

まるっと一日公益大学生体験

機械学習入門

—文字が読める人工ニューロンを作ってみよう—

カロル ノヴァコフスキ

Karol Nowakowski

東北公益文科大学

karol@koeki-u.ac.jp



東北公益文科大学
Tohoku University of Community Service and Science

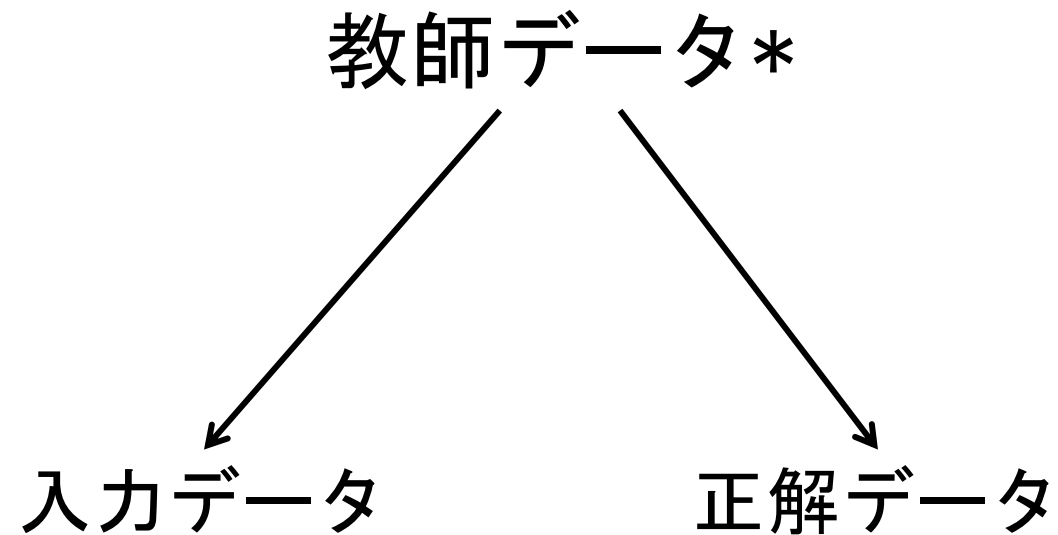
I 機械学習の基礎

機械学習の基礎

機械学習

- データから（パターンを）学習し、それを基に未知のデータに関して予測できるプログラムを開発する技術
- 人工知能の下位分野

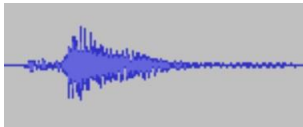
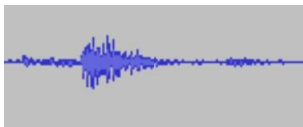
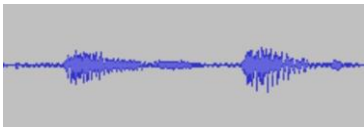
機械学習の基礎



* 「教師あり学習」の場合（「教師なし学習」や「強化学習」は「正解データ」を使わない）。

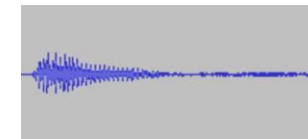
機械学習の基礎

例（音声認識）：

教師データ	
入力データ (音声)	正解データ (テキスト)
	→ dog
	→ cat
	→ hamster



(新しい) 入力:



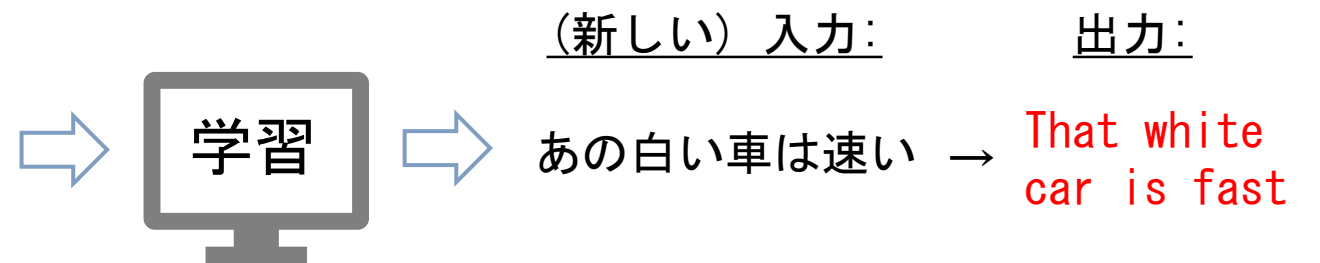
出力:

horse

機械学習の基礎

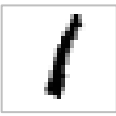
例（機械翻訳）：

教師データ	
入力データ (原言語の文)	正解データ (目的言語の文)
あれが私の車です →	That' s my car
あの白いシャツが好き →	I like that white shirt
この馬は速い →	This horse is fast



機械学習の基礎

例（文字認識）：

教師データ	
入力データ (画像)	正解データ (文字)
	→ 0
	→ 1
	→ 2



機械学習の基礎

MNISTデータベース：

7万枚の手書き数字の画像
(+正解ラベル) を含む
データベース。機械学習
による画像処理システム
の学習に広く使われている。



[“A few samples from the MNIST test dataset”](#) by Josef Steppan (CC BY-SA 4.0)



機械学習の基礎

X - 入力データ

Y - 正解データ

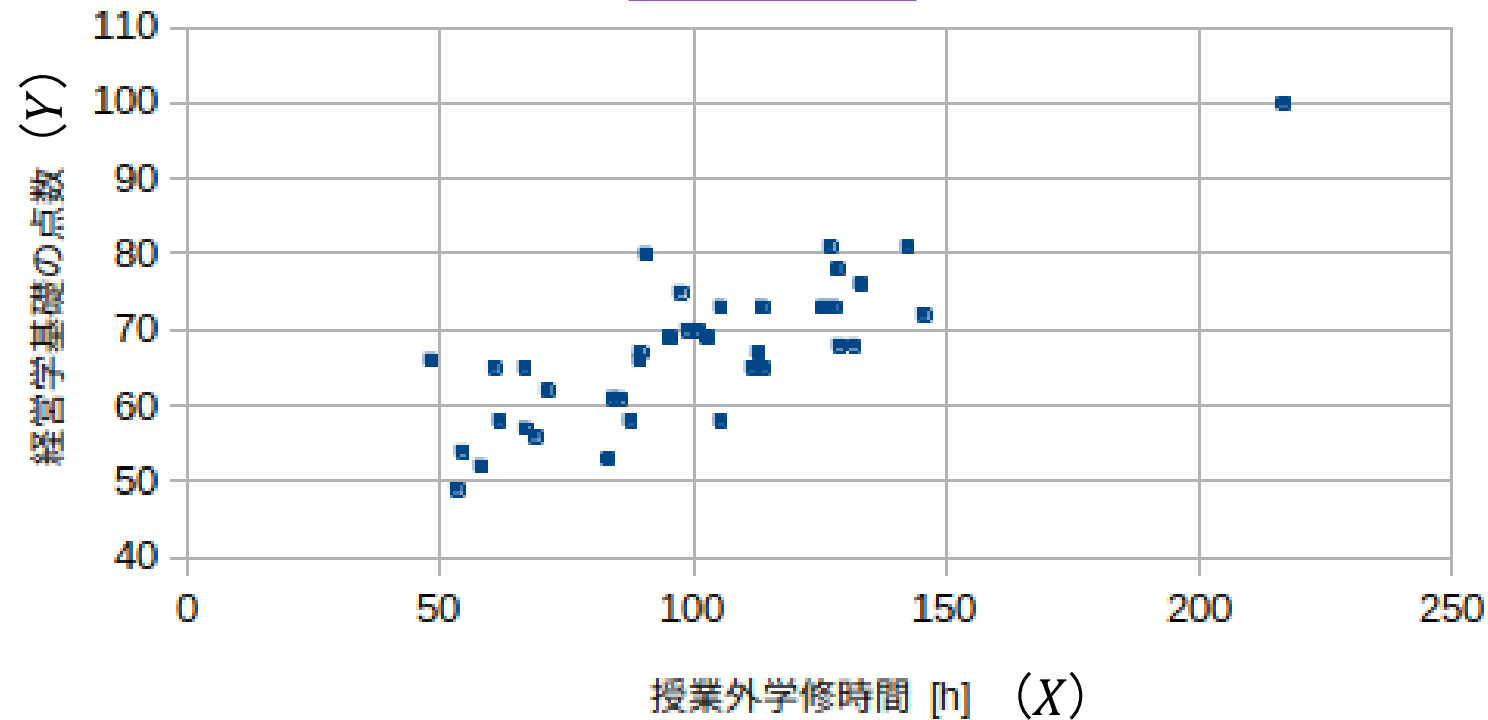
- $f: X \rightarrow Y$ という関数が存在すると仮定する（非常に複雑な関数になる場合もある）。
- **機械学習の目的**： X と Y に基づいて f 関数を（近似的に）求める。結果は h (hypothesis) 関数である（「モデル」とも呼ばれる）。

