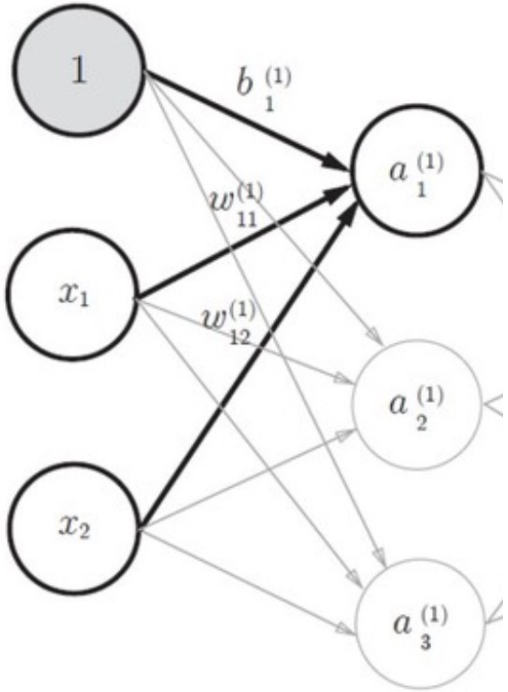
信号伝達の実装

信号伝達を行列で表現しよう



信号伝達を行列で表現しよう

重みの記号の定義

$$W_{12}^{(1)}$$

1層目の1番目のニューロンに向けて
前層の２番目のニューロンへかける重み

$$a_1^{(1)} = w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 + b_1^{(1)}$$

信号伝達を行列で表現しよう

これを行列の内積を用いるとまとめて表すことができる！

$$A^{(1)} = XW^{(1)} + B^{(1)}$$

$$W^{(1)} = \begin{pmatrix} w_{11}^{(1)} & w_{21}^{(1)} & w_{31}^{(1)} \\ w_{12}^{(1)} & w_{22}^{(1)} & w_{32}^{(1)} \end{pmatrix}$$

信号伝達を行列で表現しよう

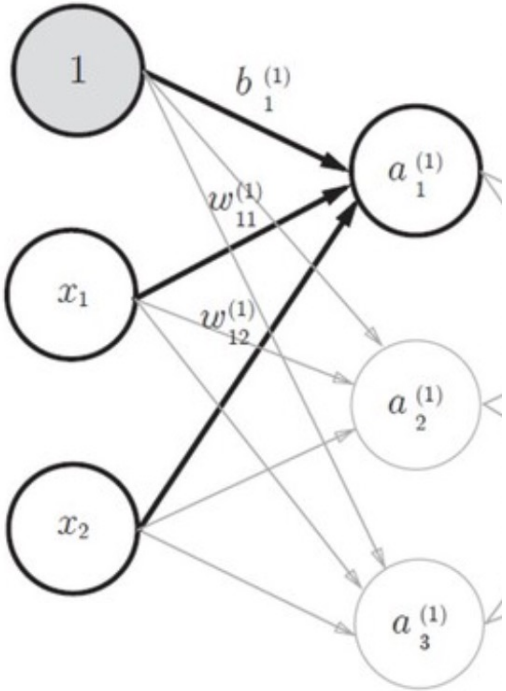
$$X = (1 \quad 0.5)$$

$$W^{(1)} = \begin{pmatrix} 0.1 & 0.3 & 0.5 \\ 0.2 & 0.4 & 0.6 \end{pmatrix}$$

$$B^{(1)} = (0.1 \quad 0.2 \quad 0.3)$$

活性化関数なしの実装

```python
import numpy as np

X = np.array([1.0,0.5])
W1 = np.array([[0.1,0.3,0.5],[0.2,0.4,0.6]])
B1 = np.array([0.1,0.2,0.3])

A1 = np.dot(X, W1) + B1
```



活性化関数の導入

ゼロイチゼミ

シグモイド関数を定義して
A1を代入したZ1を作ってみよう！

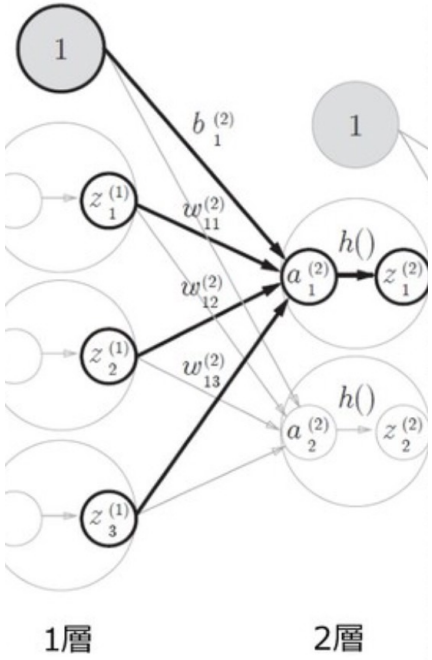
```
def sigmoid(x):
    return 1 / (1+np.exp(-x))

Z1 = sigmoid(A1)
```

１層目から２層目へ

$$W^{(1)}=\begin{pmatrix}0.1 & 0.4\\ 0.2 & 0.5\\ 0.3 & 0.6\end{pmatrix}$$

$$B^{(1)}=(0.1 \quad 0.2)$$

活性化関数：シグモイド関数



１層目から２層目へ

```
W2 = np.array([[0.1,0.4],[0.2,0.5],[0.3,0.6]])
B2 = np.array([0.1,0.2])
A2 = np.dot(Z1, W2) + B2
Z2 = sigmoid(A2)
```

2層目から出力層へ

活性化関数を恒等関数（x = x）に変更する

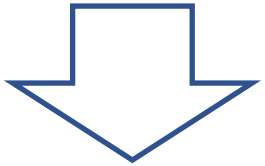
```
def identity_function(x):
    return x
```

※ 出力層で利用する活性化関数は解く問題の性質に応じて決める
回帰　　：恒等関数
２クラス分類：シグモイド関数
他クラス分類：ソフトマックス関数

全てまとめて実装してみよう

ゼロイチゼミ

ディクショナリを用いて
重みやバイアスを一つに収納



関数として定義し実行する

全てまとめて実装してみよう

ゼロイチゼミ

```
def init_network():
    network = {}
    ...

    return network
```

```
def forward(network, x):
    W1, W2, W3 =
    b1, b2, b3 =
    a1 =
    z1 =
    a2 =
    z2 =
    a3 =
    y =

    return y
```