

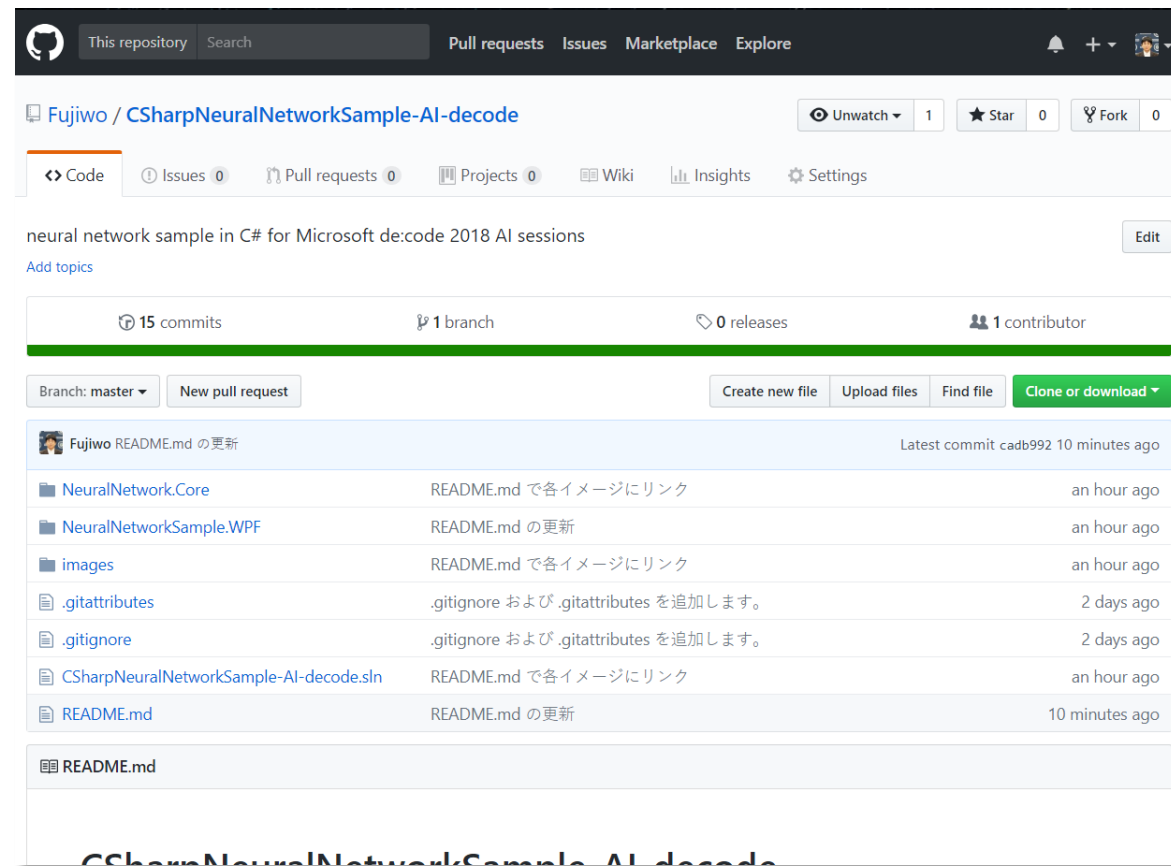
C# でニューラルネットワークをスクラッチで書いて 機械学習の原理を理解しよう

福井コンピュータホールディングス株式会社

小島 富治雄

ソースコードの場所

- 「C# でニューラルネットワークをフルスクラッチで書いて機械学習の原理を理解しよう」
for Microsoft de:code 2018 AI sessions
- <https://github.com/Fujiwo/CSharpNeuralNetworkSample-AI-decode>



de:code 2018

関連資料

- 「Microsoft Azure Machine Learning Studio による株価予想チュートリアル」
for Microsoft de:code 2018 AI sessions
- <https://github.com/Fujiwo/PredictStockPrice-AI-decode>

This repository

Pull requests Issues Marketplace Explore

Fujiwo / PredictStockPrice-AI-decode

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

PredictStockPrice Sample for Microsoft de:code 2018 AI sessions

Add topics

25 commits 1 branch 0 releases 1 contributor MIT

Branch: master New pull request

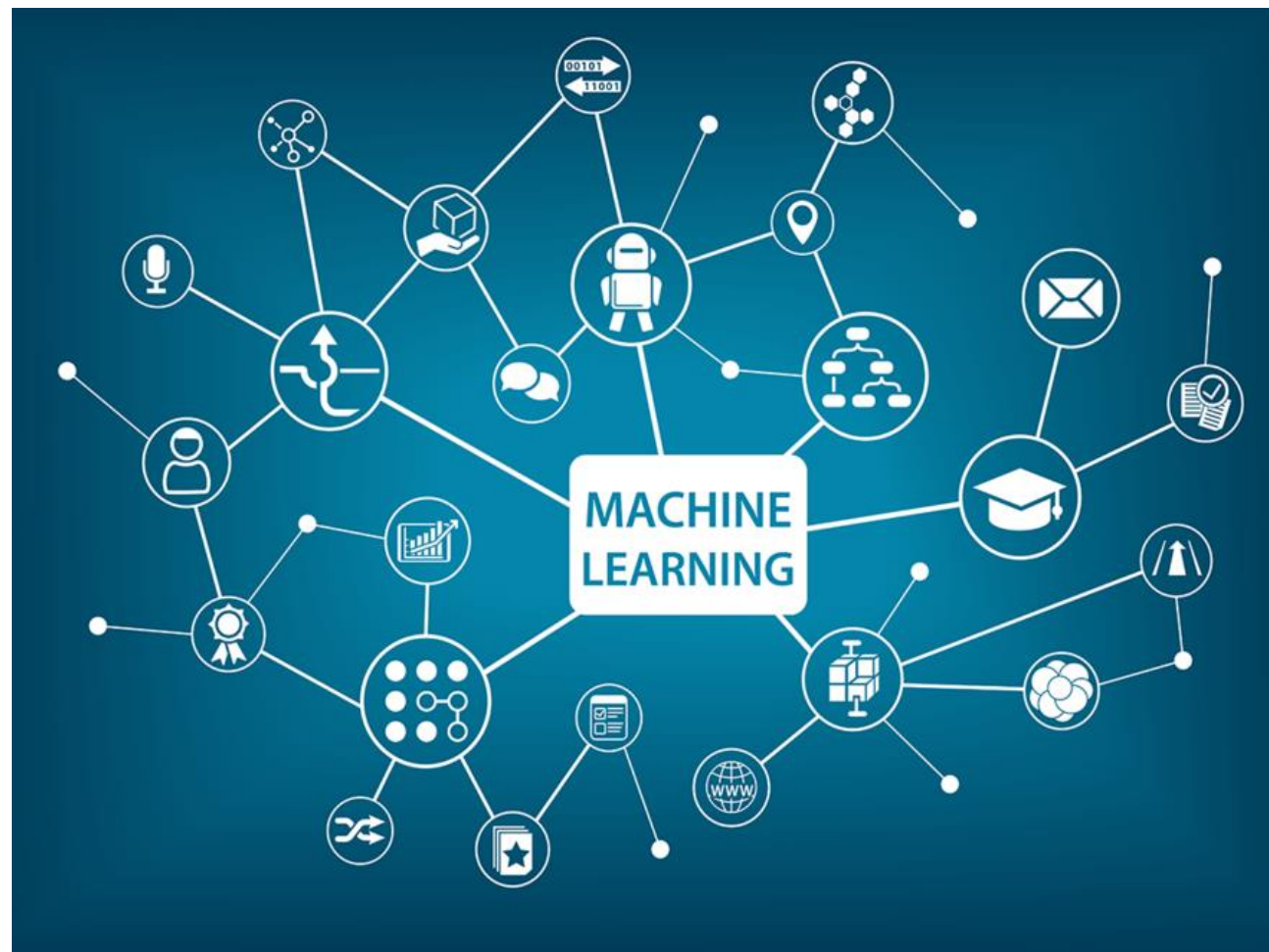
Create new file Upload files Find file Clone or download

Fujiwo README.md の更新 Latest commit f4185b8 9 minutes ago

.vs	README.md の更新	9 minutes ago
PredictStockPrice.Console	Program.cs	2 days ago
images	README.md の4章と5章を更新し、ファイル説明を追加	2 days ago
.gitattributes	Initial commit	4 days ago
.gitignore	C# コンソール アプリケーションの追加	3 days ago
9790.csv	9790.csv の追加と README.md の更新	4 days ago
AzureMachineLearningScript.1.py	README.md の更新	4 days ago
LICENSE	Initial commit	4 days ago
README.md	README.md の更新	9 minutes ago

README.md

de:code 2018



まず 人工知能 (Artificial Intelligence) とは

- 人間の知能の一部をコンピュータで再現する技術



機械学習とは

- 人工知能の一分野
- コンピュータプログラムが経験、学習を行う



機械学習の種類

- 機械学習 (Machine Learning)
- ディープラーニング
(深層学習: Deep Learning)
- 強化学習
(Reinforcement Learning)
- 深層強化学習
(Deep Reinforcement Learning)