機械学習の基礎

例：

授業外学習時間と「経営学基礎」の試験点数の関係（経営コース2年）

(Image by 山本先生)

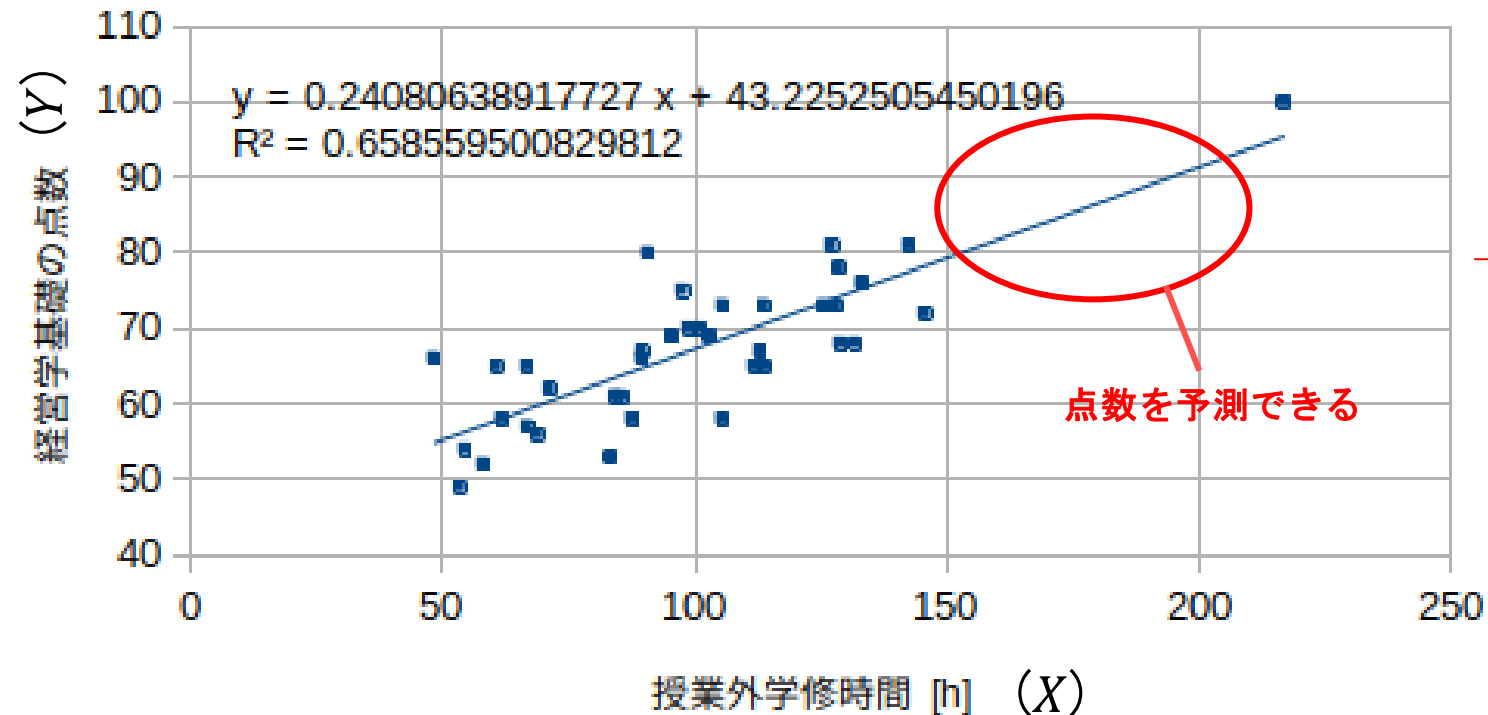


機械学習の基礎

例：

授業外学習時間と「経営学基礎」の試験点数の関係（経営コース2年）

(Image by 山本先生)



(線形回帰による) モデル：

$$h(x) = 0.24x + 43.23$$

モデルの「パラメーター」

機械学習の基礎

- h 関数の妥当性を評価するためには学習に使用されない「評価データ」が必要
- X と Y の一部（1~2割程度）を学習に使わず評価のために置いておくのが一般的である

教師データ												評価データ			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9

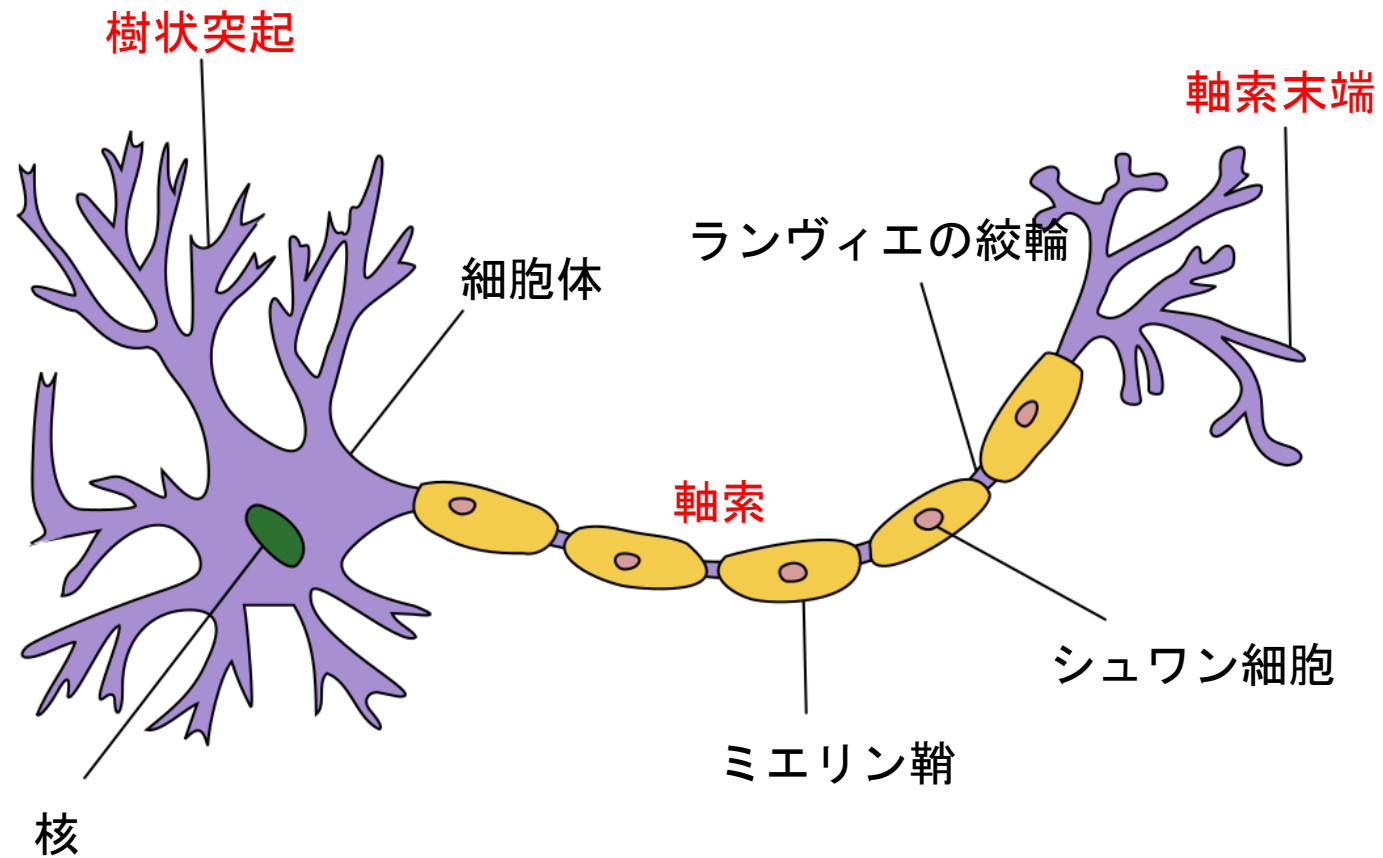
“A few samples from the MNIST test dataset” by Josef Steppan (CC BY-SA 4.0)



東北公益文科大学
Tohoku University of Community Service and Science

II 人工ニューロン (パーセプトロン)

ニューロン（神経細胞）



[Image by Quasar Jarosz, CC-BY-SA-3.0](#)

