



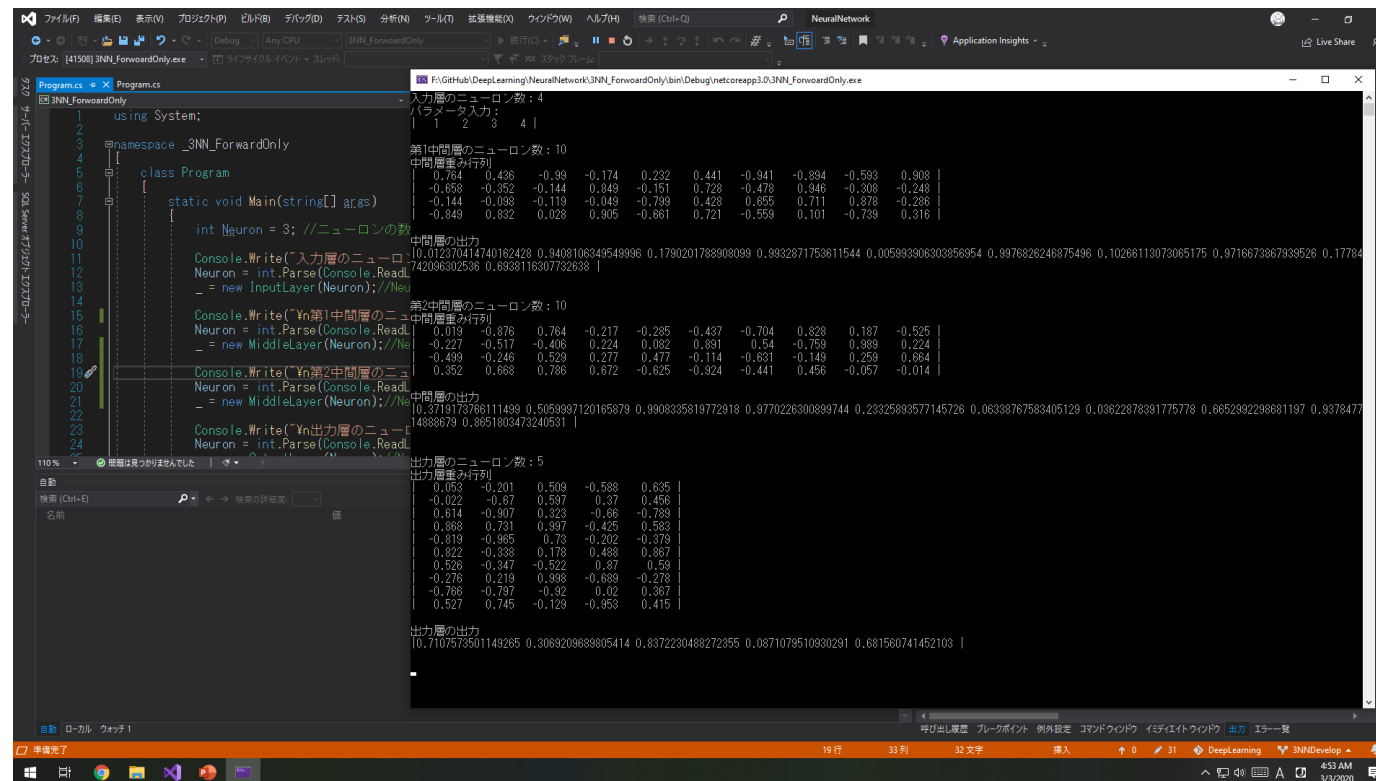
深層学習入門

#6 出力層の設計

たくのろじい / takunology

リポジトリ変更のお知らせ

- ニューラルネットワークのソリューションを1つにしました
- 中間層2つ目以降で1つ目より小さい値を入れると発生する問題を修正しました
- NNをプロジェクトごとにならべて管理するようにしました



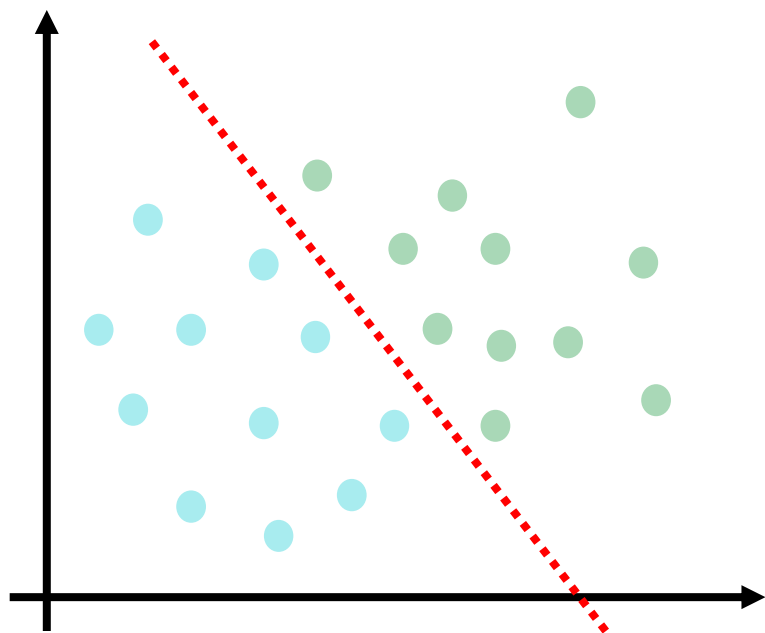
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace _3NN_ForwardOnly
{
    class Program
    {
        static void Main(string[] args)
        {
            int Neuron = 3; //ニューロンの数
            Console.WriteLine("入力層のニューロン数: 4");
            Console.WriteLine("パラメータ入力:");
            Console.WriteLine("1 2 3 4");
            Console.WriteLine("第1中間層のニューロン数: 10");
            Console.WriteLine("中間層重み行列");
            Console.WriteLine("0.764 0.436 -0.99 -0.174 0.232 0.441 -0.941 -0.894 -0.593 0.908");
            Console.WriteLine("-0.658 -0.352 -0.144 0.849 -0.151 0.728 -0.478 0.946 -0.308 -0.248");
            Console.WriteLine("-0.144 -0.098 -0.119 -0.049 -0.799 0.428 0.655 0.711 0.878 -0.286");
