

# MATLAB을 이용한 고강도강 겹치기 레이저 용접부의 모델링

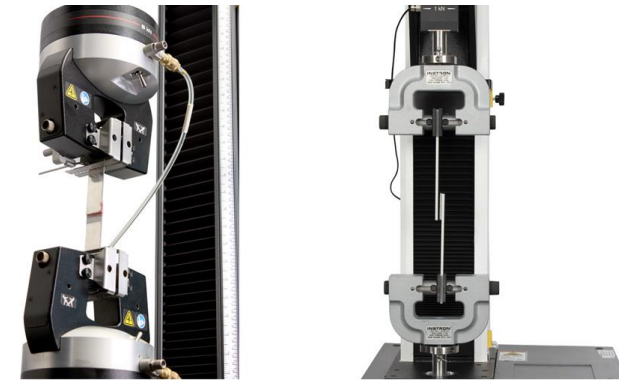
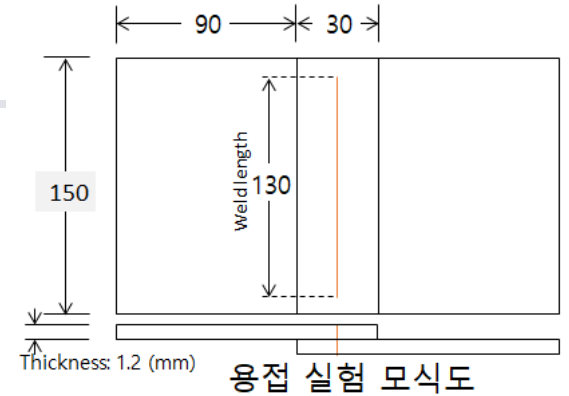
- SNN (shallow), DNN (deep) -

# 안 내

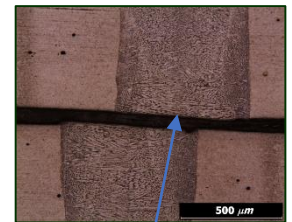
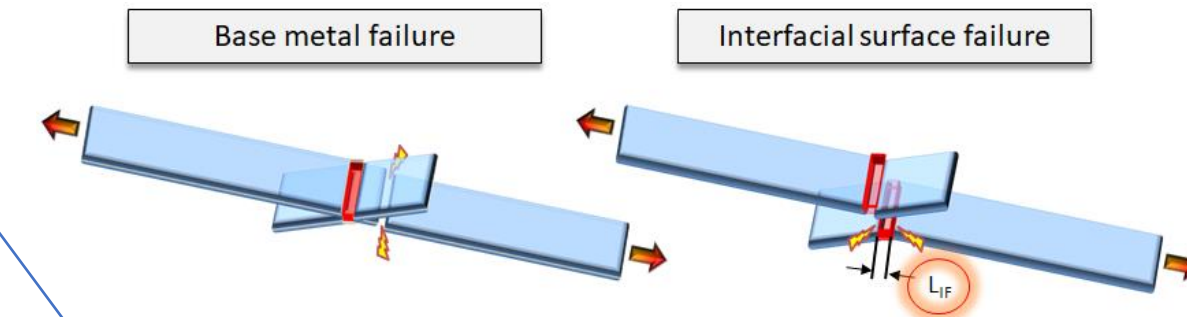
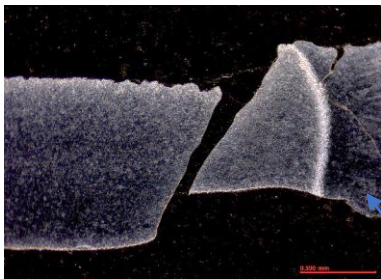
- 본 자료는 아래의 사람들이 만들었습니다.  
유현정 (Portland State University)  
김철희 (한국생산기술연구원, Portland State University)
- 예제 파일은 아래에서 받을 수 있습니다.  
<https://deepjoining.github.io/>
- 문의사항 및 의견: [deepjoining@gmail.com](mailto:deepjoining@gmail.com)
- 자료는 한국생산기술연구원 용접접합그룹 신입대학원생 교육자료입니다.  
일체의 다른 용도 사용을 금지합니다.

# 0. 풀어야 할 문제

- 소재: 인장강도 590~1500 MPa급 자동차용 강판  
(cf. 연강의 경우 인장강도 270~300 MPa)
- 용접방법: 레이저 겹치기 용접
- 용접부 시험방법: 인장-전단 강도 평가
- 품질판단 기준: 파단의 위치
- 모델링할 문제
  - \* 다양한 소재 조합 및 다양한 레이저 용접조건하에서
  - (1) 용접 후 용접 비드폭은 얼마인가? (회귀)
  - (2) 인장-전단 시험에서 강도는? (회귀)
  - (3) 인장-전단 시험에서 파단위치는? (분류)



인장-전단 시험



용접 계면파단

HPF2.0G  
용접부

# 1. 머신 러닝 모델 구축에 사용된 Input, Output parameter

- Input parameter

| No.             | 1~7                                     | 8~14                                    | 15            | 16             |
|-----------------|---|---|---------------|----------------|
| Input parameter | Chemical composition of the upper sheet | Chemical composition of the lower sheet | Welding speed | Focal position |

- Output parameter

|                  | Regression model                 |               | Classification model |
|------------------|----------------------------------|---------------|----------------------|
| Output parameter | Bead width at the faying surface | Fracture load | Fracture location    |

- Chemical compositions

| Base materials (thickness) | C     | Si    | Mn    | P     | S     | Cr    | B     |
|----------------------------|-------|-------|-------|-------|-------|-------|-------|
| 590 DP (1.2 mm)            | 0.078 | 0.363 | 1.808 | 0.011 | 0.001 | -     | -     |
| 780 DP (1.2 mm)            | 0.070 | 0.977 | 2.264 | 0.010 | 0.015 | -     | -     |
| 980 DP (1.2 mm)            | 0.170 | 1.340 | 2.000 | 0.016 | 0.001 | -     | -     |
| 1180 CP (1.2 mm)           | 0.110 | 0.110 | 2.790 | 0.019 | 0.004 | 1.040 | -     |
| 1500 HPF (1.1 mm)          | 0.216 | 0.240 | 1.255 | 0.002 | 0.002 | 0.001 | 0.003 |

# 1. 머신 러닝 모델 구축에 사용된 데이터

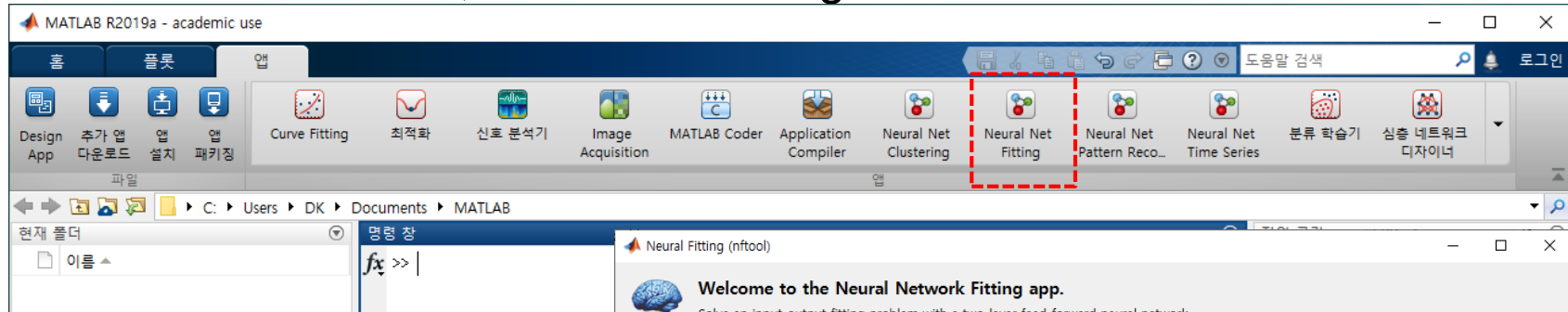
- Input parameter

- Output parameter

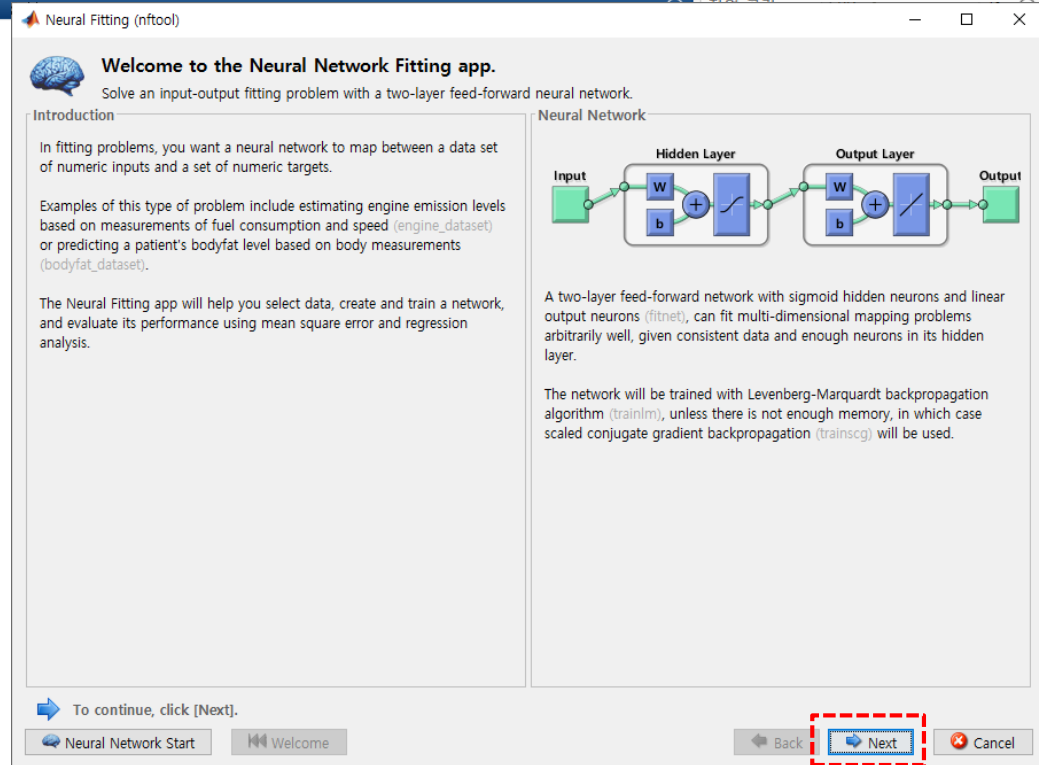
| Chemical composition of upper sheet |       |       |       |       |    |   | Chemical composition of lower sheet |       |       |       |       |    |   |           |            |            |             |             |
|-------------------------------------|-------|-------|-------|-------|----|---|-------------------------------------|-------|-------|-------|-------|----|---|-----------|------------|------------|-------------|-------------|
| C                                   | Si    | Mn    | P     | S     | Cr | B | C                                   | Si    | Mn    | P     | S     | Cr | B | Weldingsp | Focalposit | Bead width | Fracture lo | Fracture lo |
| 0.078                               | 0.363 | 1.808 | 0.011 | 0.001 | 0  | 0 | 0.078                               | 0.363 | 1.808 | 0.011 | 0.001 | 0  | 0 | 70        | 0          | 0.82       | 15659.47    | 0           |
| 0.078                               | 0.363 | 1.808 | 0.011 | 0.001 | 0  | 0 | 0.078                               | 0.363 | 1.808 | 0.011 | 0.001 | 0  | 0 | 60        | -5         | 0.87       | 16660.73    | 0           |
| 0.078                               | 0.363 | 1.808 | 0.011 | 0.001 | 0  | 0 | 0.078                               | 0.363 | 1.808 | 0.011 | 0.001 | 0  | 0 | 48        | -10        | 1.02       | 18593.63    | 0           |
| 0.078                               | 0.363 | 1.808 | 0.011 | 0.001 | 0  | 0 | 0.078                               | 0.363 | 1.808 | 0.011 | 0.001 | 0  | 0 | 37        | -15        | 1.33       | 18619.1     | 1           |
| 0.078                               | 0.363 | 1.808 | 0.011 | 0.001 | 0  | 0 | 0.078                               | 0.363 | 1.808 | 0.011 | 0.001 | 0  | 0 | 26        | -20        | 1.99       | 18859.83    | 1           |
| 0.078                               | 0.363 | 1.808 | 0.011 | 0.001 | 0  | 0 | 0.078                               | 0.363 | 1.808 | 0.011 | 0.001 | 0  | 0 | 20        | -25        | 2.02       | 18765.03    | 1           |
| 0.078                               | 0.363 | 1.808 | 0.011 | 0.001 | 0  | 0 | 0.07                                | 0.977 | 2.264 | 0.01  | 0.015 | 0  | 0 | 70        | 0          | 0.76       | 14681.73    | 0           |
| 0.078                               | 0.363 | 1.808 | 0.011 | 0.001 | 0  | 0 | 0.07                                | 0.977 | 2.264 | 0.01  | 0.015 | 0  | 0 | 60        | -5         | 0.81       | 15620.37    | 0           |
| 0.078                               | 0.363 | 1.808 | 0.011 | 0.001 | 0  | 0 | 0.07                                | 0.977 | 2.264 | 0.01  | 0.015 | 0  | 0 | 48        | -10        | 1.1        | 18561.8     | 1           |
| 0.078                               | 0.363 | 1.808 | 0.011 | 0.001 | 0  | 0 | 0.07                                | 0.977 | 2.264 | 0.01  | 0.015 | 0  | 0 | 40        | -10        | 1.09       | 18555.93    | 1           |
| 0.078                               | 0.363 | 1.808 | 0.011 | 0.001 | 0  | 0 | 0.07                                | 0.977 | 2.264 | 0.01  | 0.015 | 0  | 0 | 32        | -15        | 1.41       | 18899.8     | 1           |
| 0.078                               | 0.363 | 1.808 | 0.011 | 0.001 | 0  | 0 | 0.07                                | 0.977 | 2.264 | 0.01  | 0.015 | 0  | 0 | 24        | -20        | 2.01       | 18918.9     | 1           |
| 0.078                               | 0.363 | 1.808 | 0.011 | 0.001 | 0  | 0 | 0.17                                | 1.34  | 2     | 0.016 | 0.001 | 0  | 0 | 70        | 0          | 0.74       | 16050.63    | 0           |
| 0.078                               | 0.363 | 1.808 | 0.011 | 0.001 | 0  | 0 | 0.17                                | 1.34  | 2     | 0.016 | 0.001 | 0  | 0 | 60        | -5         | 0.86       | 17751.67    | 0           |