ニューロン（神経細胞）

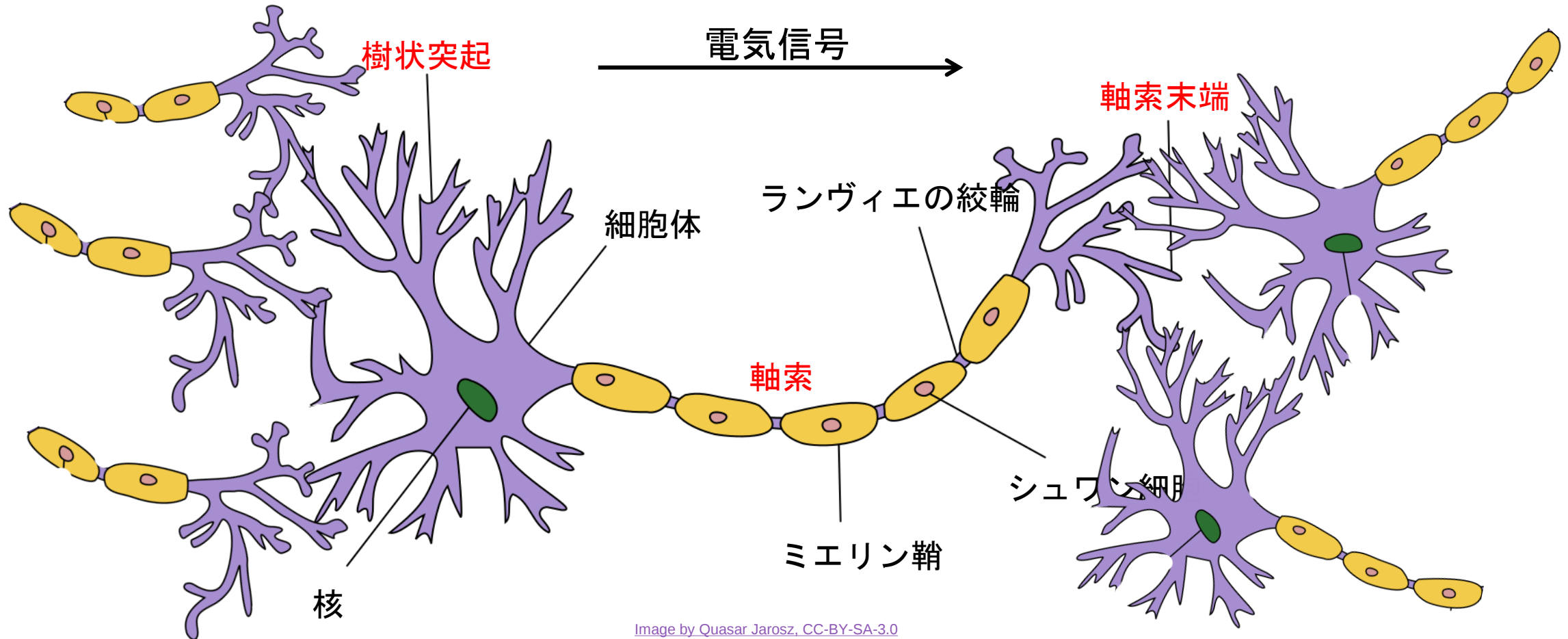
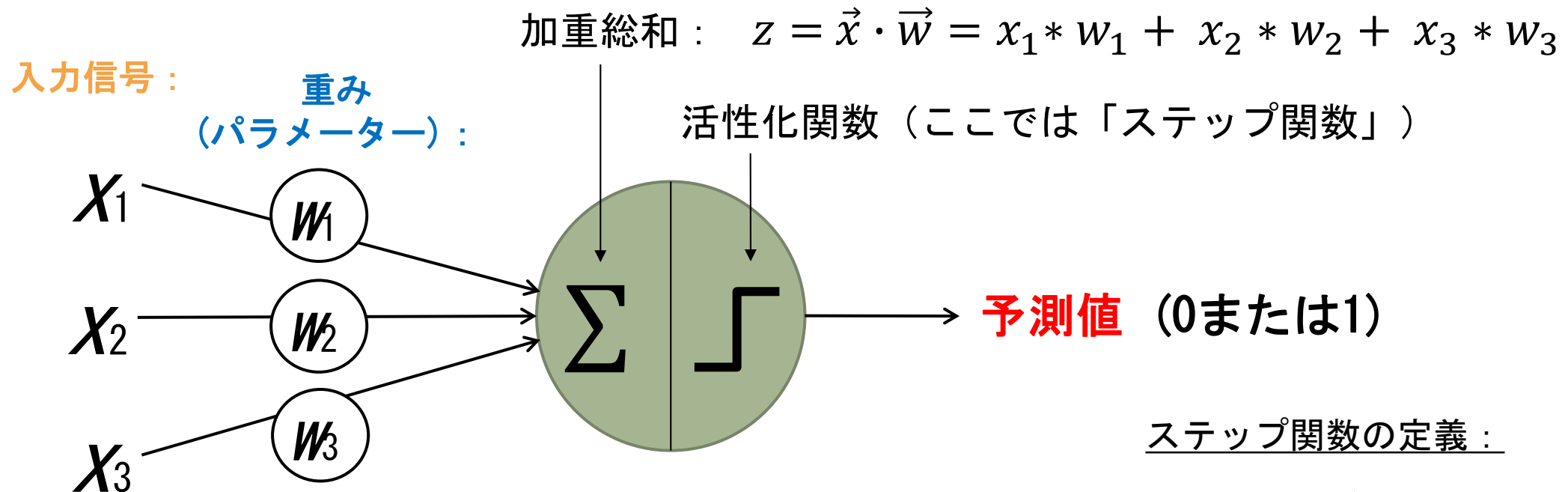


Image by Quasar Jarosz, CC-BY-SA-3.0

人工ニューロン（パーセプトロン）



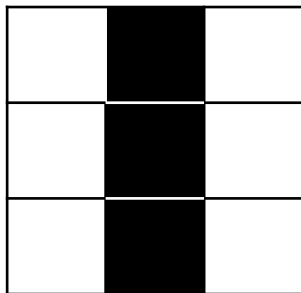
ステップ関数の定義：

$$step(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

Frank Rosenblatt (1958). "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain". *Psychological Review*.

