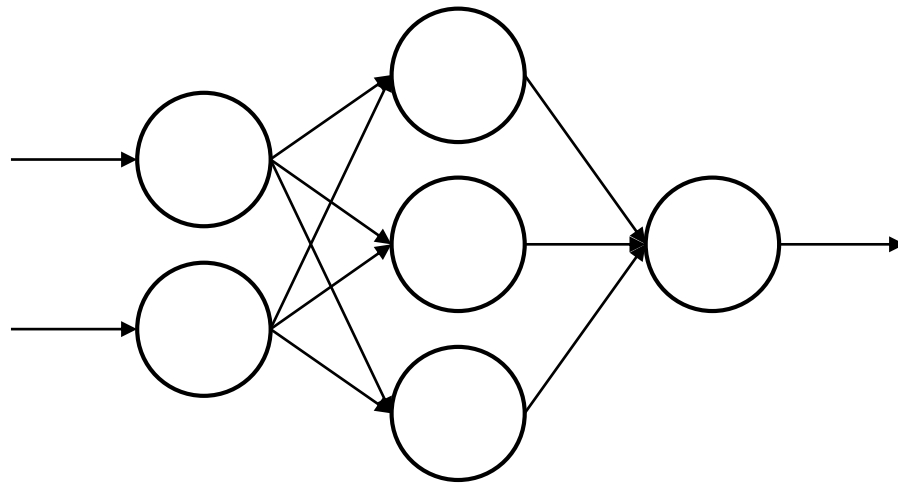


ニューラルネットワーク Neural Network (NN)

MATLAB Deep Learning Toolboxの使い方
関数値の補間・外挿



2020/7/9 平野拓一

MATLAB

パターン認識、クラスタリング、および時系列用の浅いネットワーク

<https://jp.mathworks.com/help/deeplearning/gs/shallow-networks-for-pattern-recognition-clustering-and-time-series.html>

浅いニューラル ネットワークによるデータのあてはめ

<https://jp.mathworks.com/help/deeplearning/gs/fit-data-with-a-neural-network.html>

例題(AND)

x and y

x \ y	0	1
0	0	0
1	0	1

入力ファイル

$z = x \text{ and } y$

inputs.txt

x	0	0	1	1	0	0	1	1
y	0	1	0	1	0	1	0	1

targets.txt

z	0	0	0	1	0	0	0	1
---	---	---	---	---	---	---	---	---

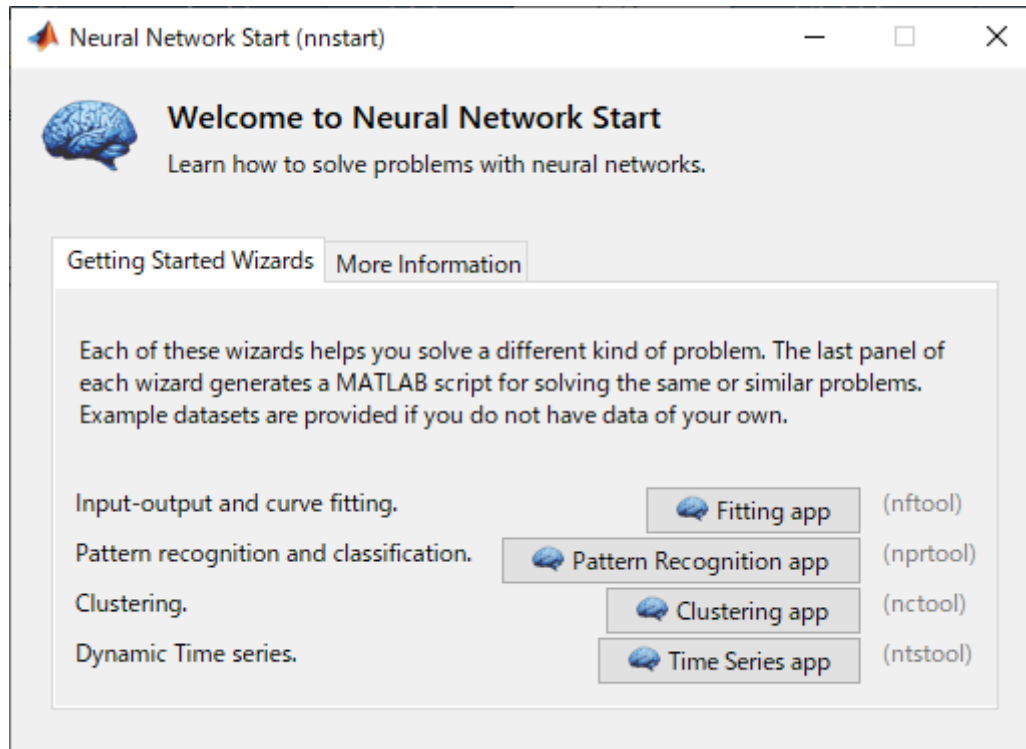
いくつかが学習用、いくつかが検証用に使われるため、同じデータを2回書いている。（今回は練習で、すべてのANDの動作を学習させたいから）

