Microsoft

AI61

# 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

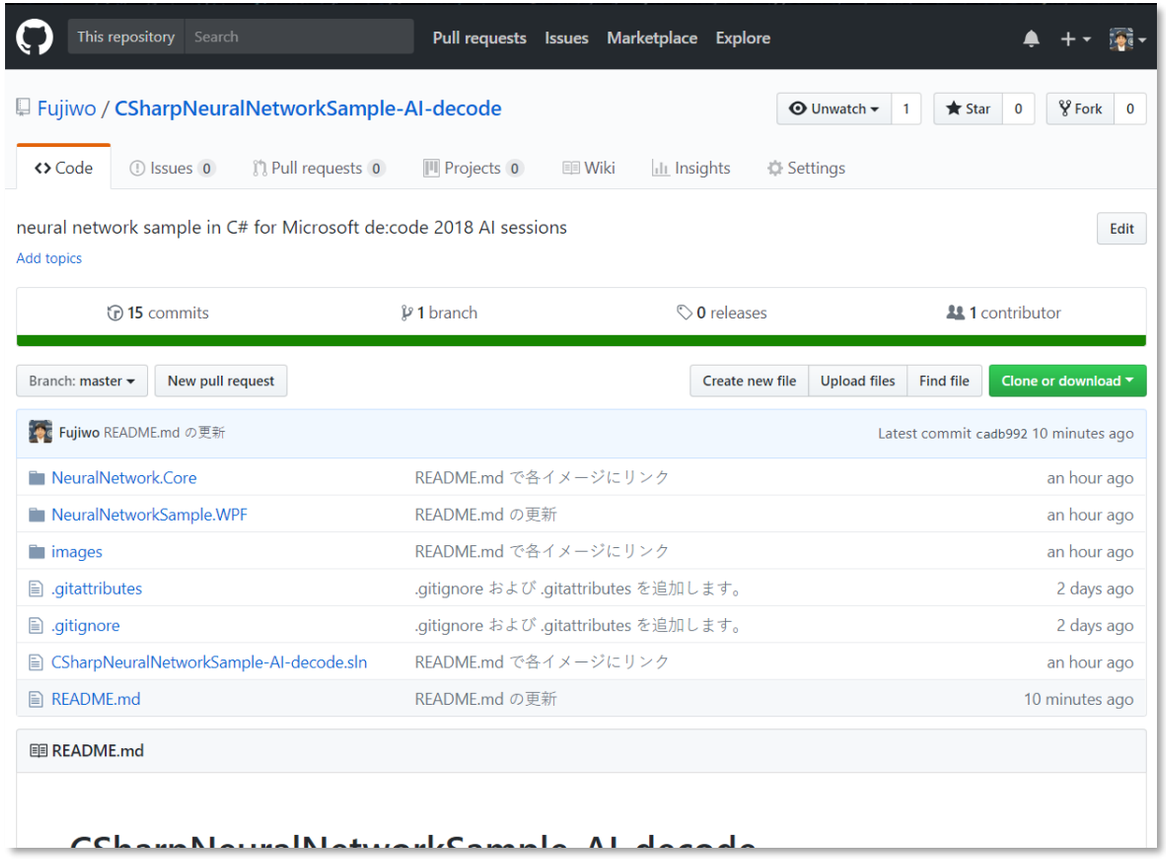
福井コンピュータホールディングス株式会社

小島 富治雄

de:code 2018

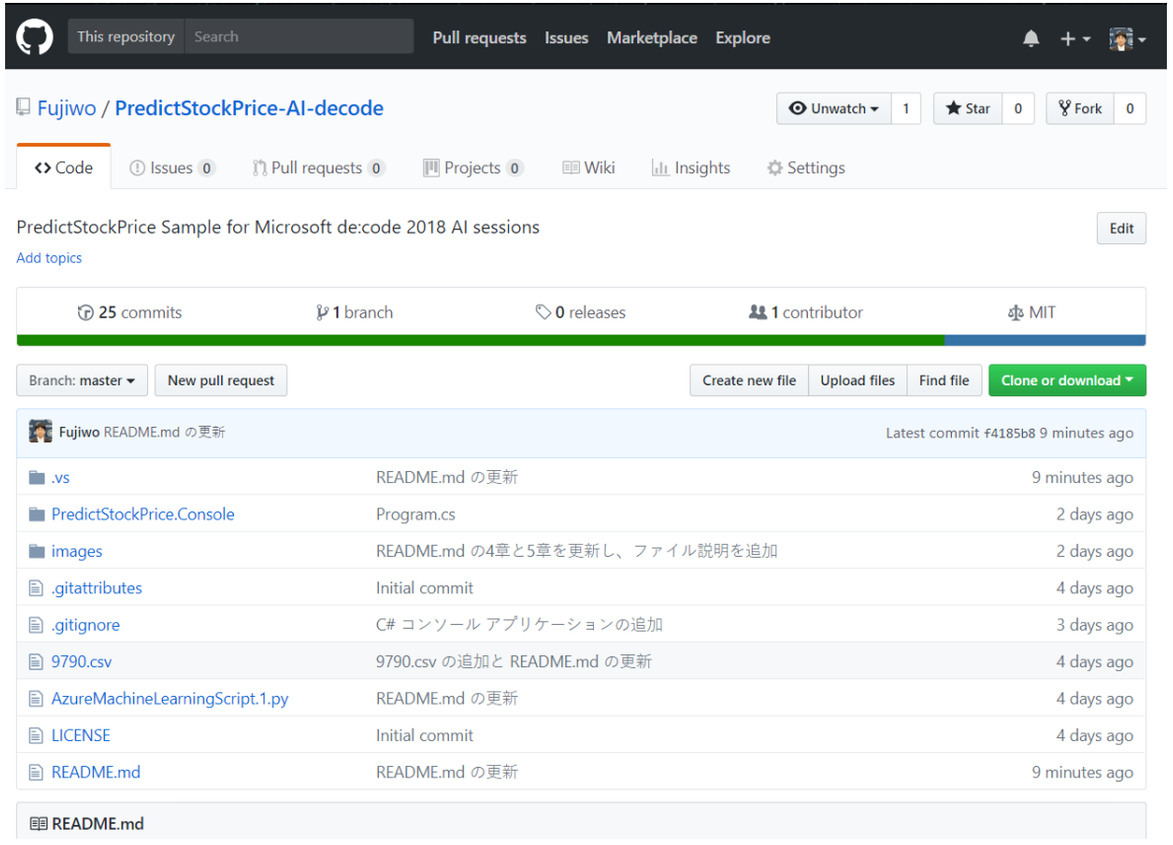
# ソース コードの場所

- 「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

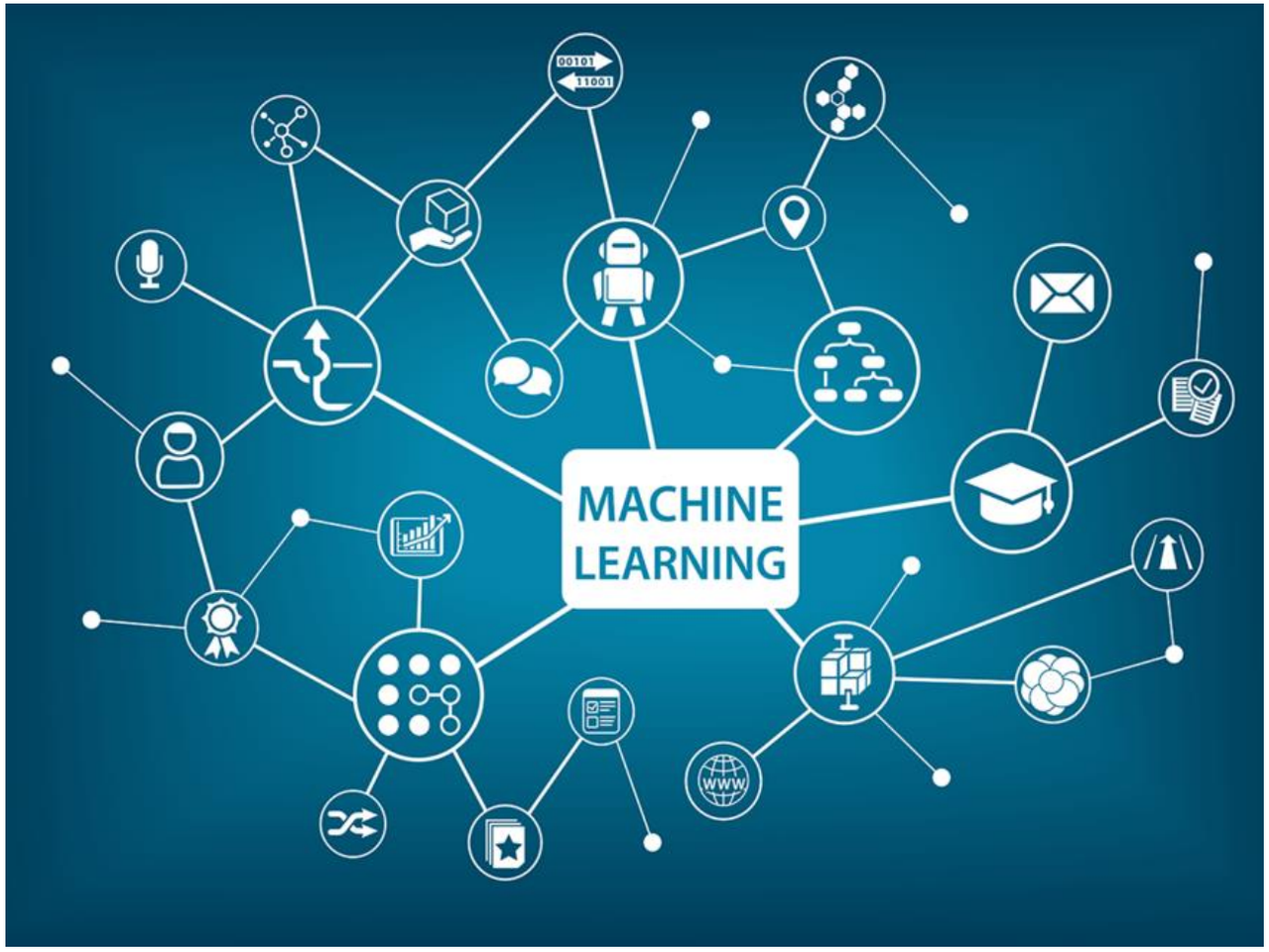


# 関連資料

- 「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



# 機械学習 (Machine Learning) とは



# 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)

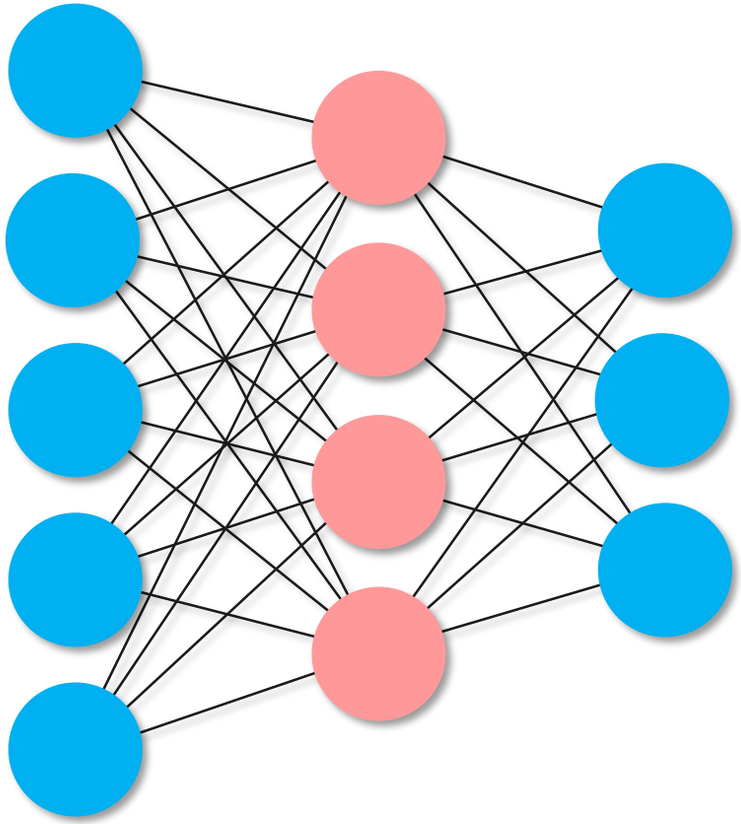
# ディープラーニング (深層学習: Deep Learning)

- 機械学習の一種
- 通常のものと違って多層になった
  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