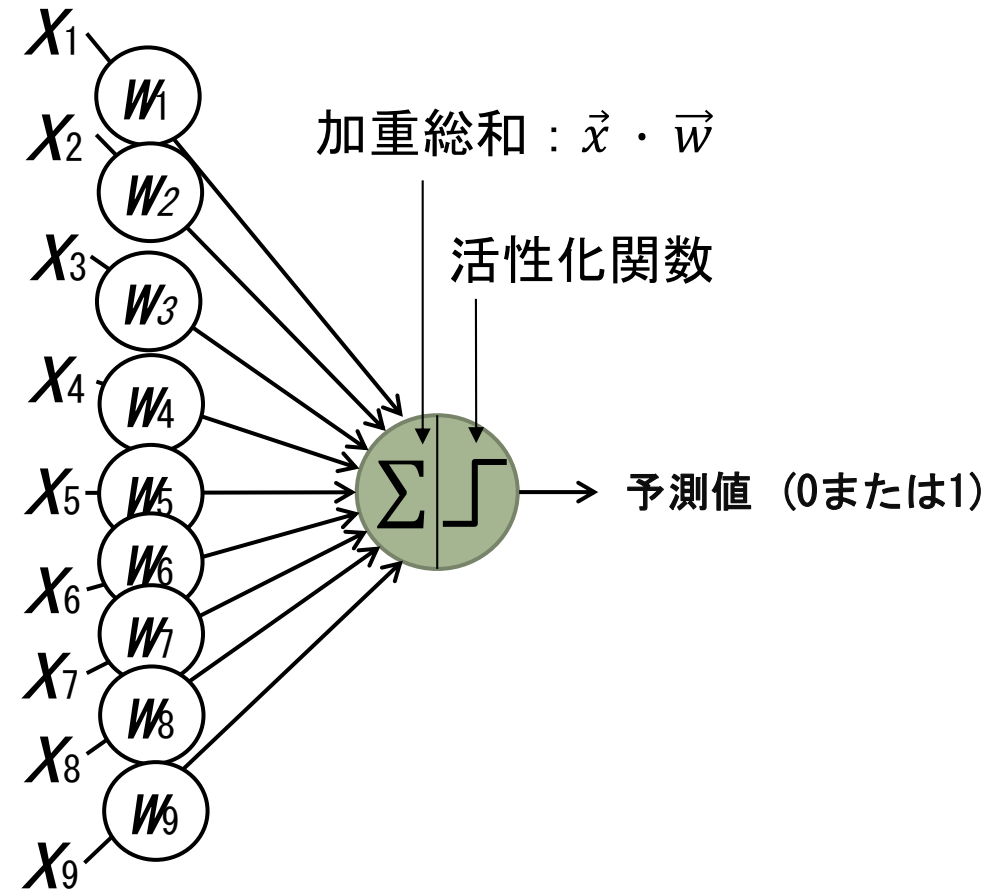
人工ニューロン（パーセプトロン）

画像（3 × 3 画素）：

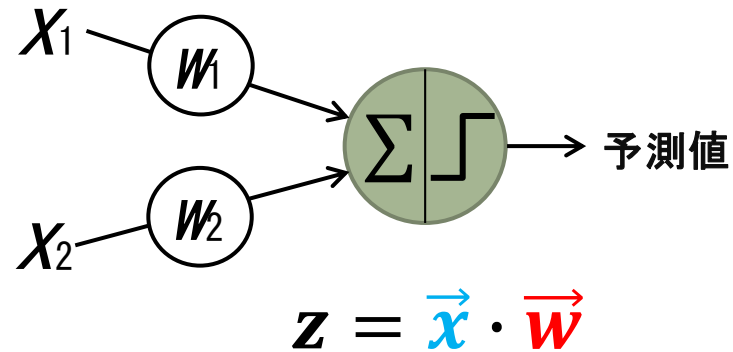


パーセプトロンの
入力値（画素値）：

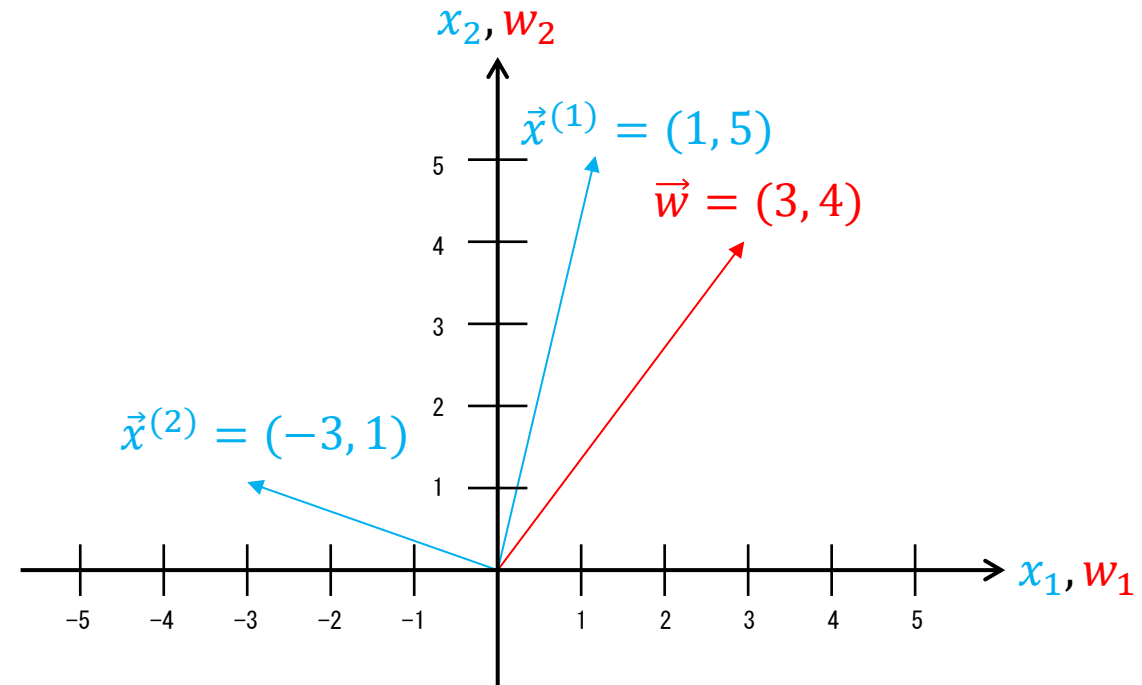
x_1 = 0	x_2 = 255	x_3 = 0
x_4 = 0	x_5 = 255	x_6 = 0
x_7 = 0	x_8 = 255	x_9 = 0



人工ニューロン (パーセプトロン)



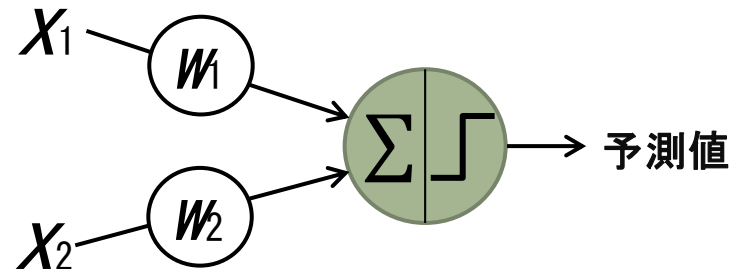
$$\text{予測値} = \text{step}(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$



$$\vec{x}^{(1)} \cdot \vec{w} = 23 \rightarrow \text{予測値: } 1$$

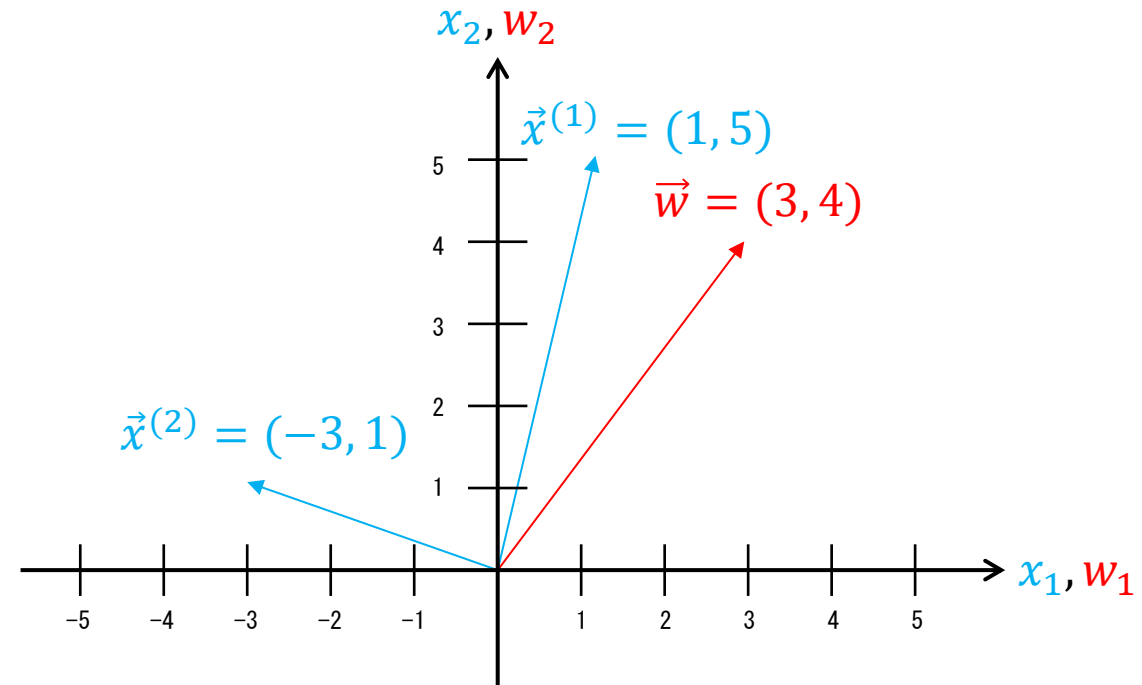
$$\vec{x}^{(2)} \cdot \vec{w} = -5 \rightarrow \text{予測値: } 0$$

人工ニューロン (パーセプトロン)



$$z = \vec{x} \cdot \vec{w}$$

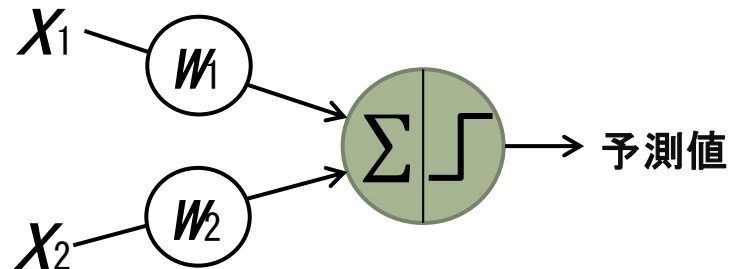
$$\text{予測値} = \text{step}(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$



$$\vec{x}^{(1)} \cdot \vec{w} = 23 \rightarrow \text{予測値: } 1$$

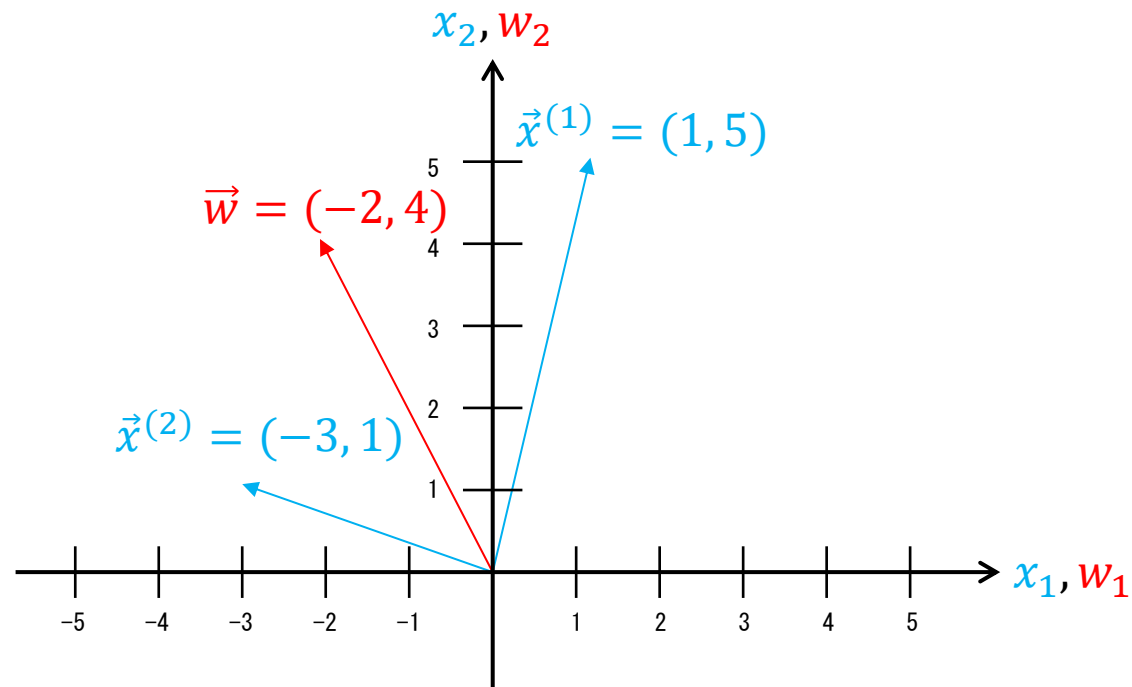
$$\vec{x}^{(2)} \cdot \vec{w} = -5 \rightarrow \text{予測値: } 0 \quad (\text{正解: } 1)$$

