

自然言語処理プログラミング勉強会 10 ニューラルネット

Graham Neubig
奈良先端科学技術大学院大学 (NAIST)

予測問題

x が与えられた時
 y を予測する

今回の例

- Wikipedia 記事の最初の 1 文が与えられた時
- その記事が人物についての記事かどうかを予測

与えられた情報

Gonso was a Sanron sect priest (754-827)
in the late Nara and early Heian periods.



予測

Yes!

Shichikuzan Chigogataki Fudomyoo is
a historical site located at Magura, Maizuru
City, Kyoto Prefecture.



No!

- これはもちろん、2 値予測

線形識別器

$$\begin{aligned} y &= \text{sign}(\mathbf{w} \cdot \boldsymbol{\varphi}(\mathbf{x})) \\ &= \text{sign}\left(\sum_{i=1}^I \mathbf{w}_i \cdot \varphi_i(\mathbf{x})\right) \end{aligned}$$

- \mathbf{x} : 入力
- $\boldsymbol{\varphi}(\mathbf{x})$: 素性関数のベクトル $\{\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), \dots, \varphi_I(\mathbf{x})\}$
- \mathbf{w} : 重みベクトル $\{w_1, w_2, \dots, w_I\}$
- y : 予測値、「yes」なら +1、「no」なら -1
 - $\text{sign}(v)$ は「 $v \geq 0$ 」の場合 +1、そうでない場合 -1

素性関数の例： 1-gram 素性

- 「事例において、ある単語が何回現れるか？」

x = A site , located in Maizuru , Kyoto

$$\varphi_{\text{unigram "A"}}(x) = 1 \quad \varphi_{\text{unigram "site"}}(x) = 1 \quad \varphi_{\text{unigram ","}}(x) = 2$$

$$\varphi_{\text{unigram "located"}}(x) = 1 \quad \varphi_{\text{unigram "in"}}(x) = 1$$

$$\varphi_{\text{unigram "Maizuru"}}(x) = 1 \quad \varphi_{\text{unigram "Kyoto"}}(x) = 1$$

$$\left. \begin{array}{l} \varphi_{\text{unigram "the"}}(x) = 0 \quad \varphi_{\text{unigram "temple"}}(x) = 0 \\ \dots \end{array} \right\} \text{残りはすべて 0}$$

- 便宜のため、素性 $\text{ID}(\varphi_1)$ の代わりに、素性の名前 ($\varphi_{\text{unigram "A"}}$) を利用