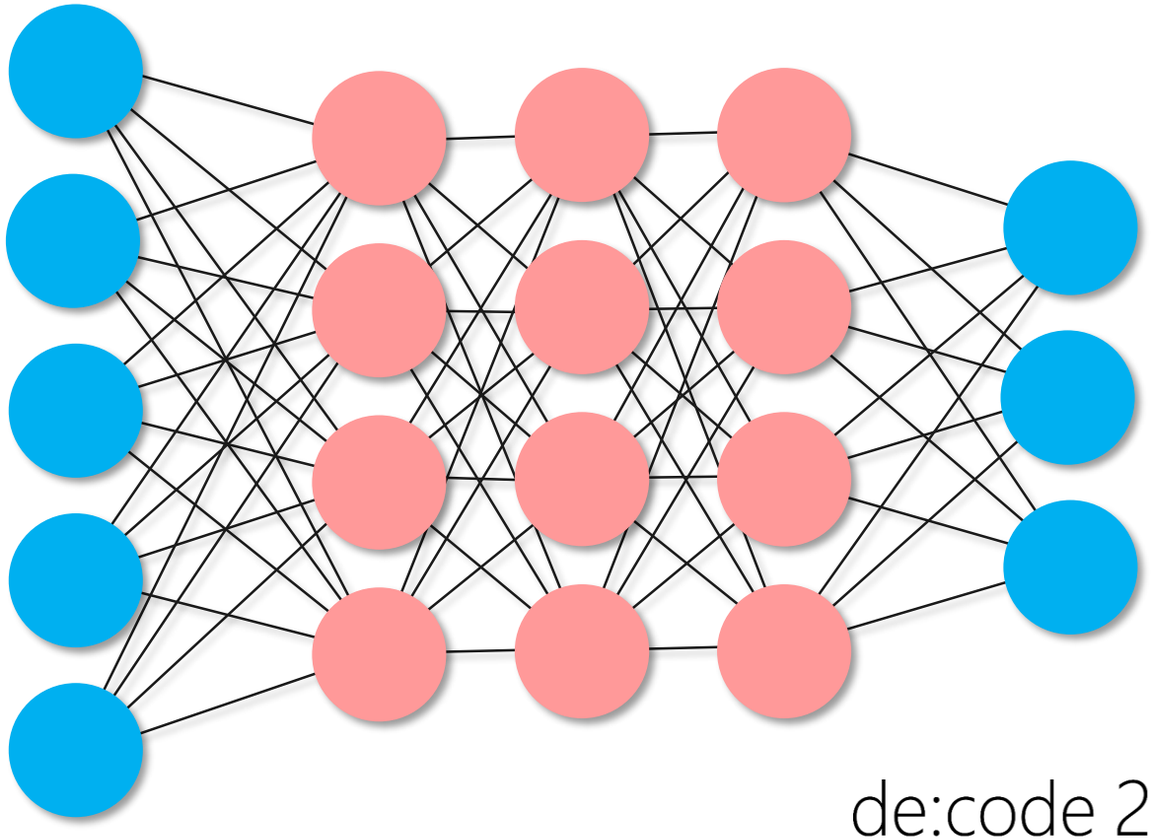
ニューラル ネットワーク

入力層　　中間層　　出力層

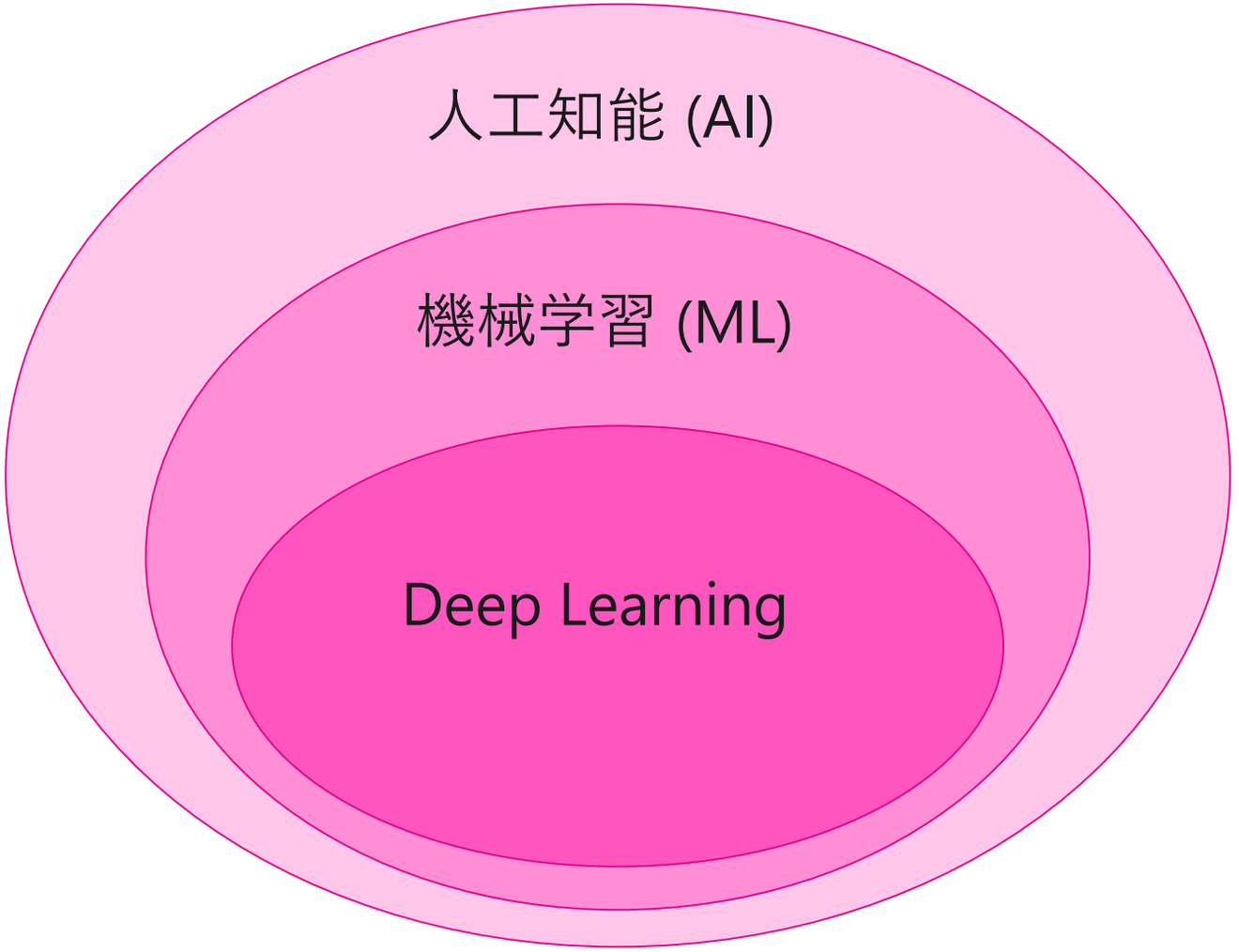


ディープ ラーニング

入力層　　中間層　　出力層

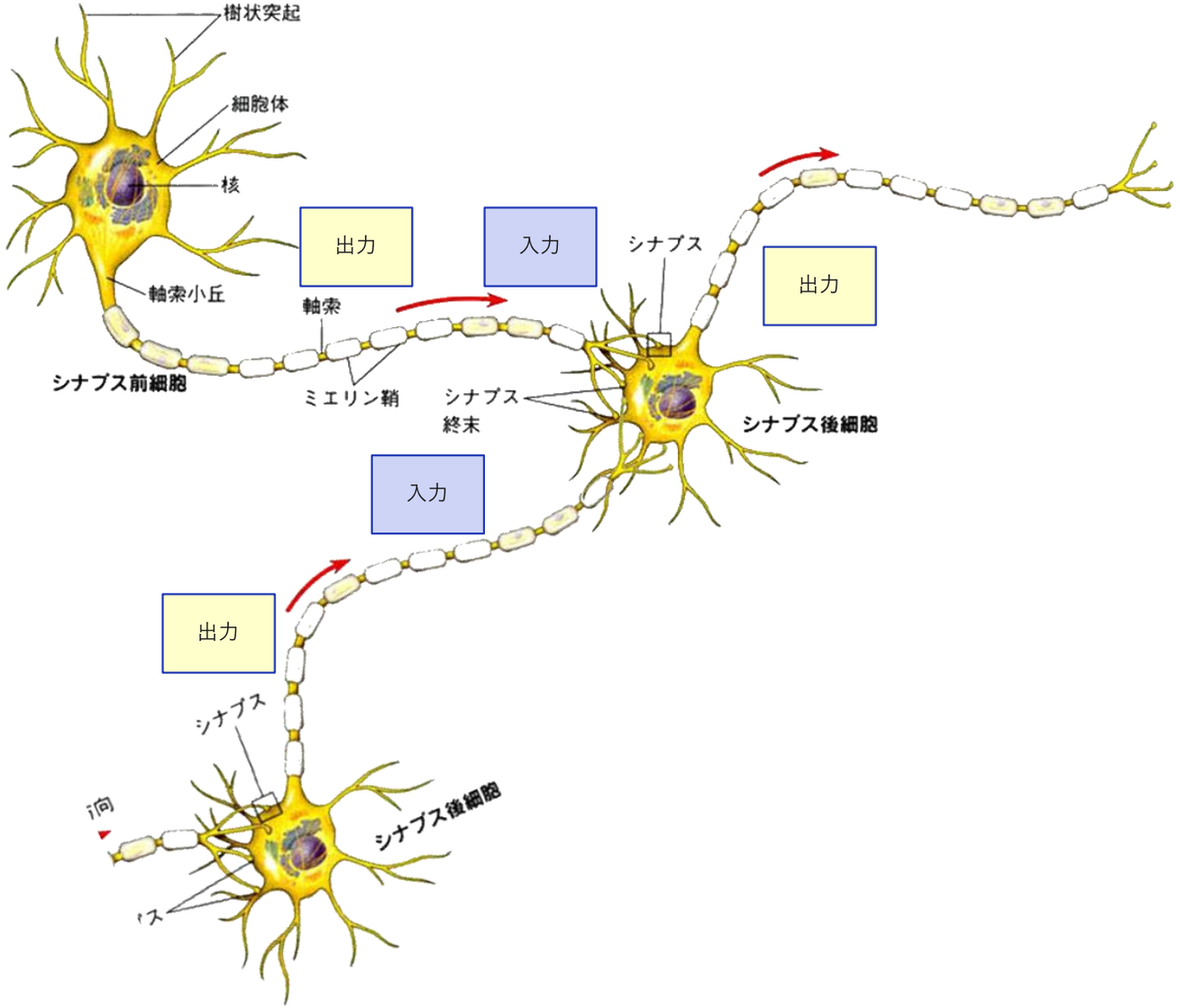


# 人工知能と機械学習



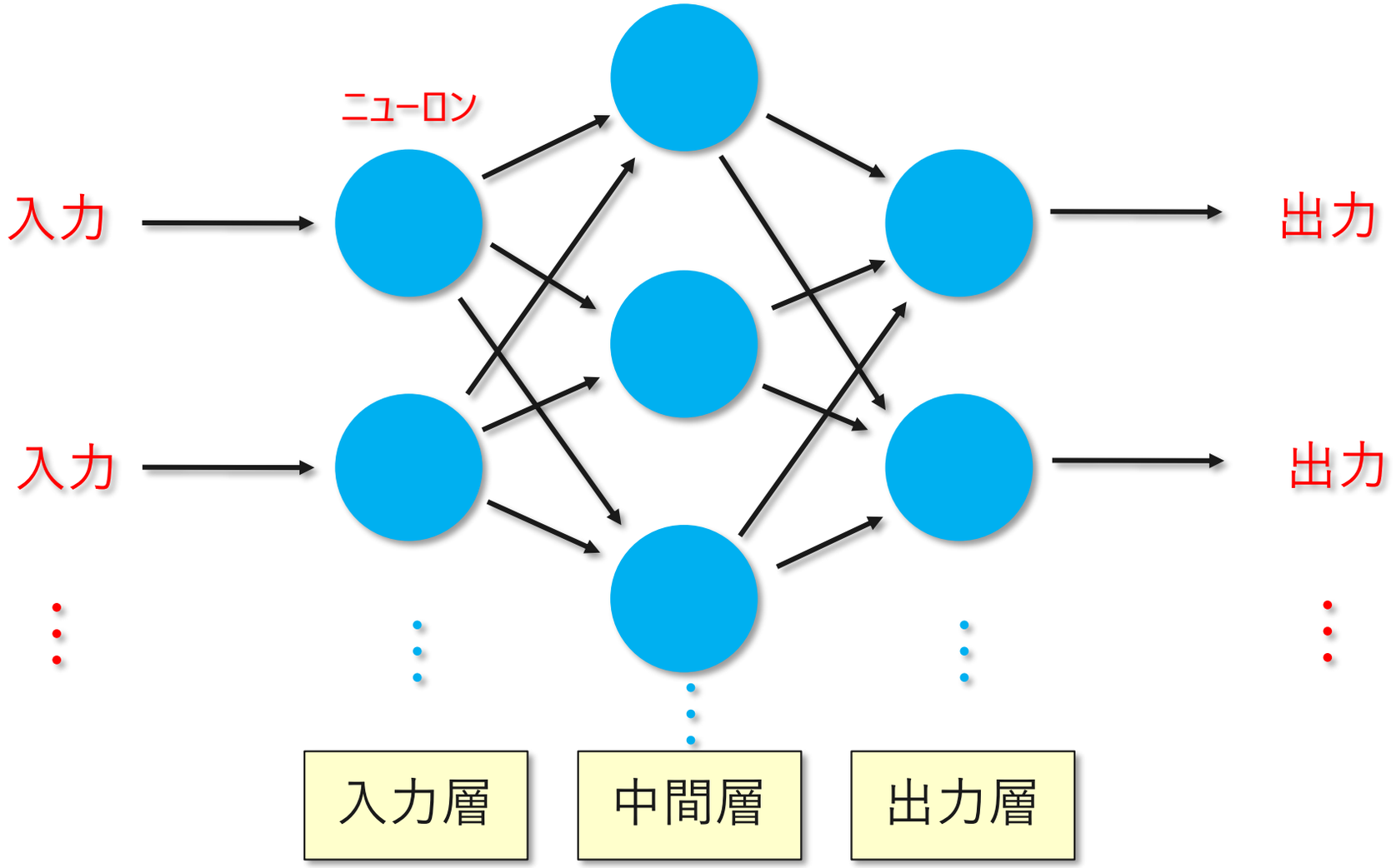