人工ニューロン (パーセプトロン)



$$z = \vec{x} \cdot \vec{w}$$

$$\text{予測値} = \text{step}(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$



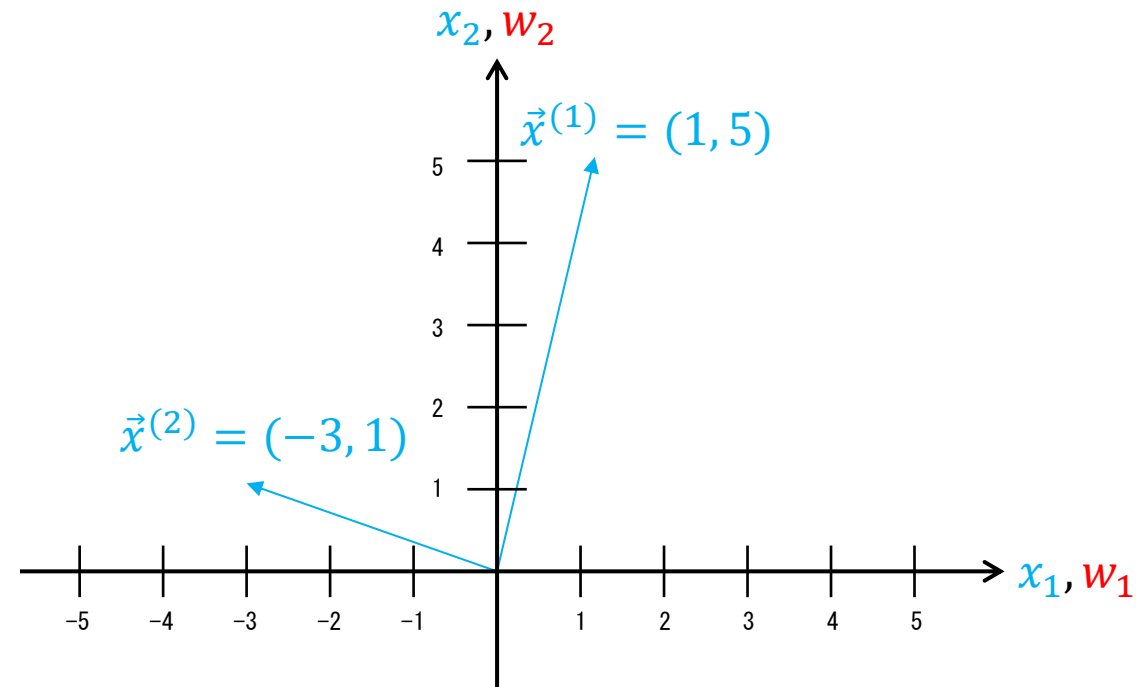
$$\vec{x}^{(1)} \cdot \vec{w} = 18 \rightarrow \text{予測値: } 1$$

$$\vec{x}^{(2)} \cdot \vec{w} = 10 \rightarrow \text{予測値: } 1 \quad (\text{正解: } 1)$$

人工ニューロン（パーセプトロン）

学習の手順：

1. 初期化：重み ($\vec{w}$) を乱数のベクトルにする
2. 最適化：
 - 各学習サンプルを使って次の処理を行う：
 - for $\vec{x}^{(i)}, y^{(i)}$ in X, Y
 - 予測を行う：
 - $\text{予測値} \leftarrow \text{step}(\vec{x}^{(i)} \cdot \vec{w})$
 - 正解と予測値の差を求める：
 - $\text{誤差} \leftarrow y^{(i)} - \text{予測値}$
 - 重みを一つ一つ更新する：
 - for w_j in $\vec{w}$
 - $w_j \leftarrow w_j + \text{誤差} \times x_j^{(i)} \times \text{学習率}^*$
3. 必要に応じて最適化を繰り返す