## 2. MATLAB에서 신경회로망 fitting (SNN)

1. Matlab 실행
2. Tab에서 앱으로 이동, Neural net fitting 선택

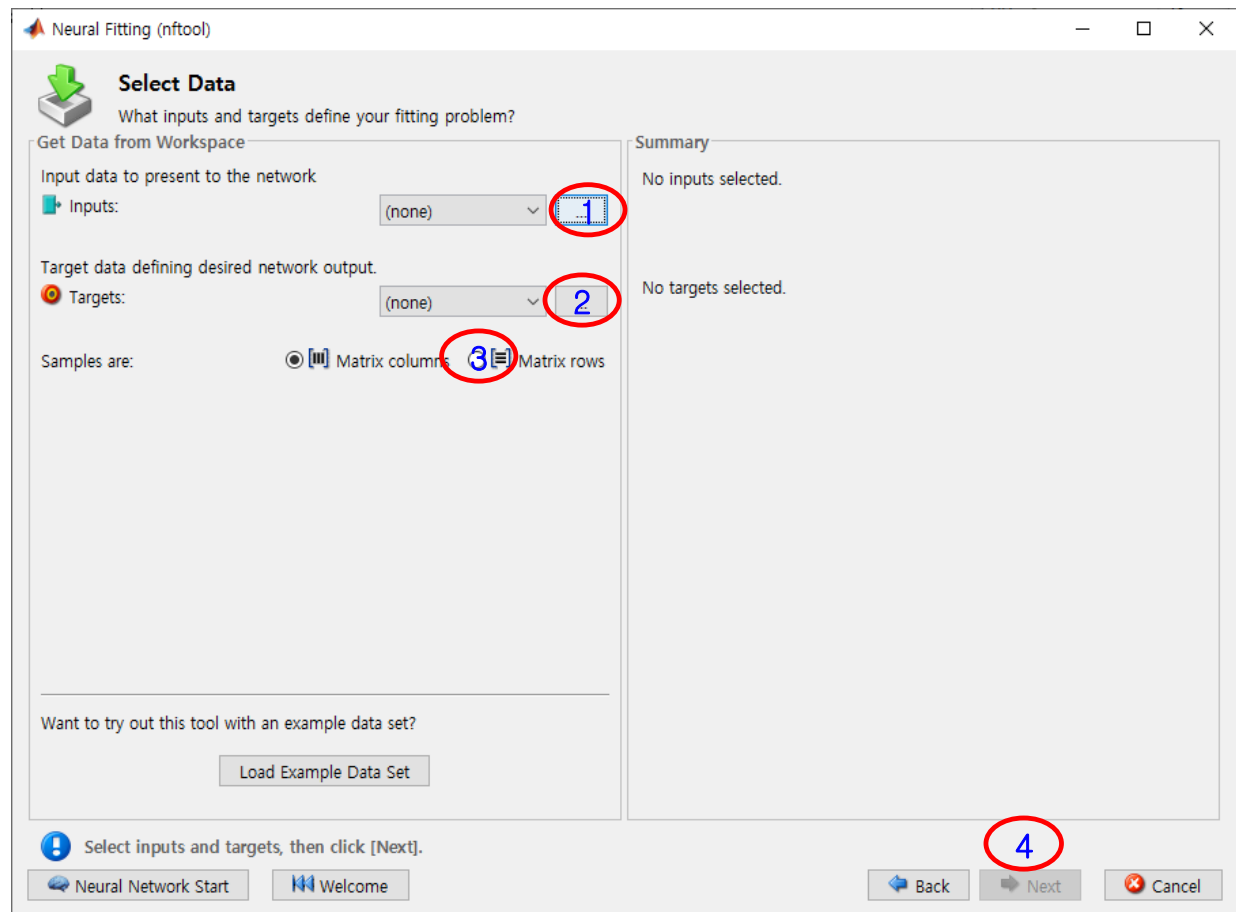


3. 새로 뜬 창에서 Next 클릭



## 2. MATLAB에서 신경회로망 fitting (SNN)

4. 1을 눌러서 input 데이터만 있는 파일을 불러옴
5. 2를 눌러서 output 데이터만 있는 파일을 불러옴
6. 3에서 Matrix rows 선택하고 4-Next 선택



## 2. MATLAB에서 신경회로망 fitting (SNN)

7. Validation과 testing data의 퍼센트 변경 가능
8. Next 로 넘어가면
9. Hidden layer의 노드 개수 변경 가능
10. Next로 넘어갈 것

Neural Fitting (nftool)

### Validation and Test Data

Set aside some samples for validation and testing.

Select Percentages

Randomly divide up the 90 samples:

- Training: 70% 62 samples
- Validation: 15% 14 samples
- Testing: 15% 14 samples

Explanation

Three Kinds of Samples:

- Training: These are presented to the network during training, and the network is adjusted according to its error.
- Validation: These are used to measure network generalization, and to halt training when generalization stops improving.
- Testing: These have no effect on training and so provide an independent measure of network performance during and after training.

Restore Defaults

Change percentages if desired, then click [Next] to continue.

Neural Network Start Welcome Back Next Cancel

Neural Fitting (nftool)

### Network Architecture

Set the number of neurons in the fitting network's hidden layer.

Hidden Layer

Define a fitting neural network. (fitnet)

Number of Hidden Neurons: 30

Recommendation

Return to this panel and change the number of neurons if the network does not perform well after training.

Restore Defaults

Neural Network

Input: 16

Hidden Layer: 30

Output Layer: 1

Output: 1

Change settings if desired, then click [Next] to continue.

Neural Network Start Welcome Back Next Cancel



# 2. MATLAB에서 신경회로망 fitting (SNN)

11. Train을 눌러서 학습

12. 결정계수 확인

**Neural Fitting (nftool)**

**Train Network**  
Train the network to fit the inputs and targets.

Choose a training algorithm:  
Levenberg-Marquardt

This algorithm typically requires more memory but less time. Training automatically stops when generalization stops improving, as indicated by an increase in the mean square error of the validation samples.

Train using Levenberg-Marquardt. (trainlm)

**Retrain**

**Results**

|             | Samples | MSE        | R          |
|-------------|---------|------------|------------|
| Training:   | 62      | 1.28744e-2 | 9.62310e-1 |
| Validation: | 14      | 1.74353e-2 | 9.40218e-1 |
| Testing:    | 14      | 2.84707e-2 | 9.25881e-1 |

Plot Fit Plot Error Histogram **Plot Regression**

Open plot window.

**Notes**

1

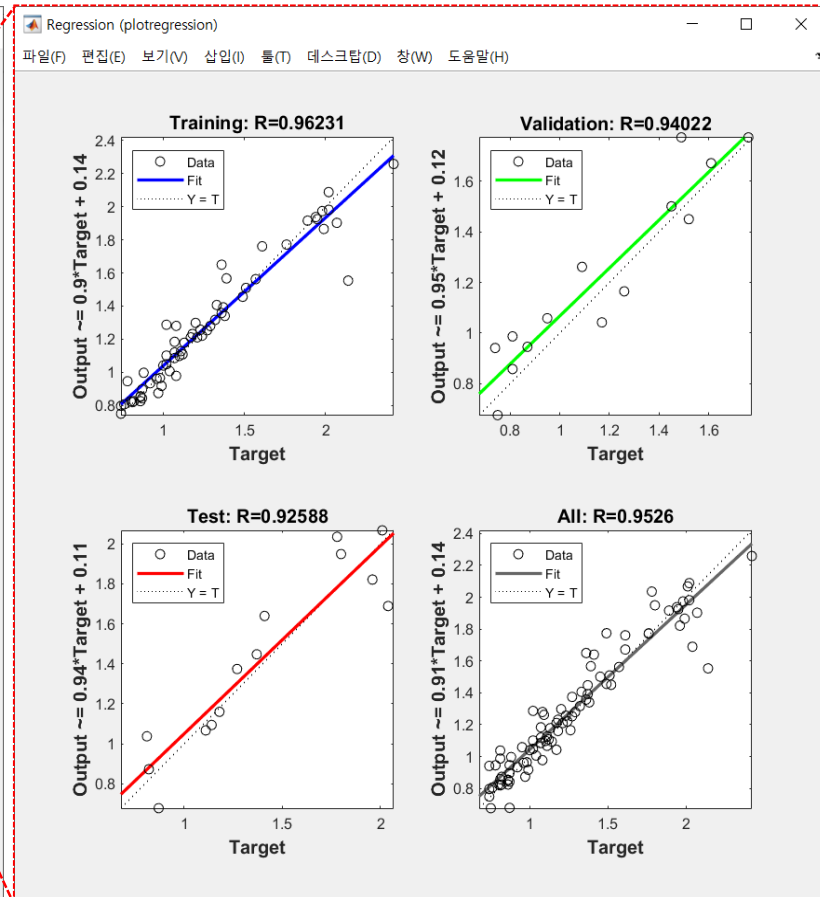
Training multiple times will generate different results due to different initial conditions and sampling.

Mean Squared Error is the average squared difference between outputs and targets. Lower values are better. Zero means no error.

Regression R Values measure the correlation between outputs and targets. An R value of 1 means a close relationship, 0 a random relationship.

Open a plot, retrain, or click [Next] to continue.