# ニューラル ネットワークとは

# 神経細胞のネットワーク

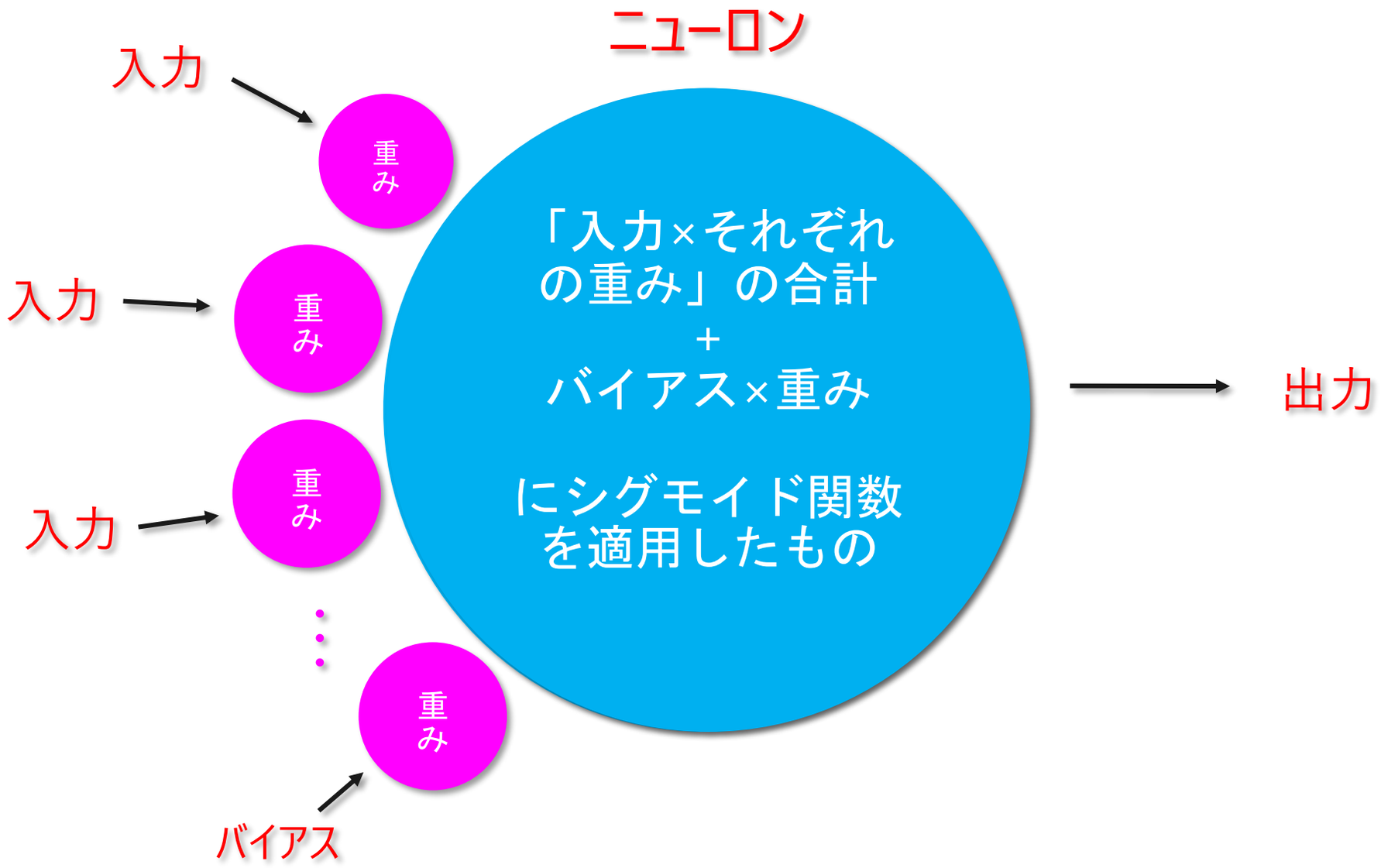


# ニューラル ネットワーク

## 神経細胞のネットワークを模倣



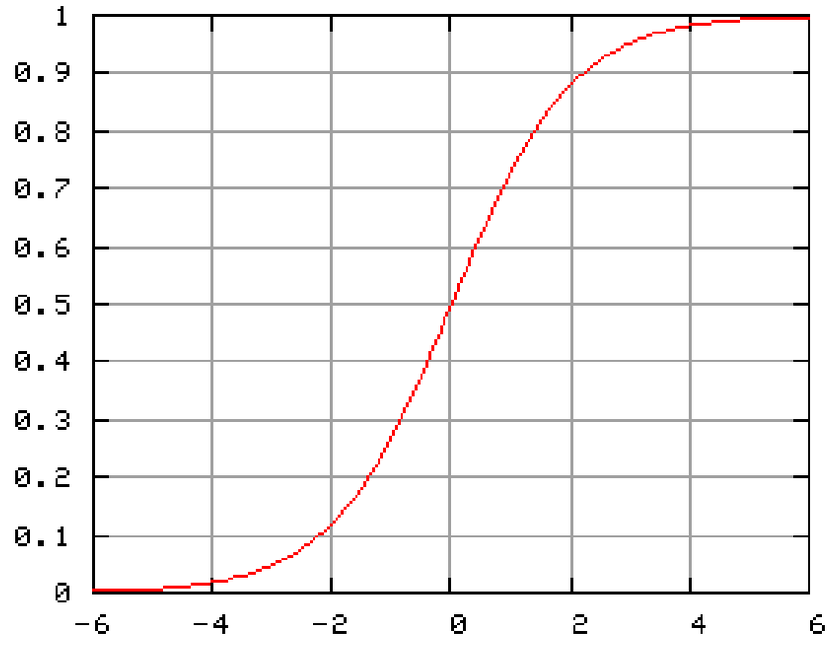
# 個々のニューロン


ニューロン
入力
入力
入力
重み
重み
重み
重み
バイアス
「入力×それぞれの重み」の合計
+
バイアス×重み
にシグモイド関数を適用したもの
出力

# シグモイド関数

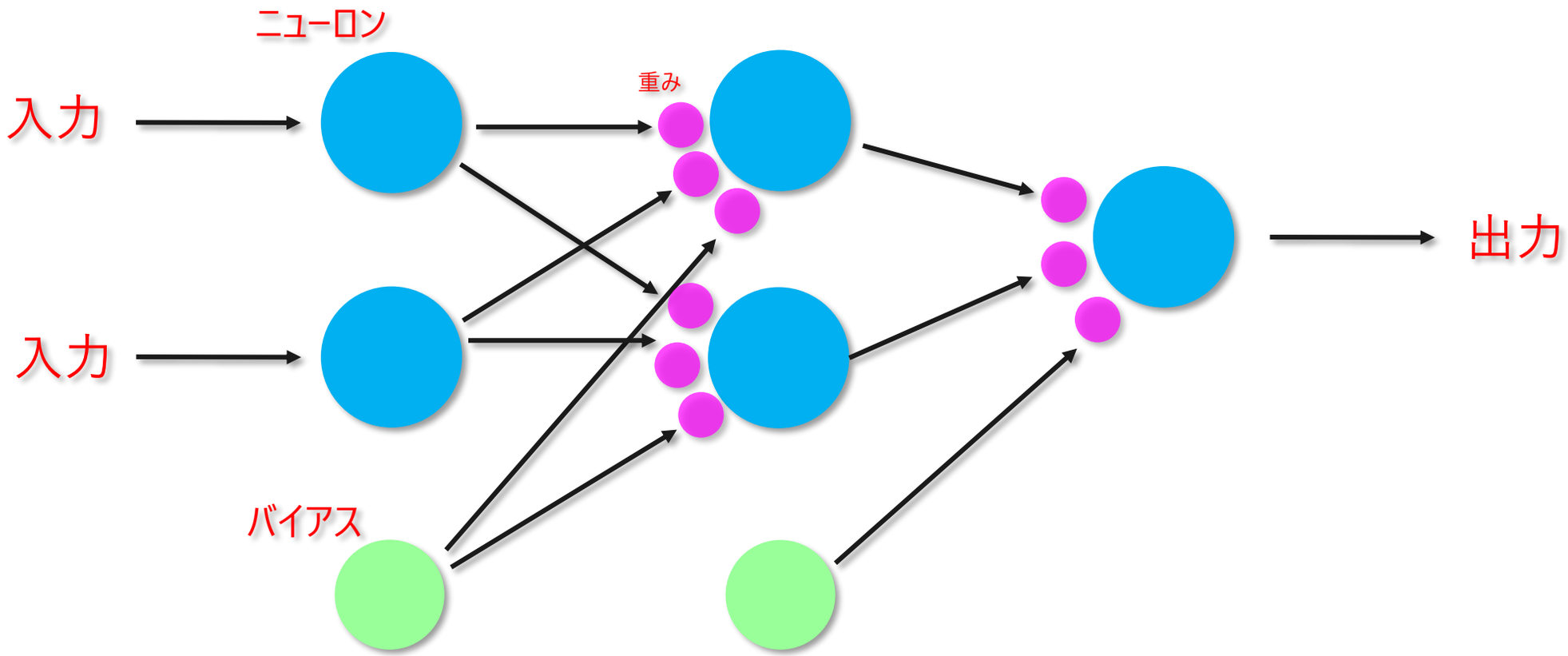