*0.1などの小さな固定値

人工ニューロン（パーセプトロン）

学習の手順：

1. 初期化：重み ($\vec{w}$) を乱数のベクトルにする

2. 最適化：

各学習サンプルを使って次の処理を行う：

for $\vec{x}^{(i)}, y^{(i)}$ in X, Y

予測を行う：

予測値 $\leftarrow \text{step}(\vec{x}^{(i)} \cdot \vec{w})$

正解と予測値の差を求める：

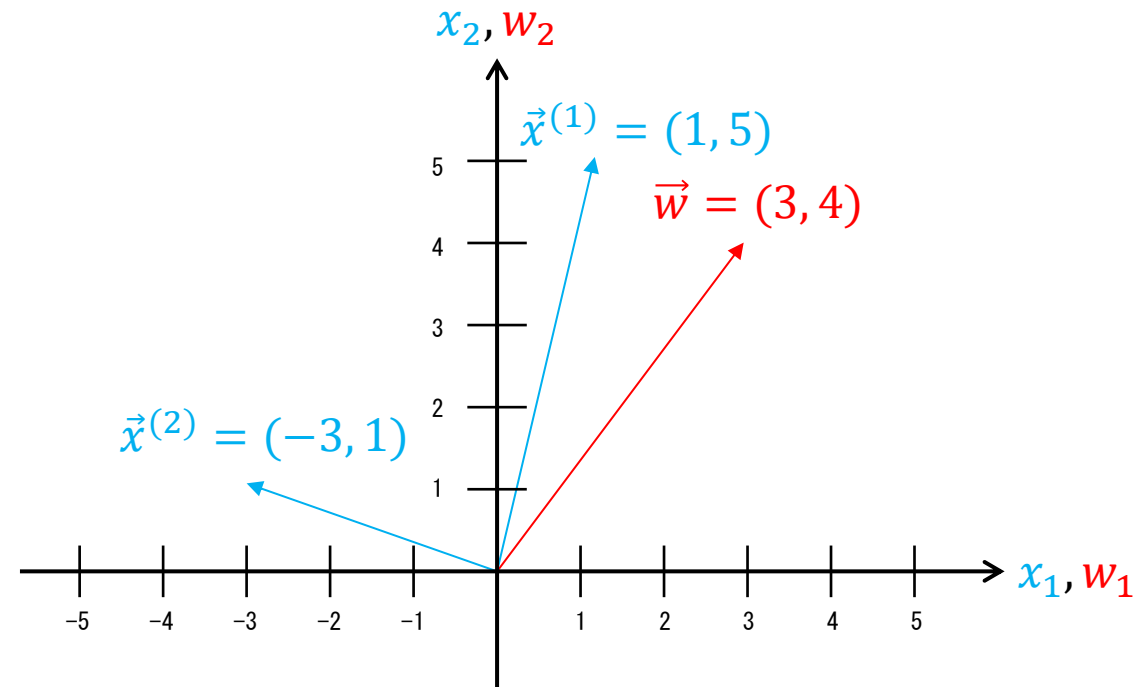
誤差 $\leftarrow y^{(i)} - \text{予測値}$

重みを一つ一つ更新する：

for w_j in $\vec{w}$

$w_j \leftarrow w_j + \text{誤差} \times x_j^{(i)} \times \text{学習率}^*$

3. 必要に応じて最適化を繰り返す



初期化： $\vec{w} \leftarrow (3, 4)$

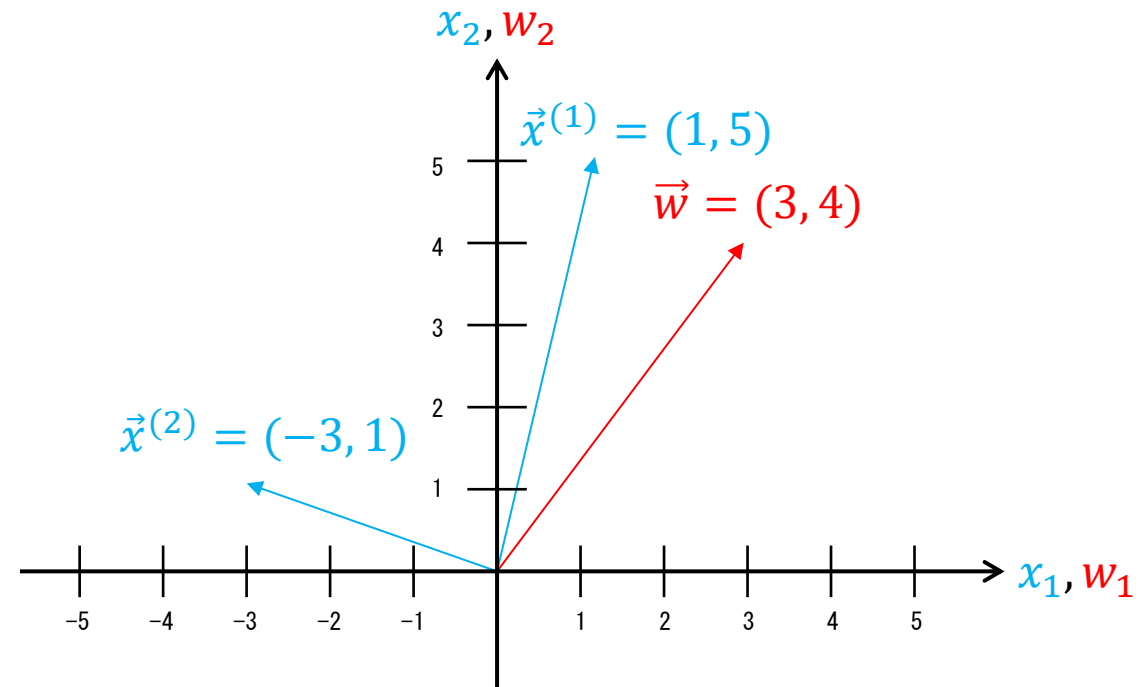
*0.1などの小さな固定値

人工ニューロン（パーセプトロン）

学習の手順：

1. 初期化：重み ($\vec{w}$) を乱数のベクトルにする
2. 最適化：

各学習サンプルを使って次の処理を行う：
 for $\vec{x}^{(i)}, y^{(i)}$ in X, Y
 予測を行う：
 $\text{予測値} \leftarrow \text{step}(\vec{x}^{(i)} \cdot \vec{w})$
 正解と予測値の差を求める：
 $\text{誤差} \leftarrow y^{(i)} - \text{予測値}$
 重みを一つ一つ更新する：
 for w_j in $\vec{w}$
 $w_j \leftarrow w_j + \text{誤差} \times x_j^{(i)} \times \text{学習率}^*$
3. 必要に応じて最適化を繰り返す



*0.1などの小さな固定値

人工ニューロン（パーセプトロン）

学習の手順：

1. 初期化：重み ($\vec{w}$) を乱数のベクトルにする
2. 最適化：

各学習サンプルを使って次の処理を行う：

for $\vec{x}^{(i)}, y^{(i)}$ in X, Y

予測を行う：

予測値 $\leftarrow \text{step}(\vec{x}^{(i)} \cdot \vec{w})$

正解と予測値の差を求める：

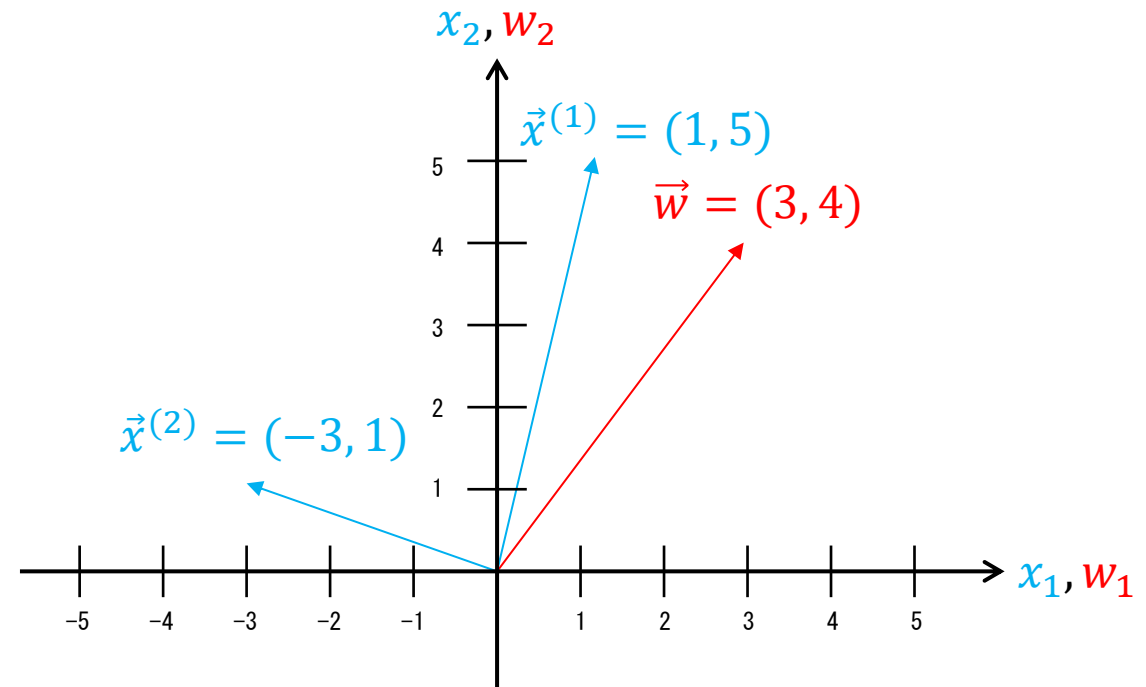
誤差 $\leftarrow y^{(i)} - \text{予測値}$

重みを一つ一つ更新する：

for w_j in $\vec{w}$

$w_j \leftarrow w_j + \text{誤差} \times x_j^{(i)} \times \text{学習率}^*$

3. 必要に応じて最適化を繰り返す



予測： $\text{step}(\vec{x}^{(1)} \cdot \vec{w}) = \text{step}(23) = 1$

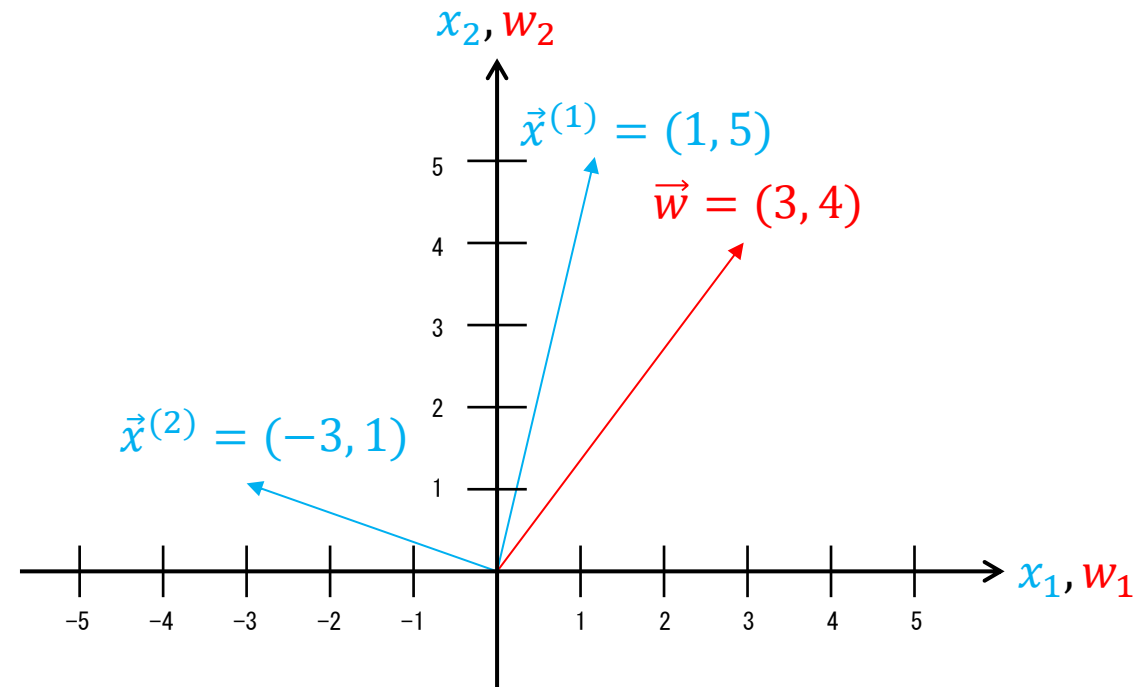
*0.1などの小さな固定値

人工ニューロン（パーセプトロン）

学習の手順：

1. 初期化：重み ($\vec{w}$) を乱数のベクトルにする
2. 最適化：
 - 各学習サンプルを使って次の処理を行う：
 - for $\vec{x}^{(i)}, y^{(i)}$ in X, Y
 - 予測を行う：
 - $\text{予測値} \leftarrow \text{step}(\vec{x}^{(i)} \cdot \vec{w})$
 - 正解と予測値の差を求める：
 - $\text{誤差} \leftarrow y^{(i)} - \text{予測値}$
 - 重みを一つ一つ更新する：
 - for w_j in $\vec{w}$
 - $w_j \leftarrow w_j + \text{誤差} \times x_j^{(i)} \times \text{学習率}^*$
3. 必要に応じて最適化を繰り返す