ディープラーニング (深層学習: Deep Learning)

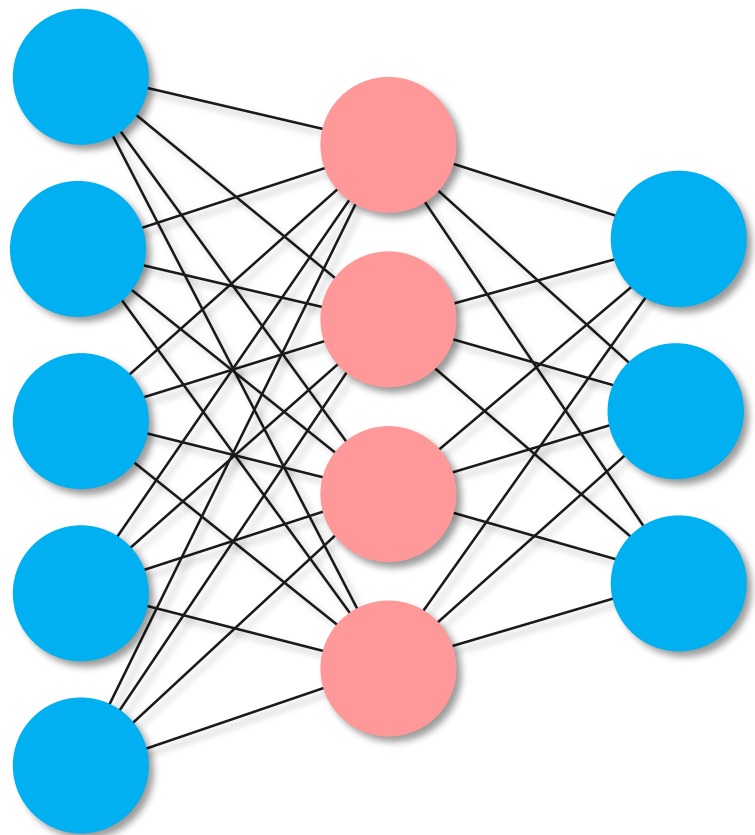
- 機械学習の一種
- 通常のものであって多層になったニューラルネットワークを使用
- 画像解析や音声認識、自然言語理解、翻訳など、複雑な処理にはこちら
- 作成のハードルは、かなり高い
- 高いコンピュータ パワーが求められるため、CPUだけでなく GPU も利用

