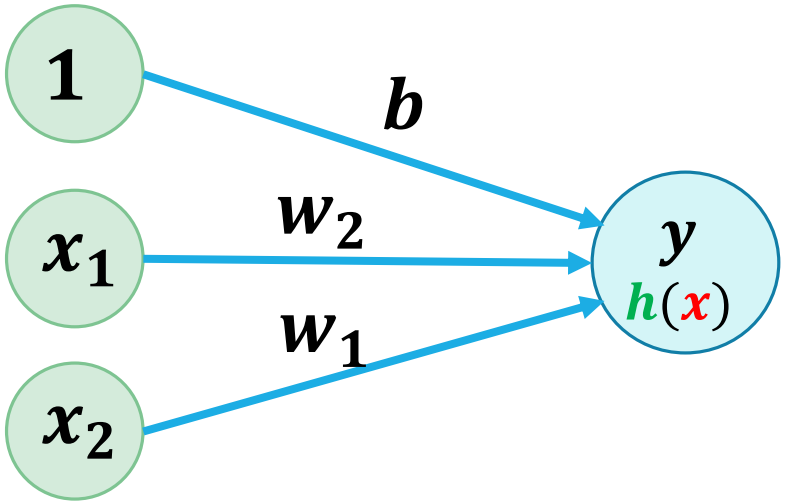
深層学習入門
#3 ニューラルネットワークと活性化関数

たくのろじぃ / takunology

パーセプトロンの関数表記

- 閾値を超えたら1を出力する関数
- パラメータ：入力(x_1, x_2), 重み(w_1, w_2), バイアス(b), 閾値 $= 0$



$y = h(x)$ という関数を導入

入力値 x はパラメータで決まる

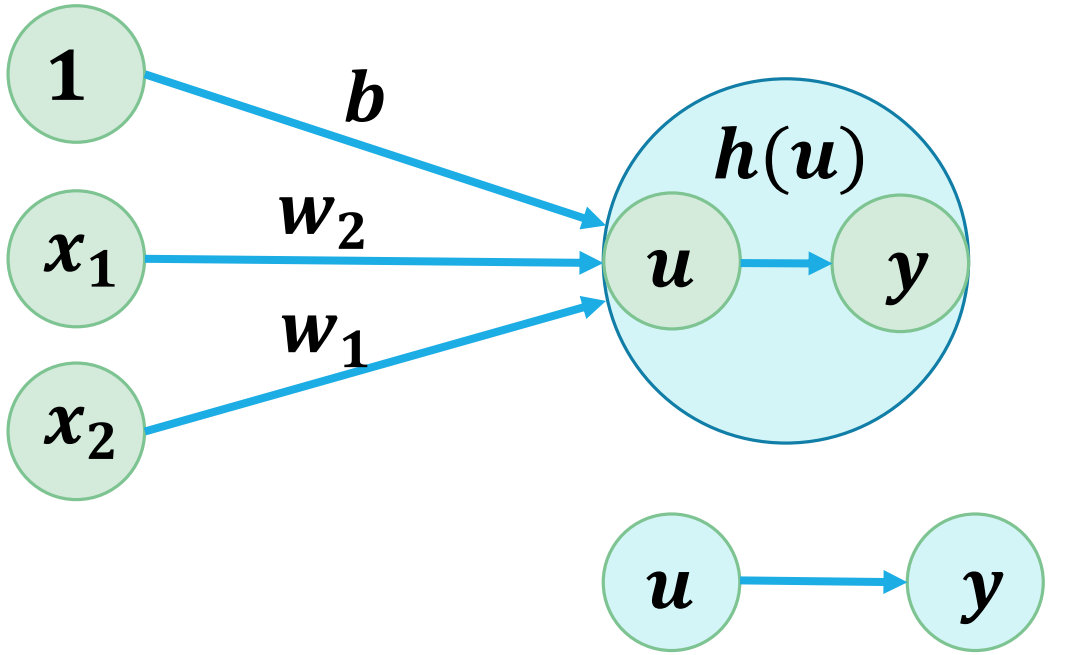
$$y = h\,(b + x_1 w_1 + x_2 w_2)$$

$$h_{(x)} = \begin{cases} 0\ (x \leq 0) \\ 1\ (x > 0) \end{cases}$$

関数 $h_{(x)}$ によって入力（総和）が変換される　→　変換する関数：活性化関数

ニューロンのプロセス

- ニューロンは入力と出力で構成される
