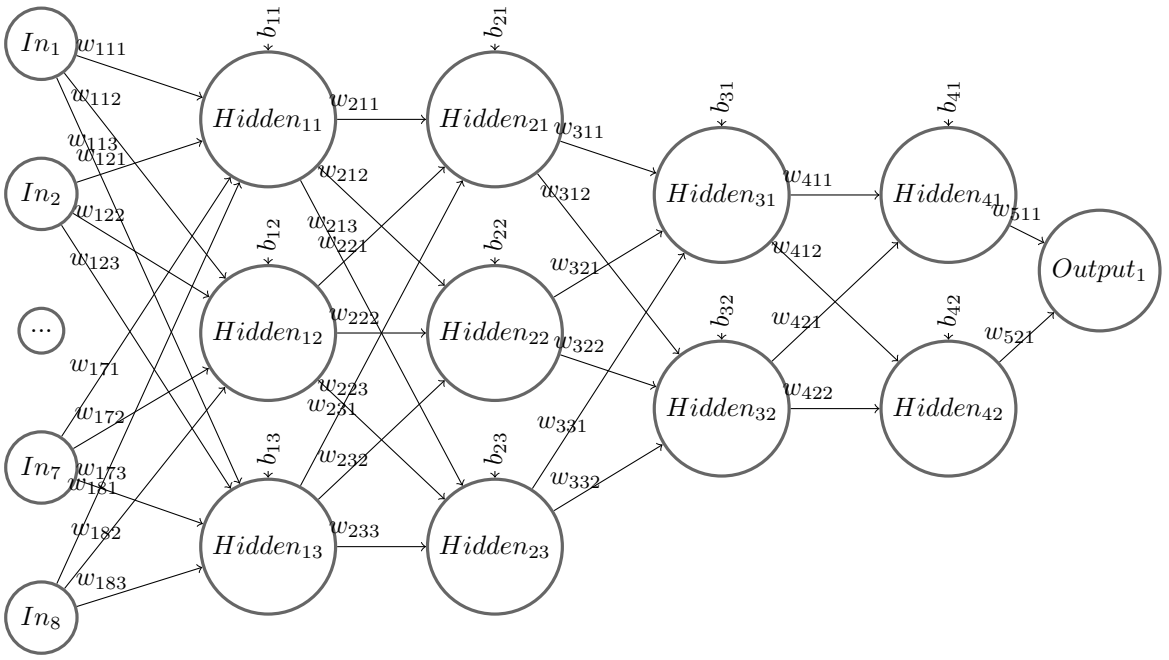


รูปที่ 6: Fold 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Fold 1 MSE $\approx$ 0.080
Fold 2 MSE $\approx$ 0.096
Fold 3 MSE $\approx$ 0.051
Fold 4 MSE $\approx$ 0.051
Fold 5 MSE $\approx$ 0.052
Fold 6 MSE $\approx$ 0.085
Fold 7 MSE $\approx$ 0.058
Fold 8 MSE $\approx$ 0.076
Fold 9 MSE $\approx$ 0.053
Fold 10 MSE $\approx$ 0.063

วิธีการทดลอง 1 ครั้งที่ 3

1. การทดลองที่ 3 ทดลองข้อ 1 โดยโครงสร้าง มี 5 Layer โดยที่ Layer ชั้นแรกมี 3 Node, ชั้นถัดไปมี 3 Nodes, ชั้นถัดไปมี 2 Nodes, 2 Nodes, 1 Output Node, และมี bias ทุกๆ Node



โดยที่ Input Node มีจำนวน Node เท่ากับจำนวน Features ของ Data จะ Predict หาค่า MSE (Mean Square Error) เพื่อหาความถูกต้องของ Data เหมือนเดิม

2. โดยที่ Model นี้มี Learning Rate = 0.1, Momentum Rate = 0.5 กำหนดจำนวน Epoch = 5 epoch

3. สร้าง Cross Validation ขึ้นมา 10 Fold เพื่อทดสอบ Model ที่สร้างขึ้นมา

ผลการทดลองข้อที่ 1 ทดลองครั้งที่ 1

```
Fold 1, Epoch = 100 epochs
MSE = 0.07704674596724286
----------------------------------------
Fold 2, Epoch = 100 epochs
MSE = 0.08606706757068844
----------------------------------------
Fold 3, Epoch = 100 epochs
MSE = 0.04808302349861191
----------------------------------------
Fold 4, Epoch = 100 epochs
MSE = 0.05199120963099378
----------------------------------------
Fold 5, Epoch = 100 epochs
MSE = 0.054376308383945166
----------------------------------------
Fold 6, Epoch = 100 epochs
MSE = 0.0891105881000198
----------------------------------------
Fold 7, Epoch = 100 epochs
MSE = 0.05966908417000785
----------------------------------------
Fold 8, Epoch = 100 epochs
MSE = 0.0763279607767713
----------------------------------------
Fold 9, Epoch = 100 epochs
MSE = 0.053775375706667315
----------------------------------------
Fold 10, Epoch = 100 epochs
MSE = 0.06077094867877956
----------------------------------------
```

รูปที่ 7: Fold 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Fold 1 MSE $\approx$ 0.077
Fold 2 MSE $\approx$ 0.086
Fold 3 MSE $\approx$ 0.048
Fold 4 MSE $\approx$ 0.052
Fold 5 MSE $\approx$ 0.054
Fold 6 MSE $\approx$ 0.089
Fold 7 MSE $\approx$ 0.059
Fold 8 MSE $\approx$ 0.076
Fold 9 MSE $\approx$ 0.053
Fold 10 MSE $\approx$ 0.060

ลองเปลี่ยน Learning Rate เป็น 0.001, Momemtum Rate เป็น 0.5 ทดลองกับ Model

เดิม

```
Fold 1, Epoch = 100 epochs
MSE = 0.06395652715795044
----------------------------------------
Fold 2, Epoch = 100 epochs
MSE = 0.09319513256378116
----------------------------------------
Fold 3, Epoch = 100 epochs
MSE = 0.04890739355476166
----------------------------------------
Fold 4, Epoch = 100 epochs
MSE = 0.05523734203354771
----------------------------------------
Fold 5, Epoch = 100 epochs
MSE = 0.052799114797383494
----------------------------------------
Fold 6, Epoch = 100 epochs
MSE = 0.09239782482364639
----------------------------------------
Fold 7, Epoch = 100 epochs
MSE = 0.05964785345453295
----------------------------------------
Fold 8, Epoch = 100 epochs
MSE = 0.08215485890786141
----------------------------------------
Fold 9, Epoch = 100 epochs
MSE = 0.05261963355531311
----------------------------------------
Fold 10, Epoch = 100 epochs
MSE = 0.05521769092084983
----------------------------------------
```

รูปที่ 8: Fold 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Fold 1 MSE $\approx$ 0.064
Fold 2 MSE $\approx$ 0.093
Fold 3 MSE $\approx$ 0.049
Fold 4 MSE $\approx$ 0.055
Fold 5 MSE $\approx$ 0.052
Fold 6 MSE $\approx$ 0.092
Fold 7 MSE $\approx$ 0.059
Fold 8 MSE $\approx$ 0.082
Fold 9 MSE $\approx$ 0.052
Fold 10 MSE $\approx$ 0.055