            Console.WriteLine("-0.849 0.832 0.028 0.905 -0.661 0.721 -0.559 0.101 -0.739 0.316");
            Console.WriteLine("中間層の出力");
            Console.WriteLine("10.012370414740162428 0.9408106349549996 0.1790201788908099 0.9932871753611544 0.005993906303856954 0.9976826246875496 0.10266113073065175 0.9716673867939526 0.17784742096302536 0.6938116307732638");
            Console.WriteLine("第2中間層のニューロン数: 10");
            Console.WriteLine("中間層重み行列");
            Console.WriteLine("0.019 -0.876 0.764 -0.217 -0.285 -0.437 -0.704 0.828 0.187 -0.525");
            Console.WriteLine("-0.227 -0.517 -0.406 0.224 0.082 0.891 0.54 -0.759 0.989 0.224");
            Console.WriteLine("-0.493 -0.246 0.539 0.277 0.477 -0.114 -0.631 0.149 0.259 0.684");
            Console.WriteLine("0.352 0.668 0.736 0.672 -0.625 -0.924 -0.441 0.456 -0.057 -0.014");
            Console.WriteLine("中間層の出力");
            Console.WriteLine("10.3719173768111499 0.5059997120165879 0.9908335819772918 0.9770226300899744 0.23325893577145726 0.06338767583405129 0.03622878391775778 0.6652992298681197 0.937847714888679 0.8651803473240531");
            Console.WriteLine("出力層のニューロン数: 5");
            Console.WriteLine("出力層重み行列");
            Console.WriteLine("0.053 -0.201 0.509 -0.588 0.635");
            Console.WriteLine("-0.022 -0.67 0.597 0.37 0.456");
            Console.WriteLine("0.614 -0.907 0.323 -0.66 -0.789");
            Console.WriteLine("0.868 0.731 0.997 -0.425 0.583");
            Console.WriteLine("-0.819 -0.965 0.73 -0.202 -0.379");
            Console.WriteLine("0.822 -0.338 0.178 0.488 0.867");
            Console.WriteLine("0.526 -0.347 -0.522 0.97 0.591");
            Console.WriteLine("-0.276 0.219 0.988 -0.689 -0.278");
            Console.WriteLine("-0.766 -0.797 -0.92 0.02 0.367");
            Console.WriteLine("0.527 0.745 -0.129 -0.953 0.415");
            Console.WriteLine("出力層の出力");
