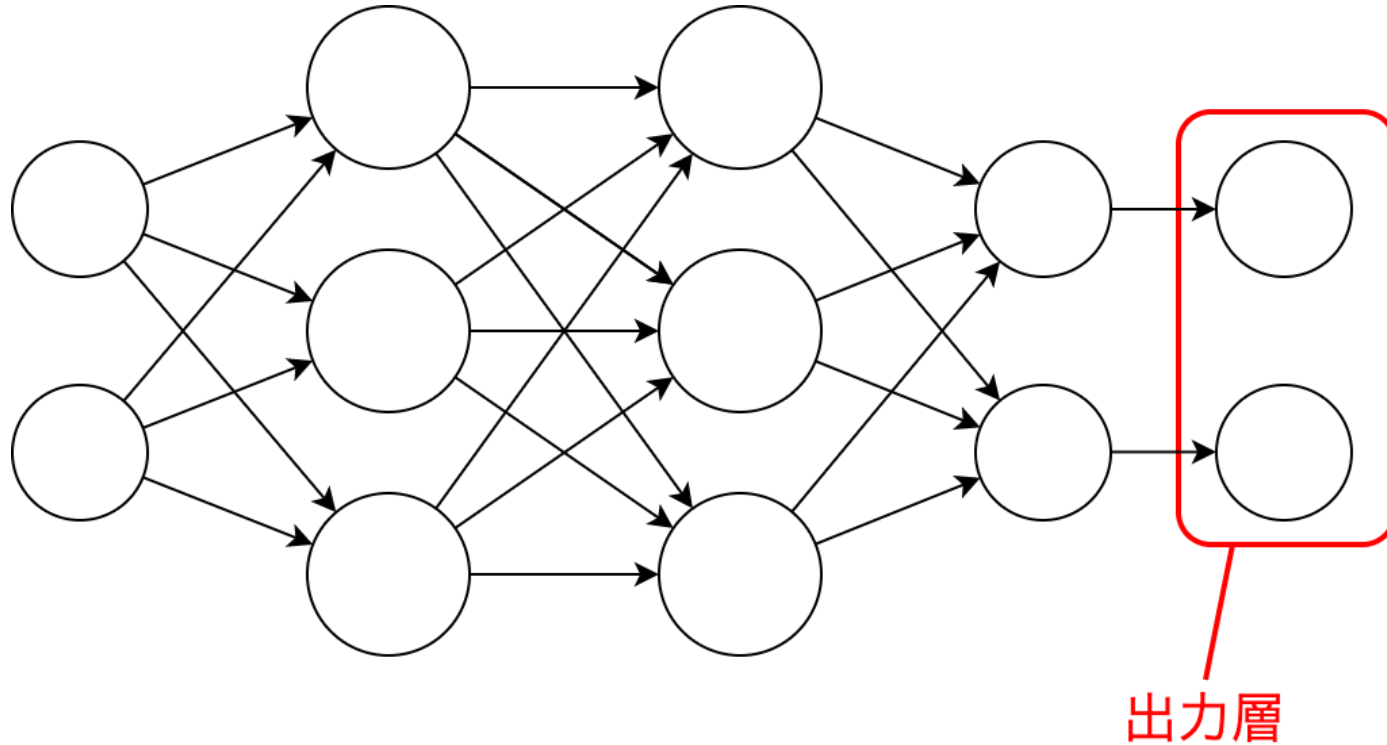


2. 出力層の設計

甲本健太

出力層とは

出力層（ニューラルネットワークの最後の部分）



出力関数の種類

恒等関数

- 与えられた入力をそのまま返す関数
- 主に**回帰問題**で利用

ソフトマックス関数

- 総和が1になるように分配する関数
- 主に**分類問題**で使用

回帰問題・分類問題って？

分類問題