ディープラーニング



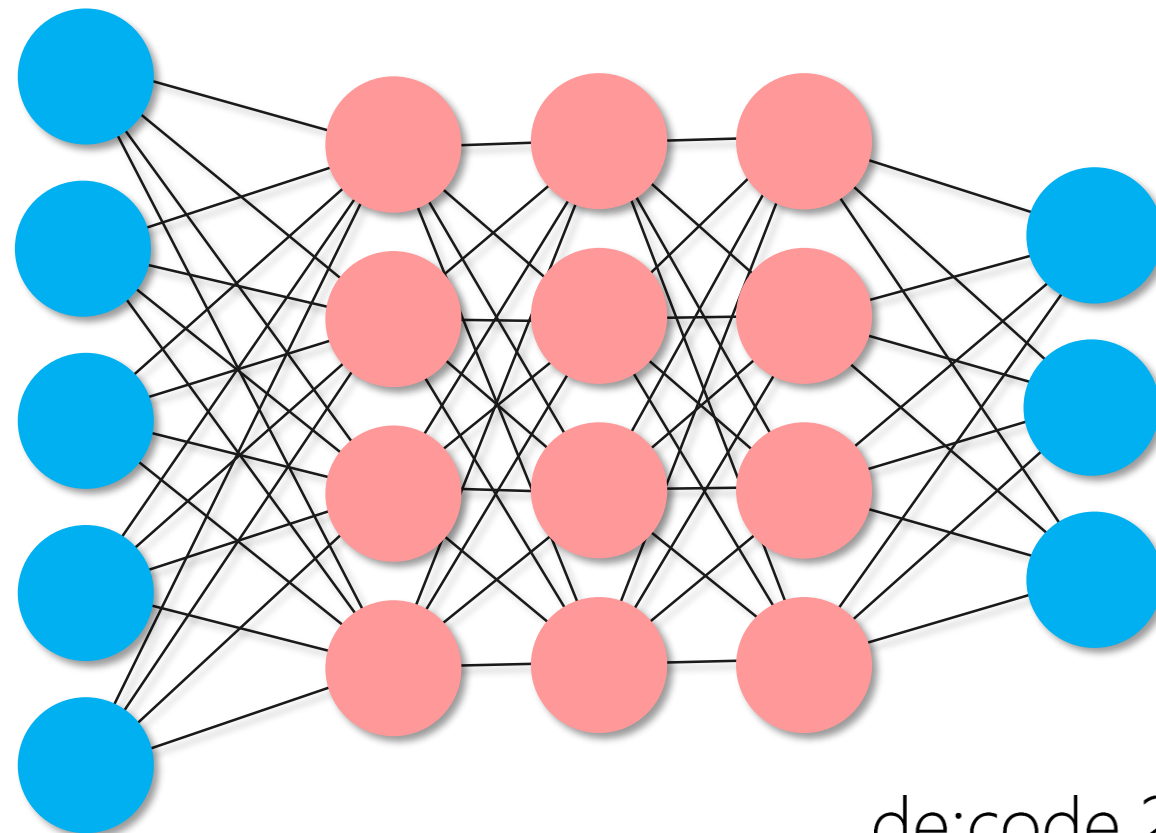
ニューラル ネットワーク

入力層 中間層 出力層

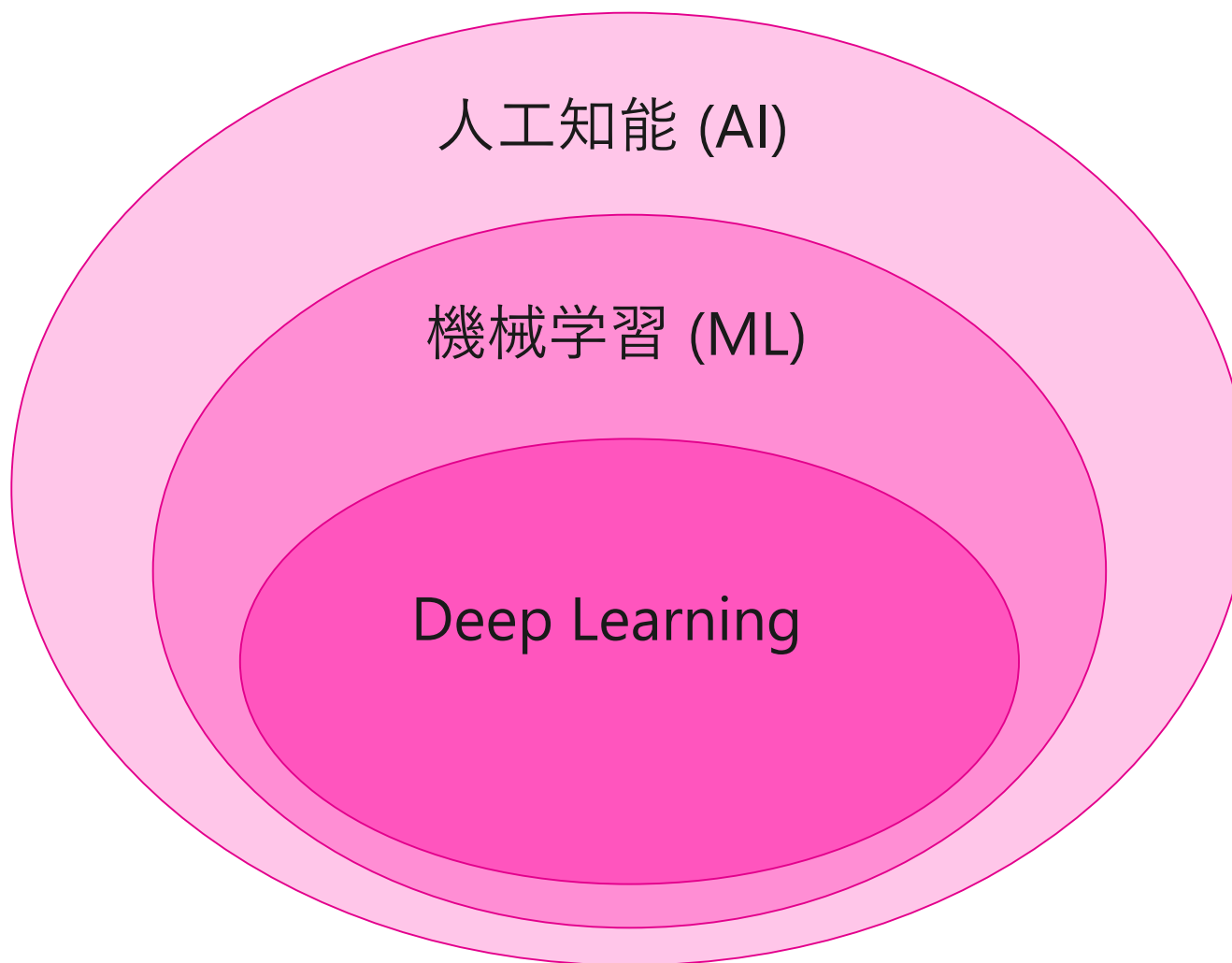


ディープラーニング

入力層 中間層 出力層



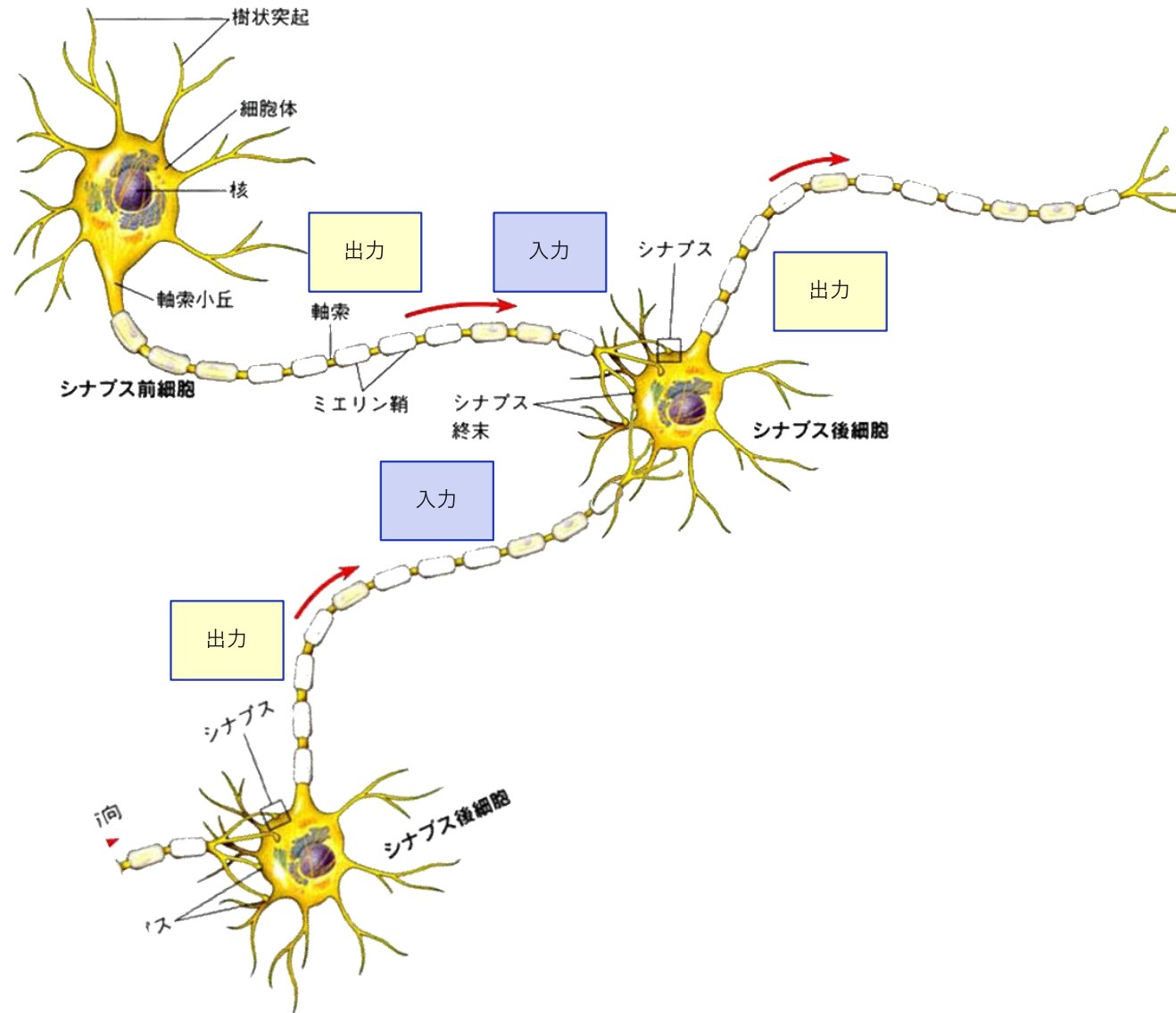
人工知能と機械学習



ニューラル ネットワークとは



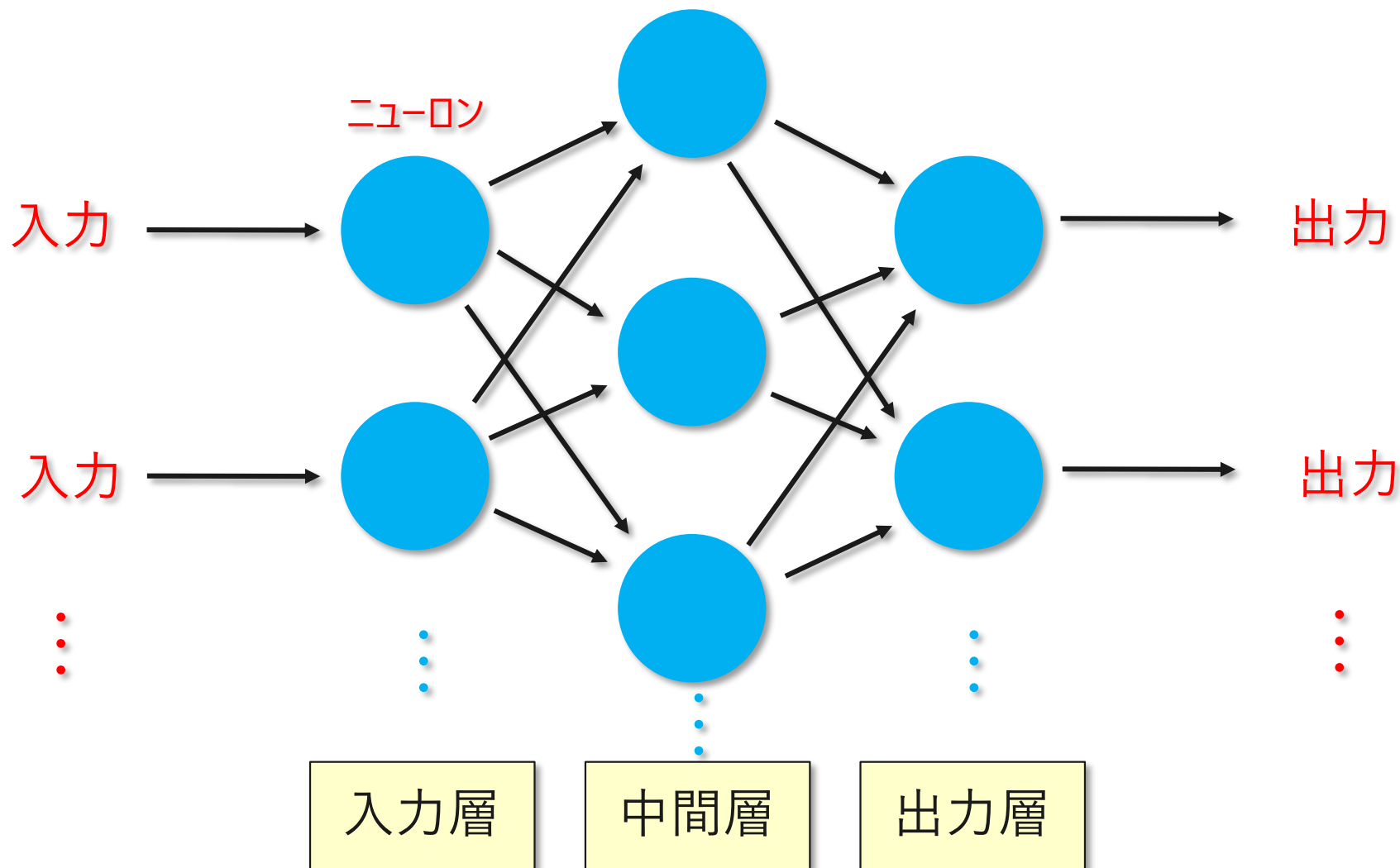