重み付き和の計算

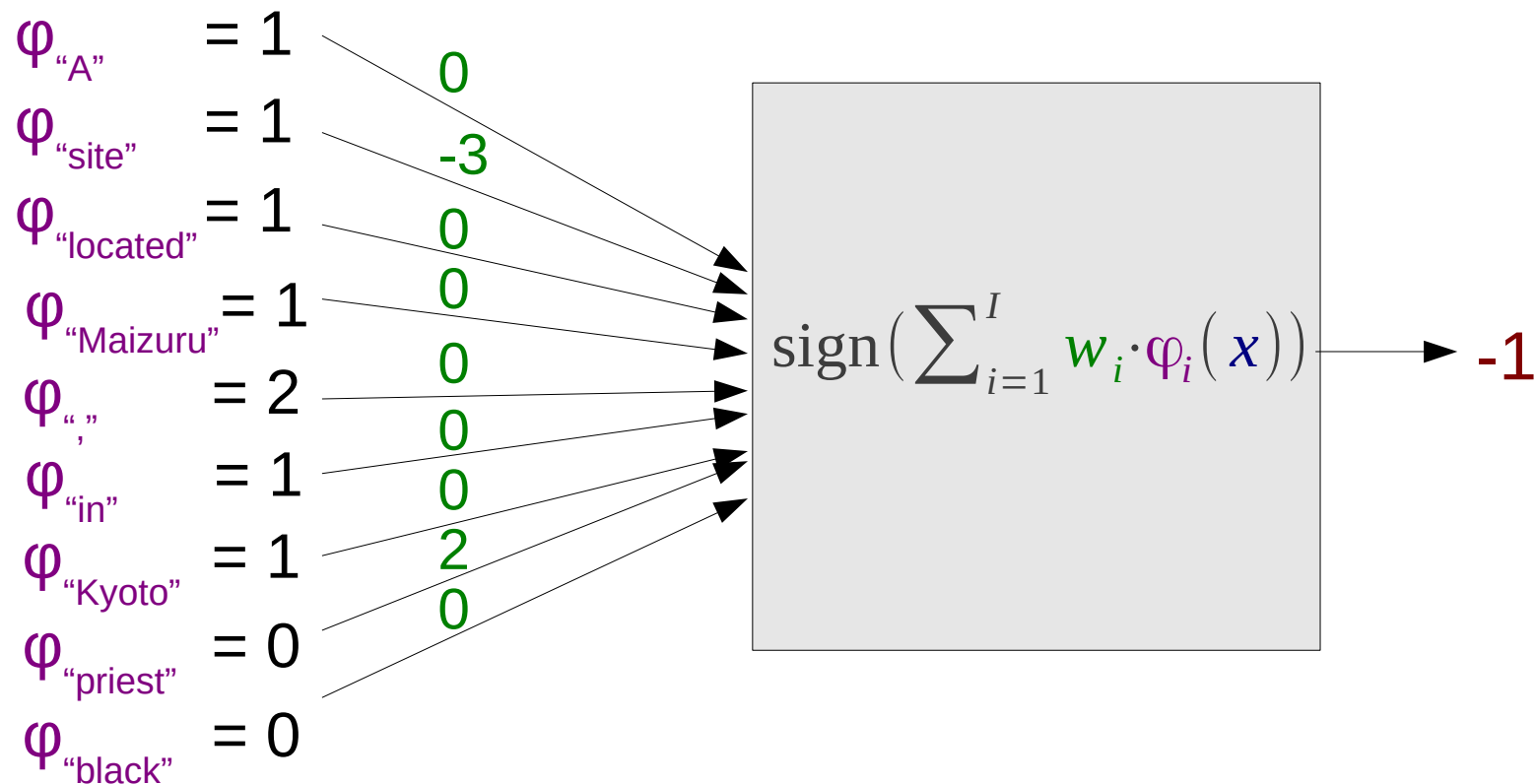
x = A site , located in Maizuru , Kyoto

$\varphi_{\text{unigram "A"}}(x) = 1$		$W_{\text{unigram "a"}} = 0$		0	+
$\varphi_{\text{unigram "site"}}(x) = 1$		$W_{\text{unigram "site"}} = -3$		-3	+
$\varphi_{\text{unigram "located"}}(x) = 1$		$W_{\text{unigram "located"}} = 0$		0	+
$\varphi_{\text{unigram "Maizuru"}}(x) = 1$		$W_{\text{unigram "Maizuru"}} = 0$		0	+
$\varphi_{\text{unigram " , "}}(x) = 2$	*	$W_{\text{unigram " , "}} = 0$	=	0	+
$\varphi_{\text{unigram "in"}}(x) = 1$		$W_{\text{unigram "in"}} = 0$		0	+
$\varphi_{\text{unigram "Kyoto"}}(x) = 1$		$W_{\text{unigram "Kyoto"}} = 0$		0	+
$\varphi_{\text{unigram "priest"}}(x) = 0$		$W_{\text{unigram "priest"}} = 2$		0	+
$\varphi_{\text{unigram "black"}}(x) = 0$		$W_{\text{unigram "black"}} = 0$		0	+
	
				=	

-3 \rightarrow No!

パーセプトロン

- 重み付き和を計算する「機械」として考える



numpy でパーセプトロン

numpy とは

- Python で利用できる、強力な計算ライブラリ
- ベクトルや行列の掛け算などが簡単
- SciPy というパッケージの一部（SciPy は機械学習アルゴリズムなども実装）

numpy の使用例 (ベクトル編)

```
import numpy as np

a = np.array( [20,30,40,50] )
b = np.array( [0,1,2,3] )
print(a - b)           # Subtract each element
print(b ** 2)          # Take the power of each element
print(10 * np.tanh(b)) # Hyperbolic tangent * 10 of each element
print(a < 35)          # Check if each element is less than 35
```

numpy の使用例 (行列編)

```
import numpy as np
```

```
A = np.array( [[1,1],
               [0,1]] )
```

```
B = np.array( [[2,0],
               [3,4]] )
```

<pre>print(A * B)</pre>	<pre># elementwise product</pre>
<pre>print(np.dot(A,B))</pre>	<pre># dot product</pre>
<pre>print(B.T)</pre>	<pre># transpose</pre>

パーセプトロンの予測

```
predict_one(w, phi)  
    score = 0  
    for each name, value in phi           # score =  $w * \phi(x)$   
        if name exists in w  
            score += value * w[name]  
    return (1 if score >= 0 else -1)
```