神経細胞が入力の合計を出力に
するときの性質をモデル化

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

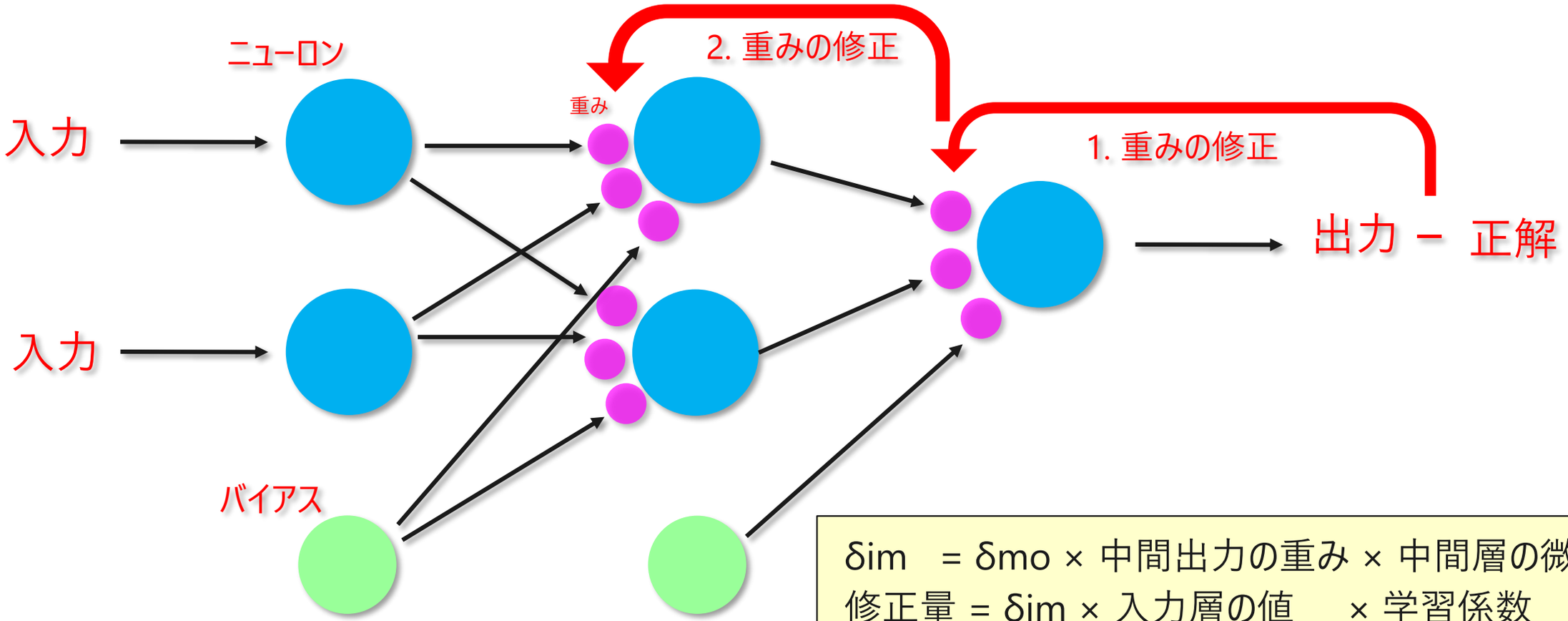


# ニューラル ネットワークによる分類

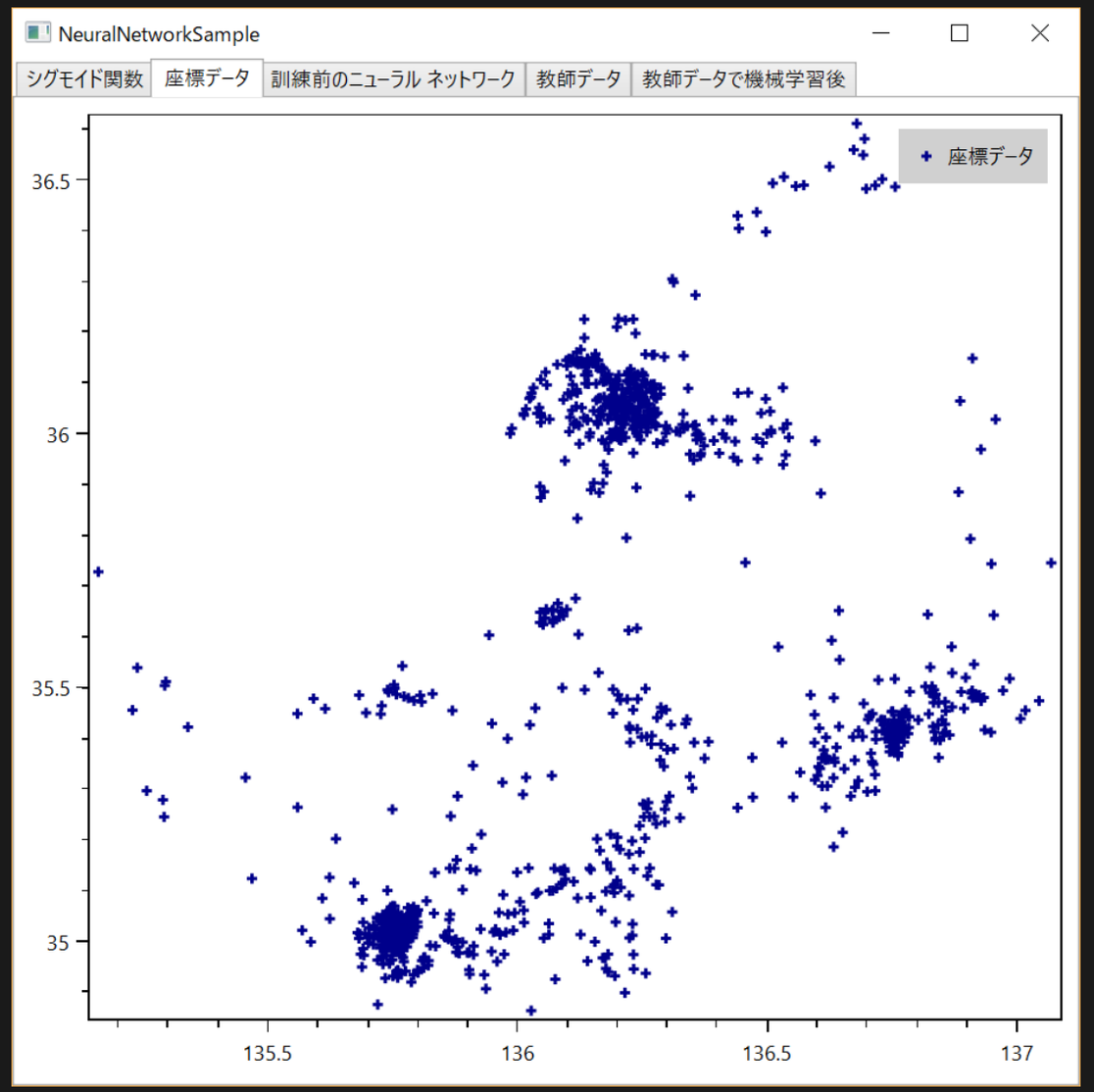
# 今回作成するニューラル ネットワーク



# 今回のニューラル ネットワークの訓練



# Demo


NeuralNetworkSample