神経細胞のネットワーク



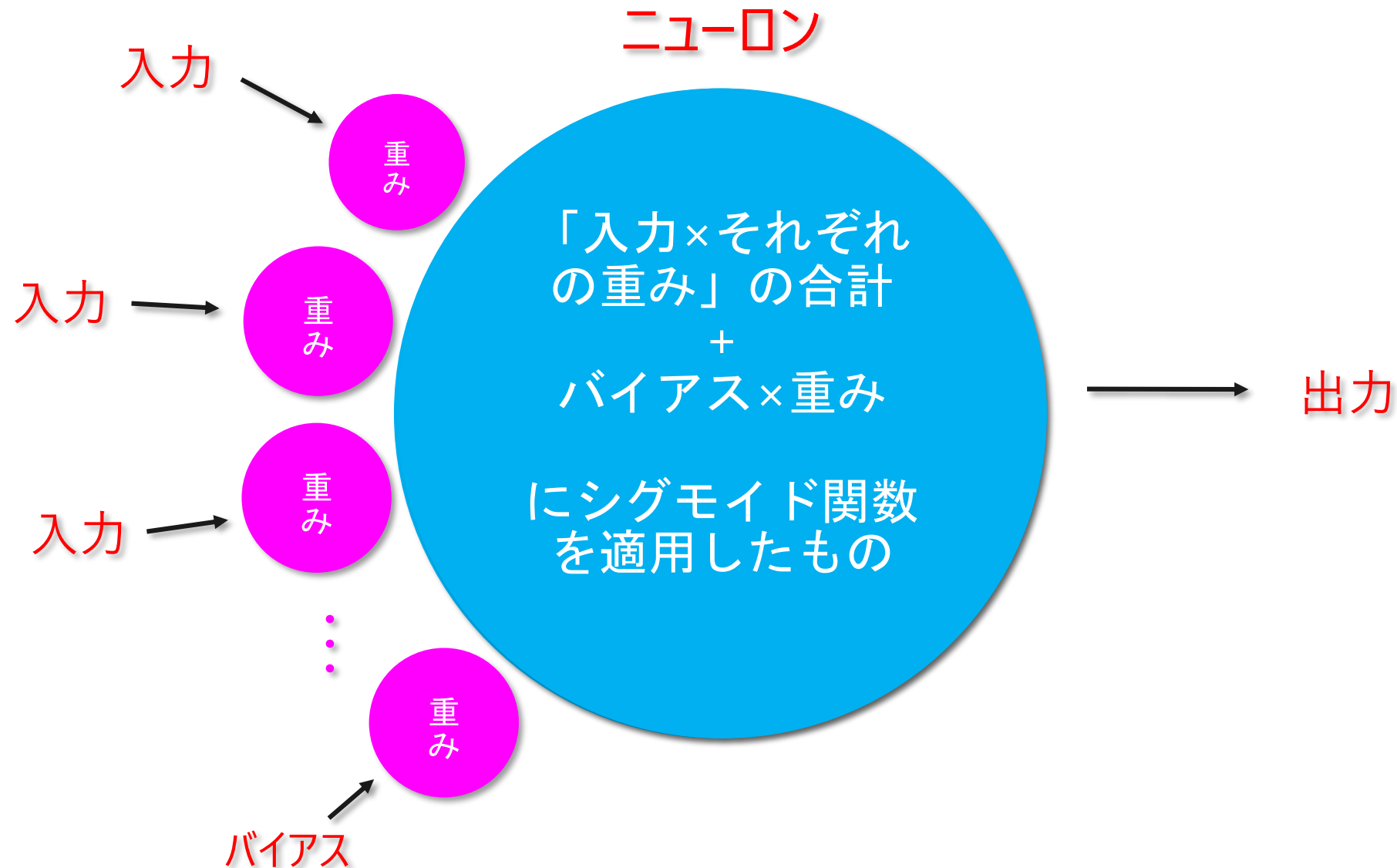
ニューラル ネットワーク



神経細胞のネットワークを模倣



個々のニューロン

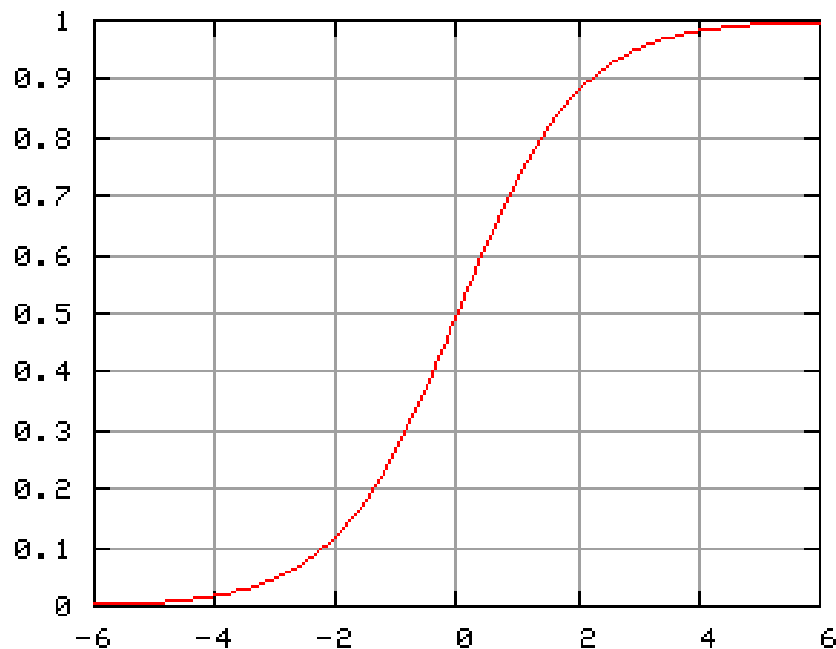


シグモイド関数



神経細胞が入力の合計を出力に
するときの性質をモデル化

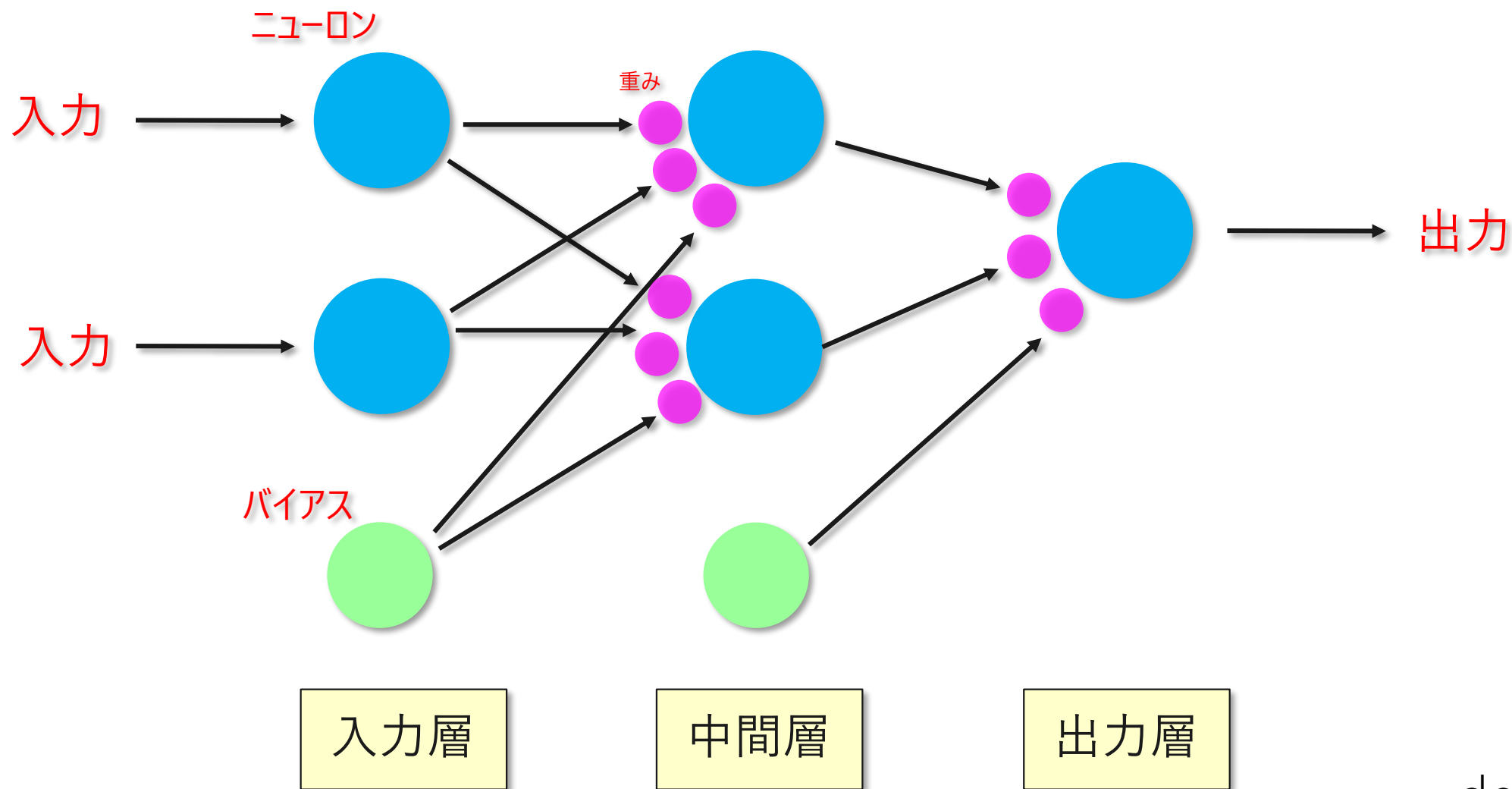
$$\varsigma_1(x) = \frac{1}{1 + e^{-x}}$$