ลองเปลี่ยน Learning Rate เป็น 0.00357, Momemtum Rate เป็น 0.5 ทดลองกับ

Model เดิม

รูปที่ 9: Fold 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Fold 1 MSE $\approx$ 0.082
Fold 2 MSE $\approx$ 0.094
Fold 3 MSE $\approx$ 0.051
Fold 4 MSE $\approx$ 0.048
Fold 5 MSE $\approx$ 0.052
Fold 6 MSE $\approx$ 0.085
Fold 7 MSE $\approx$ 0.059
Fold 8 MSE $\approx$ 0.076
Fold 9 MSE $\approx$ 0.053
Fold 10 MSE $\approx$ 0.057

จากการทดลองในข้อที่ 1 ทั้งสามครั้ง

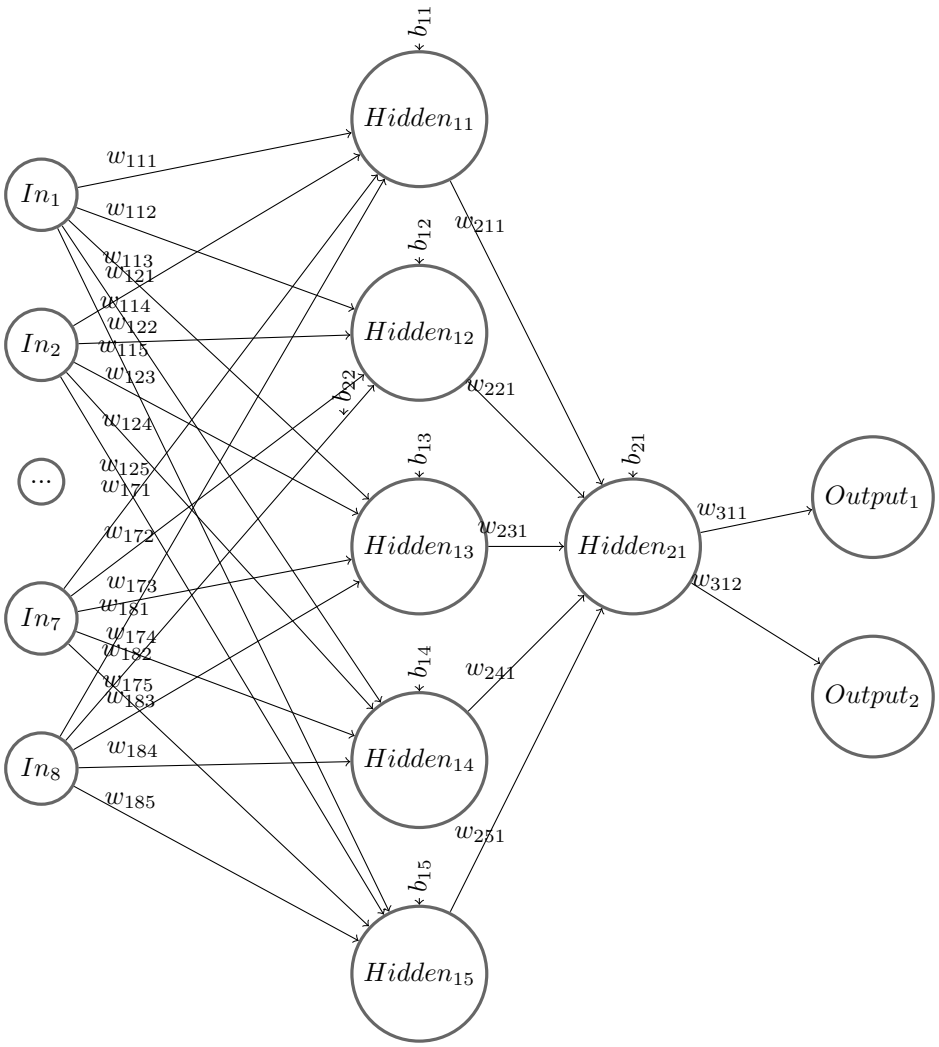
ในการทดลองครั้งที 1 Fold ที่ดีที่สุดคือ Fold ที่ 3 มีค่า MSE ≈ 0.48
ในการทดลองครั้งที่ 2 Fold ที่ดีที่สุดคือ ที่ 3 มีค่า MSE ≈ 0.49
ในการทอลองครั้งที่ 3 Fold ที่ดีที่สุดคือ Fold ที่ 4 มีค่า MSE ≈ 0.048

โดยทดลองกับ แต่ละ Model ที่มีขนาด Layer ต่างกันแต่ในแต่ละการทดลองได้ทำการทดลองปรับ Learning Rate ไว้ 3 ค่าคือ 0.1, 0.001, 0.00375 และมี Momentum เท่ากันทั้งหมดโดยที่ weight และ bias นั้นได้ทำการสุ่มใหม่ทุกครั้งที่เริ่มทำการทดลองในแต่ละครั้งโดยได้ทำการทดลองไป 9 ครั้งนั้นเอง

การทดลองที่ 2 เป็นการหาว่า Model จะตอบ Class ที่ถูกต้องรึเปล่า

วิธีการทดลอง 2 ครั้งที่ 1

1. การทดลองที่ 1 ทดลองข้อ 2 โดยโครงสร้าง มี 3 Layer โดยที่ Layer ชั้นแรกมี 5 Node, ชั้นถัดไปมี 1 Nodes, 1 Output Node, และมี bias ทุกๆ Node



โดยที่ Input Node มีจำนวน Node เท่ากับจำนวน Features ของ Data จะ Predict หา Class ที่ถูกต้อง เพื่อหาความถูกต้องของ Model

2. โดยที่ Model นี้มี Learning Rate = 100, Momentum Rate = 0.25 กำหนดจำนวน

Epoch = 100 epochs

3. สร้าง Cross Validation ขึ้นมา 10 Fold เพื่อทดสอบ Model ที่สร้างขึ้น

ผลการทดลองที่ 2 ครั้งที่ 1 Fold ที่ 1 ถึง 10

```
Fold 1, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 2, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 3, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 4, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 5, Epoch = 100 epochs

         Predict
       |-----------
Actual | [10.  0.]
       | [10.  0.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
```

รูปที่ 10: Fold 1, 2, 3, 4, 5

```
Fold 6, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 7, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 8, Epoch = 100 epochs

         Predict
       |-----------
Actual | [10.  0.]
       | [10.  0.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 9, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 10, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
```

รูปที่ 11: Fold 6, 7, 8, 9, 10

ผลการ Predict Class ของแต่ละ Fold

Fold 1 = 10/20

Fold 2 = 10/20

Fold 3 = 10/20

Fold 4 = 10/20

Fold 5 = 10/20

Fold 6 = 10/20

Fold 7 = 10/20
Fold 8 = 10/20
Fold 9 = 10/20
Fold 10 = 10/20

ลองเปลี่ยน Learning Rate เป็น 0.1, Momemtum Rate เป็น 0.25 ทดลองกับ Model เดิม

```
Fold 1, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [3. 7.]

Accuracy = 35.0 %, 7/20
-----------------------------------------
Fold 2, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [1. 9.]

Accuracy = 45.0 %, 9/20
-----------------------------------------
Fold 3, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [4. 6.]

Accuracy = 30.0 %, 6/20
-----------------------------------------
Fold 4, Epoch = 100 epochs

         Predict
       |-----------
Actual | [1. 9.]
       | [3. 7.]

Accuracy = 40.0 %, 8/20
-----------------------------------------
Fold 5, Epoch = 100 epochs

         Predict
       |-----------
Actual | [10.  0.]
       | [ 0. 10.]

Accuracy = 100.0 %, 20/20
-----------------------------------------
```