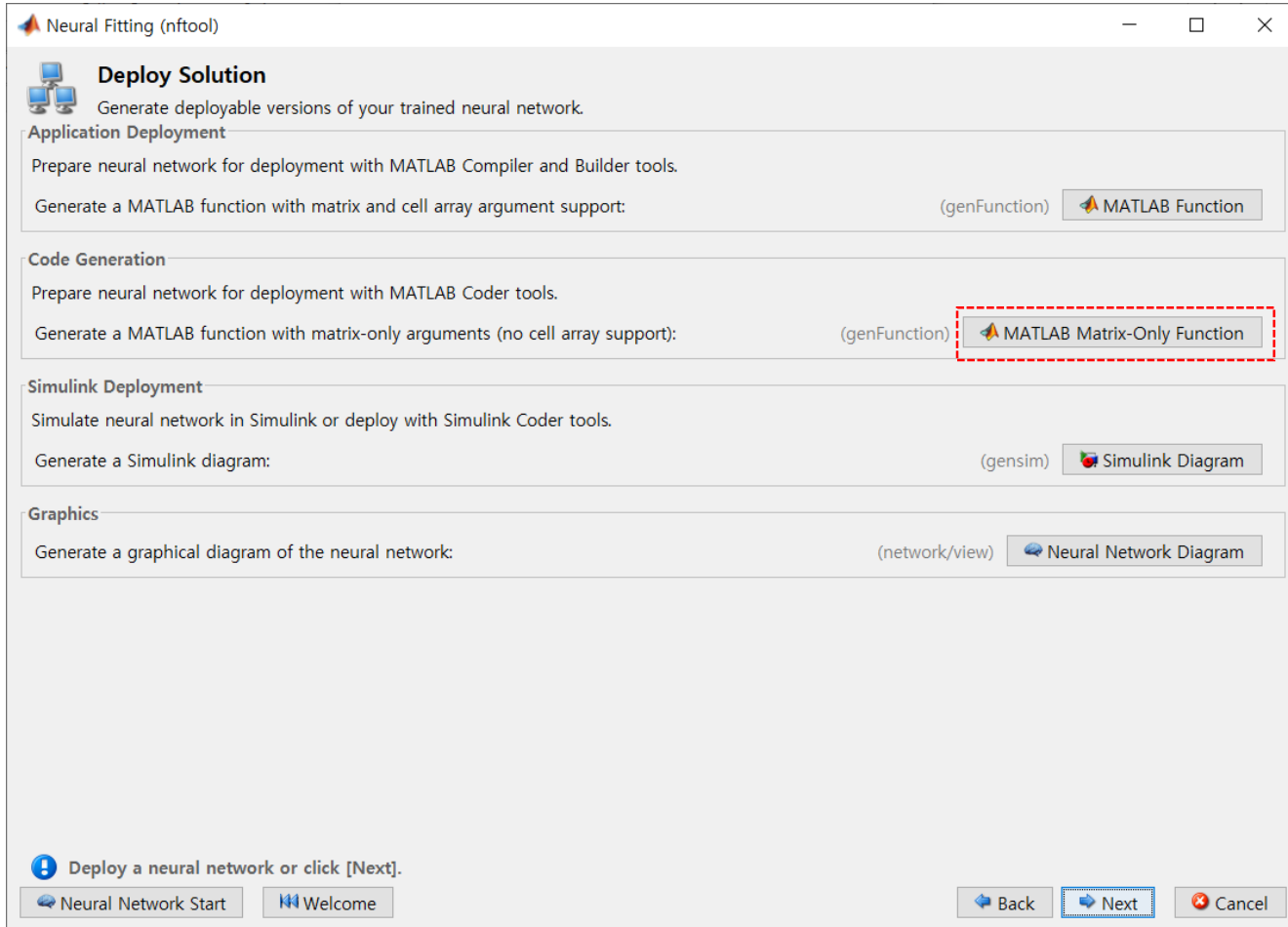
Neural Network Start Welcome Back Next Cancel



## 2. MATLAB에서 신경회로망 fitting (SNN)

11. Next를 두 번 클릭하면 아래와 같은 화면이 나옴

12. MATLAB Matrix-Only Function 선택



## 2. MATLAB에서 신경회로망 fitting (SNN)

13. 아래와 같은 편집기가 생성되며, 그 수식 위에  $y1=myNeuralNetworkFunction(x1)$ 을 작성해줘야 하며 괄호 안에 x1 대신 작업 공간에서 확인할 수 있는 실제 사용된 input data의 이름을 작성

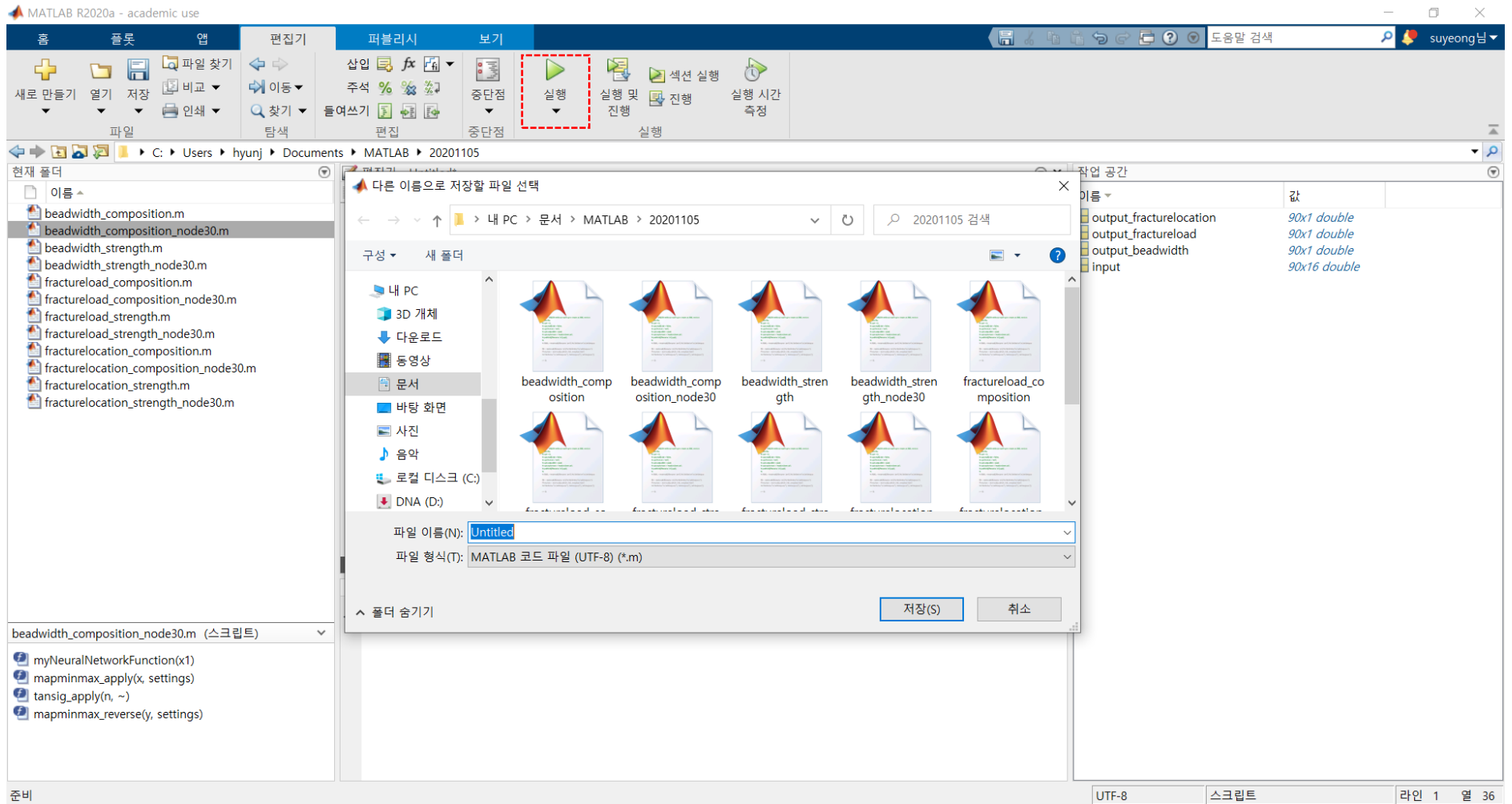
The screenshot displays the MATLAB R2020a interface. The main window is the Function Editor for 'myNeuralNetworkFunction'. The function signature is `y1 = myNeuralNetworkFunction(input)`. The workspace on the right shows the following variables:

| 이름                      | 값            |
|-------------------------|--------------|
| output_fracturelocation | 90x1 double  |
| output_fractureload     | 90x1 double  |
| output_beadwidth        | 90x1 double  |
| input                   | 90x16 double |

The command window shows the command `fx >>`.

## 2. MATLAB에서 신경회로망 fitting (SNN)

14. 실행을 클릭하면 다음과 같이 코드 파일을 저장하라는 화면이 생성



## 2. MATLAB에서 신경회로망 fitting (SNN)

### 15. y1이라는 SNN 모델을 통해 예측된 값이 생성됨

The screenshot displays the MATLAB R2020a environment. The main window shows a variable viewer for 'y1', which is a 90x1 double array. The values are listed in the '명령 창' (Command Window) below the array. The workspace window on the right lists several variables, including 'y1', 'output\_fracturelocation', 'output\_fractureload', 'output\_beadwidth', and 'input'.

| 번호 | 값      |
|----|--------|
| 1  | 0.5757 |
| 2  | 0.7062 |
| 3  | 0.9521 |
| 4  | 1.4238 |
| 5  | 1.9125 |
| 6  | 2.0602 |
| 7  | 0.8045 |
| 8  | 0.8469 |
| 9  | 0.9631 |
| 10 | 1.0342 |
| 11 | 1.4315 |
| 12 | 2.0139 |
| 13 | 0.7763 |
| 14 | 0.8977 |
| 15 | 1.0647 |
| 16 | 1.1031 |

명령 창

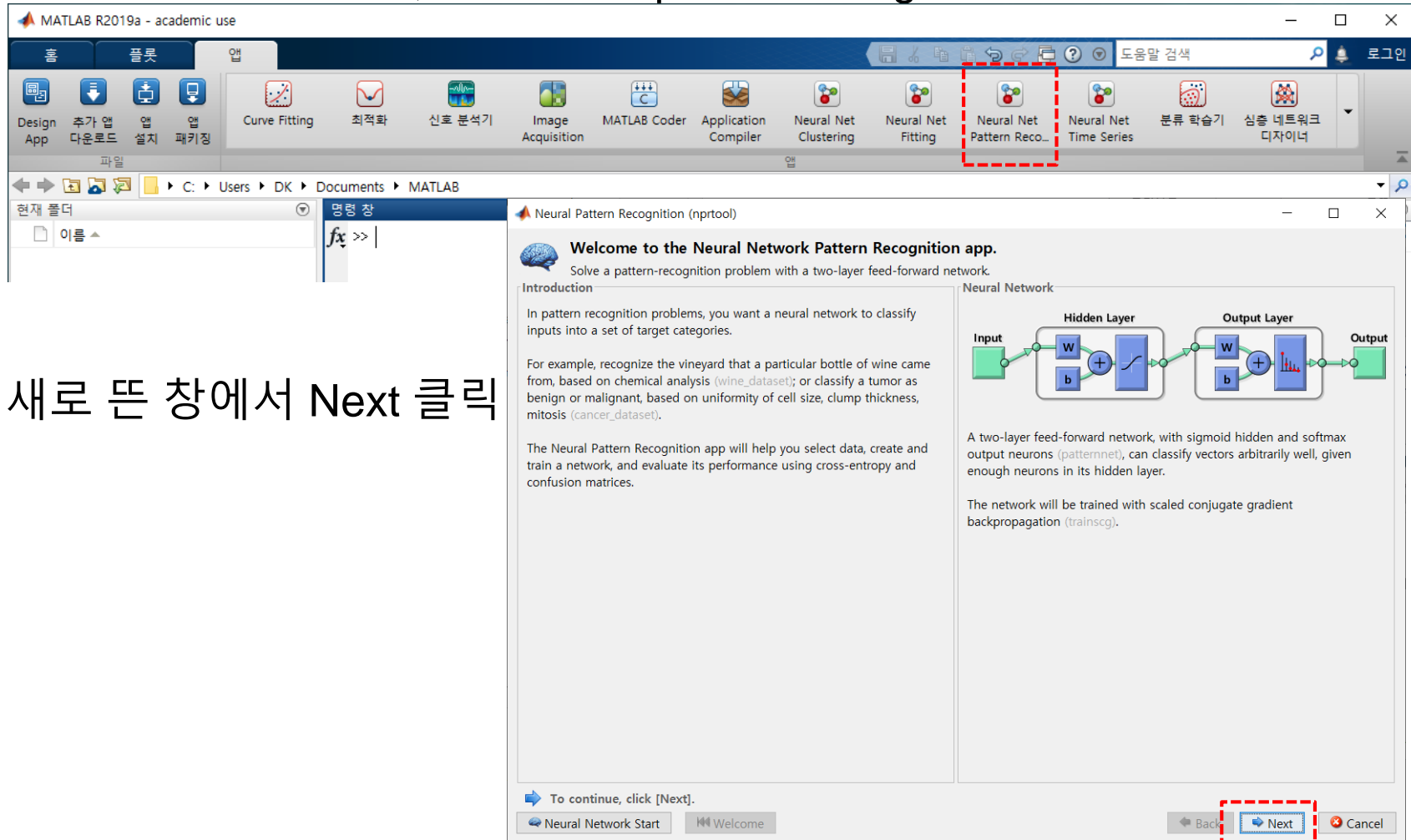
```
2.4253  
0.8712  
0.9533  
1.1326  
1.5712  
1.8197  
1.8301
```