- 入力パラメータは u (1つのニューロン)としてまとめる
- 最終的には u と y の２つのニューロンで構成できる



u を入力パラメータにする

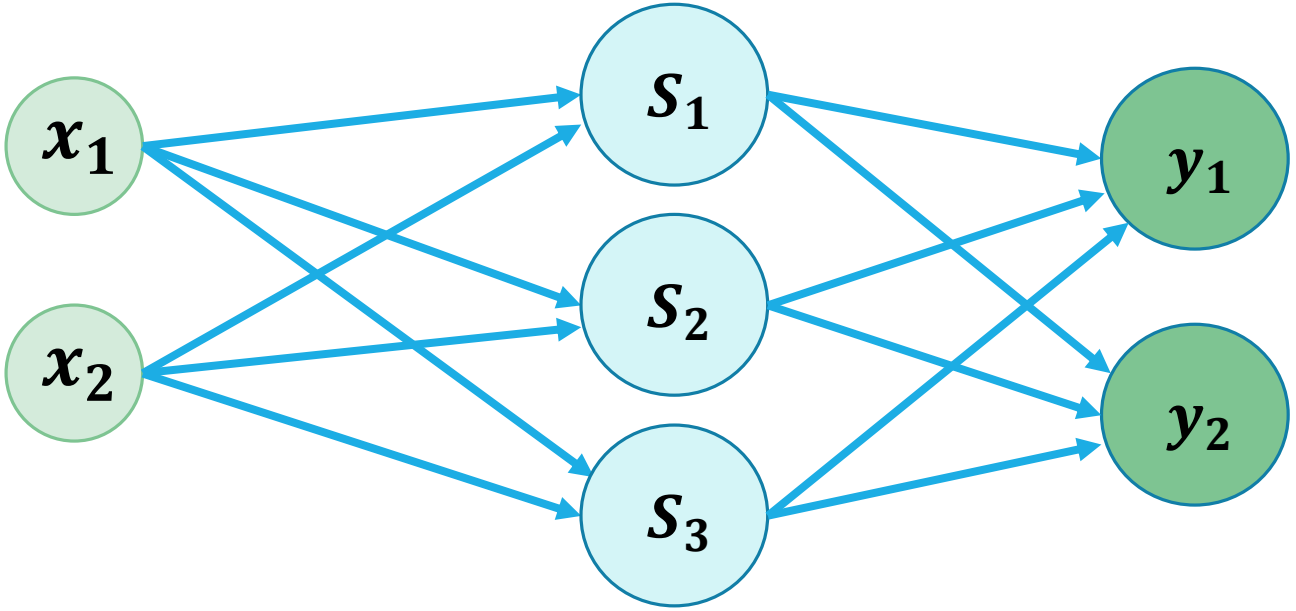
$$u = b + x_1 w_1 + x_2 w_2$$

$$y = h(u)$$

$h(u)$ は活性化関数という

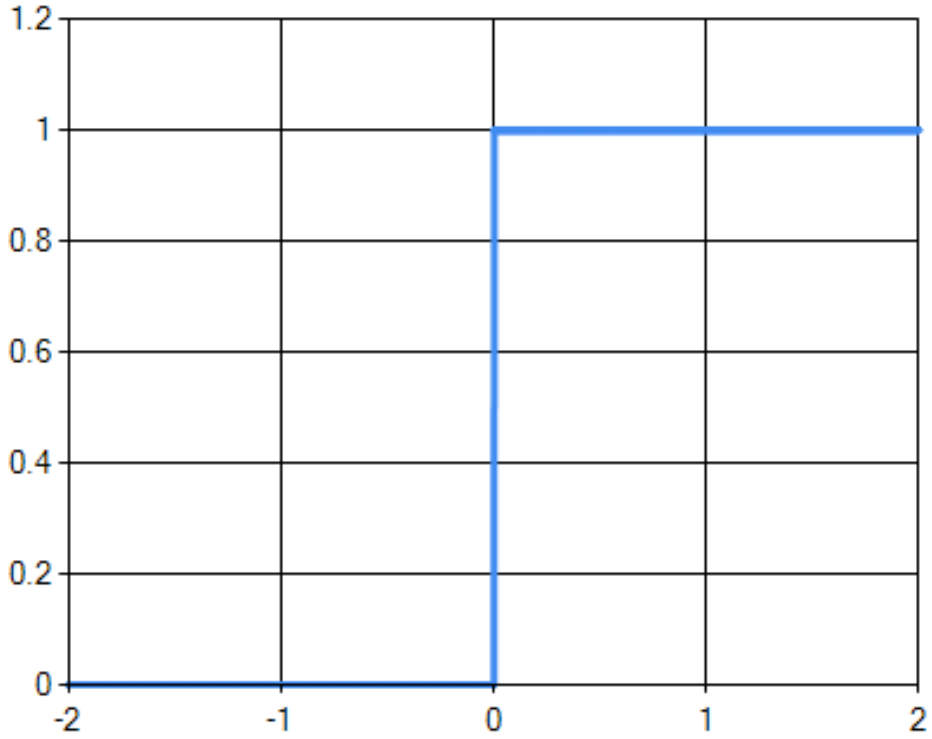
ニューラルネットワーク

- 隠れ層のノード（ニューロン）を1つ増やすと、 出力層への3入力になる
- S, y は重みとバイアス、閾値が1つになった関数として扱う
- 多層パーセプトロンに活性化関数を適用したものが、ニューラルネットワークになる



活性化関数 #1

- ニューロンを発火させる関数 $h(u)$ を決める（ニューロンを活性化させる）
- 活性化 ≒ 閾値の境界を決めるもの
- パーセプトロンはステップ関数をもとに閾値を決めていた