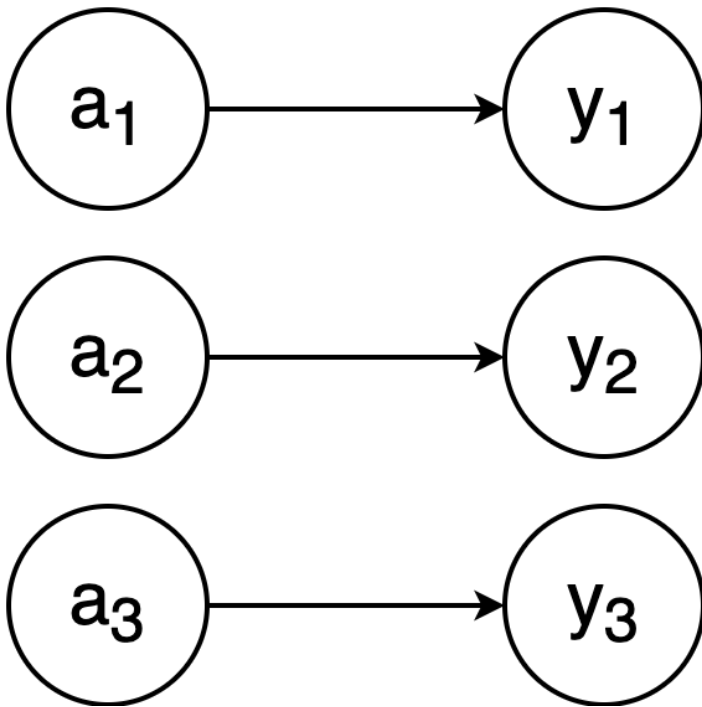
- 入力データがどのクラスに属するか分類する
- 「手書き数字の認識」など

回帰問題

- 入力データ数値の予測を行う
- 「株価の予想」など

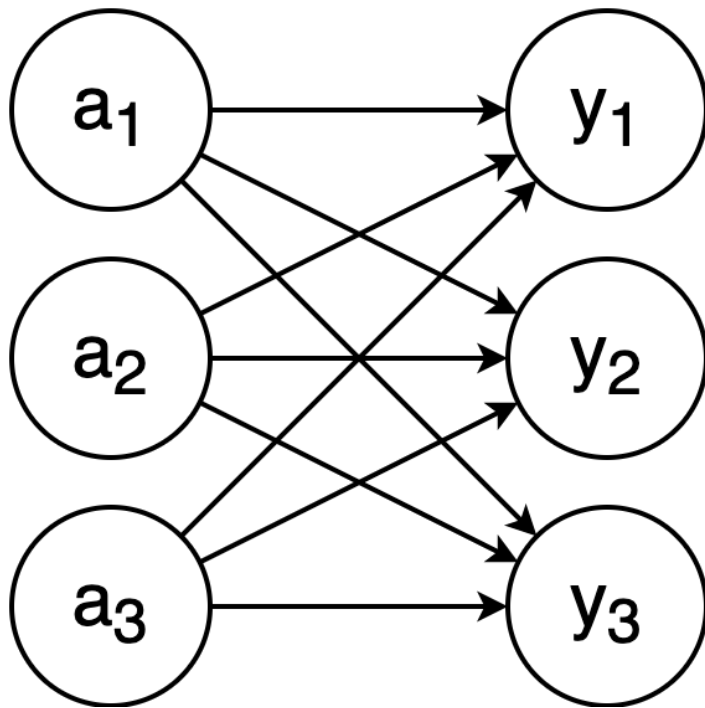
恒等関数

- 入力をそのまま返す



ソフトマックス関数 (1/2)

- 各入力を全ての入力の相対値にして返す



ソフトマックス関数 (2/2)