รูปที่ 12: Fold 1, 2, 3, 4, 5

```
Fold 6, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [7. 3.]

Accuracy = 15.0 %, 3/20
-----------------------------------------
Fold 7, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 8, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 9, Epoch = 100 epochs

         Predict
       |-----------
Actual | [6. 4.]
       | [7. 3.]

Accuracy = 45.0 %, 9/20
-----------------------------------------
Fold 10, Epoch = 100 epochs

         Predict
       |-----------
Actual | [4. 6.]
       | [4. 6.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
```

รูปที่ 13: Fold 6, 7, 8, 9, 10

ผลการ Predict Class ของแต่ละ Fold
Fold 1 = 7/20
Fold 2 = 9/20
Fold 3 = 6/20
Fold 4 = 8/20

Fold 5 = 20/20
Fold 6 = 3/20
Fold 7 = 10/20
Fold 8 = 9/20
Fold 9 = 10/20
Fold 10 = 10/20

ลองเปลี่ยน Learning Rate เป็น 0.001375, Momemtum Rate เป็น 0.25 ทดลองกับ Model เดิม

```
Fold 1, Epoch = 100 epochs

         Predict
       |-----------
Actual | [10.  0.]
       | [10.  0.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 2, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 3, Epoch = 100 epochs

         Predict
       |-----------
Actual | [10.  0.]
       | [10.  0.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 4, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 5, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
```

รูปที่ 14: Fold 1, 2, 3, 4, 5

```
Fold 6, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 7, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 8, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 9, Epoch = 100 epochs

         Predict
       |-----------
Actual | [10.  0.]
       | [10.  0.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 10, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
```

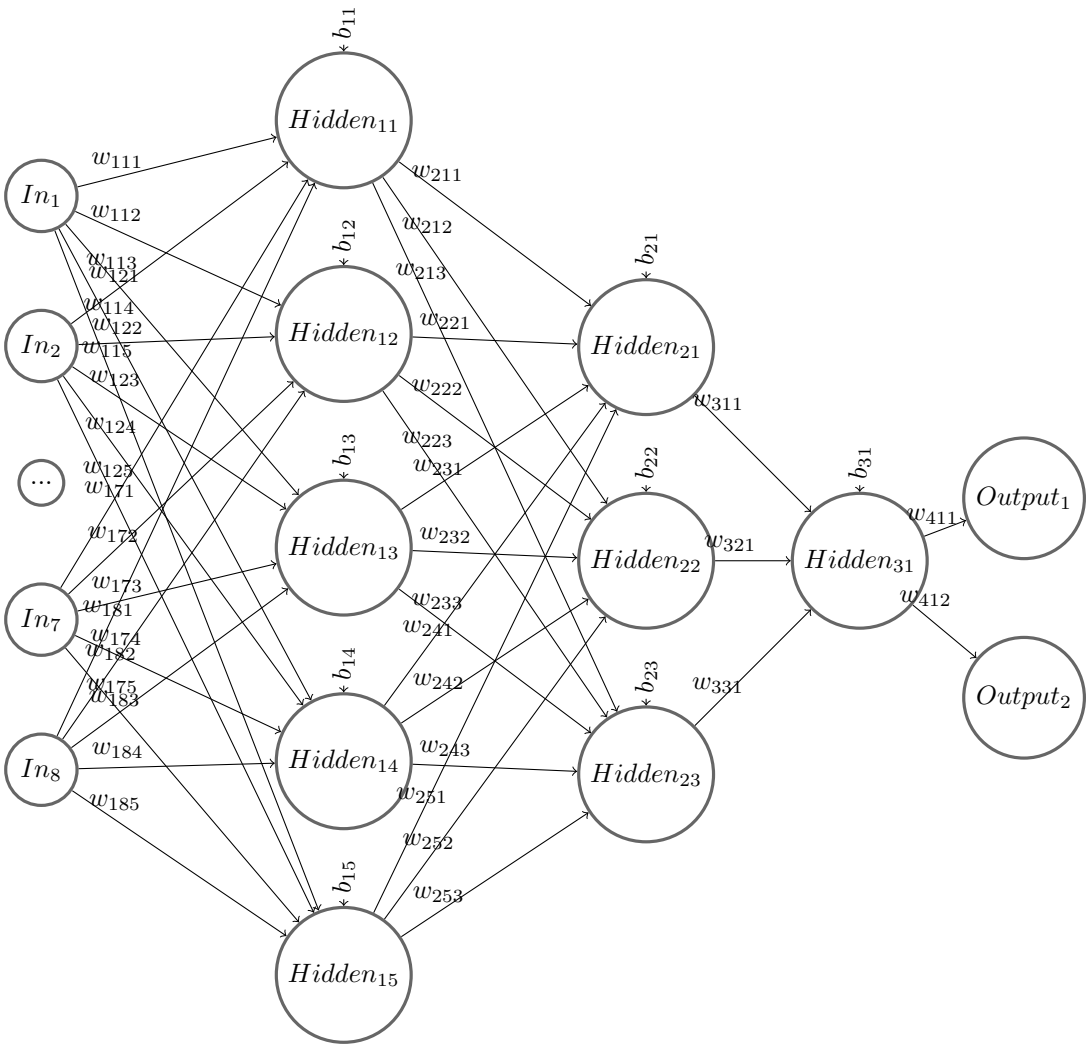
รูปที่ 15: Fold 6, 7, 8, 9, 10

ผลการ Predict Class ของแต่ละ Fold
Fold 1 = 10/20

Fold 2 = 10/20
Fold 3 = 10/20
Fold 4 = 10/20
Fold 5 = 10/20
Fold 6 = 10/20
Fold 7 = 10/20
Fold 8 = 10/20
Fold 9 = 10/20
Fold 10 = 10/20

วิธีการทดลอง 2 ครั้งที่ 2

1. การทดลองที่ 2 ทดลองข้อ 2 โดยโครงสร้าง มี 4 Layer, โดยที่ Layer ชั้นแรกมี 5 Node, ชั้นถัดไปมี 3 Nodes, ชั้นถัดไปมี 1 Nodes, 1 Output Node, และมี bias ทุกๆ Node



โดยที่ Input Node มีจำนวน Node เท่ากับจำนวน Features ของ Data จะ Predict หาค่า MSE (Mean Square Error) เพื่อหาความถูกต้องของ Data เหมือนเดิม

2. โดยที่ Model นี้มี Learning Rate = 100, Momentum Rate = 0.25 กำหนดจำนวน Epoch = 100 epochs

3. สร้าง Cross Validation ขึ้นมา 10 Fold เพื่อทดสอบ Model ที่สร้างขึ้นมา

ผลการทดลองที่ 2 ครั้งที่ 2 Fold ที่ 1 ถึง 10

```
Fold 1, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 2, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 3, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 4, Epoch = 100 epochs

         Predict
       |-----------
Actual | [10.  0.]
       | [10.  0.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 5, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
```

รูปที่ 16: Fold 1, 2, 3, 4, 5

```
Fold 6, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 7, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 8, Epoch = 100 epochs

         Predict
       |-----------
Actual | [10.  0.]
       | [10.  0.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 9, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 10, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
```