ステップ関数

$$h_{(x)} = \begin{cases} 0\ (x \leq 0) \\ 1\ (x > 0) \end{cases}$$

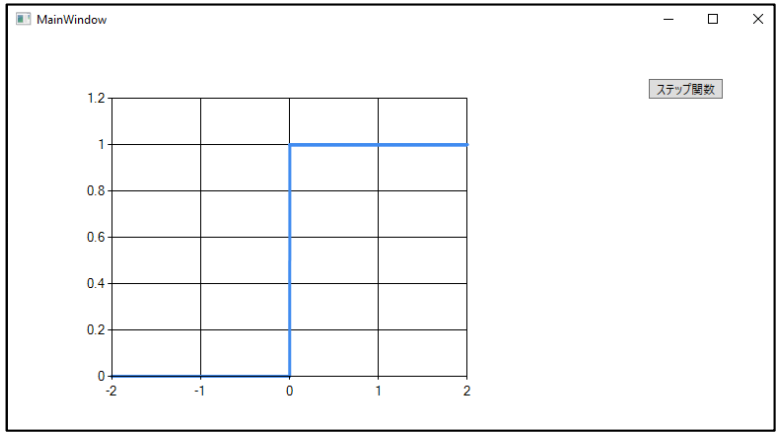
入力が 0 以下　　：0 を出力
入力が 0 を超える：1 を出力

ハンズオン ＃1

- C# (WPFアプリケーション) で ステップ関数のグラフを描画する
- グラフの描画は Windows Forms の Chart を参照

```
<Grid>
    <Button Content="ステップ関数" Hori
    <Grid HorizontalAlignment="Left" He
        <WindowsFormsHost>
            <wfc:Chart Name="Graph"/>
        </WindowsFormsHost>
    </Grid>
</Grid>
```