シグモイド関数
座標データ
訓練前のニューラル ネットワーク
教師データ
教師データで機械学習後
座標データ
36.5
36
35.5
35
135.5
136
136.5
137

# 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;
}
```

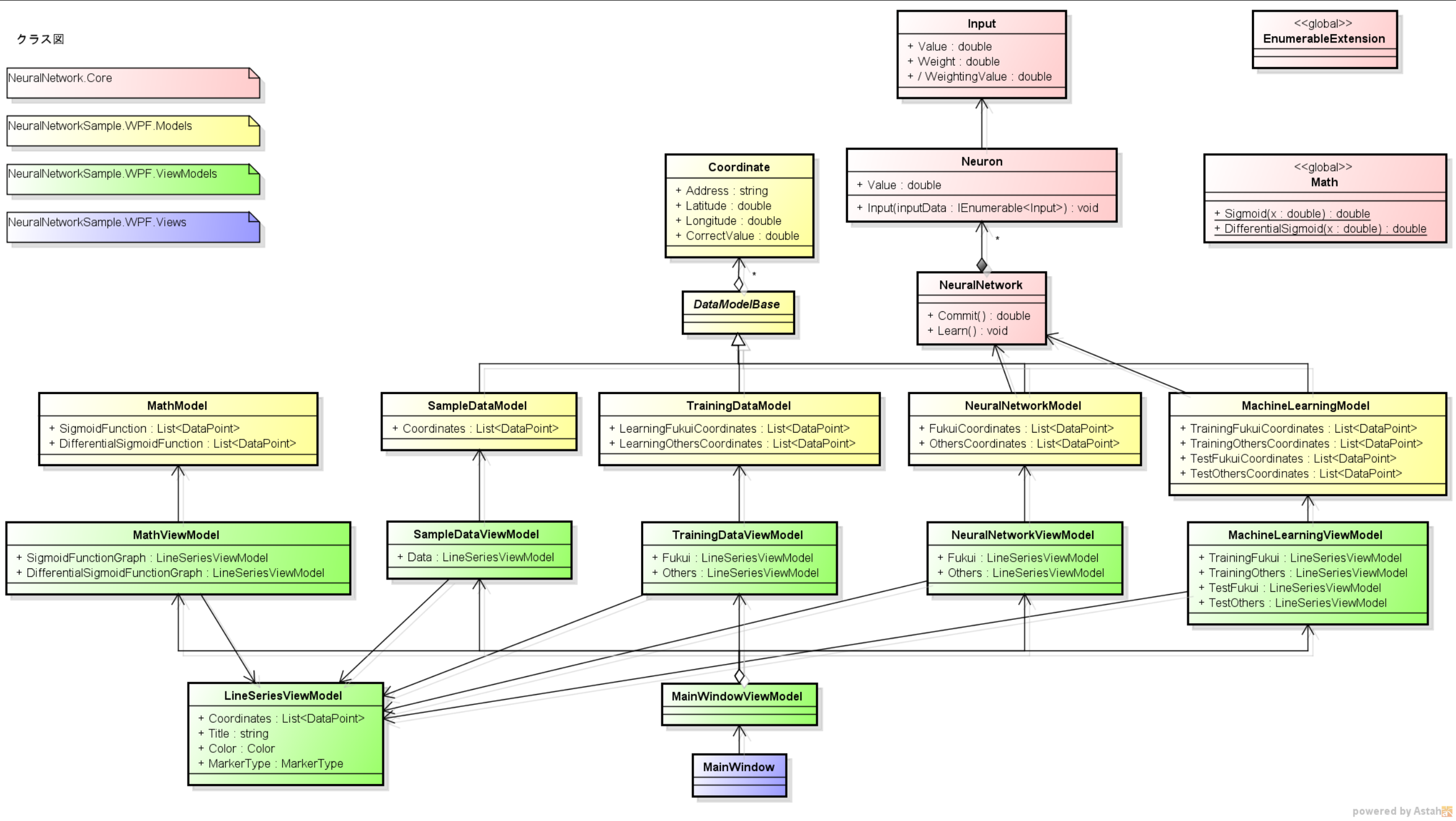
# C# によるニューラル ネットワークの実装 (続き)