上のデータではAND動作のデータを2つ、つまり8個のデータを記述している。

使い方(1)

nnstart

と入力すると次のウィンドウが開く




Fitting app
を選ぶ。

MATLABのToolboxは有料であり、そこがPython環境でGoogleが無料提供しているTensorFlowに比べると、使いたいかどうか好みに分かれるが。

使い方(2)

Neural Fitting (nftool)

 **Welcome to the Neural Network Fitting app.**
Solve an input-output fitting problem with a two-layer feed-forward neural network.

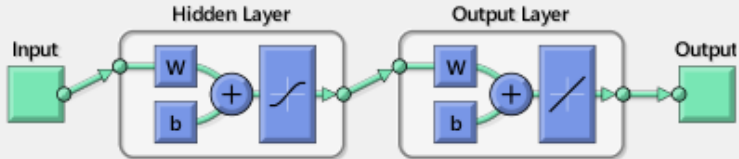
Introduction

In fitting problems, you want a neural network to map between a data set of numeric inputs and a set of numeric targets.

Examples of this type of problem include estimating engine emission levels based on measurements of fuel consumption and speed (`engine_dataset`) or predicting a patient's bodyfat level based on body measurements (`bodyfat_dataset`).

The Neural Fitting app will help you select data, create and train a network, and evaluate its performance using mean square error and regression analysis.

Neural Network



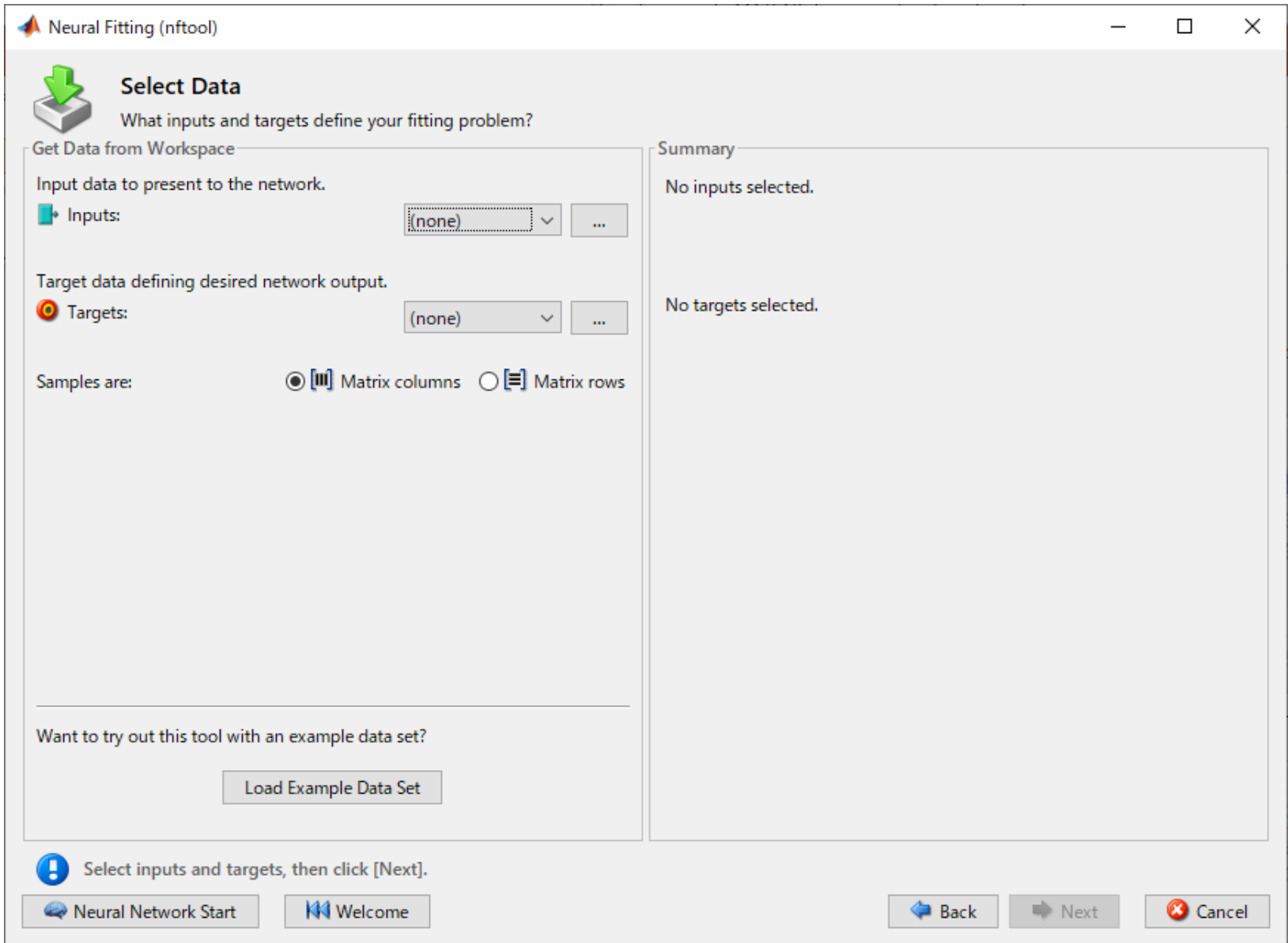
A two-layer feed-forward network with sigmoid hidden neurons and linear output neurons (`fitnet`), can fit multi-dimensional mapping problems arbitrarily well, given consistent data and enough neurons in its hidden layer.