XAML デザイン



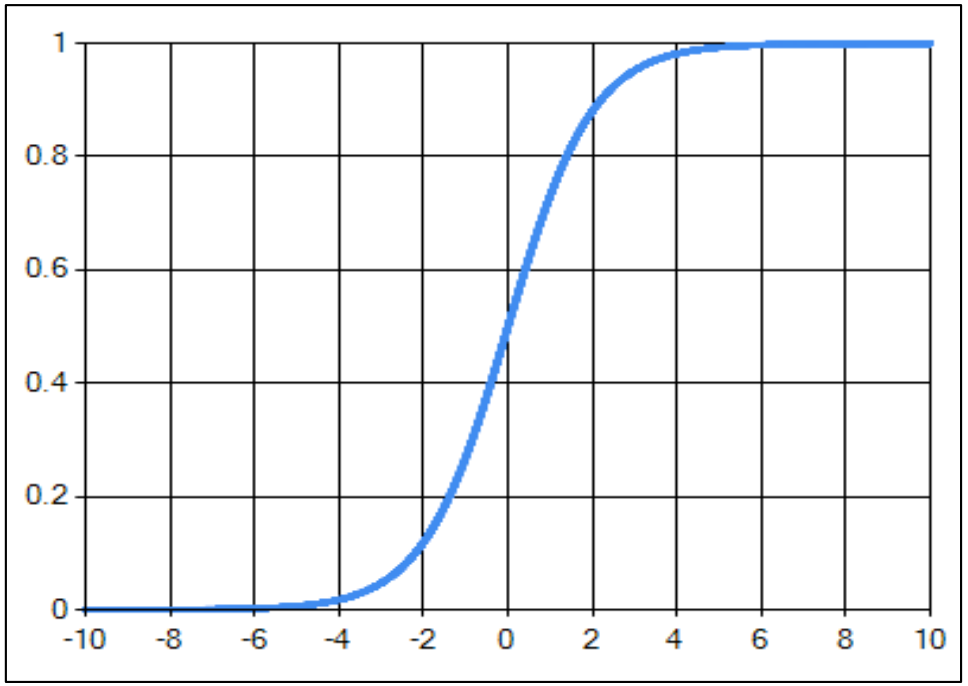
実行結果

```
private void Button_Click(object sender, RoutedEventArgs e)
{
    var windowsFormsHost = (WindowsFormsHost)GraphArea.Children[0];
    var graph = (Chart)windowsFormsHost.Child;

    // ChartArea追加
    graph.ChartAreas.Add("Graph1");
    // Seriesの作成と値の追加
    Series seriesStep = new Series();
    seriesStep.ChartType = SeriesChartType.Line;
    graph.ChartAreas[0].AxisX.Maximum = 2; //そのグラフの最小値
    graph.ChartAreas[0].AxisX.Minimum = -2; //そのグラフの最大値
    graph.ChartAreas[0].AxisX.Interval = 1; //目盛りの間隔（最大値と

    int y = 0; //ステップ関数の初期値
    for (double x = -2; x <= 2; x = x + 0.001)
    {
        if(x > 0)
        {
            y = 1;  //0を超えたら1を出力
        }
        //seriesStep.Points.AddXY(x, y);
        seriesStep.Points.AddXY(x, y);
        seriesStep.BorderWidth = 3;
    }
    graph.Series.Add(seriesStep);
}
```

ロジック (C#)