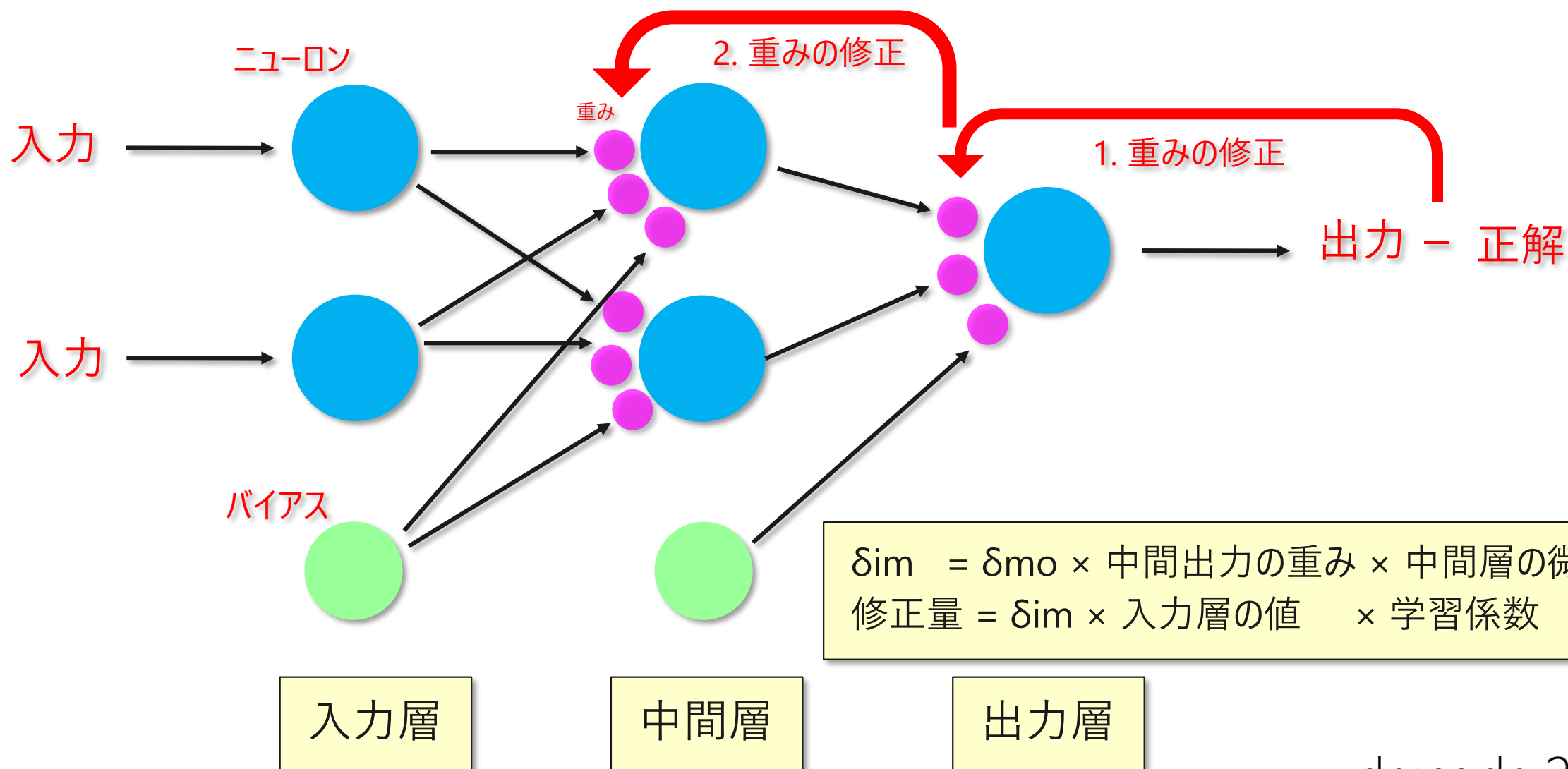
ニューラル ネットワークによる分類



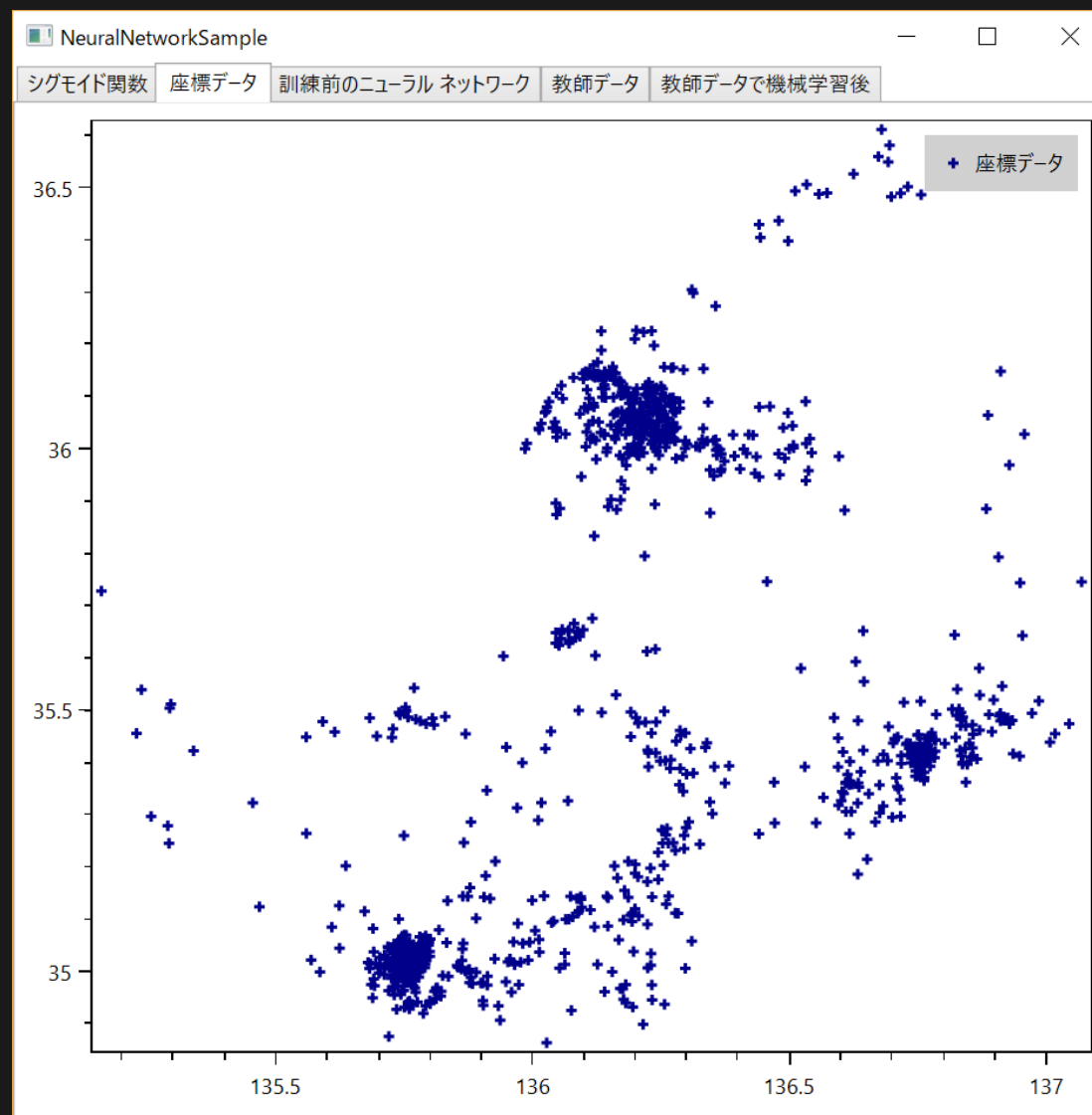
今回作成するニューラル ネットワーク



今回のニューラル ネットワークの訓練



Demo



C# によるニューロンの実装



```
public class Neuron // ニューロン
{
    double sum;

    public double Value { get; private set; } = 0.0;

    public void Input(IEnumerable<Input> inputData)
    {
        inputData.ForEach(input => Input(input.WeightingValue));
        Value = Math.Sigmoid(sum);
    }

    void Input(double value) => sum += value;
}
```

C# によるニューラル ネットワークの実装



```
public class NeuralNetwork // ニューラル ネットワーク
{
    // 各層
    double[] inputLayer;
    Neuron[] middleLayer;
    Neuron   outputLayer;

    // バイアス
    double inputLayerBias  = 1.0;
    double middleLayerBias = 1.0;

    // 各層の重み
    // 入力層 → 中間層の重み
    double[,] inputWeight = new double[,] { { 0.0, 0.0 }, { 0.0, 0.0 }, { 0.0, 0.0 } };
    // 中間層 → 出力層の重み
    double[] middleWeight = new[] { 0.0, 0.0, 0.0 };
}
```