計算方法

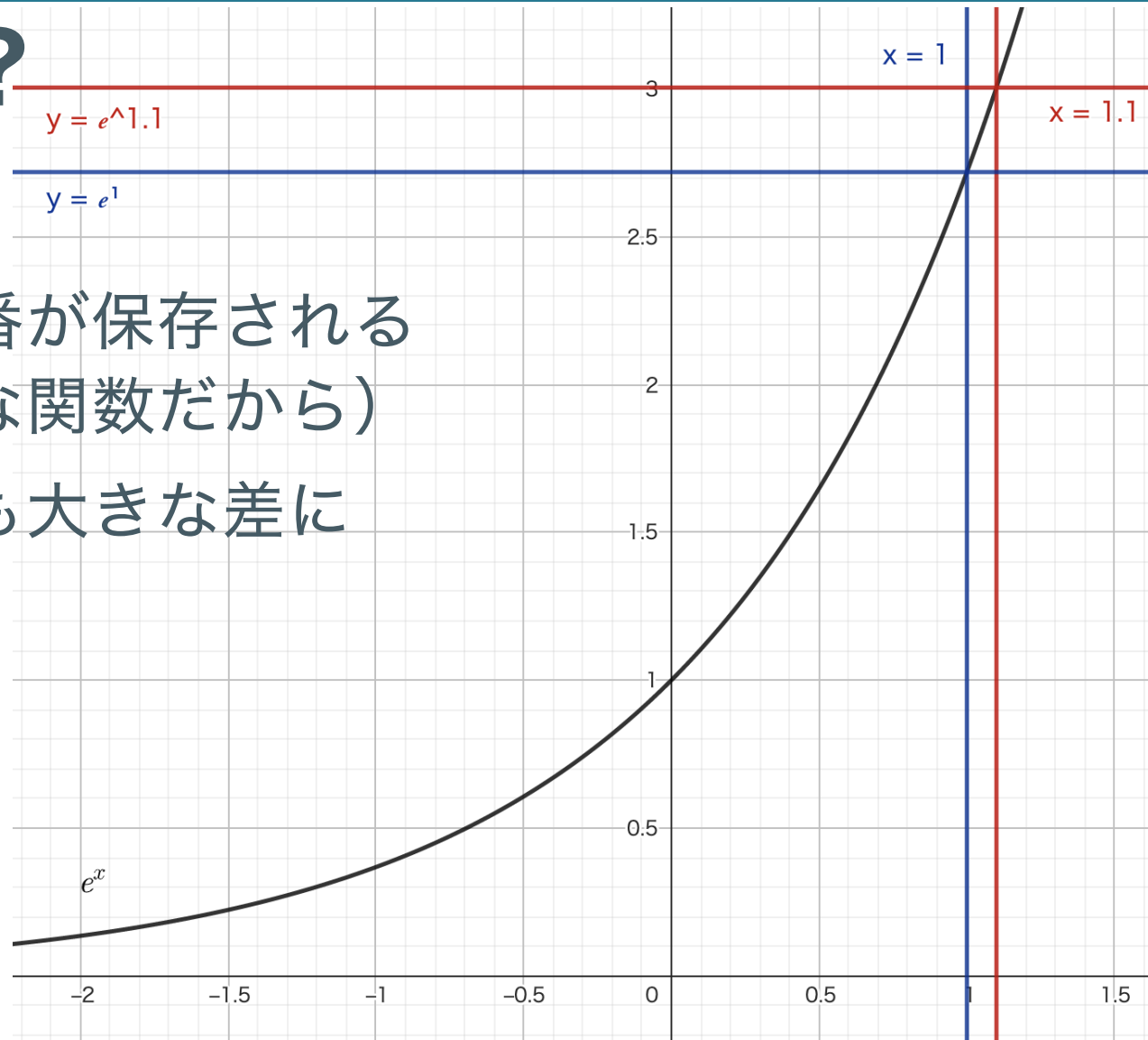
1. 全ての入力の指数を計算 : $a'_k = e^{a_k}$
2. a'_k の合計を計算 : $S = \sum_{i=1}^n a'_k$
3. 入力を合計でわる : $y_k = a'_k / S$

式

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

なぜ指数？

- 大きさの順番が保存される
(単調増加な関数だから)
- 少しの差でも大きな差に変換される



いざ実装！

```
>>> a = np.array([0.3, 2.9, 4.0])
>>>
>>> exp_a = np.exp(a)  # 指数関数
>>> exp_a
array([ 1.34985881, 18.17414537, 54.59815003])
>>>
>>> sum_exp_a = np.sum(exp_a)  # exp_a の総和
74.1221542101633
>>>
>>> y = exp_a / exp_a_sum  # 正規化
>>> y
array([0.01821127, 0.24519181, 0.73659691])  # ← 出力
```