↓ numpy

```
predict_one(w, phi)  
    score = np.dot( w, phi )  
    return (1 if score[0] >= 0 else -1)
```

素性の ID 化

- numpy は密行列を利用→ 素性を整数 ID に変えたい！

```
ids = defaultdict(lambda: len(ids)) # ID に変換するトリック！
```

```
CREATE_FEATURES(x):
```

```
    create map phi
```

```
    split x into words
```

```
    for word in words
```

```
        phi[ids["UNI:" + word]] += 1
```

```
    return phi
```

素性の初期化

- 素性の数だけのベクトルを初期化
- ゼロで初期化

```
w = np.zeros(len(ids))
```

- [-0.5,0.5] でランダムに初期化

```
w = np.random(len(ids)) - 0.5
```

パーセプトロン学習の疑似コード

```
# 素性の数を数えて、重みを初期化する
create map ids
for each labeled pair x, y in the data
    create_features(x)
w = np.zeros(len(ids))

# 学習を行う
for / iterations
    for each labeled pair x, y in the data
        phi = create_features(x)
        y' = predict_one(w, phi)
        if y' != y
            update_weights(w, phi, y)

print w to weight_file
print ids to id_file
```

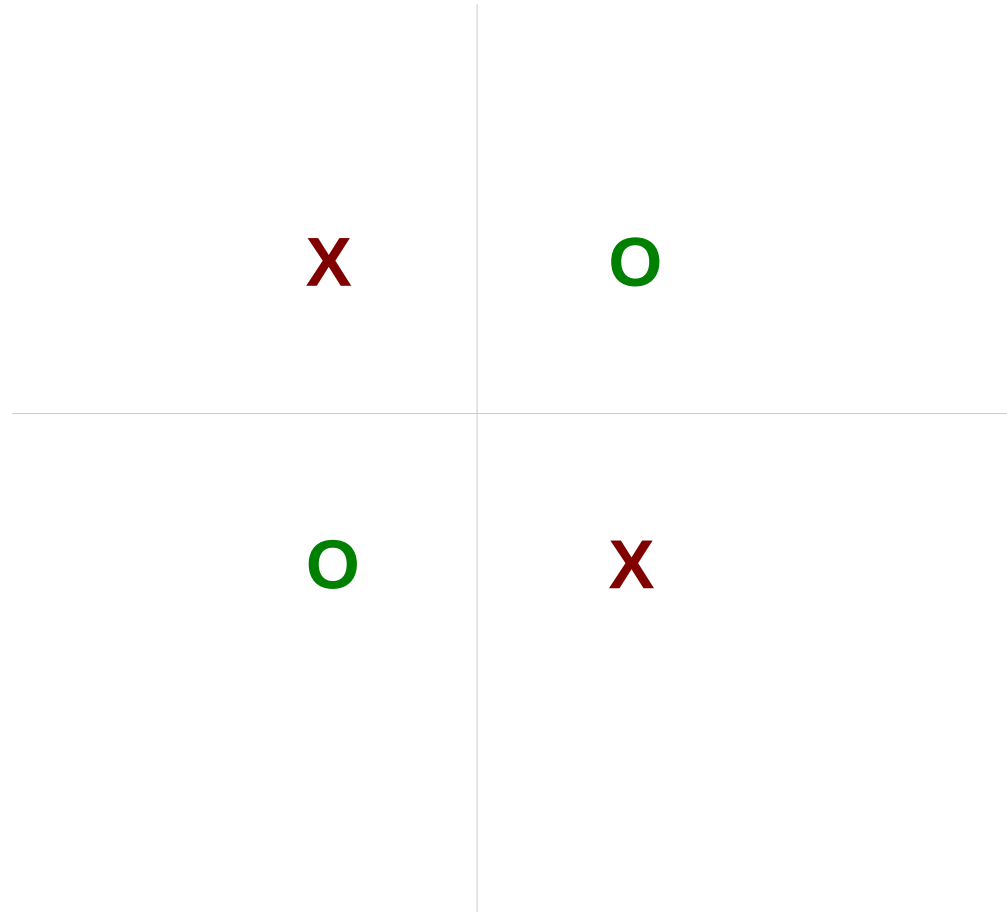
パーセプトロン予測の疑似コード

```
read ids from id_file  
read w from weights_file  
  
for each labeled pair x, y in the data  
    phi = create_features(x)  
    y' = predict_one(w, phi)
```