```
// 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
    // δmo = (出力値 - 正解値) x 出力の微分
    var daltaMO = (correctValue - outputData) * outputData * (1.0 - outputData);
    var oldMiddleWeight = middleWeight.Clone() as double[];
    // 修正量 = δmo x 中間層の値 x 学習係数
    middleLayer.For((index, neuron) => middleWeight[index] += neuron.Value * daltaMO * learningRate);
    middleWeight[2] += middleLayerBias * daltaMO * learningRate;

    // 中間層→入力層
    // δim = δmo x 中間出力の重み x 中間層の微分
    var deltaIM = middleLayer.IndexSelect(index =>
 daltaMO * oldMiddleWeight[index] * middleLayer[index].Value * (1.0 - middleLayer[index].Value)).ToArray();
     // 修正量 = δim x 入力層の値 x 学習係数
     inputWeight.For((row, column, _) =>
         inputWeight[row, column] += inputLayer[row] * deltaIM[column] * learningRate);
}
```

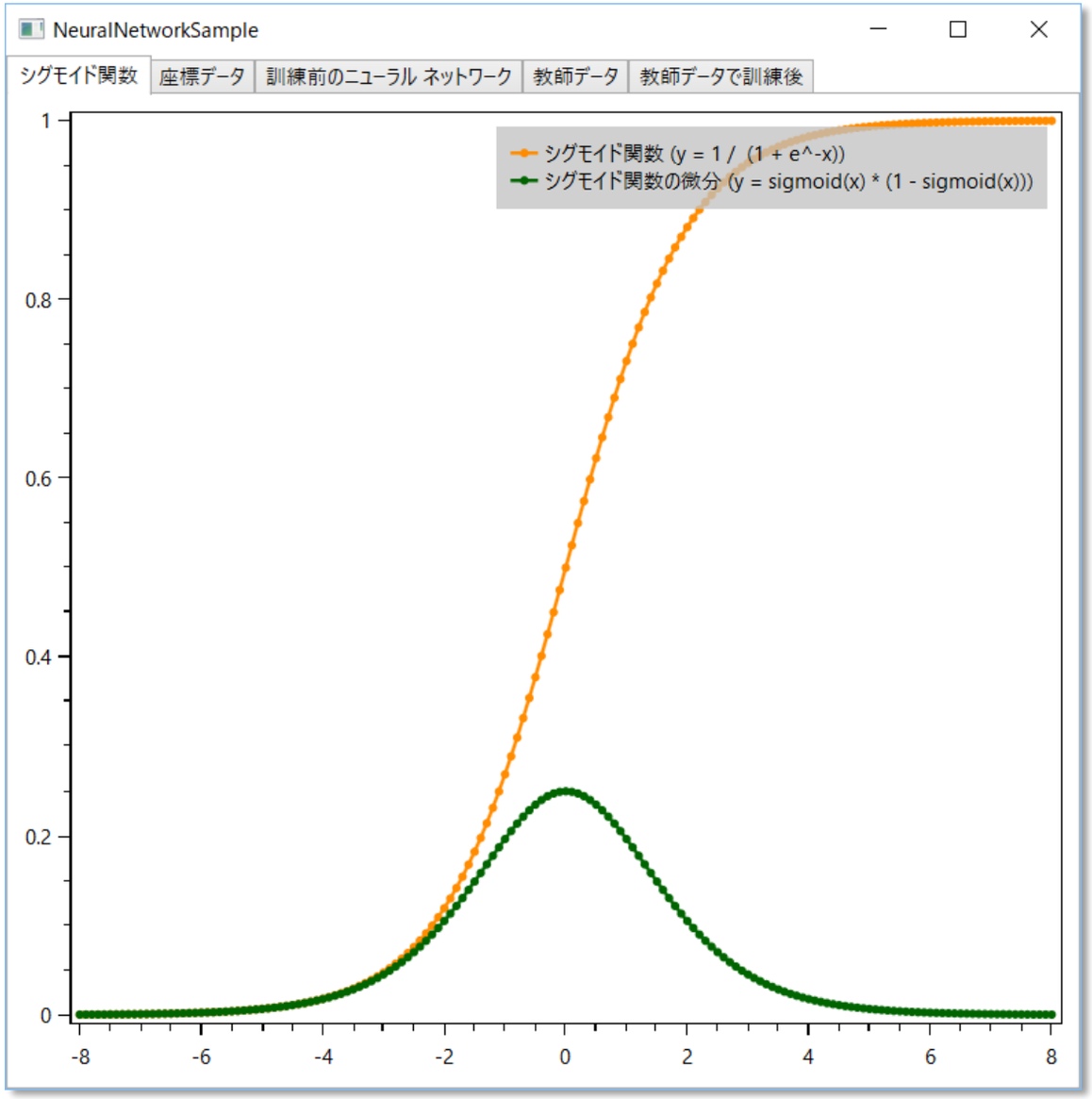
# C# によるニューラル ネットワークの実装 (続き)