活性化関数 #2

- ニューラルネットワークの活性化関数の1つにシグモイド関数がある
- シグモイド関数の出力は基本的に小数になる
- ステップ関数よりも滑らかな曲線 → 連続的で細かい値 （主に 0 ~ 1 の区間で）



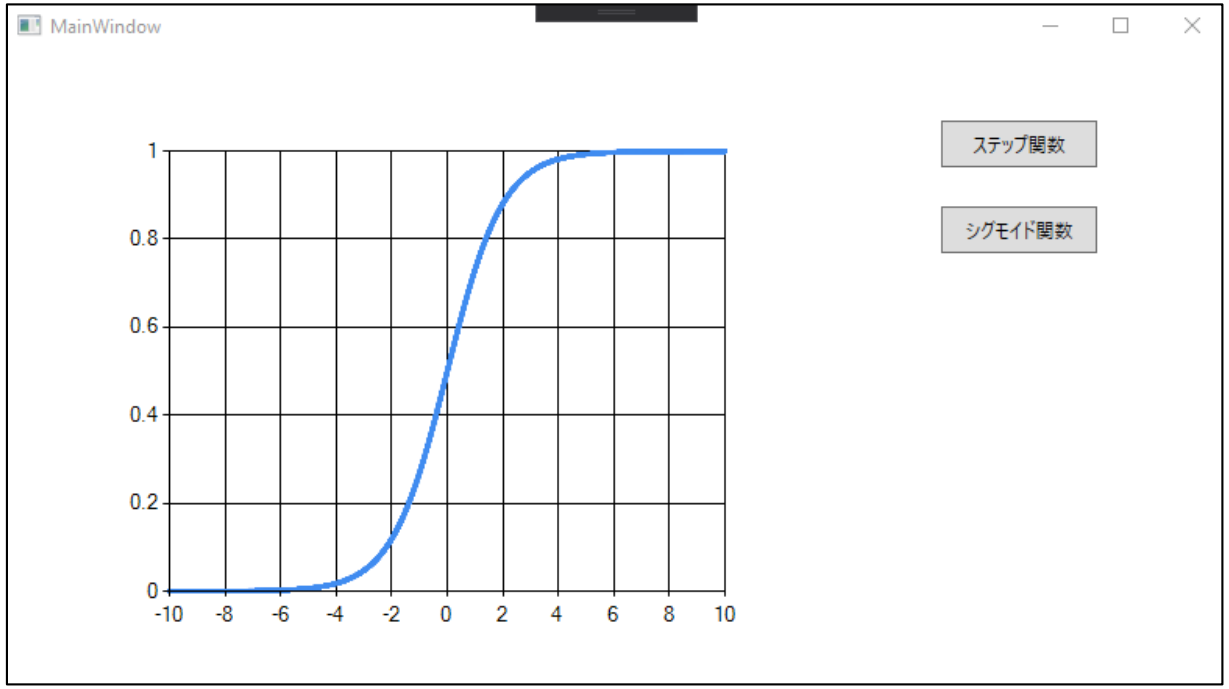
シグモイド関数

$$h_{(x)} = \frac{1}{1 + \exp(-x)}$$

例）
入力が 1：0.7310585... を出力
入力が 0.5：0.6224593... を出力
入力が 0：0.5 を出力

ハンズオン ＃2

- C# (WPFアプリケーション) で シグモイド関数のグラフを描画する
- XAMLは #1 をそのまま使用（ボタンは追加）



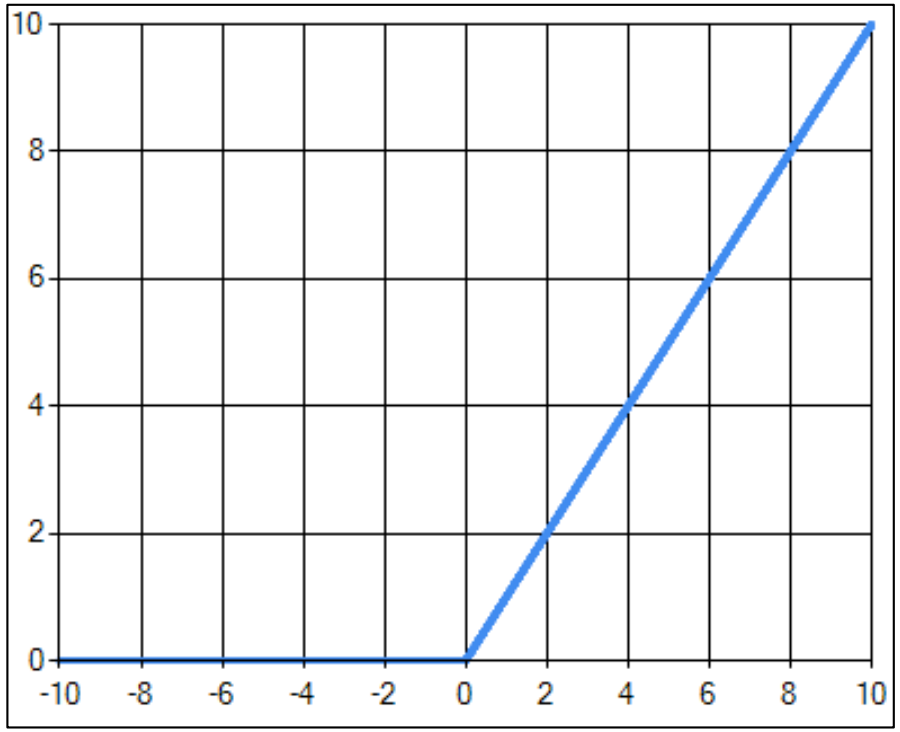
実行結果

```
private void Button_Sigmoid(object sender, RoutedEventArgs e)
{
    var windowsFormsHost = (WindowsFormsHost)GraphArea.Children[0];
    var graph = (Chart)windowsFormsHost.Child;
    graph.ChartAreas.Clear();

    // ChartArea追加
    graph.ChartAreas.Add("Graph1");
    // Seriesの作成と値の追加
    Series seriesStep = new Series();
    seriesStep.ChartType = SeriesChartType.Line;
    graph.ChartAreas[0].AxisX.Maximum = max; //そのグラフの最小値
    graph.ChartAreas[0].AxisX.Minimum = min; //そのグラフの最大値
    graph.ChartAreas[0].AxisX.Interval = interval; //目盛りの間隔 (

    double y = 0; //ステップ関数の初期値
    for (double x = min; x <= max; x = x + 0.001)
    {
        y = 1 / (1 + Math.Exp(-x)); //シグモイド関数
        seriesStep.Points.AddXY(x, y);
        seriesStep.BorderWidth = 3;
    }
    graph.Series.Add(seriesStep);
}
```

ロジック (C#)

活性化関数 #3

- 最近のニューラルネットワークはReLU関数が使用されている (ReLU は様々な種類がある)
- ReLU：Rectified Linear Unit の略
- 入力された値はそのまま出力される関数