รูปที่ 17: Fold 6, 7, 8, 9, 10

ผลการ Predict Class ของแต่ละ Fold

Fold 1 = 10/20

Fold 2 = 10/20

Fold 3 = 10/20

Fold 4 = 10/20

Fold 5 = 10/20

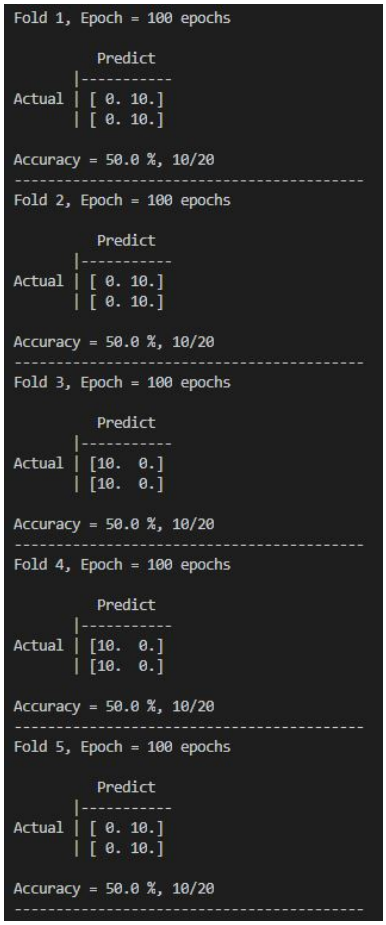
Fold 6 = 10/20

Fold 7 = 10/20

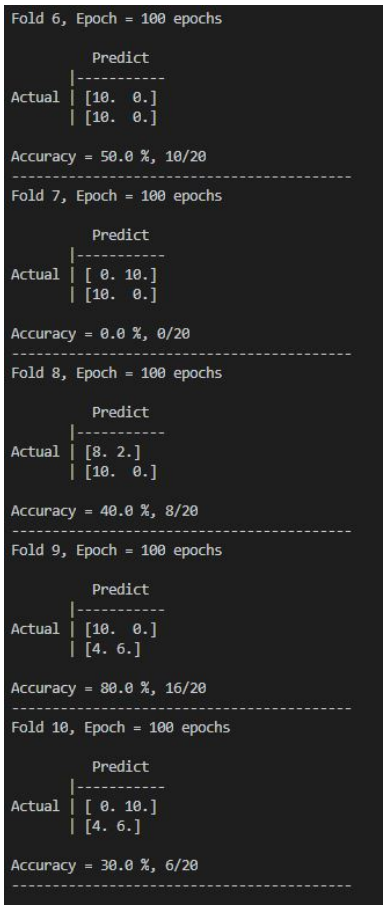
Fold 8 = 10/20

Fold 9 = 10/20
Fold 10 = 10/20

ลองเปลี่ยน Learning Rate เป็น 0.01, Momemtum Rate เป็น 0.25 ทดลองกับ Model เดิม



รูปที่ 18: Fold 1, 2, 3, 4, 5



รูปที่ 19: Fold 6, 7, 8, 9, 10

ผลการ Predict Class ของแต่ละ Fold
Fold 1 = 10/20
Fold 2 = 10/20
Fold 3 = 10/20
Fold 4 = 10/20

Fold 5 = 10/20
Fold 6 = 10/20
Fold 7 = 0/20
Fold 8 = 8/20
Fold 9 = 16/20
Fold 10 = 6/20

ลองเปลี่ยน Learning Rate เป็น 0.00375, Momemtum Rate เป็น 0.25 ทดลองกับ Model เดิม

```
Fold 1, Epoch = 100 epochs

         Predict
       |-----------
Actual | [10.  0.]
       | [10.  0.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 2, Epoch = 100 epochs

         Predict
       |-----------
Actual | [10.  0.]
       | [10.  0.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 3, Epoch = 100 epochs

         Predict
       |-----------
Actual | [10.  0.]
       | [10.  0.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 4, Epoch = 100 epochs

         Predict
       |-----------
Actual | [10.  0.]
       | [10.  0.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 5, Epoch = 100 epochs

         Predict
       |-----------
Actual | [10.  0.]
       | [10.  0.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
```

รูปที่ 20: Fold 1, 2, 3, 4, 5

```
Fold 6, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 7, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 8, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 9, Epoch = 100 epochs

         Predict
       |-----------
Actual | [10.  0.]
       | [10.  0.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
Fold 10, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
-----------------------------------------
```

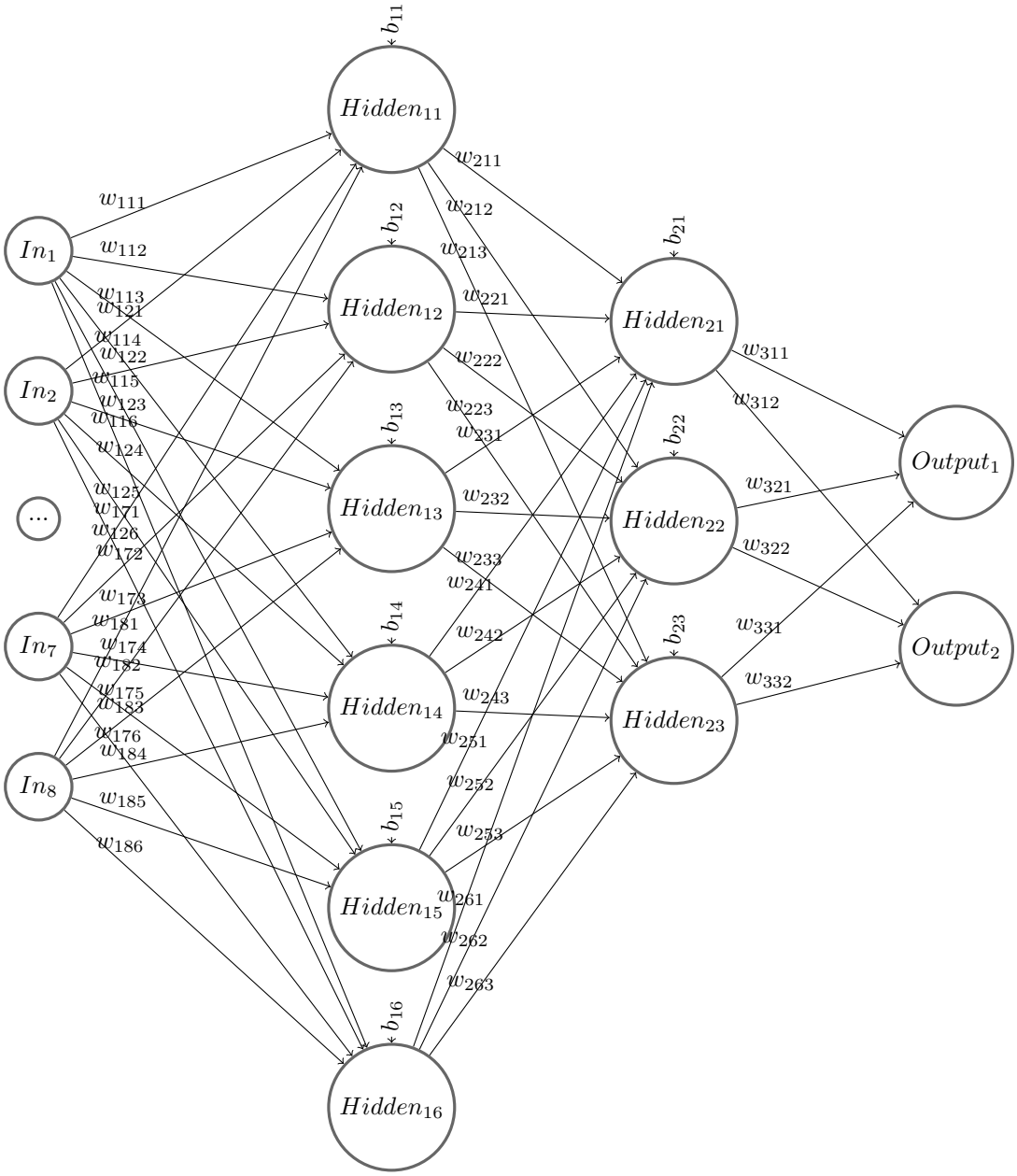
รูปที่ 21: Fold 6, 7, 8, 9, 10

ผลการ Predict Class ของแต่ละ Fold

Fold 1 = 10/20
Fold 2 = 10/20
Fold 3 = 10/20
Fold 4 = 10/20
Fold 5 = 10/20
Fold 6 = 10/20
Fold 7 = 10/20
Fold 8 = 10/20
Fold 9 = 10/20
Fold 10 = 10/20