작업 공간

| 이름                      | 값            |
|-------------------------|--------------|
| y1                      | 90x1 double  |
| output_fracturelocation | 90x1 double  |
| output_fractureload     | 90x1 double  |
| output_beadwidth        | 90x1 double  |
| input                   | 90x16 double |

# 3. MATLAB에서 신경회로망 분류

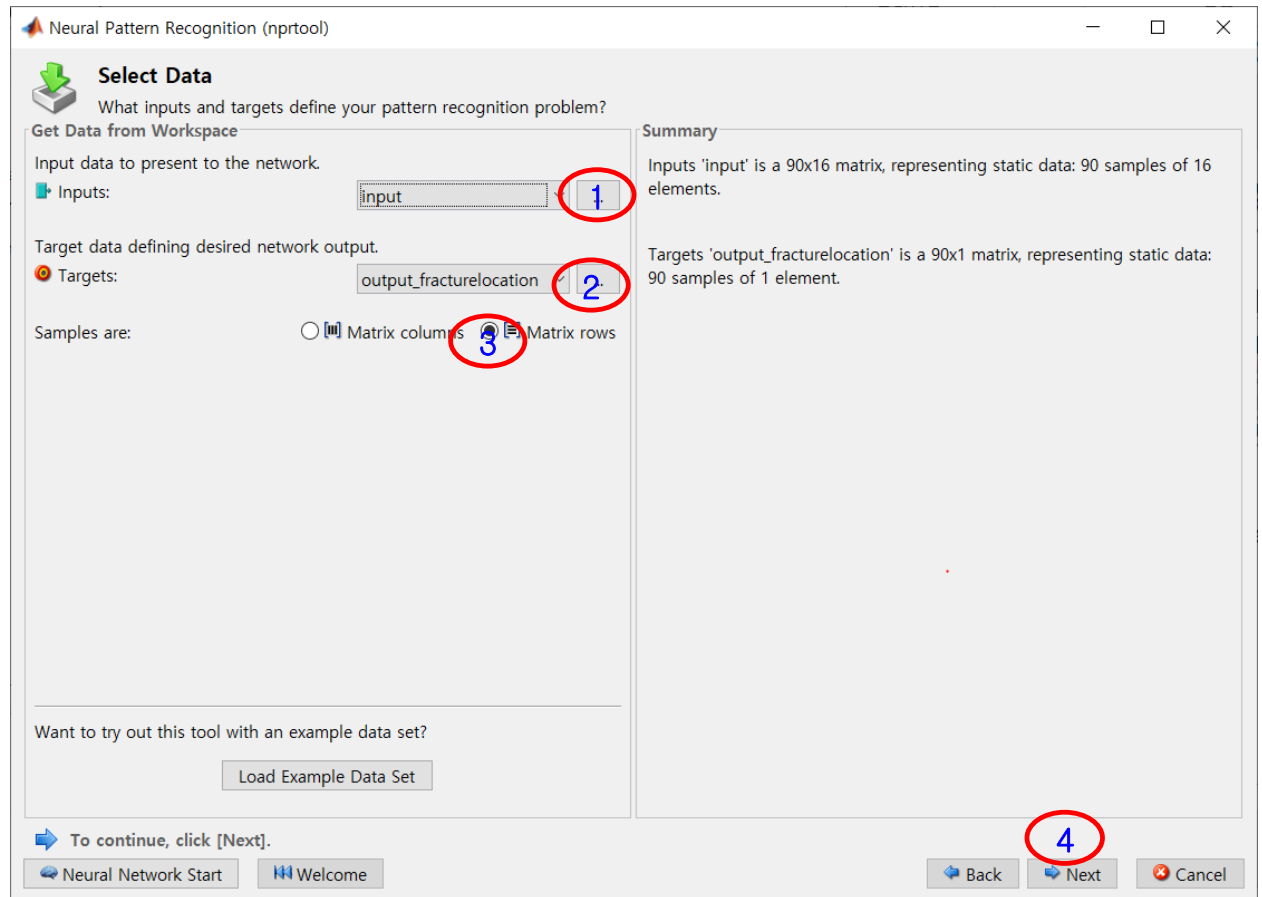
1. Matlab 실행
2. Tab에서 앱으로 이동, Neural net pattern recognition 선택



3. 새로 뜬 창에서 Next 클릭

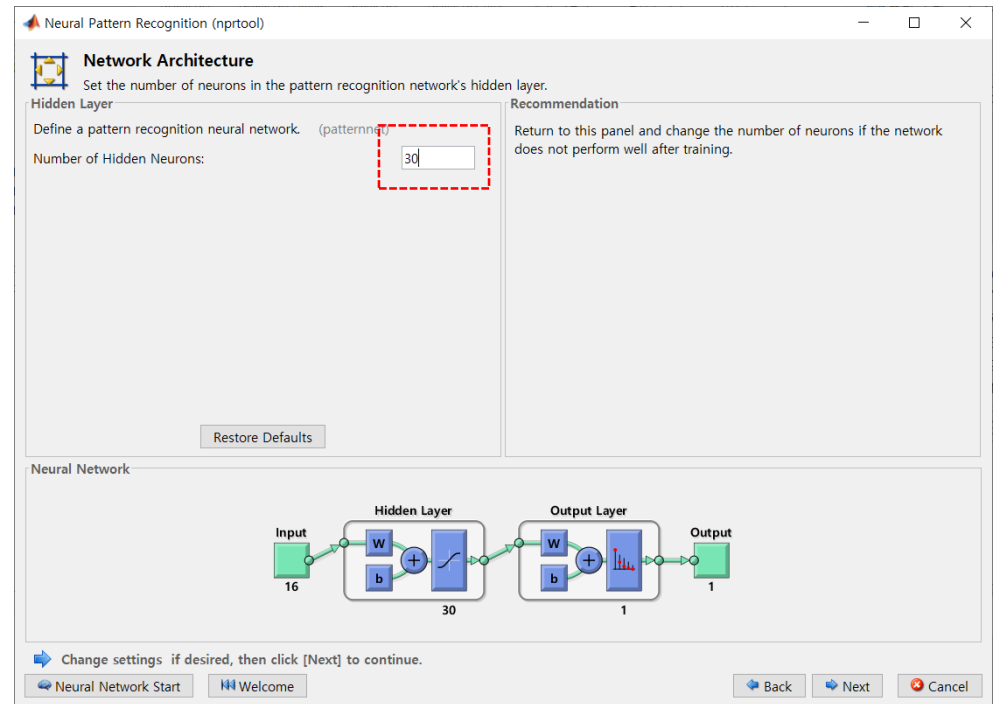
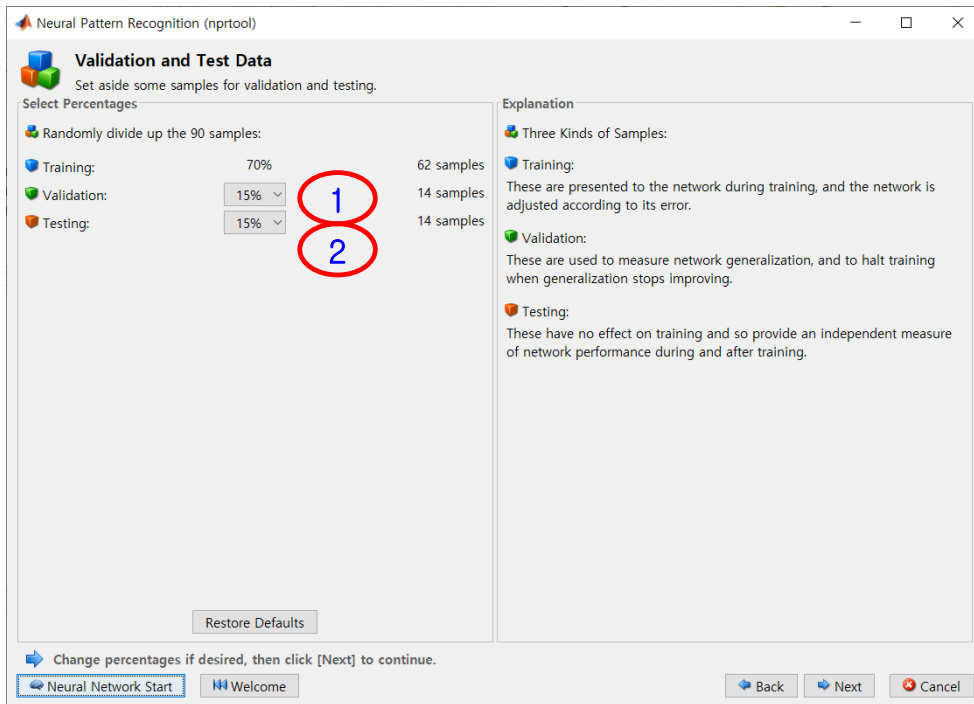
### 3. MATLAB에서 신경회로망 분류

4. 1을 눌러서 input 데이터만 있는 파일을 불러옴
5. 2를 눌러서 output 데이터만 있는 파일을 불러옴
6. 3에서 Matrix rows 선택하고 4-Next 선택



### 3. MATLAB에서 신경회로망 분류

- 7. Validation과 testing data의 퍼센트 변경 가능
- 8. Next 로 넘어가면
- 9. Hidden layer의 노드 개수 변경 가능
- 10. Next로 넘어갈 것





# 3. MATLAB에서 신경회로망 분류

11. Train을 눌러서 학습

12. 오차율 확인

The screenshot shows the MATLAB Neural Network Designer interface. The 'Train Network' window is active, displaying training results. A red circle highlights the 'Retrain' button, and a red dashed box highlights the 'Plot Confusion' button. The 'Results' table shows training, validation, and testing metrics. The 'Confusion (plotconfusion)' window displays four confusion matrices: Training, Validation, Test, and All.