C# によるニューラル ネットワークの実装 (続き)

```
// 実行
public double Commit((double, double) data)
{
    // 各層
    inputLayer = new[] { data.Item1, data.Item2, inputLayerBias };
    middleLayer = new[] { new Neuron(), new Neuron() };
    outputLayer = new Neuron();

    // 入力層→中間層
    middleLayer.For((index, neuron)
        => middleLayer[index].Input(ToInputData(inputLayer, inputWeight.GetColumn(index).ToArray())));

    // 中間層→出力層
    outputLayer.Input(new[] { new Input { Value = middleLayer[0].Value, Weight = middleWeight[0] },
                              new Input { Value = middleLayer[1].Value, Weight = middleWeight[1] },
                              new Input { Value = middleLayerBias, Weight = middleWeight[2] } });

    return outputLayer.Value;
}
```

C# によるニューラル ネットワークの実装 (続き)

```
// 学習 void Learn((double, double, double) data)
{
    var outputData    = Commit((data.Item1, data.Item2));
    var correctValue = data.Item3;
    var learningRate = 0.3; // 学習係数

    // 出力層→中間層
    //  $\delta_{mo} = (\text{出力値} - \text{正解値}) \times \text{出力の微分}$ 
    var deltaMO = (correctValue - outputData) * outputData * (1.0 - outputData);
    var oldMiddleWeight = middleWeight.Clone() as double[];
    // 修正量 =  $\delta_{mo} \times \text{中間層の値} \times \text{学習係数}$ 
    middleLayer.For((index, neuron) => middleWeight[index] += neuron.Value * deltaMO * learningRate);
    middleWeight[2] += middleLayerBias * deltaMO * learningRate;

    // 中間層→入力層
    //  $\delta_{im} = \delta_{mo} \times \text{中間出力の重み} \times \text{中間層の微分}$ 
    var deltaIM = middleLayer.IndexSelect(index =>
    deltaMO * oldMiddleWeight[index] * middleLayer[index].Value * (1.0 - middleLayer[index].Value)).ToArray();
    // 修正量 =  $\delta_{im} \times \text{入力層の値} \times \text{学習係数}$ 
    inputWeight.For((row, column, _) =>
        inputWeight[row, column] += inputLayer[row] * deltaIM[column] * learningRate);
}
```


C# によるニューラル ネットワークの実装 (続き)

// 学習

```
public void Learn(IEnumerable<(double, double, double)> dataCollection, int times)
    => times.Times(() => dataCollection.ForEach(data => Learn(data)));
```

```
static IEnumerable<Input> ToInputData(double[] inputLayer, double[] inputWeight)
    => inputLayer.IndexSelect(index =>
        new Input { Value = inputLayer[index], Weight = inputWeight[index] });
```

```
}
```