※0.1などの小さな固定値



予測： $\text{step}(\vec{x}^{(1)} \cdot \vec{w}) = \text{step}(23) = 1$

正解： $y^{(1)} = 1$

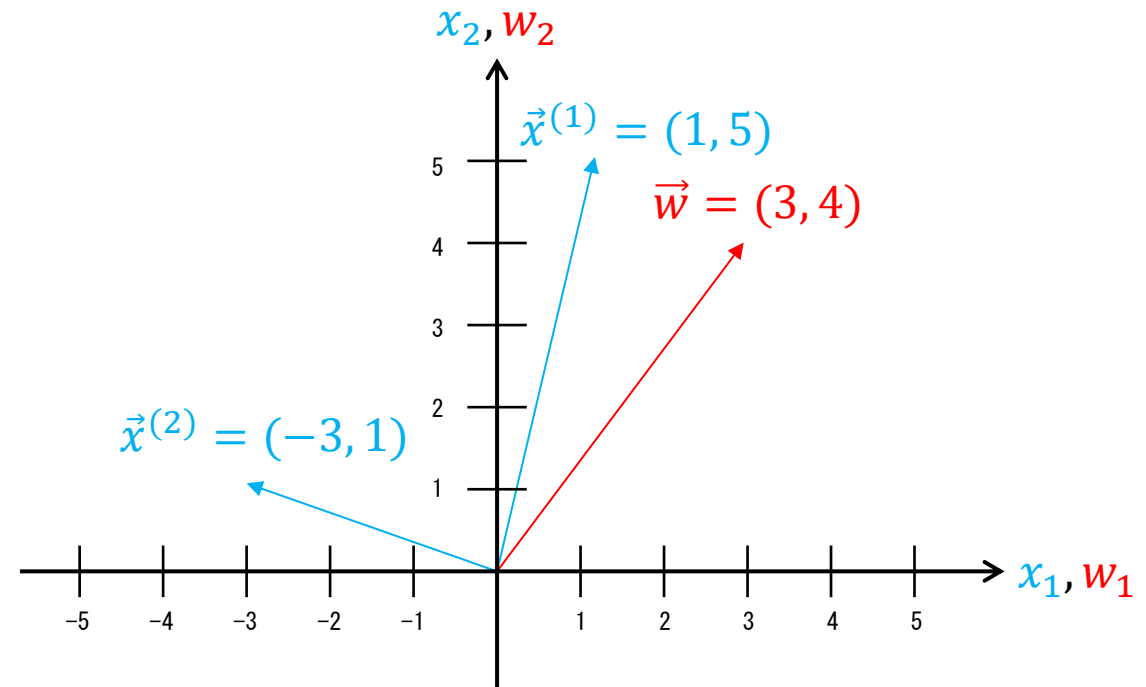
誤差： $1 - 1 = 0$

人工ニューロン（パーセプトロン）

学習の手順：

1. 初期化：重み ($\vec{w}$) を乱数のベクトルにする
2. 最適化：
 - 各学習サンプルを使って次の処理を行う：
 - for $\vec{x}^{(i)}, y^{(i)}$ in X, Y
 - 予測を行う：
 - $\text{予測値} \leftarrow \text{step}(\vec{x}^{(i)} \cdot \vec{w})$
 - 正解と予測値の差を求める：
 - $\text{誤差} \leftarrow y^{(i)} - \text{予測値}$
 - 重みを一つ一つ更新する：
 - for w_j in $\vec{w}$
 - $w_j \leftarrow w_j + \text{誤差} \times x_j^{(i)} \times \text{学習率}^*$
3. 必要に応じて最適化を繰り返す

※0.1などの小さな固定値



重みの更新：

$$w_1 \leftarrow w_1 + 0 \times 1 \times \text{学習率} \quad (\text{更新なし})$$

$$w_2 \leftarrow w_2 + 0 \times 5 \times \text{学習率} \quad (\text{更新なし})$$

人工ニューロン（パーセプトロン）

学習の手順：

1. 初期化：重み ($\vec{w}$) を乱数のベクトルにする
2. 最適化：
各学習サンプルを使って次の処理を行う：
for $\vec{x}^{(i)}, y^{(i)}$ in X, Y

予測を行う：

$$\text{予測値} \leftarrow \text{step}(\vec{x}^{(i)} \cdot \vec{w})$$

正解と予測値の差を求める：

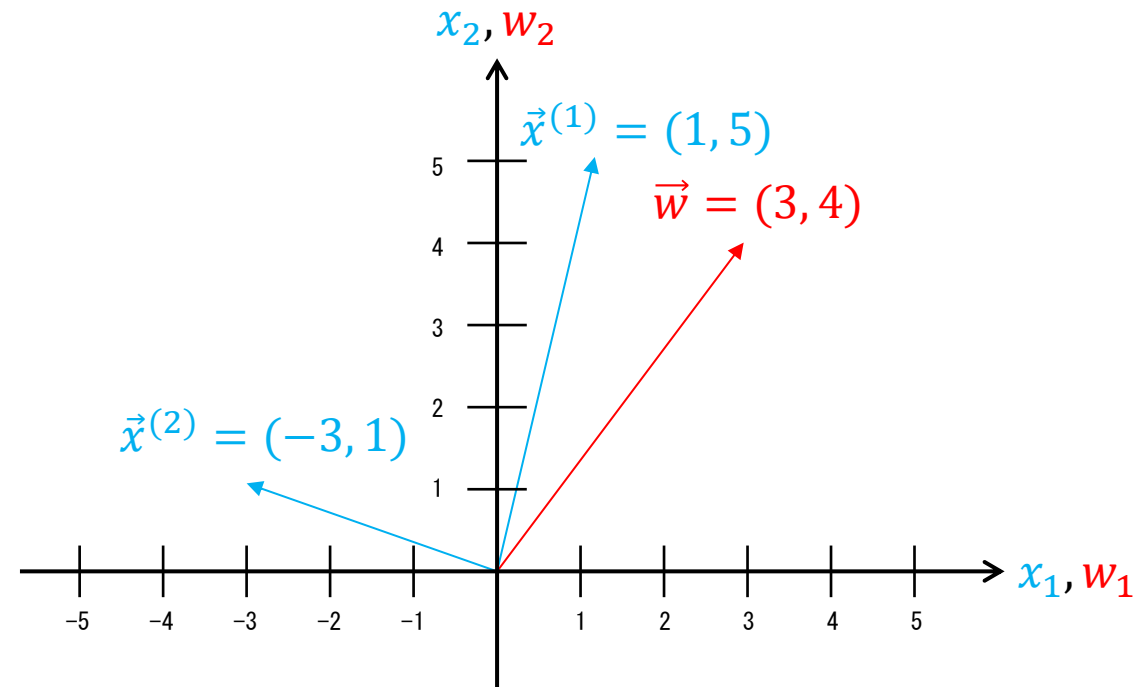
$$\text{誤差} \leftarrow y^{(i)} - \text{予測値}$$

重みを一つ一つ更新する：

for w_j in $\vec{w}$

$$w_j \leftarrow w_j + \text{誤差} \times x_j^{(i)} \times \text{学習率}^*$$

3. 必要に応じて最適化を繰り返す



$$\text{予測} : \text{step}(\vec{x}^{(2)} \cdot \vec{w}) = \text{step}(-5) = 0$$

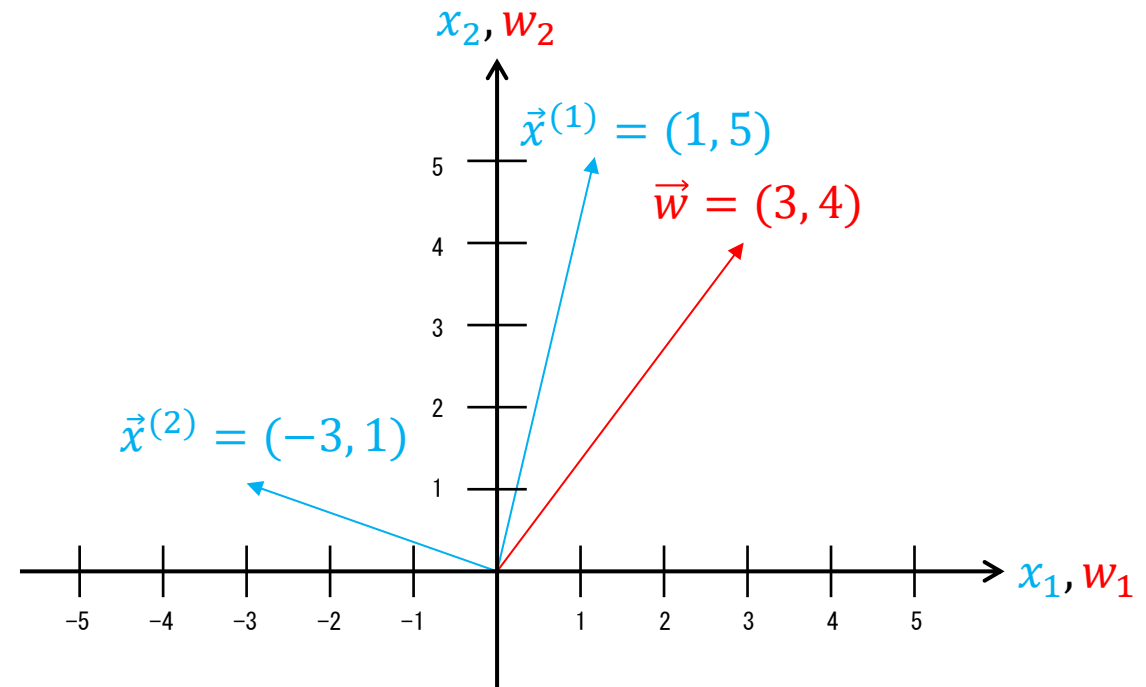
*0.1などの小さな固定値

人工ニューロン（パーセプトロン）

学習の手順：

1. 初期化：重み ($\vec{w}$) を乱数のベクトルにする
2. 最適化：
 - 各学習サンプルを使って次の処理を行う：
 - for $\vec{x}^{(i)}, y^{(i)}$ in X, Y
 - 予測を行う：
 - $\text{予測値} \leftarrow \text{step}(\vec{x}^{(i)} \cdot \vec{w})$
 - 正解と予測値の差を求める：
 - $\text{誤差} \leftarrow y^{(i)} - \text{予測値}$
 - 重みを一つ一つ更新する：
 - for w_j in $\vec{w}$
 - $w_j \leftarrow w_j + \text{誤差} \times x_j^{(i)} \times \text{学習率}^*$
3. 必要に応じて最適化を繰り返す