            Console.WriteLine("10.7107573501149265 0.3069209689805414 0.8372230468272355 0.0871079510930291 0.681560741452103");
        }
    }
}
```

<https://github.com/takunology/DeepLearning/tree/master/NeuralNetwork>

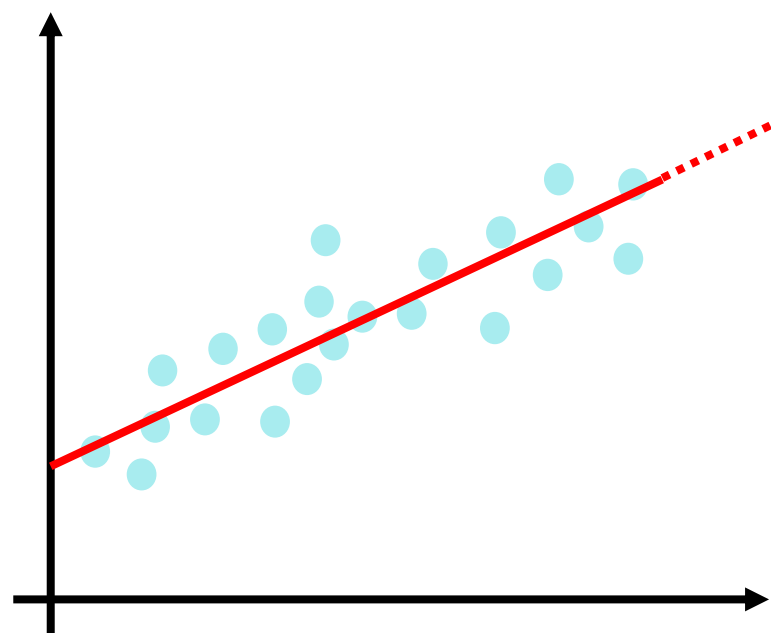
出力層の活性化関数

- 出力層の活性化関数の種類で **分類問題** or **回帰問題** に分かれる
- 回帰問題： **恒等関数** を用いる
- 分類問題： **ソフトマックス関数** を用いる



分類問題

データの散らばりを**分類**

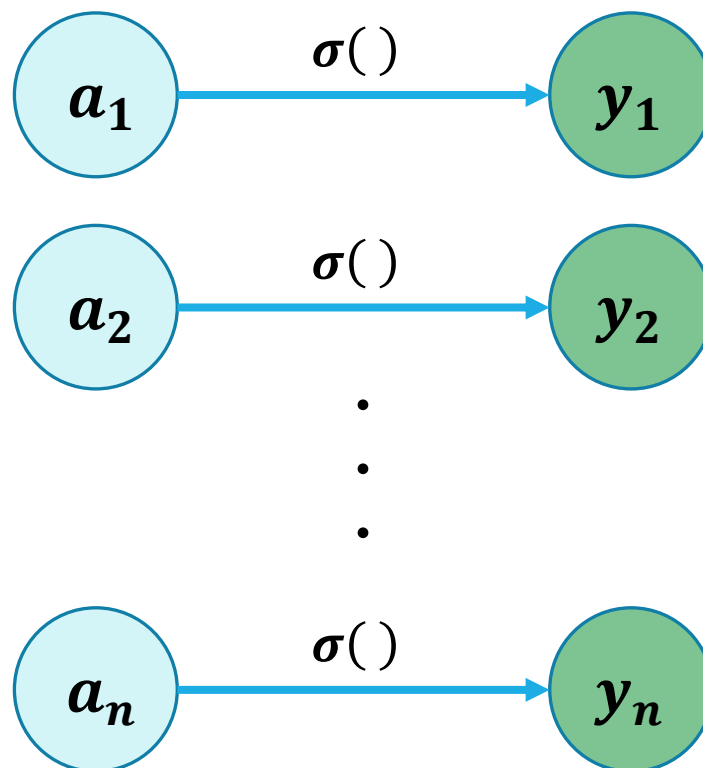


回帰問題

様々なデータをもとに**推測**

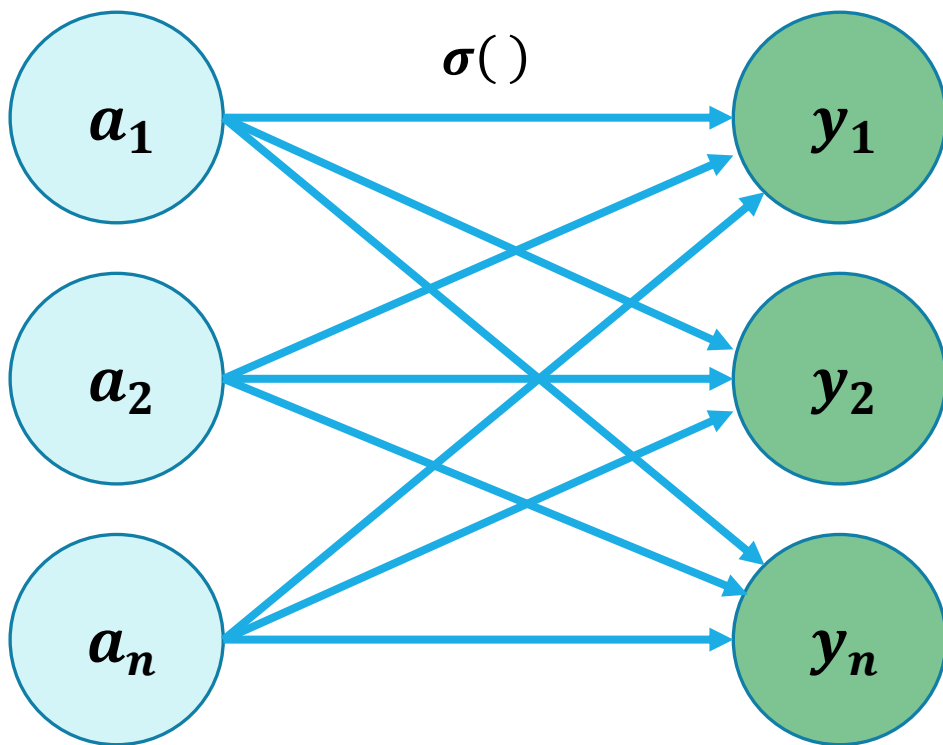
恒等関数

- 入力値をそのまま出力する関数
- 入ってきた値の総和を a , 出力層の活性化関数を $\sigma()$ とする



ソフトマックス関数

- 活性化関数の1つで主に出力層に使用する
- 出力層 n 個の中から1つ y_k を選ぶ
- 入力値 / 入力値全体の関係 \rightarrow 入力値によってそれが解である可能性が高くなる



$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^n \exp(a_i)}$$

a_k : k 番目の入力値
 y_k : k 番目の出力値
 n : 出力層ニューロンの数
 a_i : すべての入力値

ハンズオン #1

- ・ ソフトマックス関数を作って出力層に適用する
- ・ プラットフォームは C# (.NET Core 3.0 コンソールアプリケーション)
- ・ 作成した3層ニューラルネットワークを再利用（新しく作り直しました）