วิธีการทดลอง 2 ครั้งที่ 3

1. การทดลองที่ 2 ทดลองข้อ 2 โดยโครงสร้าง มี 4 Layer, โดยที่ Layer ชั้นแรกมี 5 Node, ชั้นถัดไปมี 3 Nodes, ชั้นถัดไปมี 1 Nodes, 1 Output Node, และมี bias ทุกๆ Node



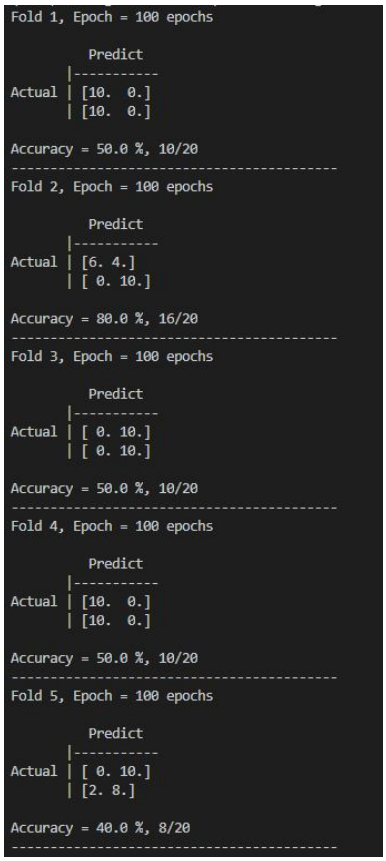
โดยที่ Input Node มีจำนวน Node เท่ากับจำนวน Features ของ Data จะ Predict หาค่า MSE (Mean Square Error) เพื่อหาความถูกต้องของ Data เหมือนเดิม

2. โดยที่ Model นี้มี Learning Rate = 100, Momentum Rate = 0.25 กำหนดจำนวน

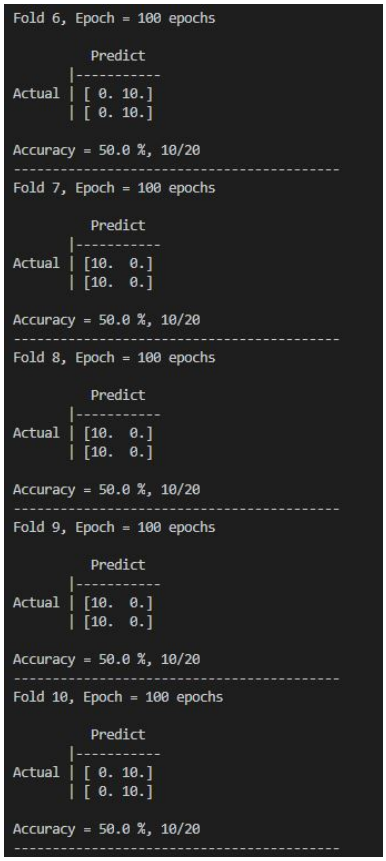
Epoch = 100 epochs

3. สร้าง Cross Validation ขึ้นมา 10 Fold เพื่อทดสอบ Model ที่สร้างขึ้นมา

ผลการทดลองที่ 2 ครั้งที่ 3 Fold ที่ 1 ถึง 10



รูปที่ 22: Fold 1, 2, 3, 4, 5



รูปที่ 23: Fold 6, 7, 8, 9, 10

ผลการ Predict Class ของแต่ละ Fold

Fold 1 = 10/20

Fold 2 = 16/20

Fold 3 = 10/20

Fold 4 = 10/20

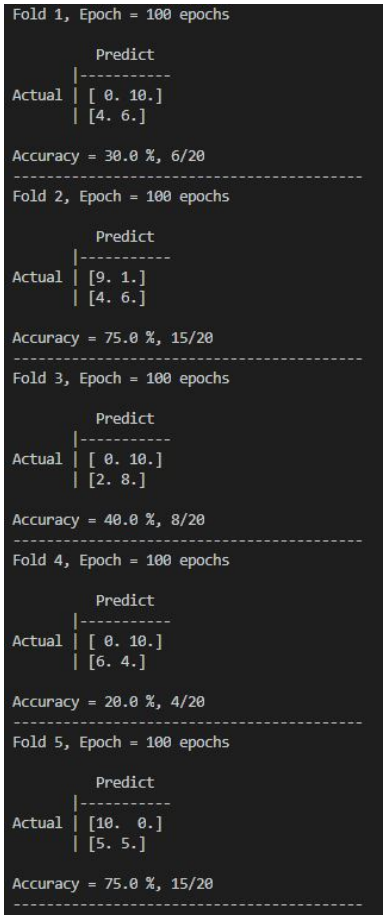
Fold 5 = 8/20

Fold 6 = 10/20

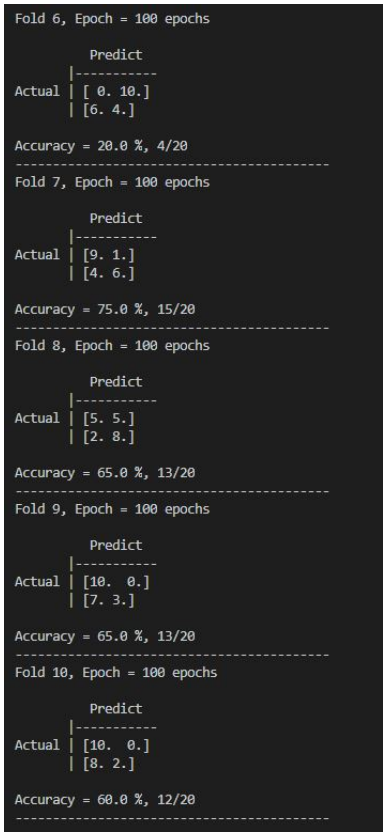
Fold 7 = 10/20

Fold 8 = 10/20
Fold 9 = 10/20
Fold 10 = 10/20

ลองเปลี่ยน Learning Rate เป็น 0.01, Momemtum Rate เป็น 0.25 ทดลองกับ Model เดิม



รูปที่ 24: Fold 1, 2, 3, 4, 5



รูปที่ 25: Fold 6, 7, 8, 9, 10

ผลการ Predict Class ของแต่ละ Fold
Fold 1 = 6/20
Fold 2 = 15/20
Fold 3 = 8/20
Fold 4 = 4/20

Fold 5 = 15/20
Fold 6 = 4/20
Fold 7 = 15/20
Fold 8 = 13/20
Fold 9 = 13/20
Fold 10 = 12/20

ลองเปลี่ยน Learning Rate เป็น 0.00375, Momemtum Rate เป็น 0.25 ทดลองกับ Model เดิม

```
Fold 1, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [1. 9.]

Accuracy = 45.0 %, 9/20
----------------------------------------
Fold 2, Epoch = 100 epochs

         Predict
       |-----------
Actual | [10.  0.]
       | [2. 8.]

Accuracy = 90.0 %, 18/20
----------------------------------------
Fold 3, Epoch = 100 epochs

         Predict
       |-----------
Actual | [5. 5.]
       | [4. 6.]

Accuracy = 55.00000000000001 %, 11/20
----------------------------------------
Fold 4, Epoch = 100 epochs

         Predict
       |-----------
Actual | [10.  0.]
       | [4. 6.]

Accuracy = 80.0 %, 16/20
----------------------------------------
Fold 5, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
----------------------------------------
```