```
    // 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] });
}
```

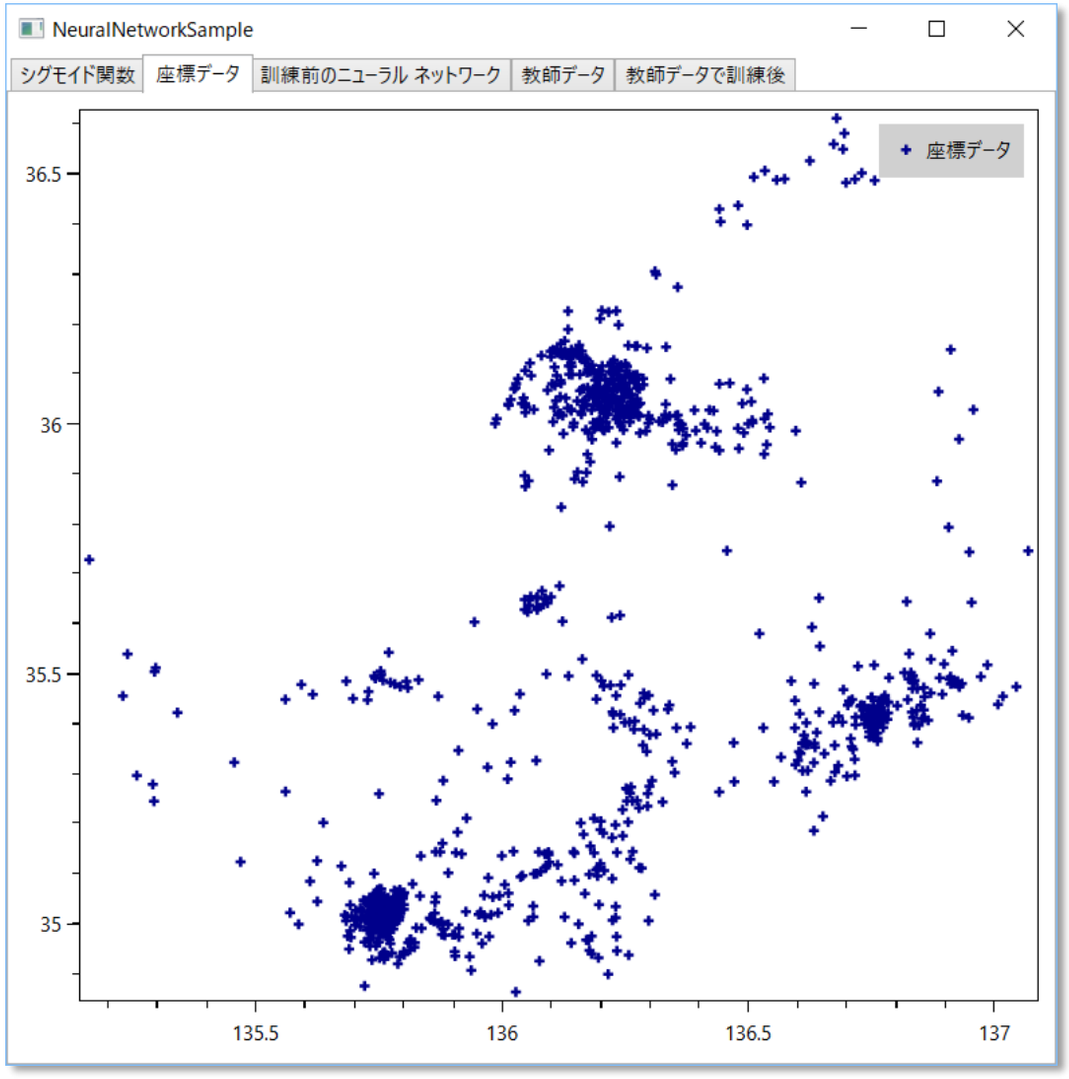
クラス図

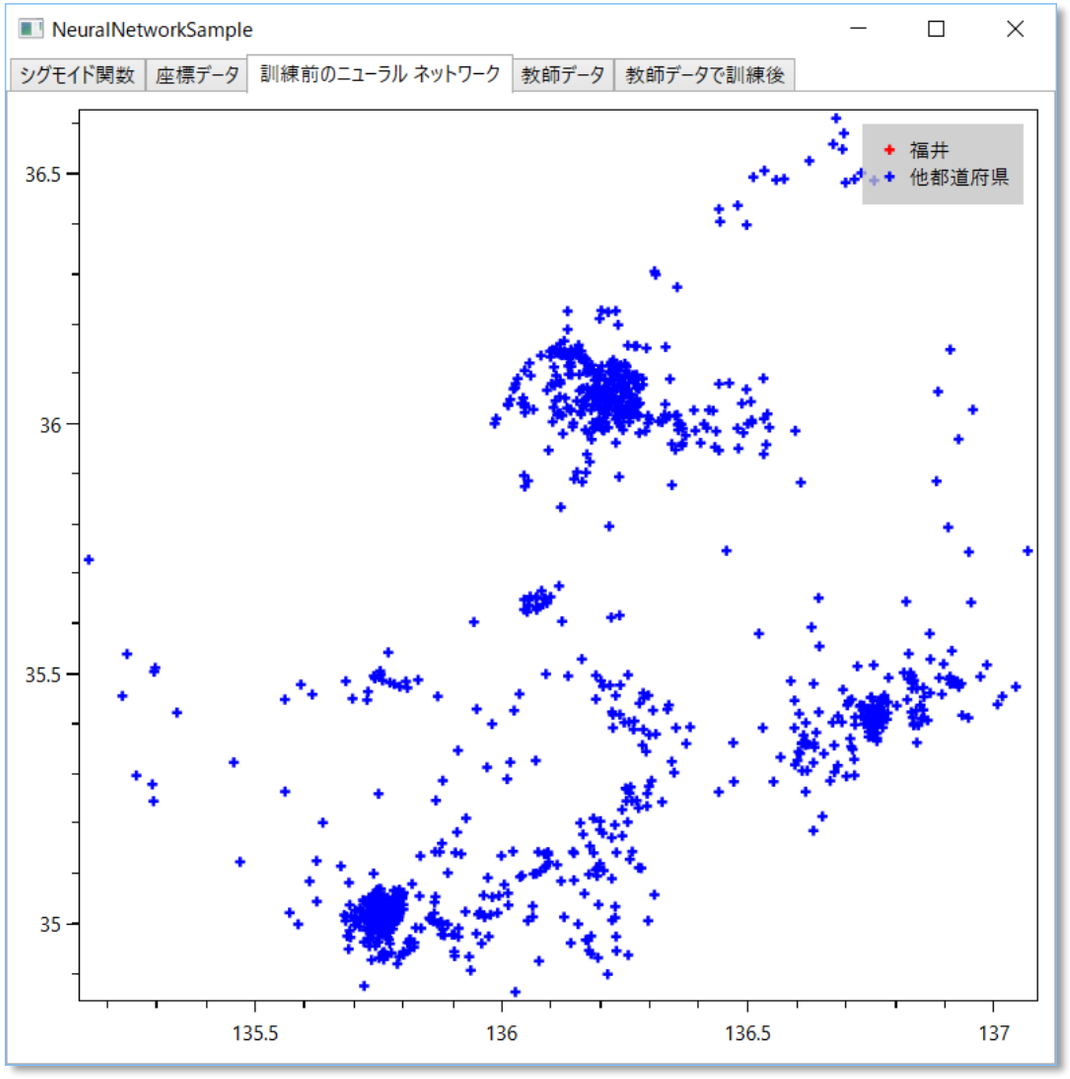


# 実行結果 (シグモイド関数)

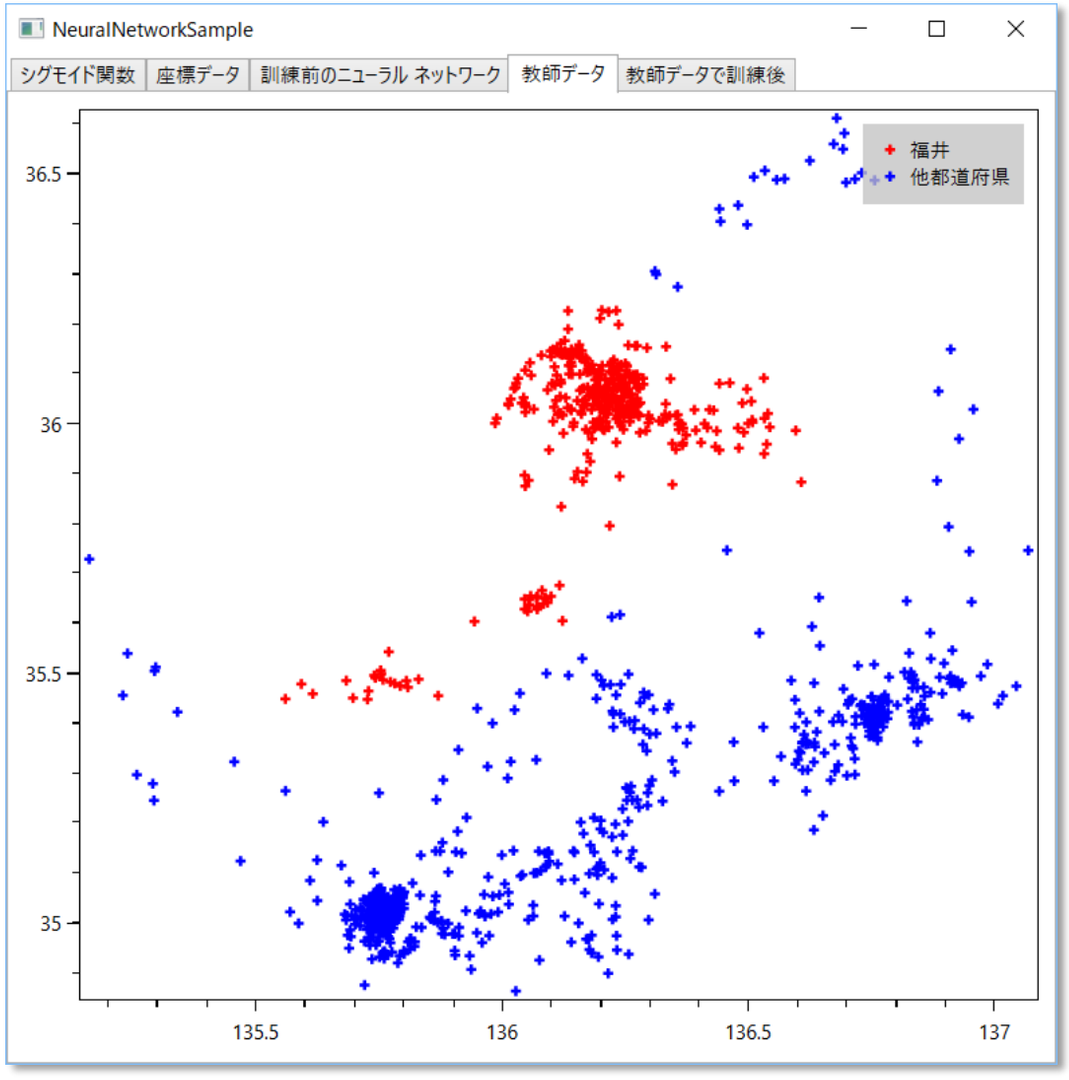


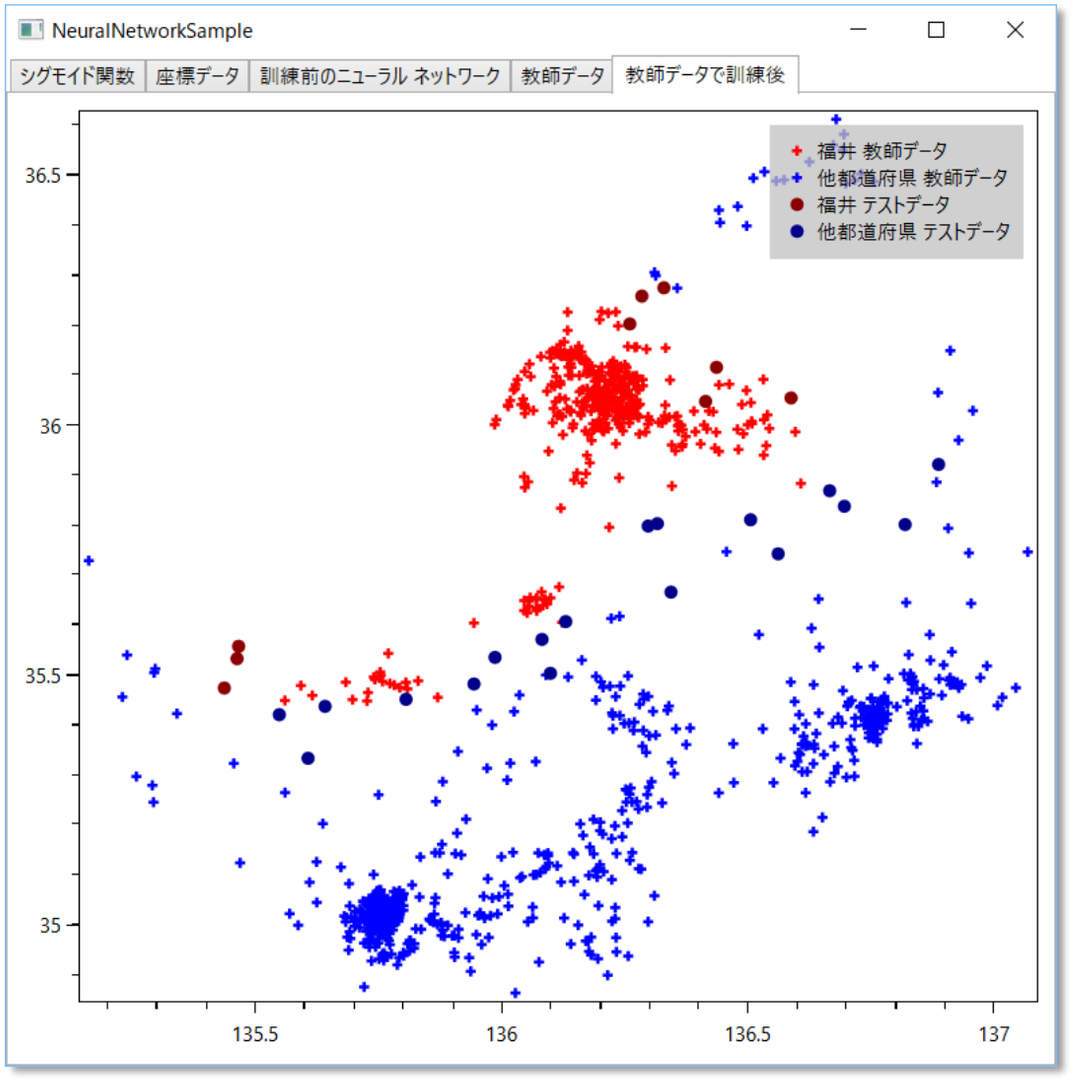
# 実行結果 (座標データと訓練前)



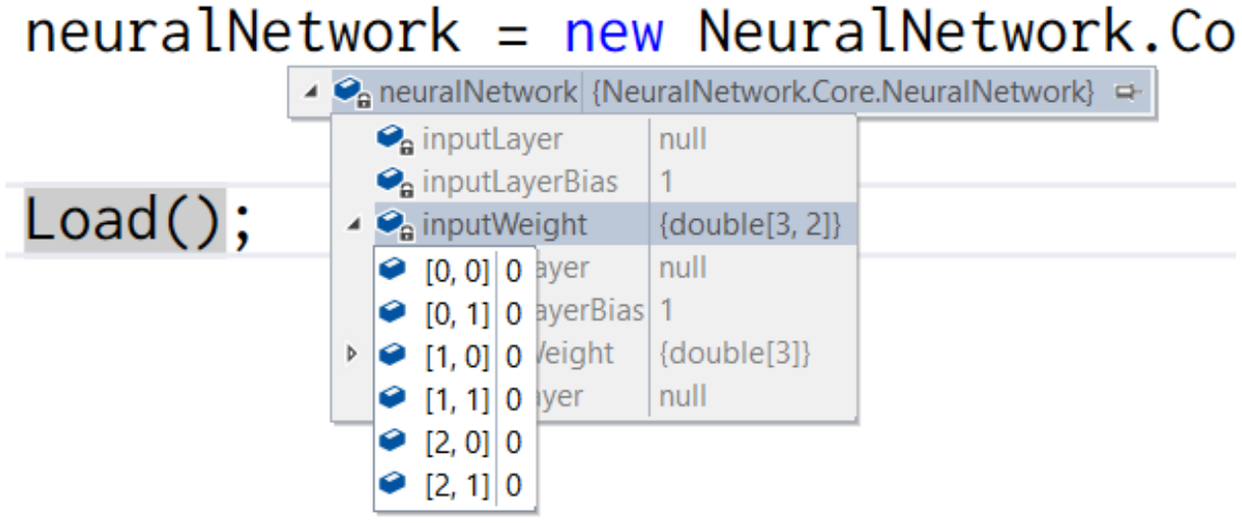