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^n \exp(a_i)}$$

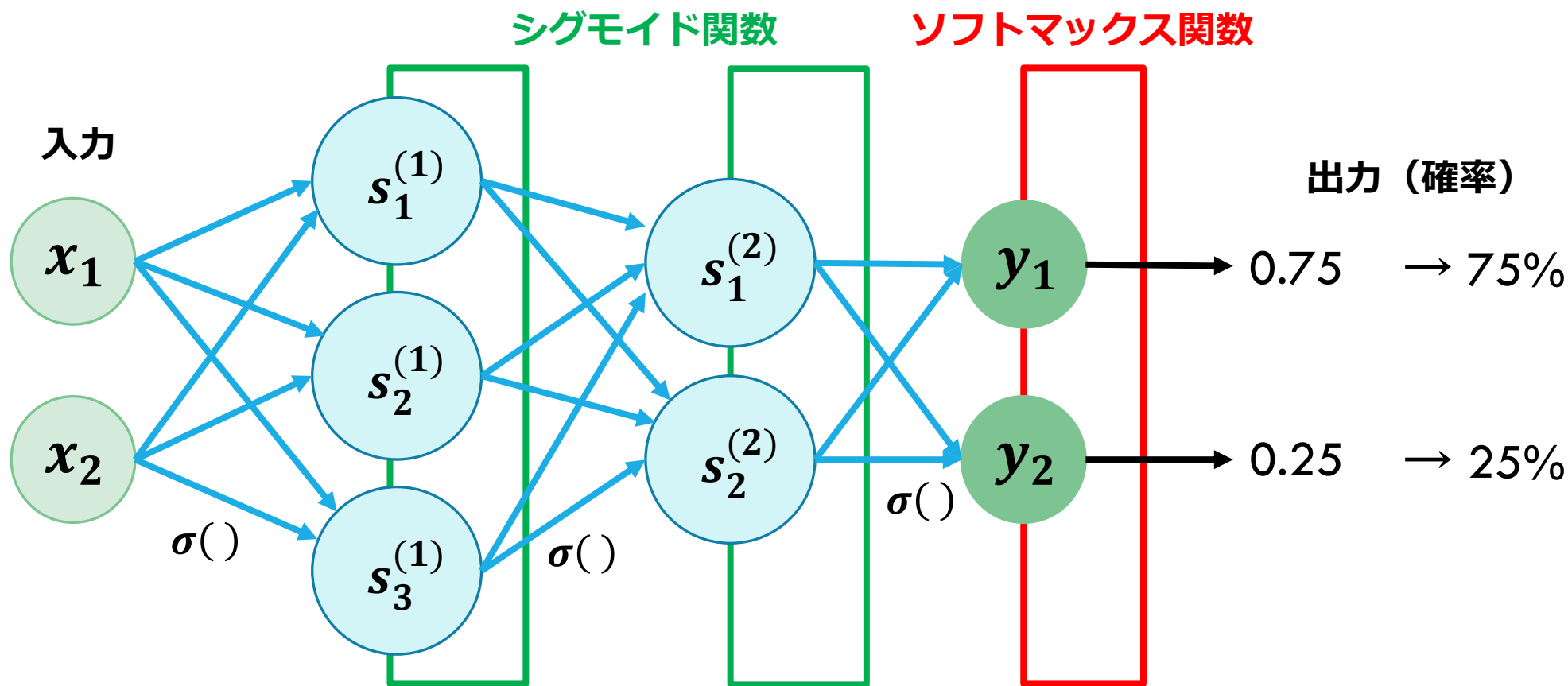
指数関数で計算してから
総和をとる

```
パラメータ入力:
| 1 2 3 |
----- 中間層(第1層)重み行列 -----
| -0.576 0.393 0.23 0.284 0.25 0.214 0.853 -0.891 0.895 0.406 -0.926 -0.985 0.288
| -0.702 0.796 |
| 0.758 -0.198 0.13 -0.735 -0.551 -0.274 0.697 0.084 0.901 -0.955 -0.629 -0.671 -0.173
| -0.629 -0.875 |
| 0.07 0.32 -0.196 0.715 0.025 -0.364 0.713 0.873 0.467 0.653 0.977 0.525 -0.854
| 0.695 -0.429 |
----- 中間層(第1層)の出力 -----
| 0.759510916949111 0.7225207532716214 0.4755195893532669 0.7229215437983317 0.31496681462772297 0.1937226957705712 0.987
| 7026755202027 0.8694381343821441 0.9836653961257537 0.6118273690017372 0.6785246631166437 0.3203856670948647 0.067862293
| 57699451 0.531209373737563 0.09612861870485127 |
----- 中間層(第2層)重み行列 -----
| -0.025 -0.326 0.454 -0.912 -0.492 0.784 0.979 0.988 0.256 -0.458 |
| 0.181 -0.443 -0.118 0.511 0.799 -0.207 -0.336 -0.928 0.151 -0.69 |