รูปที่ 26: Fold 1, 2, 3, 4, 5

```
Fold 6, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [9. 1.]

Accuracy = 5.0 %, 1/20
----------------------------------------
Fold 7, Epoch = 100 epochs

         Predict
       |-----------
Actual | [2. 8.]
       | [3. 7.]

Accuracy = 45.0 %, 9/20
----------------------------------------
Fold 8, Epoch = 100 epochs

         Predict
       |-----------
Actual | [ 0. 10.]
       | [ 0. 10.]

Accuracy = 50.0 %, 10/20
----------------------------------------
Fold 9, Epoch = 100 epochs

         Predict
       |-----------
Actual | [8. 2.]
       | [3. 7.]

Accuracy = 75.0 %, 15/20
----------------------------------------
Fold 10, Epoch = 100 epochs

         Predict
       |-----------
Actual | [10.  0.]
       | [10.  0.]

Accuracy = 50.0 %, 10/20
----------------------------------------
```

รูปที่ 27: Fold 6, 7, 8, 9, 10

ผลการ Predict Class ของแต่ละ Fold

Fold 1 = 9/20
Fold 2 = 18/20
Fold 3 = 11/20
Fold 4 = 16/20
Fold 5 = 10/20
Fold 6 = 1/20
Fold 7 = 9/20
Fold 8 = 10/20
Fold 9 = 15/20
Fold 10 = 10/20

จากการทดลองในข้อที่ 2 ทั้งสามครั้ง

ในการทดลองครั้งที 1 จะไม่มี Fold ที่เดาผิดพลาดเลย
ในการทดลองครั้งที่ 2 จะมีแค่ Fold ที่ 5 ที่เดาผิดไป 50 %
ในการทอลองครั้งที่ 3 เกือบทุก Fold นั้นเดาผิดไปเยอะมาก มีแค่ Fold 5 ที่เดาถูก Class หมด

โดยทดลองกับ แต่ละ Model ที่ขนาดของ Layer และ Node ไม่เท่ากันและ Learning Rate, Momentum Rate, weight, bias ก็สุ่มใหม่ทุกครั้งในการทดลองแต่ละครั้ง ในแต่ละครั้งไม่เท่ากันจะมีค่าต่างกันไม่มาก ใน การทดลองสองตอนทดลอง Learning Rate 100 ทุก Model จะได้ ทุก Fold ตอบได้ 50 % คิดว่าเพราะ ตัว Learning Rate, Momentum Rate นั้นมีแค่มากเกินไปจึง Learn แบบก้าวกระโดดมากเกินไปจะเห็นได้ว่าพอปรับ Learning Rate น้อยลงทำให้การทดลองนั้น มีการตอบที่หลากหลายมากขึ้นเรื่อยๆเพราะว่าปรับ weight ทีละน้อยๆก็จะทำให้ Learn แบบไม่ก้าวกระโดดมากจึงตอบได้หลากหลายมากขึ้น

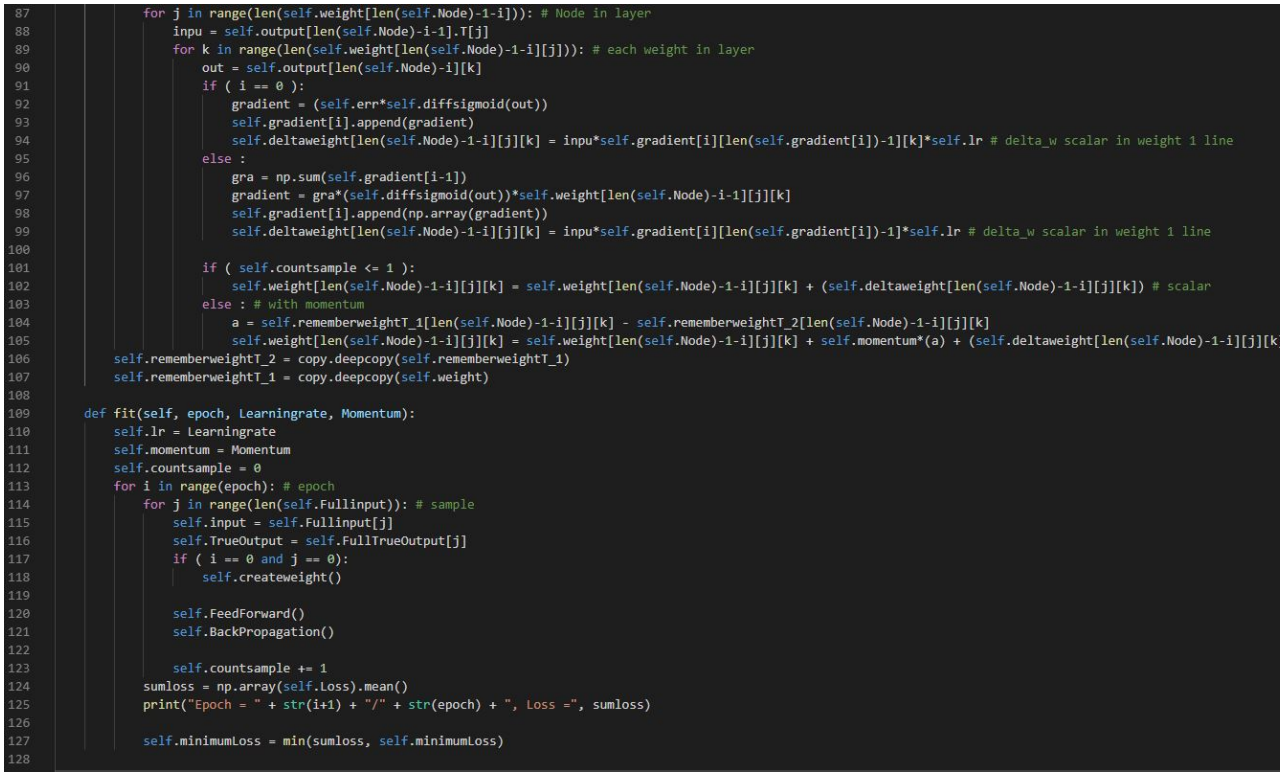
ภาคผนวก

```python
import numpy as np
import csv
import copy

data = []
abc = 2

def Normailize(x):
    mi = x.min()
    ma = x.max()
    x = (x - mi)/(ma - mi)
    return x

class NeuralNetwork:
    def __init__(self, inpu, out):
        # np.random.seed(1)
        self.Fullinput = inpu
        self.FullTrueOutput = out
        self.input = None
        self.weight = []
        self.bias = []
        self.predict = 0
        self.lr = 0
        self.E = 0
        self.countLayer = 0
        self.Node = []
        self.output = []
        self.TrueOutput= None
        self.cou = 0
        self.deltaweight = []
        self.err = []
        self.deltabias = []
        self.gradient = [] # back to front
        self.momentum = 0
        self.Loss = []
        self.minimumLoss = 10000000
        self.rememberweightT_1 = []
        self.countsample = 0
        self.rememberweightT_2 = []

    def addLayer(self, node):
        self.countLayer+=1
        self.Node.append(node)