The network will be trained with Levenberg-Marquardt backpropagation algorithm (`trainlm`), unless there is not enough memory, in which case scaled conjugate gradient backpropagation (`trainscg`) will be used.

➔ To continue, click [Next].

Neural Network Start Welcome Back **Next** Cancel

使い方(3)



使い方(4)

Neural Fitting (nftool)

Select Data

What inputs and targets define your fitting problem?

Get Data from Workspace

Input data to present to the network.

Inputs: ...

Target data defining desired network output.

Targets: ...

Samples are: Matrix columns Matrix rows

Summary

Inputs 'inputs' is a 2x8 matrix, representing static data: 8 samples of 2 elements.

Targets 'targets' is a 1x8 matrix, representing static data: 8 samples of 1 element.

Want to try out this tool with an example data set?

To continue, click [Next].

入力ファイルのスライドで書いたテキストデータを指定

使い方(5)

Neural Fitting (nftool)

Validation and Test Data

Set aside some samples for validation and testing.

Select Percentages

Randomly divide up the 8 samples:

Training:	70%	6 samples
Validation:	15% ▾	1 samples
Testing:	15% ▾	1 samples

Restore Defaults

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.

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

Neural Network Start | Welcome | Back | Next | Cancel

入力データセットの何%を教師データ、確認データ、テストデータとして使用するか指定

使い方(6)

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:

Recommendation

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

Restore Defaults

Neural Network

The diagram illustrates a neural network architecture with the following components:

- Input Layer:** 2 neurons.
- Hidden Layer:** 10 neurons, using a sigmoid function (S-shaped curve).
- Output Layer:** 1 neuron, using a linear function (straight line).

Each layer is represented by a box containing weights (W) and bias (b), followed by an addition (+) operation and the activation function. The output of the hidden layer is connected to the input of the output layer.

Annotations:

- 隠れ層のニューロン数 (Number of hidden layer neurons) points to the input field containing '10'.
- シグモイド関数 (Sigmoid function) points to the sigmoid curve in the hidden layer.
- 線形関数 (Linear function) points to the linear curve in the output layer.

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

Neural Network Start Welcome Back Next Cancel

使い方(7)

Neural Fitting (nftool)

Train Network

Train the network to fit the inputs and targets.

Train Network

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)

Train

Results

	Samples	MSE	R
Training:	6	-	-
Validation:	1	-	-
Testing:	1	-	-

Plot Fit Plot Error Histogram Plot Regression

Notes

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.

Train network, then click [Next].

Neural Network Start Welcome Back Next Cancel

使い方(8)

収束状況

The screenshot shows the Neural Network Training (nntool) interface. At the top, the title bar reads "Neural Network Training (nntool)". Below it, the "Neural Network" section displays a diagram of a feedforward network with 2 input nodes, 10 hidden nodes, and 1 output node. The hidden and output layers each contain weight (W) and bias (b) blocks, followed by an addition (+) block and an activation function block. The "Algorithms" section lists: Data Division: Random (dividerand), Training: Levenberg-Marquardt (trainlm), Performance: Mean Squared Error (mse), and Calculations: MEX. The "Progress" section contains a table with the following data:

Epoch:	0	3 iterations	1000
Time:		0:00:00	
Performance:	0.961	6.99e-20	0.00
Gradient:	1.48	2.10e-10	1.00e-07
Mu:	0.00100	1.00e-05	1.00e+10
Validation Checks:	0	0	6

The "Plots" section lists several plot types: Performance (plotperform), Training State (plottrainstate), Error Histogram (ploterrhist), Regression (plotregression), and Fit (plotfit). A "Plot Interval" slider is set to 1 epochs. At the bottom, a green checkmark indicates "Minimum gradient reached." and two buttons, "Stop Training" and "Cancel", are visible.