※0.1などの小さな固定値



予測： $\text{step}(\vec{x}^{(2)} \cdot \vec{w}) = \text{step}(-5) = 0$

正解： $y^{(2)} = 1$

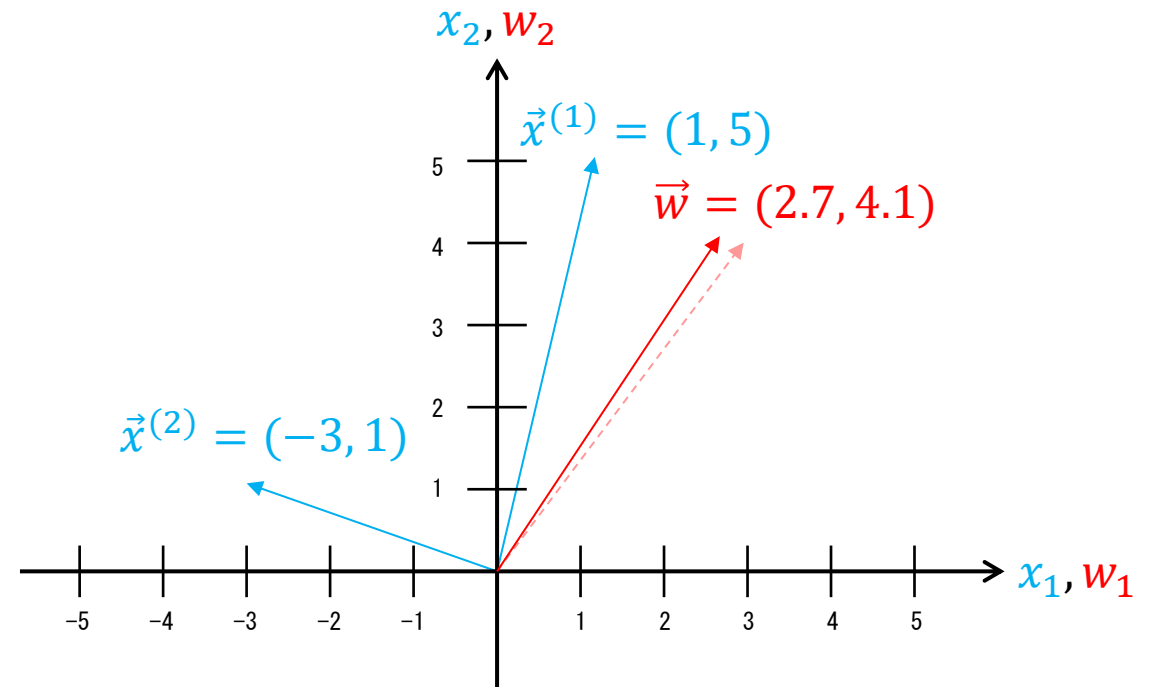
誤差： $1 - 0 = 1$

人工ニューロン（パーセプトロン）

学習の手順：

1. 初期化：重み ($\vec{w}$) を乱数のベクトルにする
2. 最適化：
 - 各学習サンプルを使って次の処理を行う：
 - for $\vec{x}^{(i)}, y^{(i)}$ in X, Y
 - 予測を行う：
 - $\text{予測値} \leftarrow \text{step}(\vec{x}^{(i)} \cdot \vec{w})$
 - 正解と予測値の差を求める：
 - $\text{誤差} \leftarrow y^{(i)} - \text{予測値}$
 - 重みを一つ一つ更新する：
 - for w_j in $\vec{w}$
 - $w_j \leftarrow w_j + \text{誤差} \times x_j^{(i)} \times \text{学習率}^*$
3. 必要に応じて最適化を繰り返す

※0.1などの小さな固定値



重みの更新：

$$w_1 \leftarrow w_1 + 1 \times -3 \times \text{学習率}(0.1) = 3 + (-0.3)$$

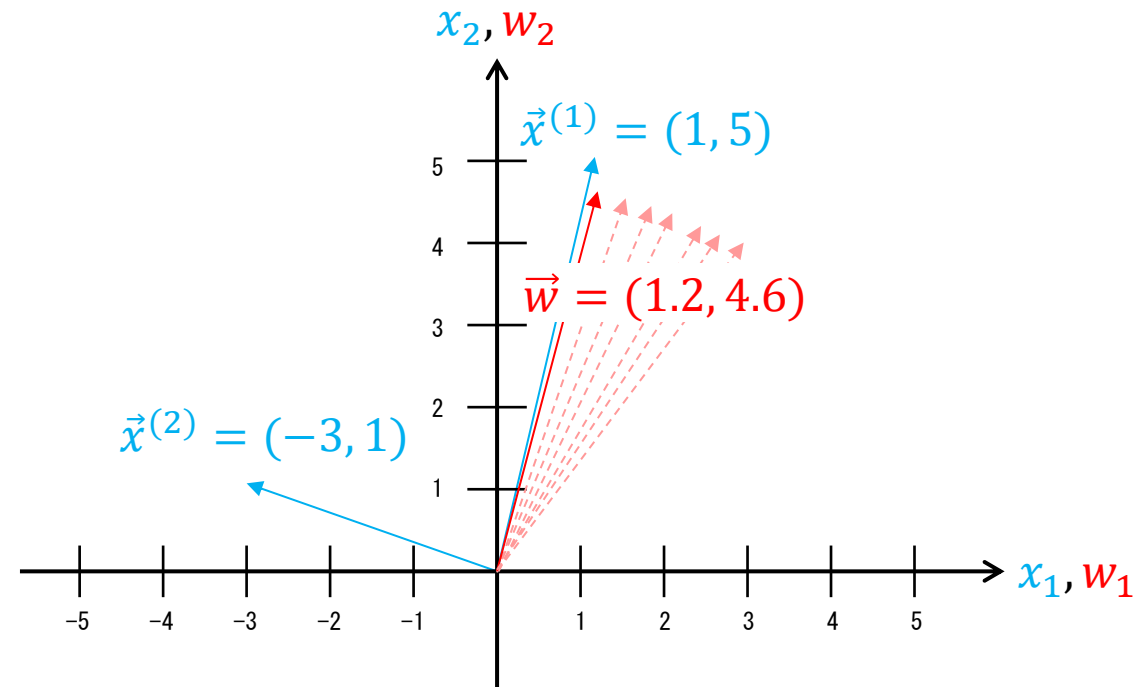
$$w_2 \leftarrow w_2 + 1 \times 1 \times \text{学習率}(0.1) = 4 + 0.1$$

人工ニューロン（パーセプトロン）

学習の手順：

1. 初期化：重み ($\vec{w}$) を乱数のベクトルにする
2. 最適化：
 - 各学習サンプルを使って次の処理を行う：
 - for $\vec{x}^{(i)}, y^{(i)}$ in X, Y
 - 予測を行う：
 - $\text{予測値} \leftarrow \text{step}(\vec{x}^{(i)} \cdot \vec{w})$
 - 正解と予測値の差を求める：
 - $\text{誤差} \leftarrow y^{(i)} - \text{予測値}$
 - 重みを一つ一つ更新する：
 - for w_j in $\vec{w}$
 - $w_j \leftarrow w_j + \text{誤差} \times x_j^{(i)} \times \text{学習率}^*$

3. 必要に応じて最適化を繰り返す



$$\vec{x}^{(1)} \cdot \vec{w} = 24.2 \rightarrow \text{予測値: } 1 \quad (\text{正解: } 1)$$

$$\vec{x}^{(2)} \cdot \vec{w} = 1 \rightarrow \text{予測値: } 1 \quad (\text{正解: } 1)$$

*0.1などの小さな固定値

人工ニューロン（パーセプトロン）

パーセプトロンを用いて手書き数字認識システムを実装したPythonのソースコード:

[Colab Notebook](#)

[Jupyter Notebook](#)