```

รูปที่ 28: Function Normalize, สร้าง Neural Network

```
45      def sigmoid(self, v):
46          return 1/(1+np.exp(-v))
47
48      def diffsigmoid(self, y):
49          return y*(1-y)
50
51      def createweight(self):
52          for i in range(len(self.Node)):
53              if ( i == 0 ):
54                  self.weight.append(2*np.random.rand(len(self.input), self.Node[i]) - 1)
55                  self.bias.append(2*np.random.rand(self.Node[i]) - 1)
56                  self.deltaweight.append(np.ones((len(self.input), self.Node[i])))
57                  self.deltabias.append(np.ones(self.Node[i]))
58                  self.rememberweightT_1.append(np.zeros((len(self.input), self.Node[i])))
59                  self.rememberweightT_2.append(np.zeros((len(self.input), self.Node[i])))
60              else:
61                  self.weight.append(2*np.random.rand(self.Node[i-1], self.Node[i]) - 1)
62                  self.bias.append(2*np.random.rand(self.Node[i]) - 1)
63                  self.deltaweight.append(np.ones((self.Node[i-1], self.Node[i])))
64                  self.deltabias.append(np.ones(self.Node[i]))
65                  self.rememberweightT_1.append(np.zeros((len(self.input), self.Node[i])))
66                  self.rememberweightT_2.append(np.zeros((len(self.input), self.Node[i])))
67
68      def FeedForward(self):
69          # each sample, each epoch
70          self.output = []
71          self.output.append(self.input.T)
72          out = np.array(self.output[0])
73          for i in range(len(self.Node)): # feed in each layer
74              v = np.dot(out, self.weight[i]) + self.bias[i]
75              out = self.sigmoid(v)
76              self.output.append(out)
77
78      def BackPropagation(self):
79          self.gradient = []
80          self.err = self.TrueOutput - self.output[len(self.output)-1]
81          self.Loss = []
82          Loss = (0.5)*((self.err)**2) # scalar
83          #calculate gradient
84          self.Loss.append(Loss)
85          for i in range(len(self.Node)): # for layer
86              self.gradient.append([])
```

รูปที่ 29: Function sigmoid, diffsigmoid, สร้าง Weight, FeedForward, Backprob (เริ่ม-ต้น)



```
            for j in range(len(self.weight[len(self.Node)-1-i])): # Node in layer
                inpu = self.output[len(self.Node)-i-1].T[j]
                for k in range(len(self.weight[len(self.Node)-1-i][j])): # each weight in layer
                    out = self.output[len(self.Node)-i][k]
                    if ( i == 0 ):
                        gradient = (self.err*self.diffsigmoid(out))
                        self.gradient[i].append(gradient)
                        self.deltaweight[len(self.Node)-1-i][j][k] = inpu*self.gradient[i][len(self.gradient[i])-1][k]*self.lr # delta_w scalar in weight 1 line
                    else :
                        gra = np.sum(self.gradient[i-1])
                        gradient = gra*(self.diffsigmoid(out))*self.weight[len(self.Node)-i-1][j][k]
                        self.gradient[i].append(np.array(gradient))
                        self.deltaweight[len(self.Node)-1-i][j][k] = inpu*self.gradient[i][len(self.gradient[i])-1]*self.lr # delta_w scalar in weight 1 line

                    if ( self.countsample <= 1 ):
                        self.weight[len(self.Node)-1-i][j][k] = self.weight[len(self.Node)-1-i][j][k] + (self.deltaweight[len(self.Node)-1-i][j][k]) # scalar
                    else : # with momentum
                        a = self.rememberweightT_1[len(self.Node)-1-i][j][k] - self.rememberweightT_2[len(self.Node)-1-i][j][k]
                        self.weight[len(self.Node)-1-i][j][k] = self.weight[len(self.Node)-1-i][j][k] + self.momentum*(a) + (self.deltaweight[len(self.Node)-1-i][j][k
        self.rememberweightT_2 = copy.deepcopy(self.rememberweightT_1)
        self.rememberweightT_1 = copy.deepcopy(self.weight)

    def fit(self, epoch, Learningrate, Momentum):
        self.lr = Learningrate
        self.momentum = Momentum
        self.countsample = 0
        for i in range(epoch): # epoch
            for j in range(len(self.Fullinput)): # sample
                self.input = self.Fullinput[j]
                self.TrueOutput = self.FullTrueOutput[j]
                if ( i == 0 and j == 0):
                    self.createweight()

                self.FeedForward()
                self.BackPropagation()

                self.countsample += 1
            sumloss = np.array(self.Loss).mean()
            print("Epoch = " + str(i+1) + "/" + str(epoch) + ", Loss =", sumloss)

            self.minimumLoss = min(sumloss, self.minimumLoss)

```

รูปที่ 30: Backprob (ต่อ), fit (function train)

```
129     def predict_data(self, inpu, output):
130         self.Fullinput = inpu
131         self.FullTrueOutput = output
132         self.Loss = []
133         for j in range(len(self.Fullinput)):
134             self.input = self.Fullinput[j]
135             self.TrueOutput = self.FullTrueOutput[j]
136 
137             self.FeedForward()
138 
139             self.err = self.TrueOutput - self.output[len(self.output)-1]
140             Loss = (0.5)*((self.err)**2) # scalar
141             self.Loss.append(Loss)
142 
143         MeanLoss = np.array(self.Loss).mean()
144         return(MeanLoss)
145 
146     def confusion_matrix(self, inpu, output):
147         self.Fullinput = inpu
148         self.FullTrueOutput = output
149         self.Loss = []
150         sizeclass = len(output[0])
151         classconfusion = np.zeros((sizeclass, sizeclass))
152         for j in range(len(self.Fullinput)):
153             self.input = self.Fullinput[j]
154             self.TrueOutput = self.FullTrueOutput[j]
155 
156             self.FeedForward()
157 
158             self.err = self.TrueOutput - self.output[len(self.output)-1]
159             classconfusion[self.TrueOutput.argmax()][self.err.argmax()] += 1
160 
161         print()
162         print("          Predict")
163         print("       |-----------")
164         for i in range(sizeclass):
165             if ( i == 0 ) :
166                 print("Actual |", classconfusion[i])
167             else :
168                 print("       |", classconfusion[i])
169         accuracy = 0
170         for i in range(len(classconfusion)):
```