**Train Network Results**

|             | Samples | CE         | %E         |
|-------------|---------|------------|------------|
| Training:   | 62      | 1.88525e-0 | 0          |
| Validation: | 14      | 4.47216e-0 | 0          |
| Testing:    | 14      | 4.74828e-0 | 7.14285e-0 |

**Confusion Matrices**

**Training Confusion Matrix**

| Output Class \ Target Class | 0          | 1          | Accuracy    |
|-----------------------------|------------|------------|-------------|
| 0                           | 28 (45.2%) | 0 (0.0%)   | 100% (0.0%) |
| 1                           | 0 (0.0%)   | 34 (54.8%) | 100% (0.0%) |
| Overall                     | 100%       | 100%       | 100%        |

**Validation Confusion Matrix**

| Output Class \ Target Class | 0         | 1         | Accuracy    |
|-----------------------------|-----------|-----------|-------------|
| 0                           | 8 (57.1%) | 0 (0.0%)  | 100% (0.0%) |
| 1                           | 0 (0.0%)  | 6 (42.9%) | 100% (0.0%) |
| Overall                     | 100%      | 100%      | 100%        |

**Test Confusion Matrix**

| Output Class \ Target Class | 0             | 1           | Accuracy      |
|-----------------------------|---------------|-------------|---------------|
| 0                           | 6 (42.9%)     | 0 (0.0%)    | 100% (0.0%)   |
| 1                           | 1 (7.1%)      | 7 (50.0%)   | 87.5% (12.5%) |
| Overall                     | 85.7% (14.3%) | 100% (0.0%) | 92.9% (7.1%)  |

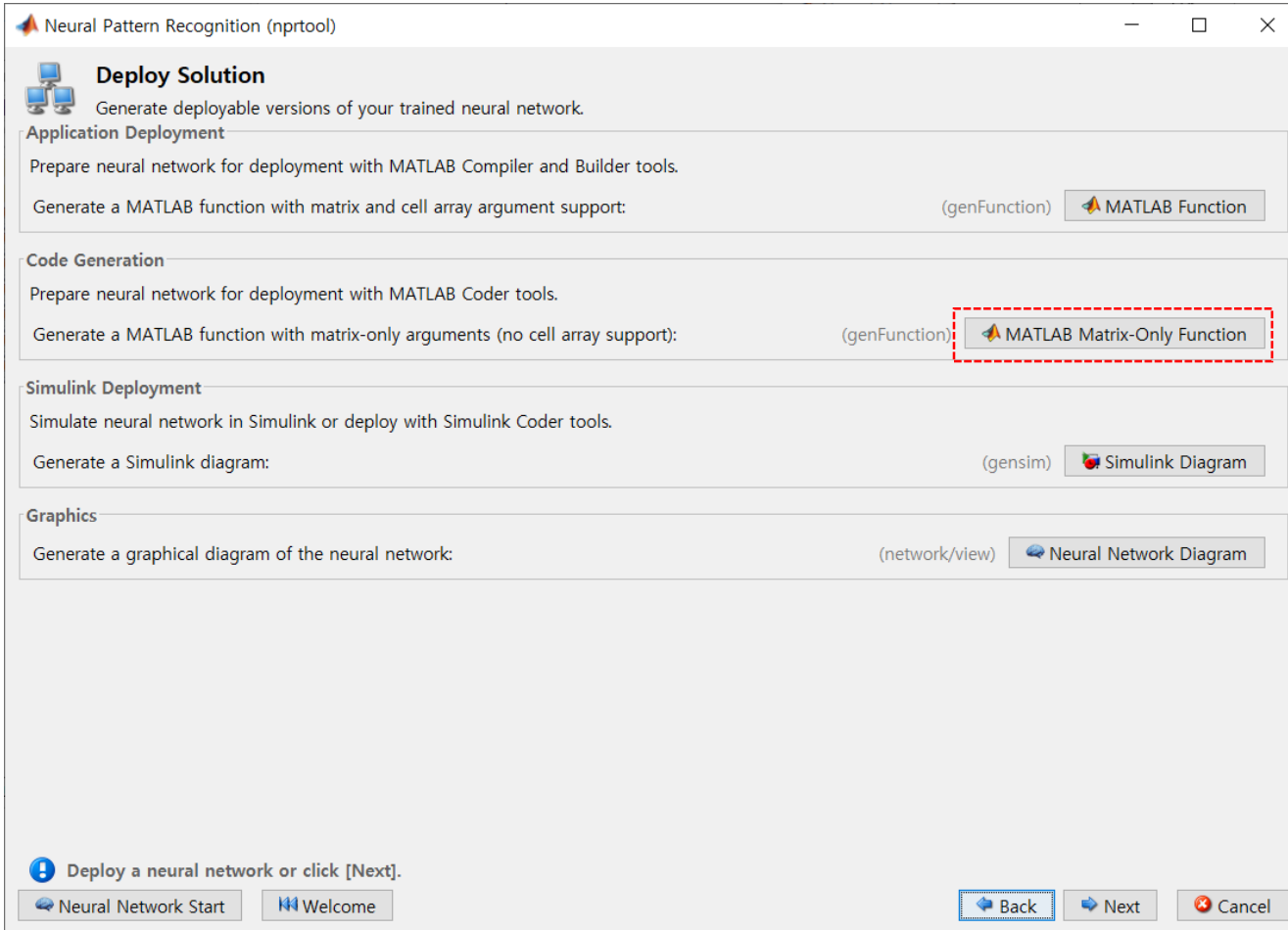
**All Confusion Matrix**

| Output Class \ Target Class | 0            | 1           | Accuracy     |
|-----------------------------|--------------|-------------|--------------|
| 0                           | 42 (46.7%)   | 0 (0.0%)    | 100% (0.0%)  |
| 1                           | 1 (1.1%)     | 47 (52.2%)  | 97.9% (2.1%) |
| Overall                     | 97.7% (2.3%) | 100% (0.0%) | 98.9% (1.1%) |

### 3. MATLAB에서 신경회로망 분류

11. Next를 두 번 클릭하면 아래와 같은 화면이 나옴

12. MATLAB Matrix-Only Function 선택



# 3. MATLAB에서 신경회로망 분류

13. 아래와 같은 편집기가 생성되며, 그 수식 위에  $y1=myNeuralNetworkFunction(x1)$ 을 작성해줘야 하며 괄호 안에  $x1$  대신 작업 공간에서 확인할 수 있는 실제 사용된 input data의 이름을 작성

The screenshot displays the MATLAB R2020a editor with a script titled "Untitled\*" open. The script content is as follows:

```
1 y1 = myNeuralNetworkFunction( input)
2
3 function [y1] = myNeuralNetworkFunction(x1)
4
5 % MYNEURALNETWORKFUNCTION neural network simulation function.
6
7 % Auto-generated by MATLAB, 27-Dec-2020 18:26:58.
8
9 % [y1] = myNeuralNetworkFunction(x1) takes these arguments:
10 % x = 0x16 matrix, input #1
11 % and returns:
12 % y = 0x1 matrix, output #1
13 % where 0 is the number of samples.
14
15 %#ok<<RPMT0>
16
17 % ===== NEURAL NETWORK CONSTANTS =====
18
19 % Input 1
20 x1 = step1_offset = [0.07;0.11;1.255;0.002;0.001;0.0;0.07;0.11;1.255;0.002;0.001;0.0;10;2.5
```

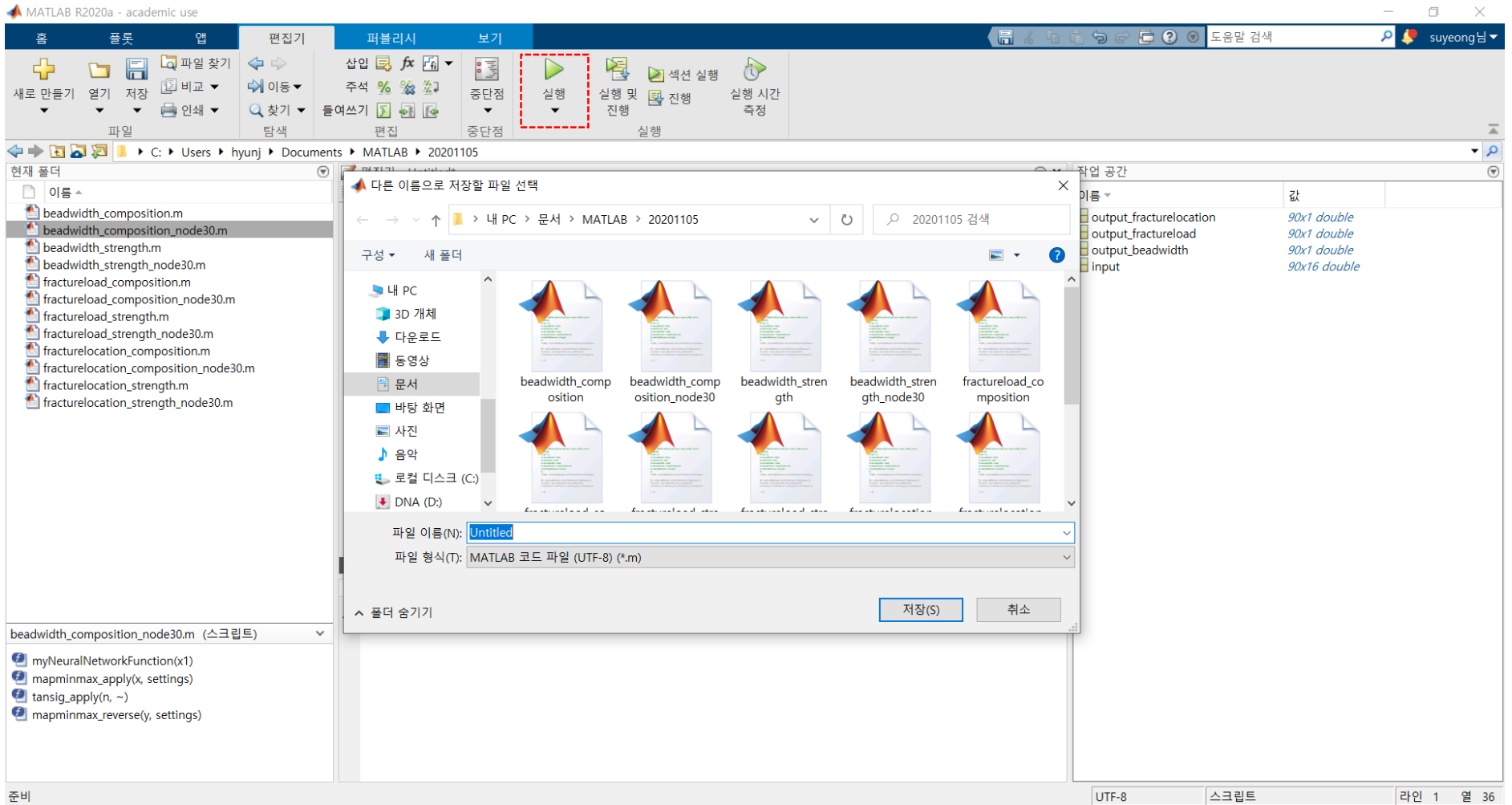
The workspace (작업 공간) on the right lists the following variables:

| 이름                      | 값            |
|-------------------------|--------------|
| output_fracturelocation | 90x1 double  |
| output_fractureload     | 90x1 double  |
| output_beadwidth        | 90x1 double  |
| input                   | 90x16 double |

The Command Window (명령 창) at the bottom shows the prompt `fx >>`.