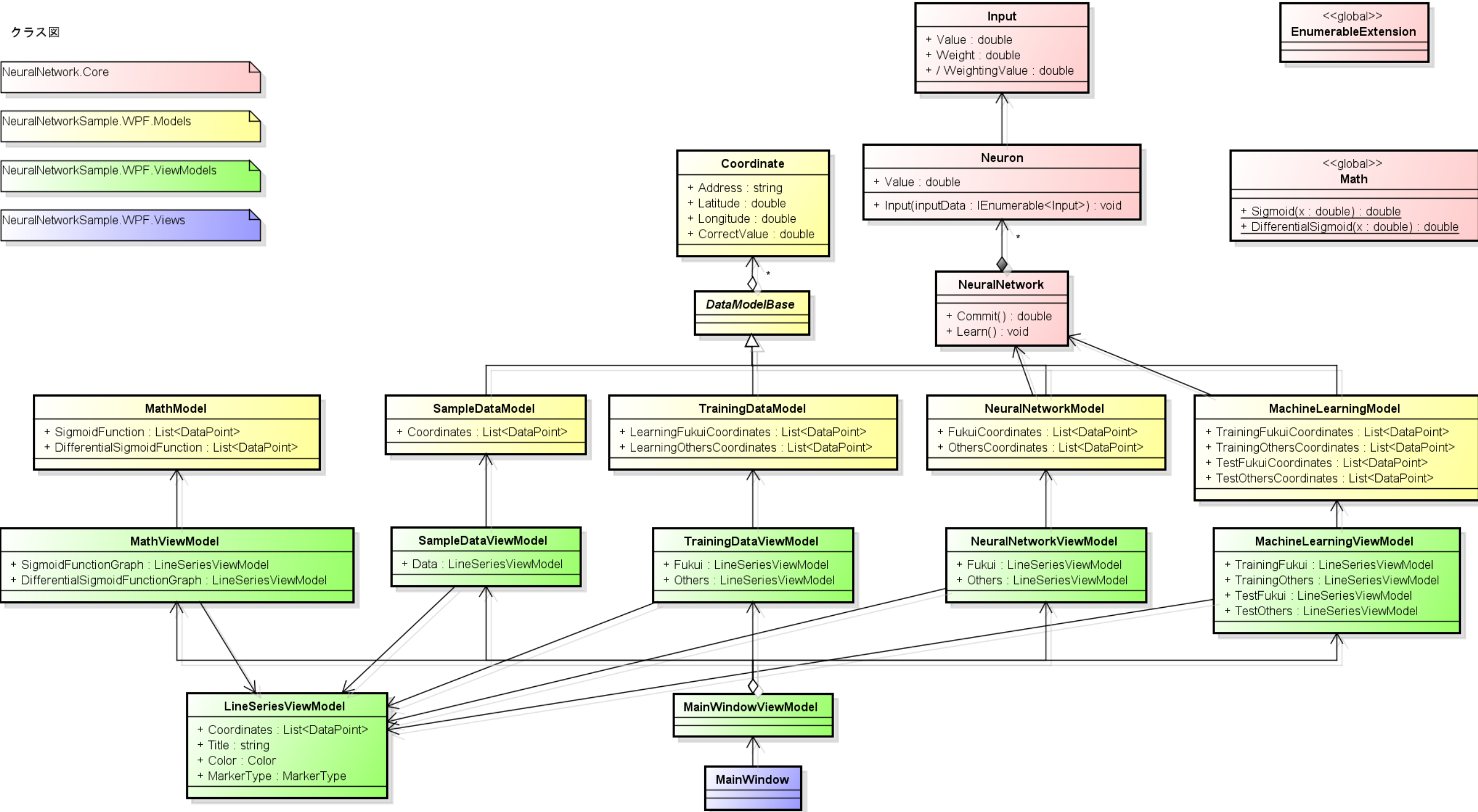
クラス図

NeuralNetwork.Core

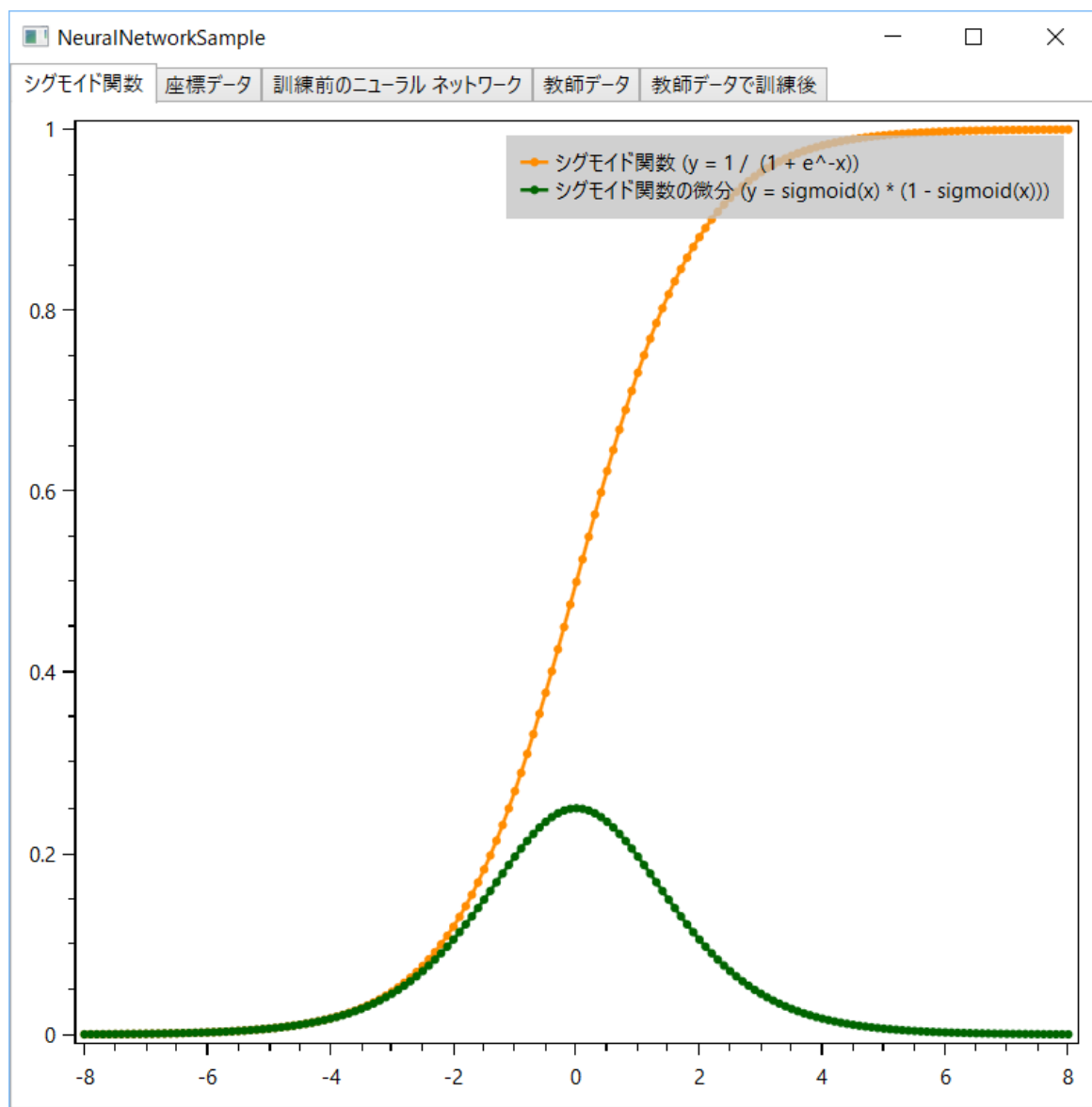
NeuralNetworkSample.WPF.Models

NeuralNetworkSample.WPF.ViewModels

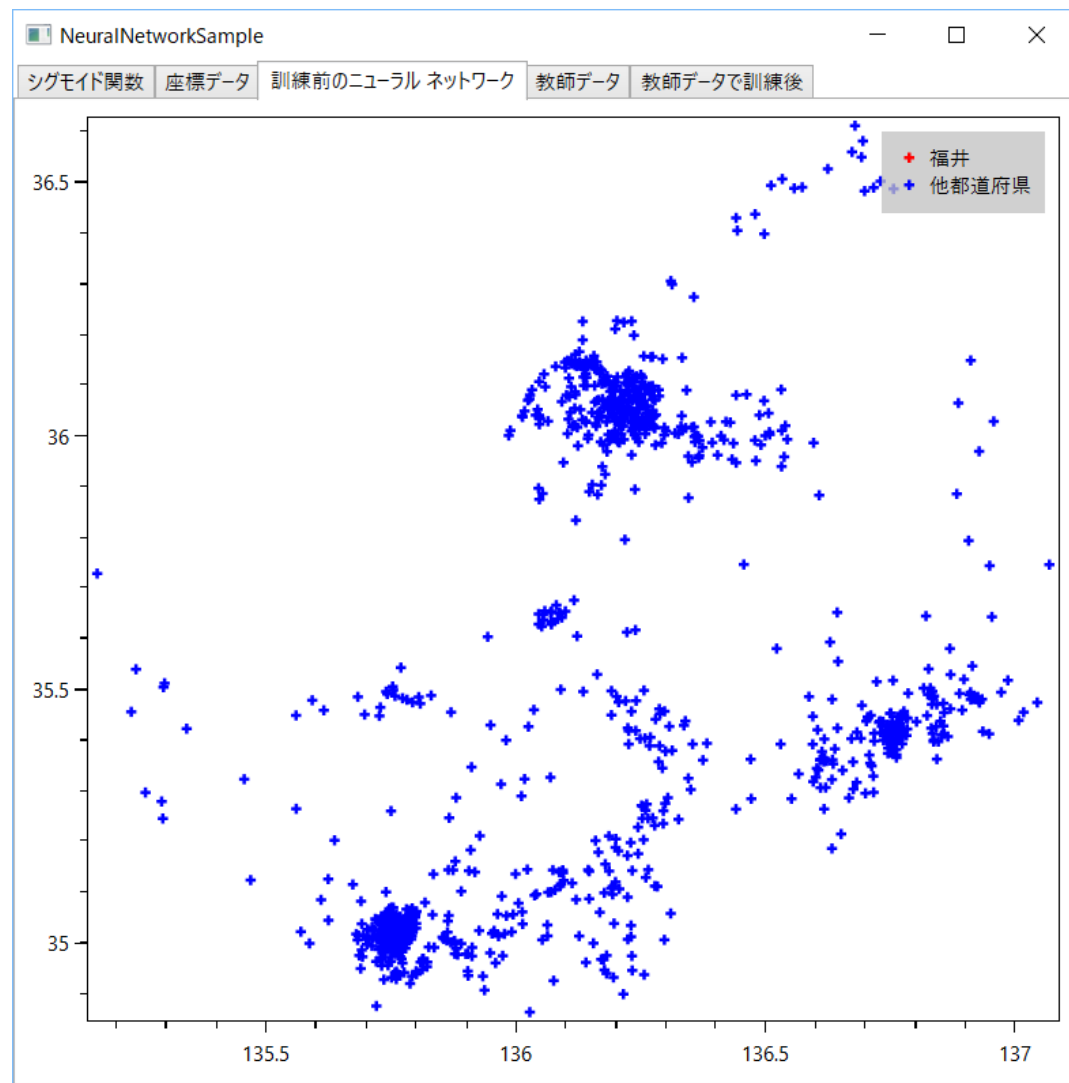
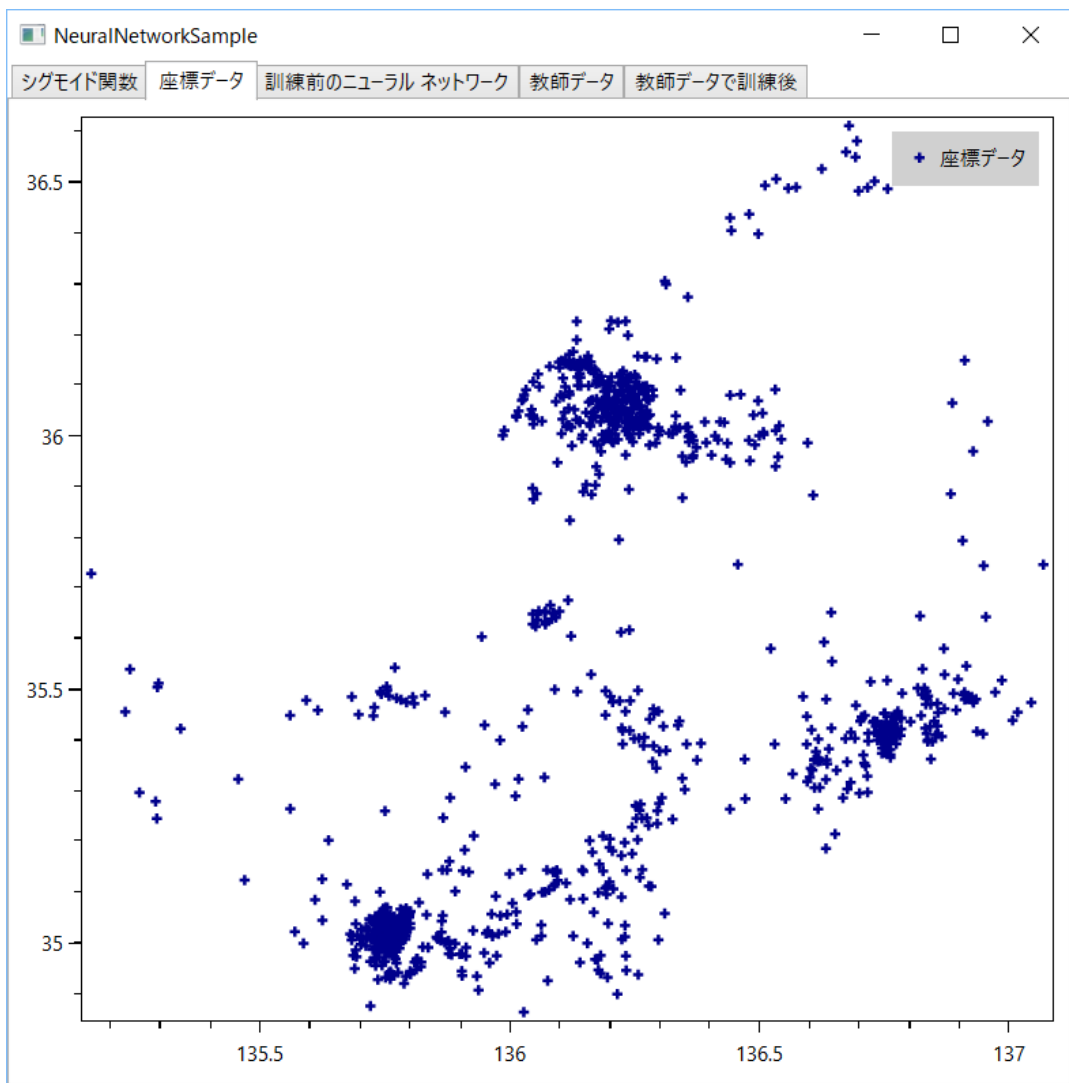
NeuralNetworkSample.WPF.Views