| -0.789 -0.636 0.848 -0.219 0.19 0.617 0.316 -0.88 -0.628 -0.292 |
----- 中間層(第2層)の出力 -----
| 0.11204703855699032 0.042289771842033794 0.9405874978712752 0.36656071159352904 0.8423741383953062 0.9021195310991816 0
| .7781641849247005 0.029085461669219492 0.20982178186118203 0.062152285111320926 |
----- 出力層重み行列 -----
| -0.458 -0.989 -0.795 |
| 0.589 0.633 0.044 |
| 0.269 0.66 0.361 |
| -0.313 0.336 0.155 |
| -0.024 0.795 0.005 |
| 0.561 0.5 0.205 |
| 0.938 0.297 0.71 |
| 0.737 0.198 -0.854 |
| -0.671 0.125 0.037 |
| 0.389 -0.18 -0.482 |
----- 出力層の出力 -----
| 0.248677990886879 0.5534871555141275 0.1978348535989934 |
出力層の確率合計 (DEBUG): 1
```

ソフトマックス関数を適用したプログラムはGithubを参考に
<https://github.com/takunology/DeepLearning/tree/master/NeuralNetwork>

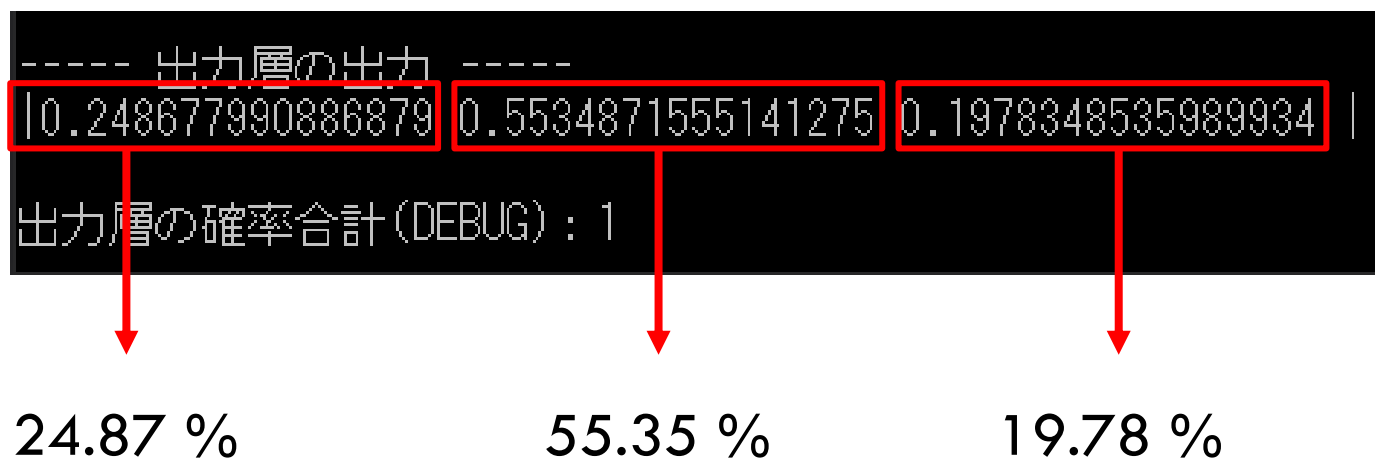
ソフトマックス関数 - 続き

- 作用させると必ず0 ~ 1の範囲内で結果が出力される
- 得られた結果を全て足すと 1 になる → 確率として解を得ることができる



出力結果を見てみる

- ソフトマックス関数による出力は確率としてみなせる



確率とすれば全て足すと 1 (100%) になるはず

計算で切り捨てる可能性もあるので 0.99999999... となることもある