# 3. MATLAB에서 신경회로망 분류

## 14. 실행을 클릭하면 다음과 같이 코드 파일을 저장하라는 화면이 생성



# 3. MATLAB에서 신경회로망 분류

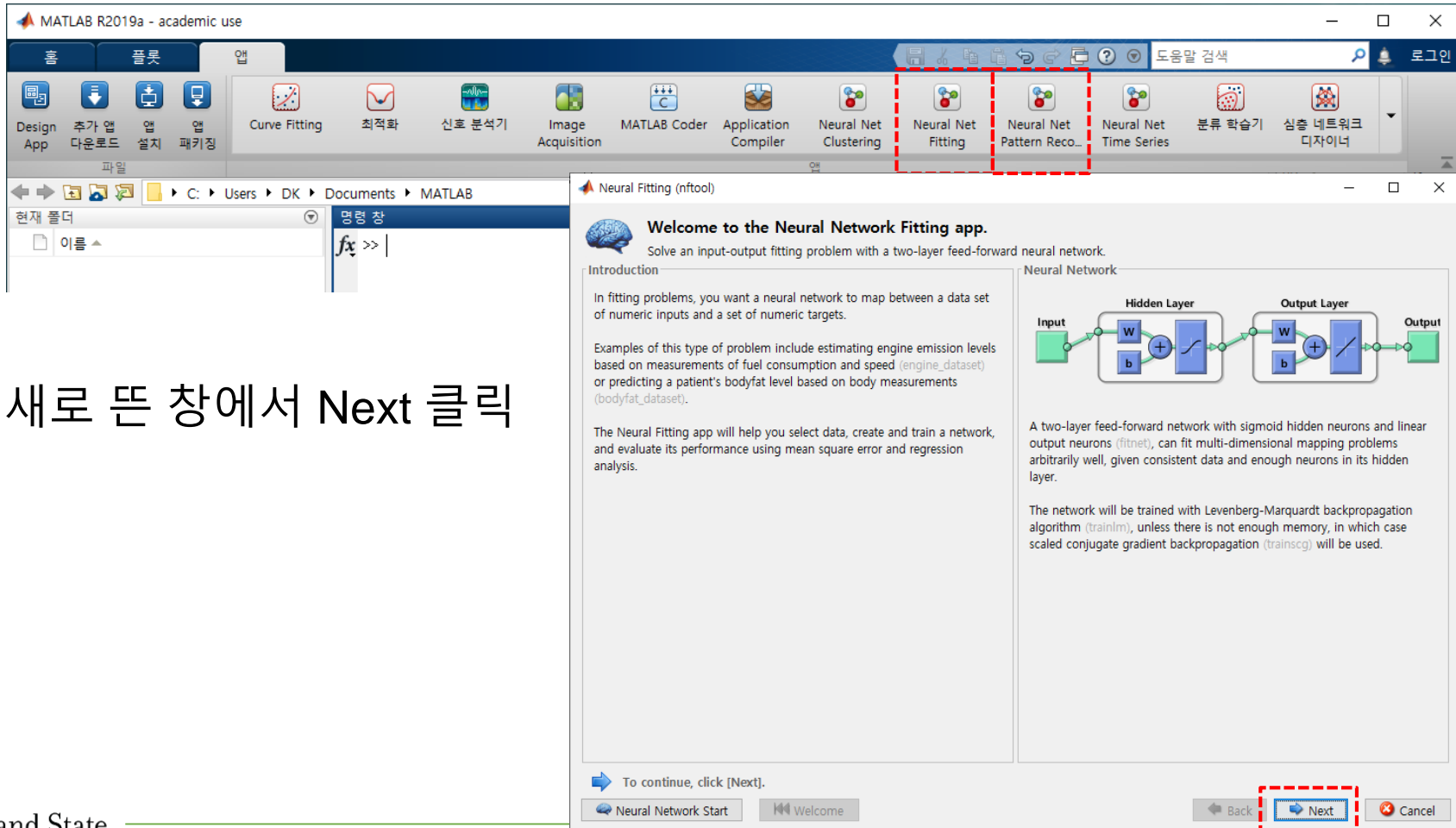
## 15. y1이라는 SNN 모델을 통해 예측된 값이 생성됨

The screenshot shows the MATLAB R2020a interface. The main window displays a variable editor for 'y1', which is a 90x1 double array. The values are listed in a table below. The workspace window on the right shows the variable 'y1' and other variables like 'output\_fracturelocation', 'output\_fractureload', 'output\_beadwidth', and 'input'.

|    | 1          | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|----|------------|---|---|---|---|---|---|---|---|----|
| 1  | 6.9323e-08 |   |   |   |   |   |   |   |   |    |
| 2  | 7.8357e-07 |   |   |   |   |   |   |   |   |    |
| 3  | 0.0029     |   |   |   |   |   |   |   |   |    |
| 4  | 0.9999     |   |   |   |   |   |   |   |   |    |
| 5  | 1.0000     |   |   |   |   |   |   |   |   |    |
| 6  | 1.0000     |   |   |   |   |   |   |   |   |    |
| 7  | 9.3838e-07 |   |   |   |   |   |   |   |   |    |
| 8  | 0.0026     |   |   |   |   |   |   |   |   |    |
| 9  | 0.9998     |   |   |   |   |   |   |   |   |    |
| 10 | 1.0000     |   |   |   |   |   |   |   |   |    |
| 11 | 1.0000     |   |   |   |   |   |   |   |   |    |
| 12 | 1.0000     |   |   |   |   |   |   |   |   |    |
| 13 | 2.9148e-07 |   |   |   |   |   |   |   |   |    |
| 14 | 1.3990e-04 |   |   |   |   |   |   |   |   |    |
| 15 | 0.9813     |   |   |   |   |   |   |   |   |    |
| 16 | 0.9997     |   |   |   |   |   |   |   |   |    |

# 4. MATLAB에서 신경회로망 fitting (DNN)

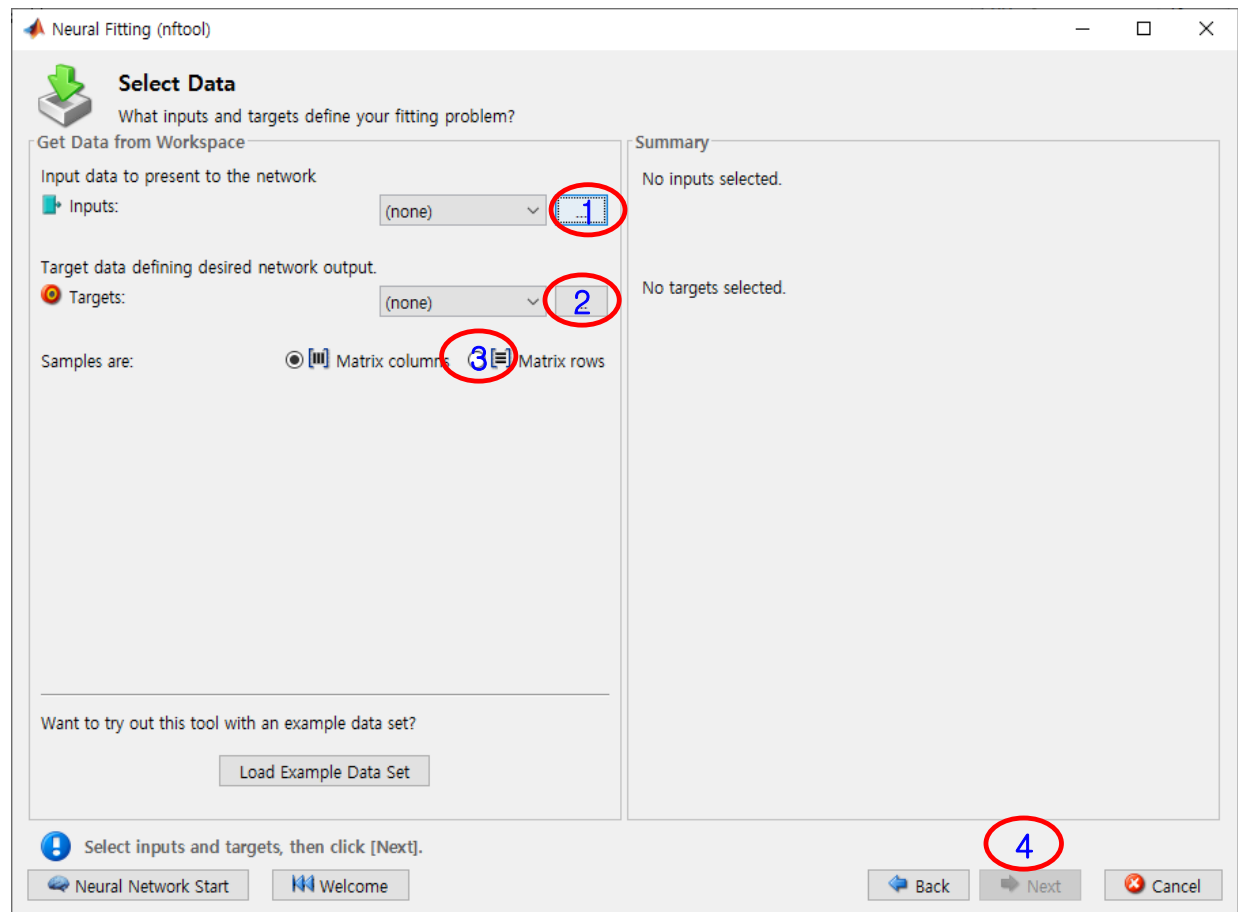
1. Matlab 실행
2. Tab에서 앱으로 이동, 회귀 문제의 경우 Neural net fitting을 분류 문제의 경우 Neural net pattern recognition 선택



3. 새로 뜬 창에서 Next 클릭

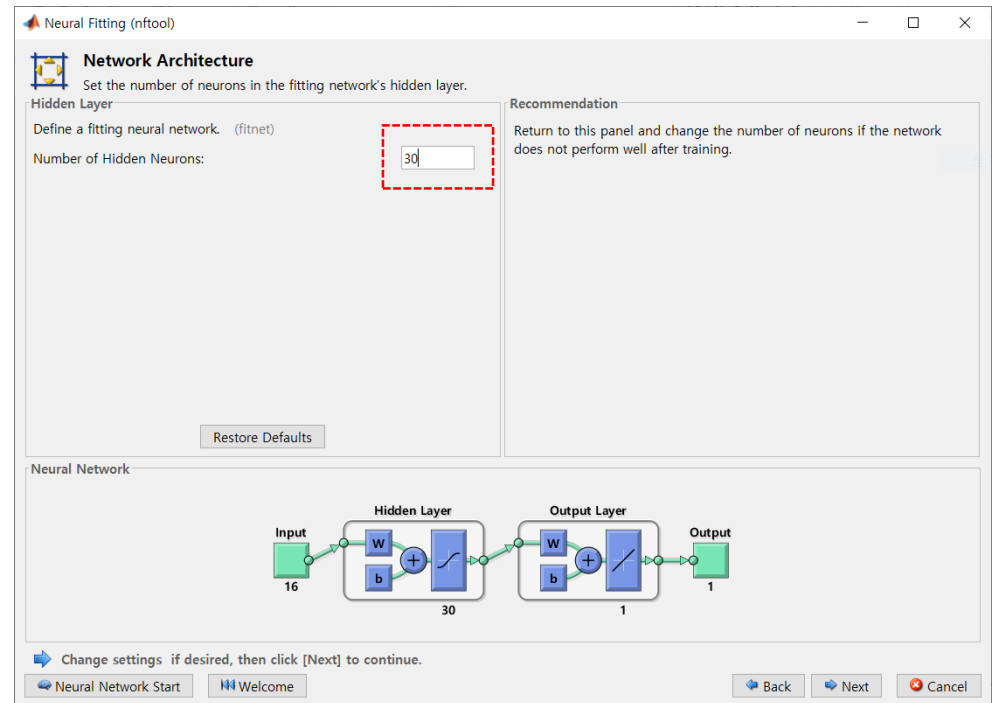
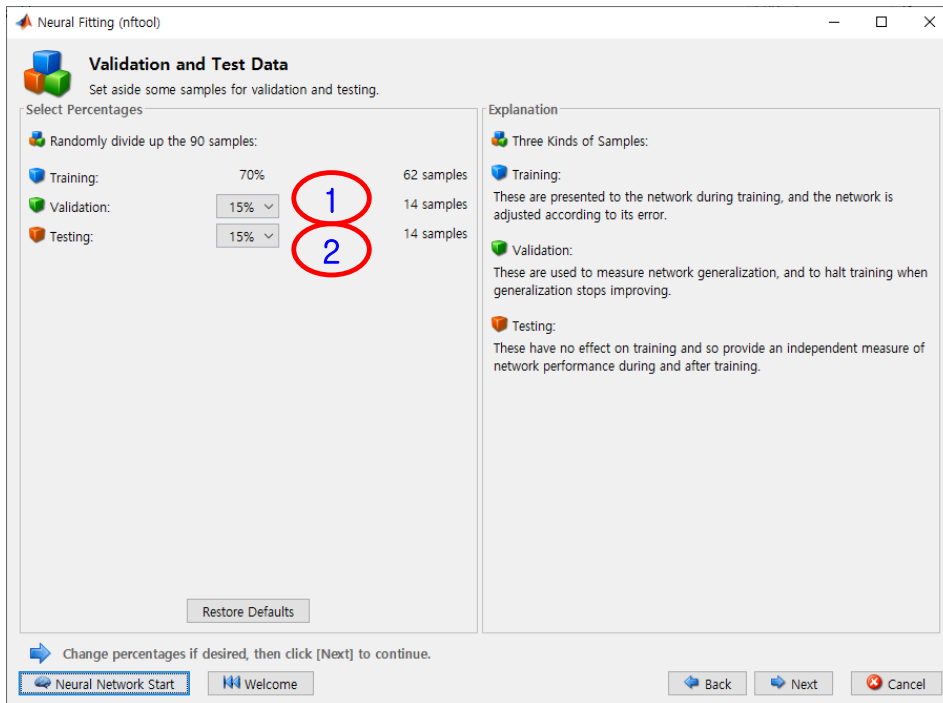
## 4. MATLAB에서 신경회로망 fitting (DNN)