使い方(9)

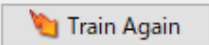
Neural Fitting (nftool)

Evaluate Network

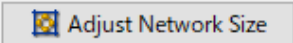
Optionally test network on more data, then decide if network performance is good enough.

Iterate for improved performance

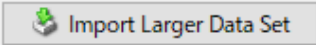
Try training again if a first try did not generate good results or you require marginal improvement.



Increase network size if retraining did not help.



Not working? You may need to use a larger data set.



Optionally perform additional tests

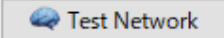
Inputs: ...

Targets: ...

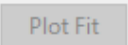
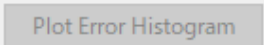
Samples are: Matrix columns Matrix rows

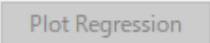
Inputs 'inputs_1' is a 2x8 matrix, representing static data: 8 samples of 2 elements.

Targets 'targets_1' is a 1x8 matrix, representing static data: 8 samples of 1 element.

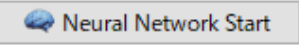
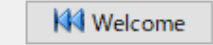





MSE
 R



Click an improvement button, test, or click [Next]

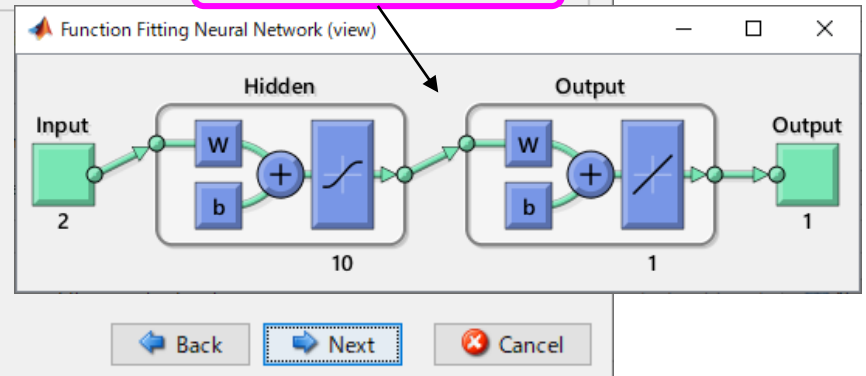
    

使い方(10)

The screenshot shows the 'Neural Fitting (nftool)' window with the following sections and options:

- Deploy Solution**
 - Generate deployable versions of your trained neural network.
 - Application Deployment**
 - Prepare neural network for deployment with MATLAB Compiler and Builder tools.
 - Generate a MATLAB function with matrix and cell array argument support: (genFunction) **MATLAB Function**
 - Code Generation**
 - Prepare neural network for deployment with MATLAB Coder tools.
 - Generate a MATLAB function with matrix-only arguments (no cell array support): (genFunction) **MATLAB Matrix-Only Function**
 - Simulink Deployment**
 - Simulate neural network in Simulink or deploy with Simulink Coder tools.
 - Generate a Simulink diagram: (gensim) **Simulink Diagram**
 - Graphics**
 - Generate a graphical diagram of the neural network: (network/view) **Neural Network Diagram**
- Buttons: **Neural Network Start**, **Welcome**
- Footer: **Deploy a neural network or click [Next].**

次スライドへ



使い方(11)

関数名"myNeuralNetworkFunction"と同じファイル名のmファイル(拡張子が.m)で保存 (次スライドへ)

```
function [y1] = myNeuralNetworkFunction(x1)
%MYNEURALNETWORKFUNCTION neural network simulation function.
%
% Auto-generated by MATLAB, 09-Jul-2020 12:58:42.
%
% [y1] = myNeuralNetworkFunction(x1) takes these arguments:
%   x = 2xQ matrix, input #1
% and returns:
%   y = 1xQ matrix, output #1
-% where Q is the number of samples.
%#ok<*RPMT0>
% ===== NEURAL NETWORK CONSTANTS =====
% Input 1
x1_step1.xoffset = [0;0];
x1_step1.gain = [2;2];
x1_step1.ymin = -1;
```

名前	値
inputs	2x8 double
inputs_1	2x8 double
targets	[0,0,0,1,0,0,0,1]
targets_1	[0,0,0,1,0,0,0,1]

```
>> nnstart
fx >>
```

使い方(12)

```
>> myNeuralNetworkFunction([0;0])
```

```
ans =
```

```
-3.3887e-10
```

```
>> myNeuralNetworkFunction([0;1])
```

```
ans =
```

```
1.8194e-10
```

```
>> myNeuralNetworkFunction([1;0])
```

```
ans =
```

```
3.8701e-11
```

```
>> myNeuralNetworkFunction([1;1])
```

```
ans =
```

```
1.0000
```

しっかりANDの動作をしている