# 実行結果 (教師データと訓練後)



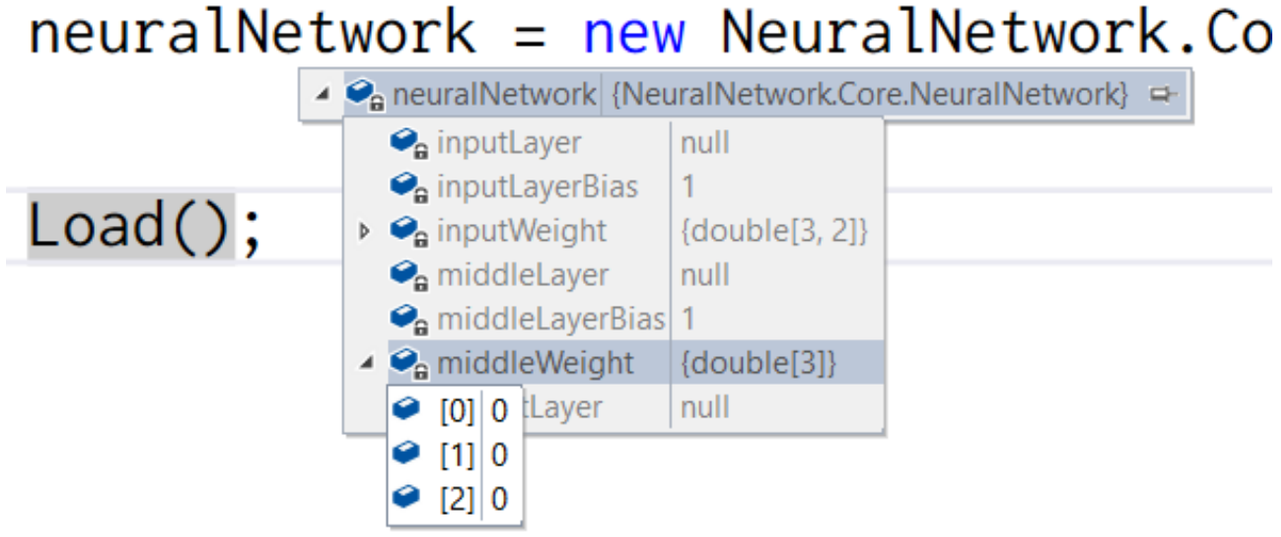



# 訓練前の重みの値

入力層 → 中間層の重み



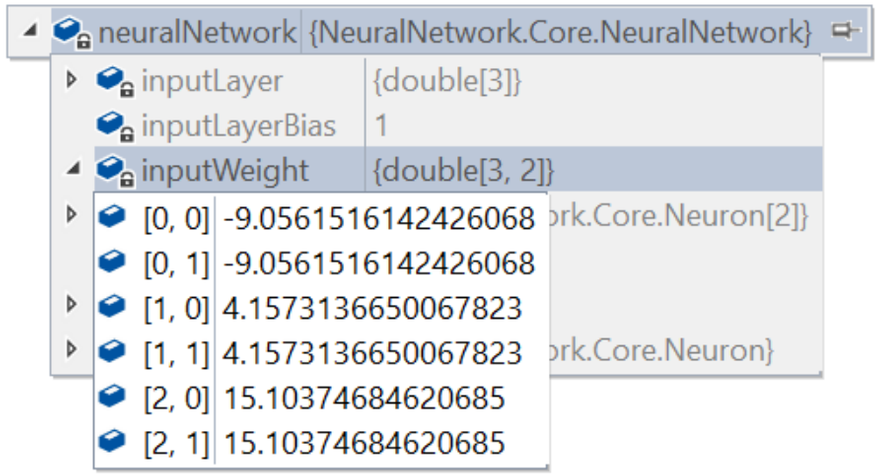

中間層 → 出力層の重み



# 訓練後の重みの値

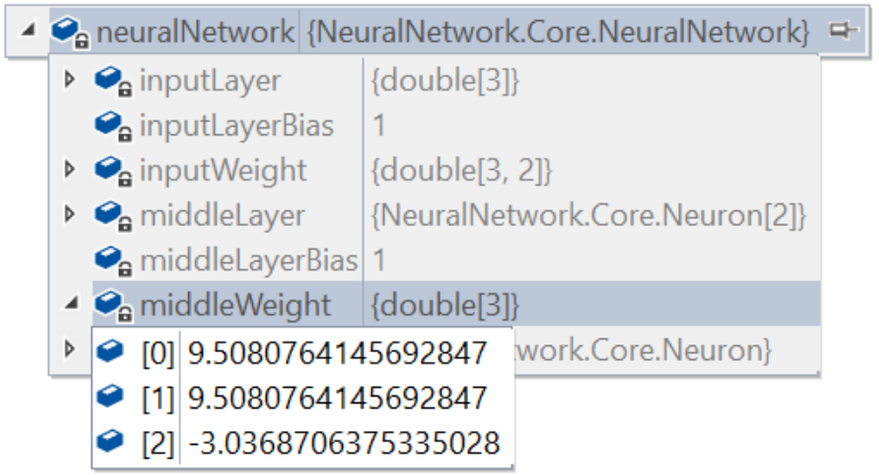
入力層 → 中間層の重み

```
neuralNetwork = new NeuralNetwork.Co
牛の変更
Load();
牛の変更
```




中間層 → 出力層の重み

```
neuralNetwork = new NeuralNetwork.C
牛の変更
Load();
牛の変更
```



Microsoft



de:code 2018