- 1을 눌러서 input 데이터만 있는 파일을 불러옴
- 2를 눌러서 output 데이터만 있는 파일을 불러옴
- 3에서 Matrix rows 선택하고 4-Next 선택



# 4. MATLAB에서 신경회로망 fitting (DNN)

7. Validation과 testing data의 퍼센트 변경 가능
8. Next 로 넘어가면
9. Hidden layer의 노드 개수 변경 가능
10. Next로 넘어갈 것





# 4. MATLAB에서 신경회로망 fitting (DNN)

11. Train을 눌러서 학습

12. 결정계수 확인

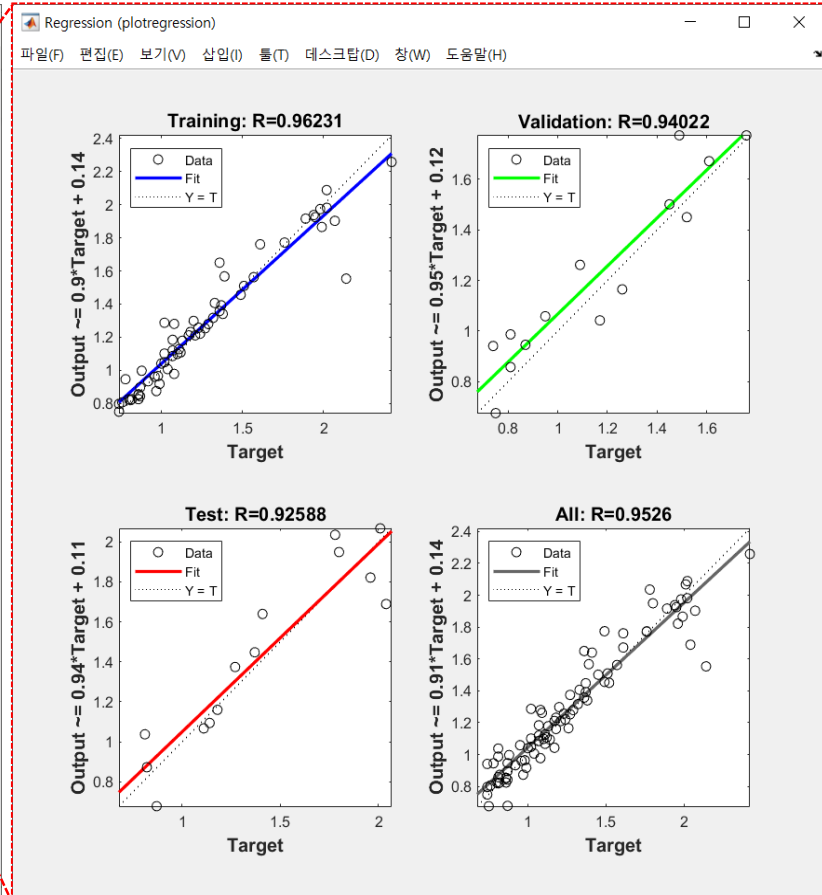
The screenshot shows the 'Neural Fitting (nftool)' window. The 'Train Network' section is active, with 'Levenberg-Marquardt' selected as the training algorithm. A red circle highlights the 'Retrain' button, with the number '1' inside it. The 'Results' table shows the following data:

|             | Samples | MSE        | R          |
|-------------|---------|------------|------------|
| Training:   | 62      | 1.28744e-2 | 9.62310e-1 |
| Validation: | 14      | 1.74353e-2 | 9.40218e-1 |
| Testing:    | 14      | 2.84707e-2 | 9.25881e-1 |

The 'Notes' section contains the following text:

- Training multiple times will generate different results due to different initial conditions and sampling.
- Mean Squared Error is the average squared difference between outputs and targets. Lower values are better. Zero means no error.
- Regression R Values measure the correlation between outputs and targets. An R value of 1 means a close relationship, 0 a random relationship.

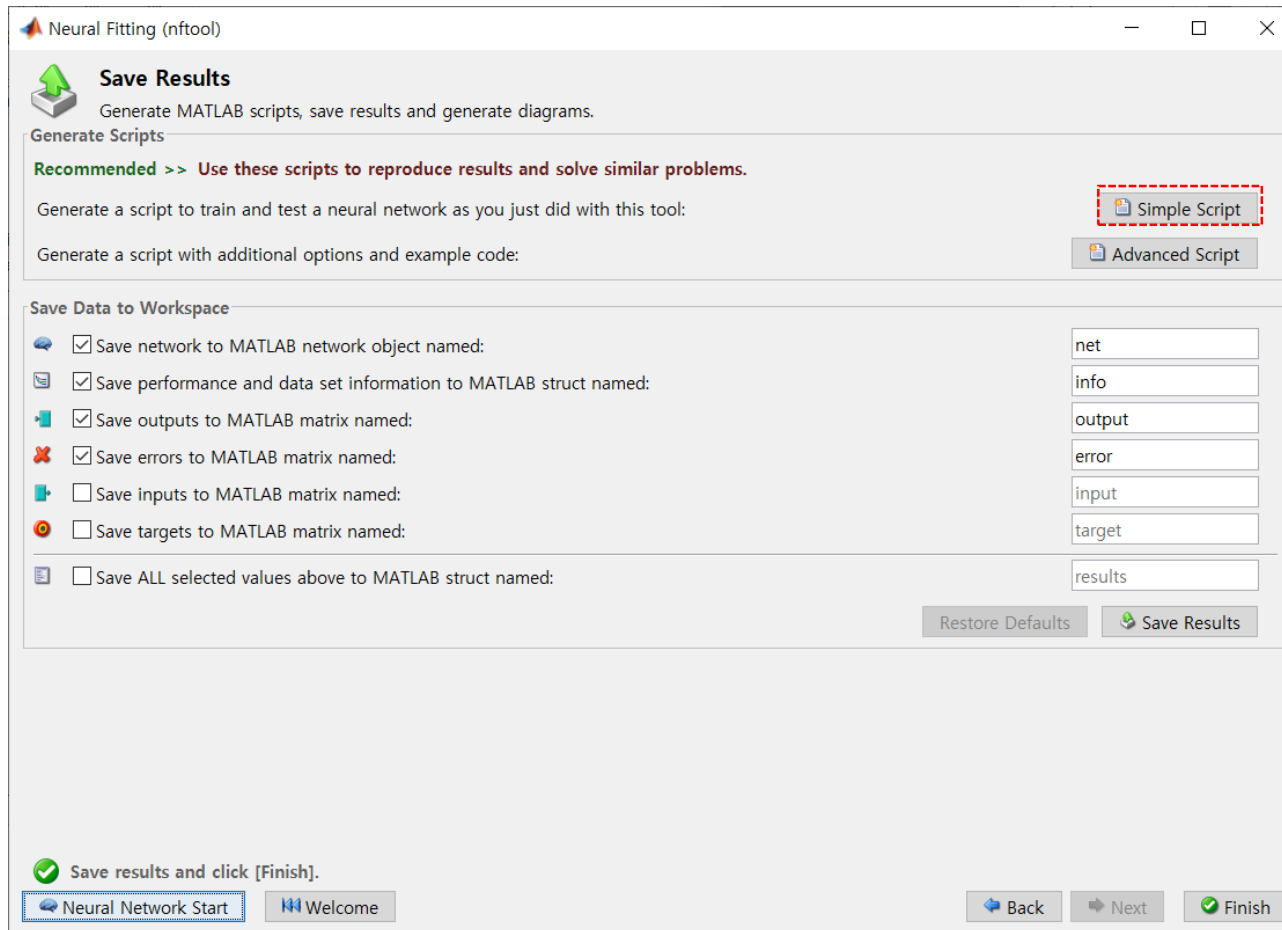
Buttons for 'Plot Fit', 'Plot Error Histogram', and 'Plot Regression' are visible. A red dashed box highlights the 'Plot Regression' button, with a tooltip that says 'Open plot window.'



## 4. MATLAB에서 신경회로망 fitting (DNN)

13. Next를 세 번 클릭하면 아래와 같은 화면이 나옴

14. Simple script 선택



# 4. MATLAB에서 신경회로망 fitting (DNN)

## 15. 아래와 같은 편집기가 생성됨

MATLAB R2020a - academic use

Home | Plots | Apps | Edit | Publish | View

새로 만들기 | 열기 | 저장 | 비교 | 이동 | 중지점 | 실행 | 실행 및 진행 | 실행 시간 측정

파일 | 탐색 | 편집 | 중지점 | 실행

C:\Users\hyunj\Documents\MATLAB

현재 폴더

- 이름 ^
- 20201105
- 20201229
- Classification\_yfit.m
- NeuralNetworkFunction.m
- Regression\_yfit.m
- testingdata.m

편집기 - Untitled\*

```
1 % Solve an Input-Output Fitting problem with a Neural Network
2 % Script generated by Neural Fitting app
3 % Created 12-Jan-2021 22:48:22
4 %
5 % This script assumes these variables are defined:
6 %
7 % inputdata - input data.
8 % outputbeadwidth - target data.
9
10 x = inputdata';
11 t = outputbeadwidth';
12
13 % Choose a Training Function
14 % For a list of all training functions type: help ntrain
15 % 'trainlm' is usually fastest.
16 % 'trainbr' takes longer but may be better for challenging problems.
17 % 'trainscg' uses less memory. Suitable in low memory situations.
18 trainFcn = 'trainlm'; % Levenberg-Marquardt backpropagation.
```

작업 공간

| 이름 ^            | 값            |
|-----------------|--------------|
| inputdata       | 90x16 double |
| outputbeadwidth | 90x1 double  |

명령 창

```
fx >>
```

세부 정보

세부 정보를 볼 파일 선택

준비

UTF-8 | 스크립트 | 라인 1 | 열 1

## 4. MATLAB에서 신경회로망 fitting (DNN)

16. Hidden Layer의 개수와 Training, Validation, Test 데이터의 비율을 확인할 수 있음
17. 여기서 hidden layer의 층을 하나 이상을 추가하게 되면 DNN 모델을 구축할 수 있음

The image shows two screenshots of the MATLAB editor side-by-side, illustrating the modification of a neural network model. A blue arrow points from the left screenshot to the right one.