ReLU関数

$$h_{(x)} = \max(0, x)$$

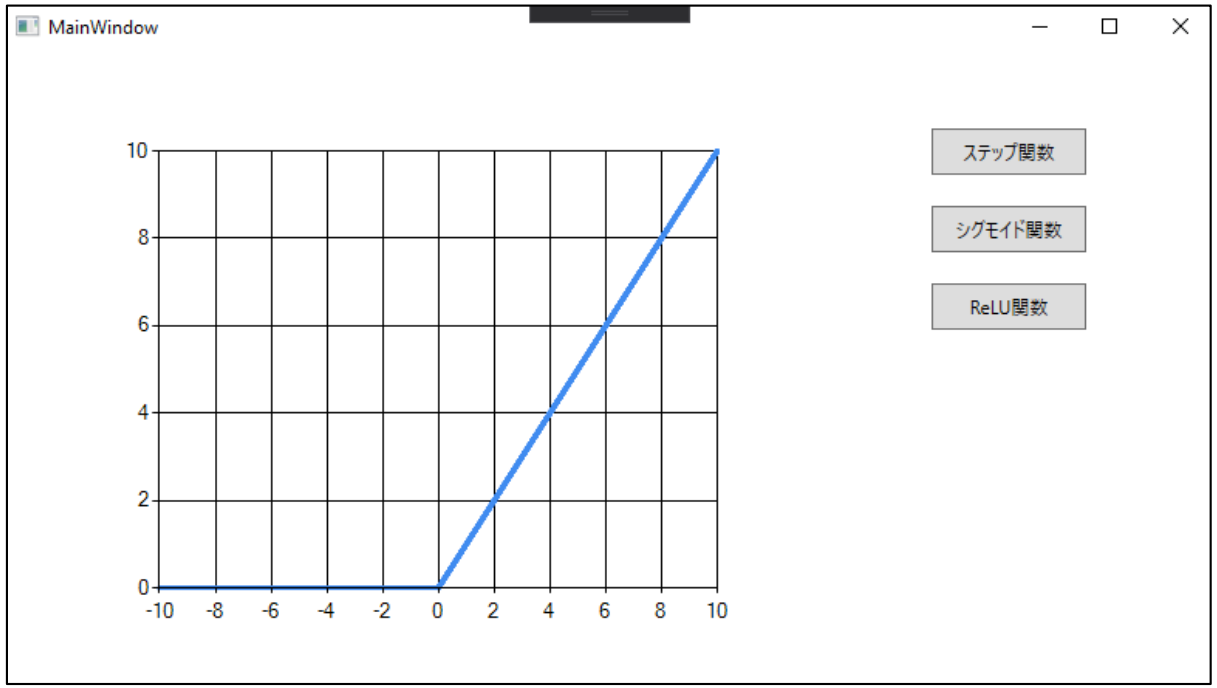
または

$$h_{(x)} = \begin{cases} 0\ (x \leq 0) \\ x\ (x > 0) \end{cases}$$

入力が 0 を超える：そのまま出力
入力が 0 以下：出力はすべて 0

ハンズオン ＃3

- C# (WPFアプリケーション) で ReLU関数のグラフを描画する
- XAMLは #2 をそのまま使用（ボタンは追加）



実行結果

```
private void Button_Relu(object sender, RoutedEventArgs e)
{
    var windowsFormsHost = (WindowsFormsHost)GraphArea.Children[0];
    var graph = (Chart)windowsFormsHost.Child;
    graph.ChartAreas.Clear();

    // ChartArea追加
    graph.ChartAreas.Add("Graph1");
    // Seriesの作成と値の追加
    Series seriesStep = new Series();
    seriesStep.ChartType = SeriesChartType.Line;
    graph.ChartAreas[0].AxisX.Maximum = max; //そのグラフの最小値
    graph.ChartAreas[0].AxisX.Minimum = min; //そのグラフの最大値
    graph.ChartAreas[0].AxisX.Interval = interval; //目盛りの間隔 (

    double y = 0; //ステップ関数の初期値
    for (double x = min; x <= max; x = x + 0.001)
    {
        if (x > 0)
        {
            y = x;  //0を超えたらxを出力
        }
        else
        {
            y = 0; //0未満は0を出力
        }
        seriesStep.Points.AddXY(x, y);
        seriesStep.BorderWidth = 3;
    }
    graph.Series.Add(seriesStep);
}
```

ロジック (C#)

ニューラルネットワークの例

- 各層にReLU活性化関数を適用した場合
- 入力値が 0（活性化関数）を超えたとき、そのまま出力
- 入力が (1, 1) のとき、出力は (0.56, 0) になる
- これは2層ニューラルネットワークの例（パラメータは適当）