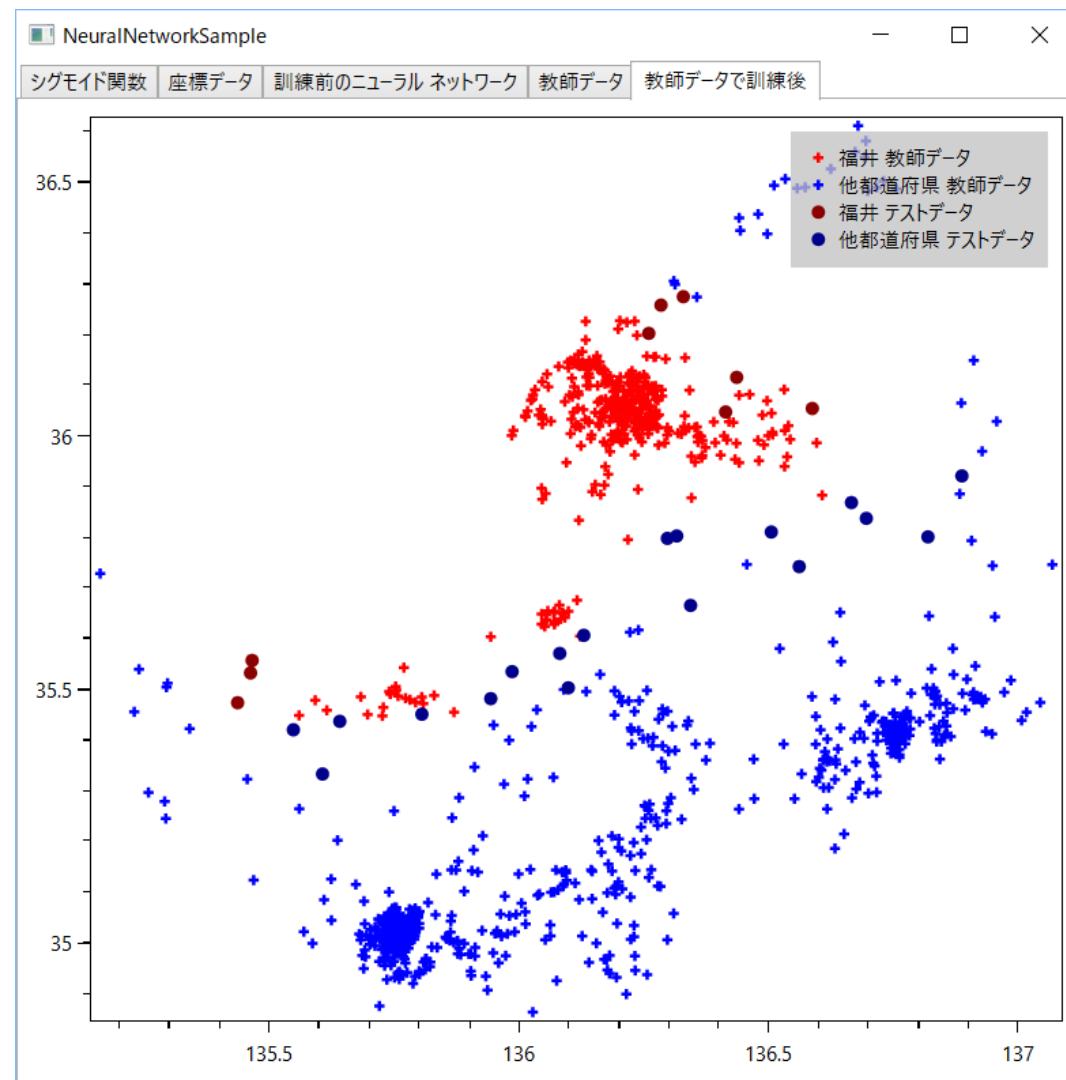
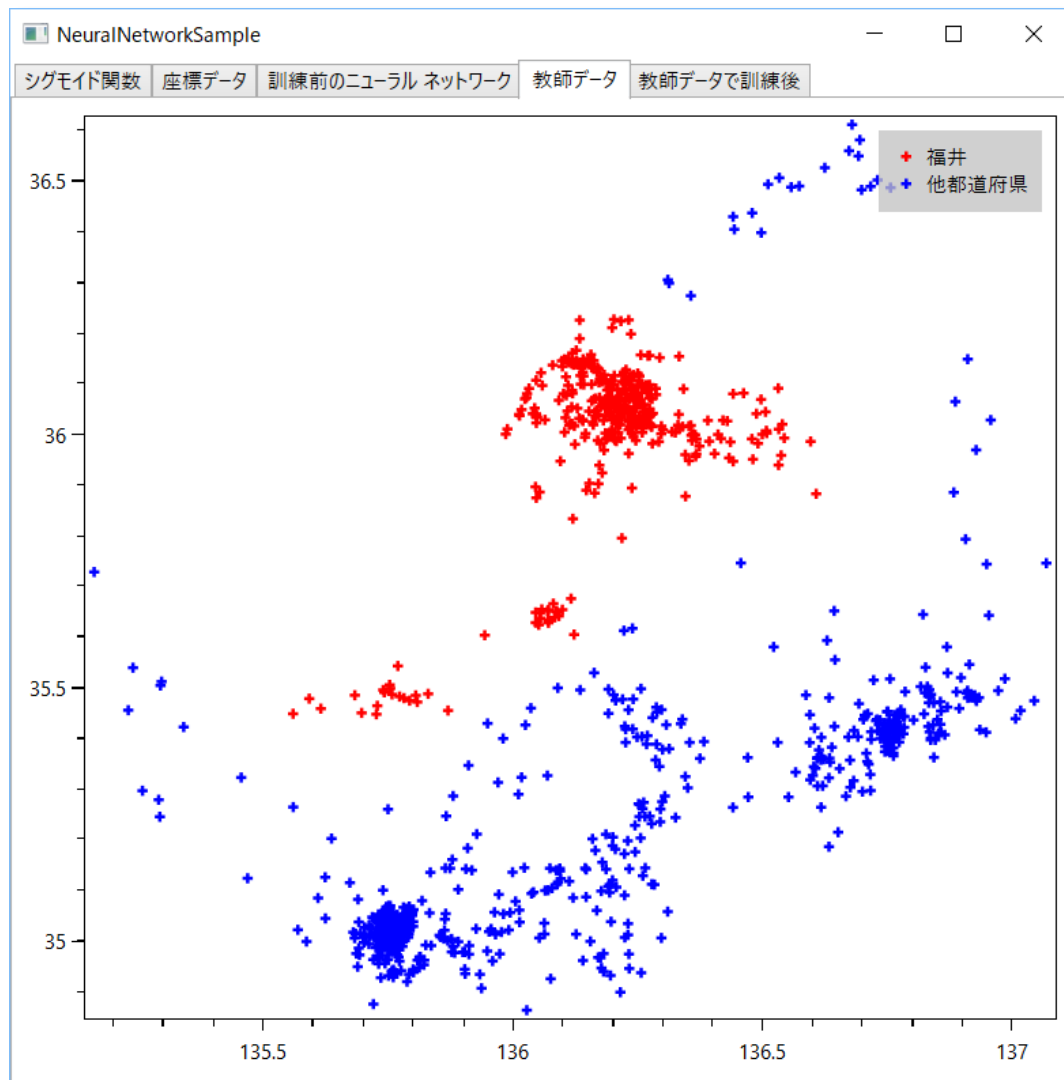
実行結果 (シグモイド関数)



実行結果 (座標データと訓練前)



実行結果 (教師データと訓練後)



訓練前の重みの値



入力層 → 中間層の重み

中間層 → 出力層の重み

```
neuralNetwork = new NeuralNetwork.Co neuralNetwork = new NeuralNetwork.Co
```

Load();

neuralNetwork {NeuralNetwork.Core.NeuralNetwork}		
inputLayer		null
inputLayerBias		1
inputWeight		{double[3, 2]}
[0, 0]	0	layer
[0, 1]	0	layerBias
[1, 0]	0	weight
[1, 1]	0	layer
[2, 0]	0	
[2, 1]	0	

Load();

neuralNetwork {NeuralNetwork.Core.NeuralNetwork}		
inputLayer		null
inputLayerBias		1
inputWeight		{double[3, 2]}
middleLayer		null
middleLayerBias		1
middleWeight		{double[3]}
[0]	0	Layer
[1]	0	
[2]	0	

訓練後の重みの値

入力層 → 中間層の重み

中間層 → 出力層の重み

```
neuralNetwork = new NeuralNetwork.Co neuralNetwork = new NeuralNetwork.C
```

半の変更

Load();

半の変更

neuralNetwork	{NeuralNetwork.Core.NeuralNetwork}
inputLayer	{double[3]}
inputLayerBias	1
inputWeight	{double[3, 2]}
[0, 0]	-9.0561516142426068
[0, 1]	-9.0561516142426068
[1, 0]	4.1573136650067823
[1, 1]	4.1573136650067823
[2, 0]	15.10374684620685
[2, 1]	15.10374684620685

半の変更

Load();

半の変更

neuralNetwork	{NeuralNetwork.Core.NeuralNetwork}
inputLayer	{double[3]}
inputLayerBias	1
inputWeight	{double[3, 2]}
middleLayer	{NeuralNetwork.Core.Neuron[2]}
middleLayerBias	1
middleWeight	{double[3]}
[0]	9.5080764145692847
[1]	9.5080764145692847
[2]	-3.0368706375335028