**Left Screenshot:** The code in the editor is as follows:

```
19 % Create a Fitting Network
20 hiddenLayerSize = 30;
21 net = fitnet(hiddenLayerSize,trainFcn);
22
23
24 % Setup Division of Data for Training, Validation, Testing
25 net.divideParam.trainRatio = 70/100;
26 net.divideParam.valRatio = 15/100;
27 net.divideParam.testRatio = 15/100;
28
29 % Train the Network
30 [net,tr] = train(net,x,t);
31
32 % Test the Network
33 y = net(x);
34 e = gsubtract(t,y);
35 performance = perform(net,t,y);
36
```

**Right Screenshot:** The code in the editor is modified to create a DNN with two hidden layers:

```
19 % Create a Fitting Network
20 hiddenLayer1Size = 30;
21 hiddenLayer2Size = 30;
22 net = fitnet([hiddenLayer1Size hiddenLayer2Size],trainFcn);
23
24
25 % Setup Division of Data for Training, Validation, Testing
26 net.divideParam.trainRatio = 70/100;
27 net.divideParam.valRatio = 15/100;
28 net.divideParam.testRatio = 15/100;
29
30 % Train the Network
31 [net,tr] = train(net,x,t);
32
33 % Test the Network
34 y = net(x);
35 e = gsubtract(t,y);
36 performance = perform(net,t,y);
37
```

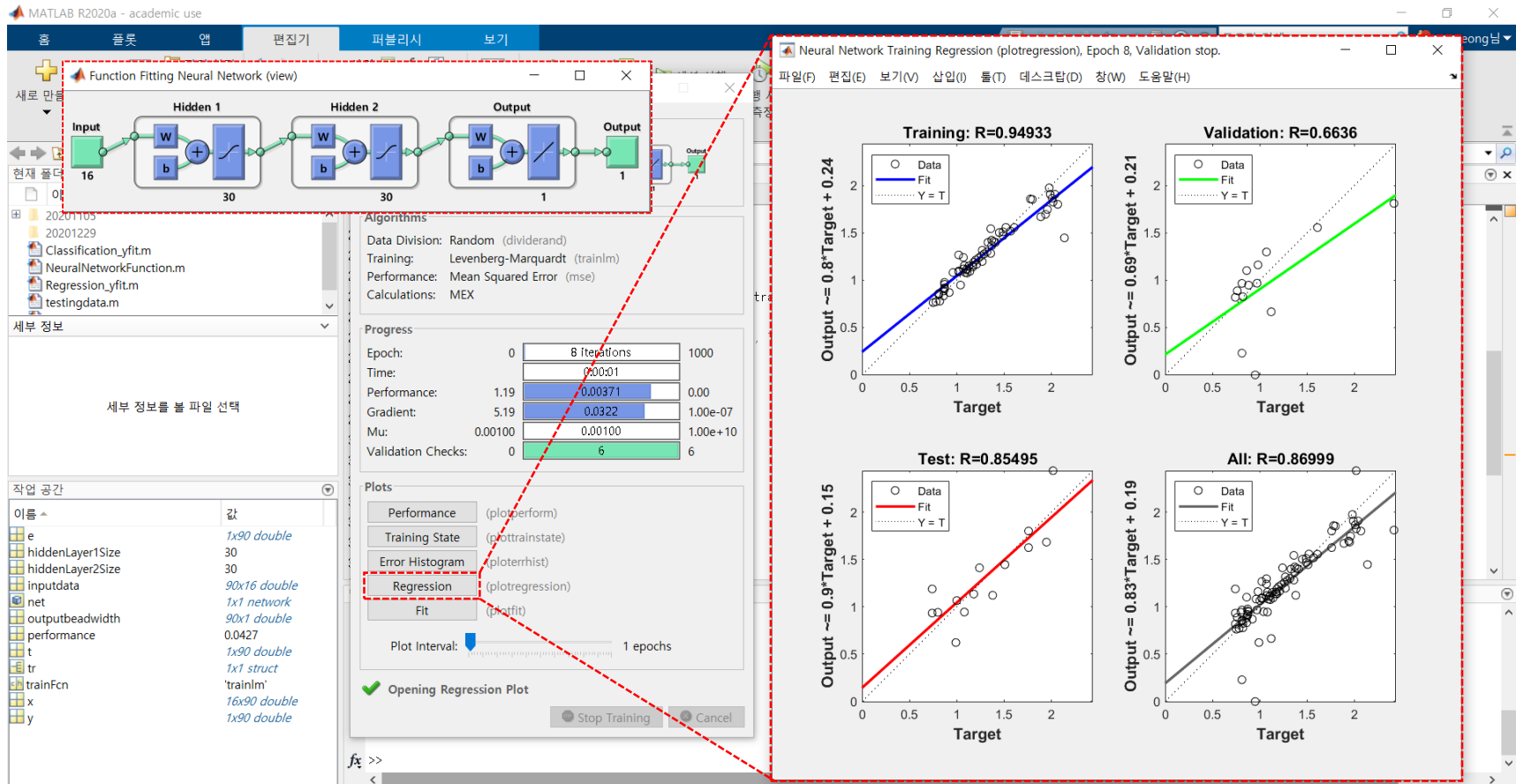
The workspace on the right shows the updated variables: `hiddenLayer1Size` (30), `hiddenLayer2Size` (30), `inputdata` (90x16 double), and `outputbeadwidth` (90x1 double).

18. 상단에 실행 버튼 클릭

# 4. MATLAB에서 신경회로망 fitting (DNN)

19. 사용한 모델의 구조를 보여주는 창이 뜬

20. Regression을 클릭하면 구축된 모델의 training, validation, test의 결정계수를 확인할 수 있음



# 4. MATLAB에서 신경회로망 fitting (DNN)

21. 왼쪽 하단의 작업공간에 생성된 y를 클릭하면 DNN을 통해 학습된 값을 확인할 수 있음

The screenshot shows the MATLAB R2020a workspace. The 'Workspace' pane on the left lists variables including 'y' with a size of 1x90 double. The main workspace area displays the values for 'y' in a grid format. The Command Window at the bottom shows the prompt 'fx >>'.

|    | 1      | 2      | 3      | 4      | 5      | 6      | 7      | 8      | 9      | 10     | 11     | 12     | 13     | 14     | 15     | 16     | 17  |
|----|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|--------|-----|
| 1  | 0.7687 | 0.9347 | 1.1987 | 1.4279 | 1.6219 | 1.7754 | 0.5689 | 0.6846 | 0.9060 | 0.9065 | 1.2230 | 1.6365 | 0.7958 | 0.9435 | 1.2480 | 1.2927 | 1.4 |
| 2  |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |     |
| 3  |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |     |
| 4  |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |     |
| 5  |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |     |
| 6  |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |     |
| 7  |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |     |
| 8  |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |     |
| 9  |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |     |
| 10 |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |     |
| 11 |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |     |
| 12 |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |     |
| 13 |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |     |
| 14 |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |     |
| 15 |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |     |
| 16 |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |        |     |

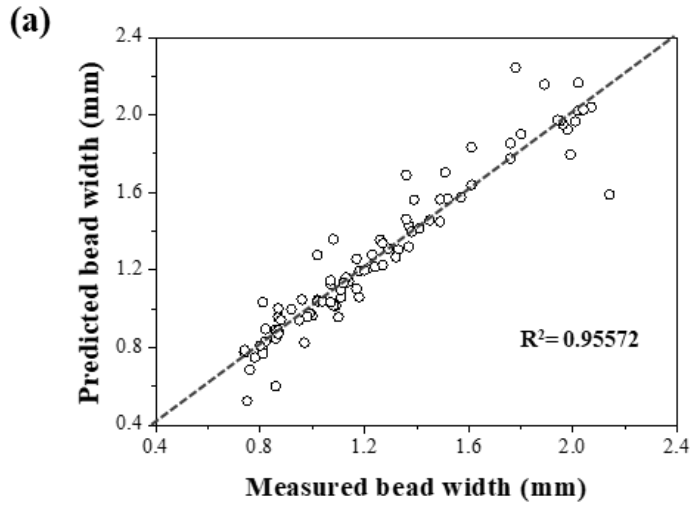
# 4. MATLAB에서 신경회로망 fitting (DNN)

22. 데이터의 행과 열을 바꾸기 위해서  $y2=y'$  라는 명령어를 작성해주면 작업공간에 y2가 생기면서 행과 열이 바뀐 데이터를 받을 수 있음

The screenshot shows the MATLAB R2020a interface. The Command Window at the bottom contains the command `>> y2=y'`, which is highlighted with a red dashed box. The Workspace on the left shows the variable `y2` as a `90x1 double`. The Editor window shows a script named `DNN_laser.m` with a variable `y2` defined. The Command Window also displays the output of the command, showing a list of values for `y2`.

| 1      | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|--------|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|
| 0.7687 |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |
| 0.9347 |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |
| 1.1987 |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |
| 1.4279 |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |
| 1.6219 |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |
| 1.7754 |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |
| 0.5689 |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |
| 0.6846 |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |
| 0.9060 |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |
| 0.9065 |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |
| 1.2230 |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |
| 1.6365 |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |
| 0.7958 |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |
| 0.9435 |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |
| 1.2480 |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |
| 1.2927 |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |

# 5. 모델의 정확성 판단하기

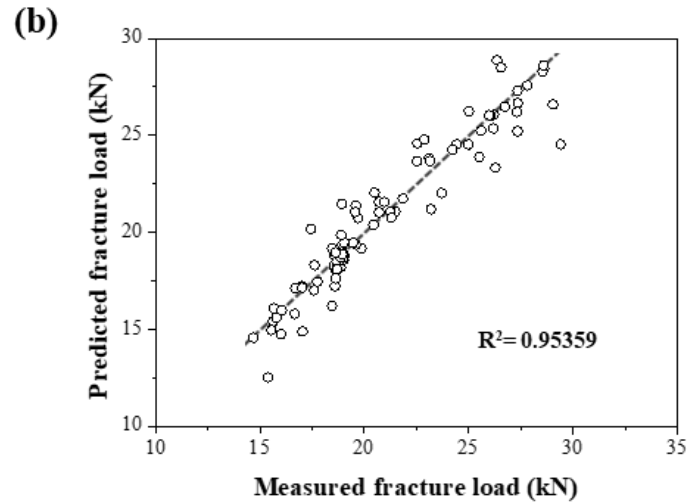


**Training Confusion Matrix**

|              |   |              |              |              |
|--------------|---|--------------|--------------|--------------|
| Output Class | 0 | 27<br>43.5%  | 0<br>0.0%    | 100%<br>0.0% |
|              | 1 | 0<br>0.0%    | 35<br>56.5%  | 100%<br>0.0% |
|              |   | 100%<br>0.0% | 100%<br>0.0% | 100%<br>0.0% |
|              |   | 0            | 1            |              |
|              |   | Target Class |              |              |

**Validation Confusion Matrix**

|              |   |              |              |              |
|--------------|---|--------------|--------------|--------------|
| Output Class | 0 | 8<br>57.1%   | 0<br>0.0%    | 100%<br>0.0% |
|              | 1 | 0<br>0.0%    | 6<br>42.9%   | 100%<br>0.0% |
|              |   | 100%<br>0.0% | 100%<br>0.0% | 100%<br>0.0% |
|              |   | 0            | 1            |              |
|              |   | Target Class |              |              |



**Test Confusion Matrix**

|              |   |              |              |              |
|--------------|---|--------------|--------------|--------------|
| Output Class | 0 | 8<br>57.1%   | 0<br>0.0%    | 100%<br>0.0% |
|              | 1 | 0<br>0.0%    | 6<br>42.9%   | 100%<br>0.0% |
|              |   | 100%<br>0.0% | 100%<br>0.0% | 100%<br>0.0% |
|              |   | 0            | 1            |              |
|              |   | Target Class |              |              |

**All Confusion Matrix**

|              |   |              |              |              |
|--------------|---|--------------|--------------|--------------|
| Output Class | 0 | 43<br>47.8%  | 0<br>0.0%    | 100%<br>0.0% |
|              | 1 | 0<br>0.0%    | 47<br>52.2%  | 100%<br>0.0% |
|              |   | 100%<br>0.0% | 100%<br>0.0% | 100%<br>0.0% |
|              |   | 0            | 1            |              |
|              |   | Target Class |              |              |



# 감사합니다

다음 강의 내용 : 파이썬을 이용한 SNN, DNN 모델 분석 방법