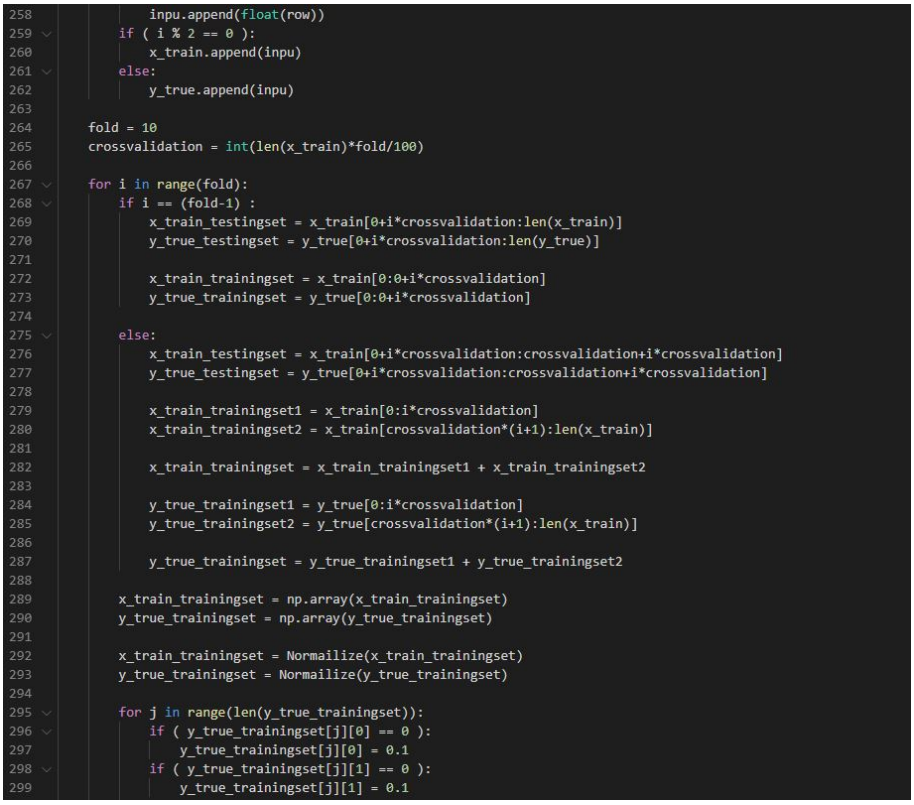
รูปที่ 31: Function Predict (คิด MSE), confusion_matrix

```
                accuracy += classconfusion[i][i]
            print()
            print("Accuracy = " + str(int(accuracy)) + "/" + str(len(output)))

    # first
    if ( abc == 1 ):
        with open('dataset.csv', 'rt')as f:
            d = csv.reader(f)
            for row in d:
                data.append(row)

        x_train = []
        y_true = []

        for i in range(len(data)):
            if ( i > 1 ):
                inpu = []
                for j in range(len(data[i])):
                    if ( j < 8 ):
                        inpu.append(float(data[i][j]))
                    else:
                        y_true.append(float(data[i][j]))
                x_train.append(inpu)

        fold = 10
        crossvalidation = int(len(x_train)*fold/100)

        for i in range(fold):
            if i == (fold-1) :
                x_train_testingset = x_train[0+i*crossvalidation:len(x_train)]
                y_true_testingset = y_true[0+i*crossvalidation:len(y_true)]

                x_train_trainingset = x_train[0:0+i*crossvalidation]
                y_true_trainingset = y_true[0:0+i*crossvalidation]

            else:
                x_train_testingset = x_train[0+i*crossvalidation:crossvalidation+i*crossvalidation]
                y_true_testingset = y_true[0+i*crossvalidation:crossvalidation+i*crossvalidation]

                x_train_trainingset1 = x_train[0:i*crossvalidation]
                x_train_trainingset2 = x_train[crossvalidation*(i+1):len(x_train)]

                x_train_trainingset = x_train_trainingset1 + x_train_trainingset2
```

รูปที่ 32: confusion$_matrix(), TrainTestsetData$

```
            y_true_trainingset1 = y_true[0:i*crossvalidation]
            y_true_trainingset2 = y_true[crossvalidation*(i+1):len(x_train)]

            y_true_trainingset = y_true_trainingset1 + y_true_trainingset2

        x_train_trainingset = np.array(x_train_trainingset)
        y_true_trainingset = np.array(y_true_trainingset)

        x_train_trainingset = Normailize(x_train_trainingset)
        y_true_trainingset = Normailize(y_true_trainingset)

        nn = NeuralNetwork(x_train_trainingset, y_true_trainingset)

        # nn.addLayer(5)
        # nn.addLayer(3)
        # nn.addLayer(1)

        x_train_testingset = np.array(x_train_testingset)
        y_true_testingset = np.array(y_true_testingset)

        x_train_testingset = Normailize(x_train_testingset)
        y_true_testingset = Normailize(y_true_testingset)

        print("Fold " + str(i+1))
        nn.fit(10, 10, 10)
        print("MSE =", nn.predict_data(x_train_testingset, y_true_testingset))
        print("----------------------------------------")

# second
elif ( abc == 2 ):
    cou = 0
    with open('cross.pat', 'rt')as f:
        d = csv.reader(f)
        for row in d:
            if ( cou % 3 != 0 ):
                data.append(row)
            cou += 1
    x_train = []
    y_true = []
    for i in range(len(data)):
        data[i][0] = data[i][0].split()
        inpu = []
        for row in data[i][0]:
```

รูปที่ 33: แบ่ง Train กับ Test set ของ Data ข้อแรก(ต่อ), เริ่มต้นข้อสอง



```
            inpu.append(float(row))
        if ( i % 2 == 0 ):
            x_train.append(inpu)
        else:
            y_true.append(inpu)

    fold = 10
    crossvalidation = int(len(x_train)*fold/100)

    for i in range(fold):
        if i == (fold-1) :
            x_train_testingset = x_train[0+i*crossvalidation:len(x_train)]
            y_true_testingset = y_true[0+i*crossvalidation:len(y_true)]

            x_train_trainingset = x_train[0:0+i*crossvalidation]
            y_true_trainingset = y_true[0:0+i*crossvalidation]

        else:
            x_train_testingset = x_train[0+i*crossvalidation:crossvalidation+i*crossvalidation]
            y_true_testingset = y_true[0+i*crossvalidation:crossvalidation+i*crossvalidation]

            x_train_trainingset1 = x_train[0:i*crossvalidation]
            x_train_trainingset2 = x_train[crossvalidation*(i+1):len(x_train)]

            x_train_trainingset = x_train_trainingset1 + x_train_trainingset2

            y_true_trainingset1 = y_true[0:i*crossvalidation]
            y_true_trainingset2 = y_true[crossvalidation*(i+1):len(x_train)]

            y_true_trainingset = y_true_trainingset1 + y_true_trainingset2

        x_train_trainingset = np.array(x_train_trainingset)
        y_true_trainingset = np.array(y_true_trainingset)

        x_train_trainingset = Normailize(x_train_trainingset)
        y_true_trainingset = Normailize(y_true_trainingset)

        for j in range(len(y_true_trainingset)):
            if ( y_true_trainingset[j][0] == 0 ):
                y_true_trainingset[j][0] = 0.1
            if ( y_true_trainingset[j][1] == 0 ):
                y_true_trainingset[j][1] = 0.1
```

รูปที่ 34: แบ่ง Train กับ Test set ของ Data ข้อสอง(ต่อ)

```
            if ( y_true_trainingset[j][0] == 1 ):
                y_true_trainingset[j][0] = 0.9
            if ( y_true_trainingset[j][1] == 1 ):
                y_true_trainingset[j][1] = 0.9

        x_train_testingset = np.array(x_train_testingset)
        y_true_testingset = np.array(y_true_testingset)

        nn = NeuralNetwork(x_train_trainingset, y_true_trainingset)

        x_train_testingset = Normailize(x_train_testingset)
        y_true_testingset = Normailize(y_true_testingset)

        for j in range(len(y_true_testingset)):
            if ( y_true_testingset[j][0] == 0 ):
                y_true_testingset[j][0] = 0.1
            if ( y_true_testingset[j][1] == 0 ):
                y_true_testingset[j][1] = 0.1
            if ( y_true_testingset[j][0] == 1 ):
                y_true_testingset[j][0] = 0.9
            if ( y_true_testingset[j][1] == 1 ):
                y_true_testingset[j][1] = 0.9

        # nn.addLayer(6)
        # nn.addLayer(3)
        # # nn.addLayer(1)
        # nn.addLayer(2)

        print("Fold " + str(i+1))
        nn.fit(1, 100, 5)
        nn.confusion_matrix(x_train_testingset, y_true_testingset)
        print("-----------------------------------------")
```

รูปที่ 35: แบ่ง Train กับ Test set ของ Data ข้อสอง(จบ)