関数にまとめよう (問題)

```
def softmax(a):  
    # ↓ここにコードを書いてね  
  
    return y
```

↓こうなればOK

```
>>> a = np.array([1.2, 6.0, 4.3, 2.1])  
>>> softmax(a)  
array([0.09003057, 0.24472847, 0.66524096])
```

関数にまとめよう (解答)

```
def softmax(a):  
    exp_a = np.exp(a)           # 指数を求める  
    sum_exp_a = np.sum(exp_a)   # 合計を求める  
    y = exp_a / sum_exp_a       # 合計で割る  
  
    return y
```

オーバーフローに注意！（1/3）

“ オーバーフローとは

数字がコンピュータに表現できる限界を超えてしまうこと

”

指数関数は簡単に桁が大きくなるため、オーバーフローが起きやすい

（Pythonでは小数型 `float` に無限大を示す値 `inf` が存在するのでエラーは起きません。代わりにWarningが出るようになっています。）

オーバーフローに注意！ (2/3)

- あらかじめ最大値を引いておきます
- 先に引いてからlogをとっても結果は変わらない！

$$\begin{aligned} y_k &= \frac{\exp(a_k)}{\sum_{i=1}^n \exp(a_i)} = \frac{\exp(a_k) / C}{\sum_{i=1}^n \exp(a_i) / C} \\ &= \frac{\exp(a_k) / \exp(a_{max})}{\sum_{i=1}^n \exp(a_i) / \exp(a_{max})} \\ &= \frac{\exp(a_k - a_{max})}{\sum_{i=1}^n \exp(a_i - a_{max})} \end{aligned}$$

オーバーフローに注意！ (3/3)

```
>>> a = np.array([1010, 1000, 900])
>>> np.exp(a) / np.sum(np.exp(a))
Warning...
array([nan, nan, nan])
>>>
>>> a_max = np.max(a) # a_max = 1010
>>> a - a_max
array([ 0, -10, -110])
>>>
>>> np.exp(a - a_max) / np.sum(np.exp(a - a_max))
array([9.99954602e-01, 4.53978687e-05, 1.68883521e-48])
```

↑ちゃんと計算できた！

関数にまとめよう！（問題）

```
def softmax(a):  
    # ↓ここにコードを書いてね  
  
    return y
```

↓こうなればOK

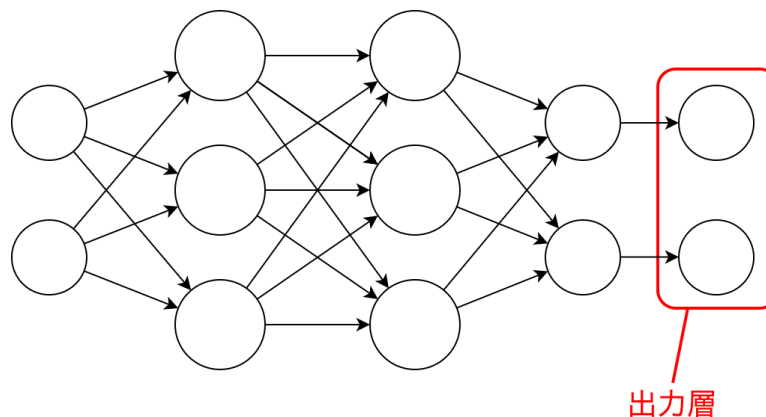
```
>>> a = np.array([1000, 1001, 999])  
>>> softmax(a)  
array([0.24472847, 0.66524096, 0.09003057])
```

関数にまとめよう！（解答）

```
def softmax(a):  
    a_max = np.max(a)  
    exp_a = np.exp(a - a_max)  
    sum_exp_a = np.sum(exp_a)  
    y = exp_a / sum_exp_a  
  
    return y
```


出力層のニューロン数は？

n クラス分類（入力を n 種類に分類する問題）では n 個に設定する



例

- 手書き数字の認識（0～9）では10個のニューロンを設定
- アルファベット（ $a \sim z$ ）の認識では26個に設定

まとめ

- 今回までで、パーセプトロンを勉強してきました
 1. 論理回路
 2. 行列、信号伝達の仕組み
- 次回は第3章の最終回**手書き数字の分類**です

今までで不安なところを復習する時間！