ニューラルネット

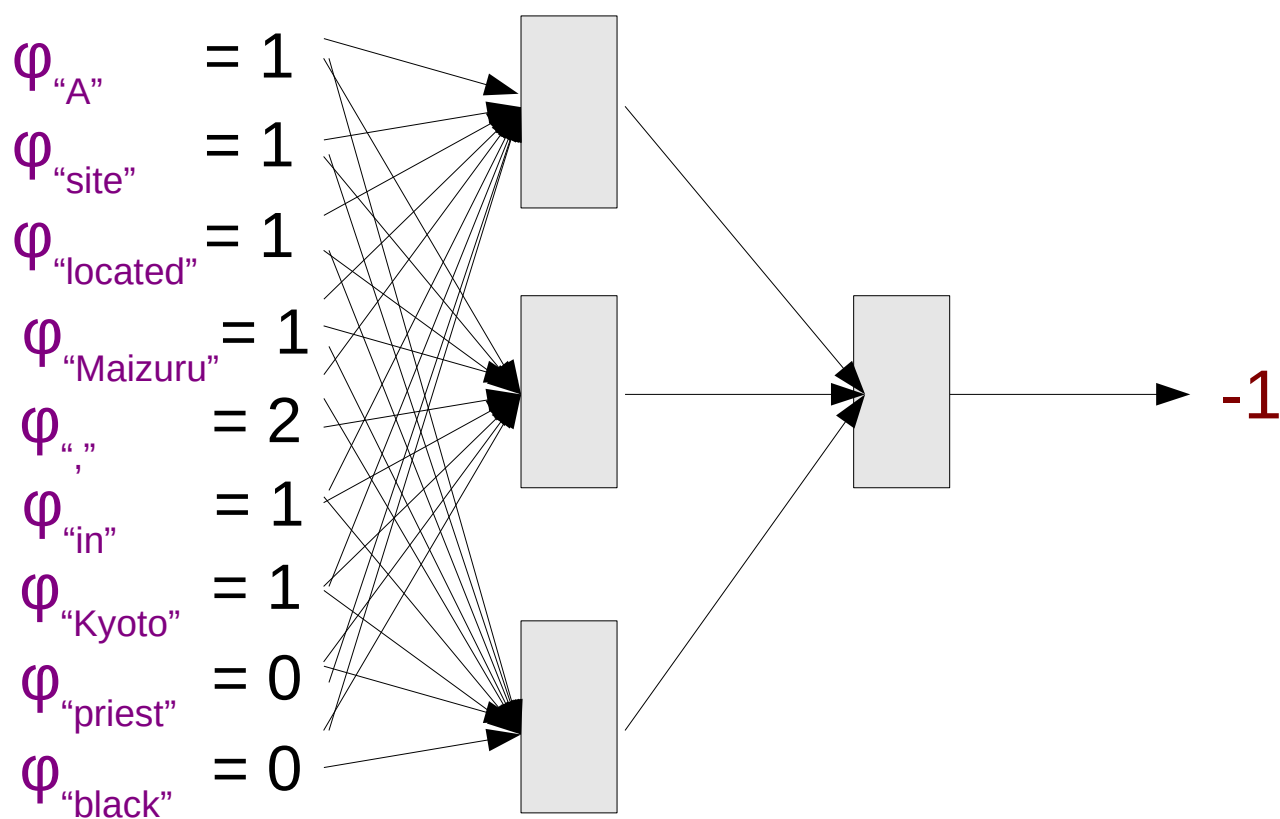
問題：線形分類のみ

- 線形分離不可能な問題に対して高い精度は実現不可



ニューラルネット

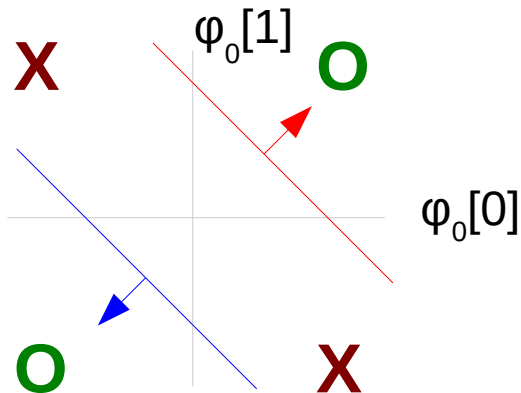
- 複数のパーセプトロンをつなげる



- モチベーション：線形でない関数も表現可能！

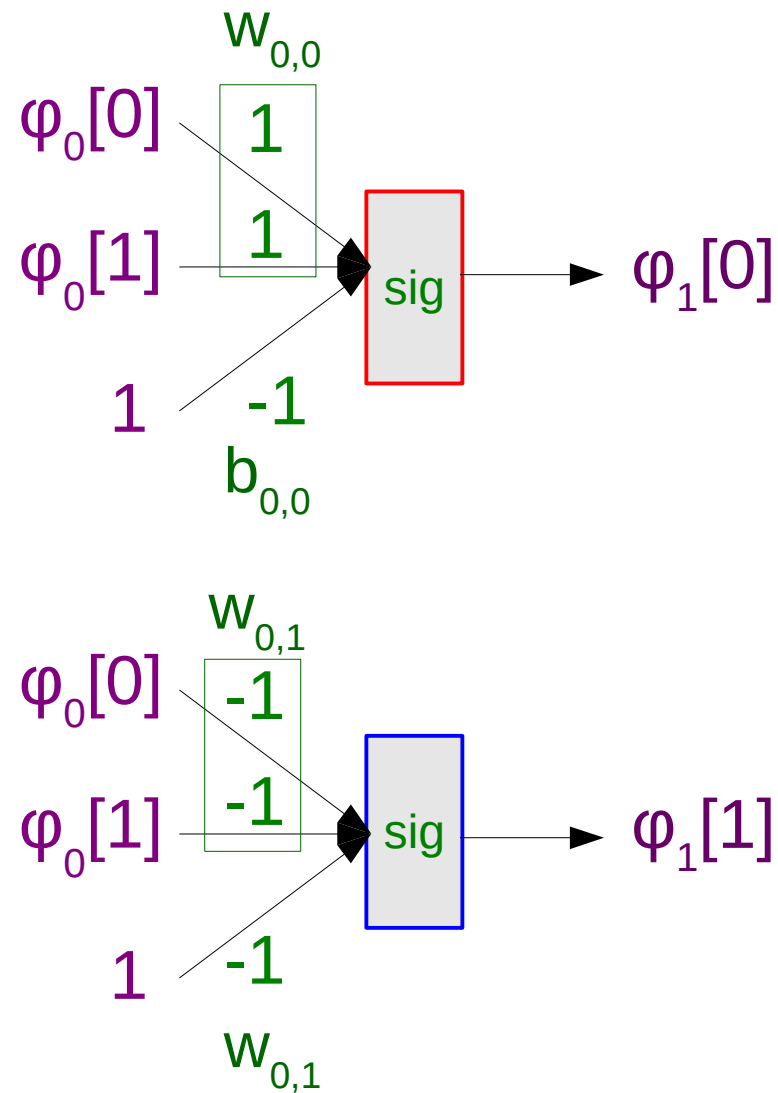
- 2つの分類器を作成

$$\varphi_0(x_1) = \{-1, 1\} \quad \varphi_0(x_2) = \{1, 1\}$$



$$\varphi_0(x_3) = \{-1, -1\} \quad \varphi_0(x_4) = \{1, -1\}$$

例：

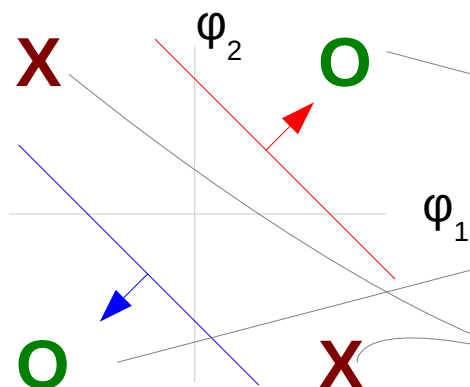


例：

- 分類器は新しい素性空間へマッピング

$$\varphi_0(x_1) = \{-1, 1\} \quad \varphi_0(x_2) = \{1, 1\}$$

$$\varphi_1(x_3) = \{-1, 1\}$$

 $\varphi_1[1]$


$$\varphi_0(x_3) = \{-1, -1\} \quad \varphi_0(x_4) = \{1, -1\}$$

$$\varphi_1(x_1) = \{-1, -1\}$$

$$\varphi_1(x_4) = \{-1, -1\}$$

$$\varphi_1(x_2) = \{1, -1\}$$

 $\varphi_1[0]$

1
1
-1

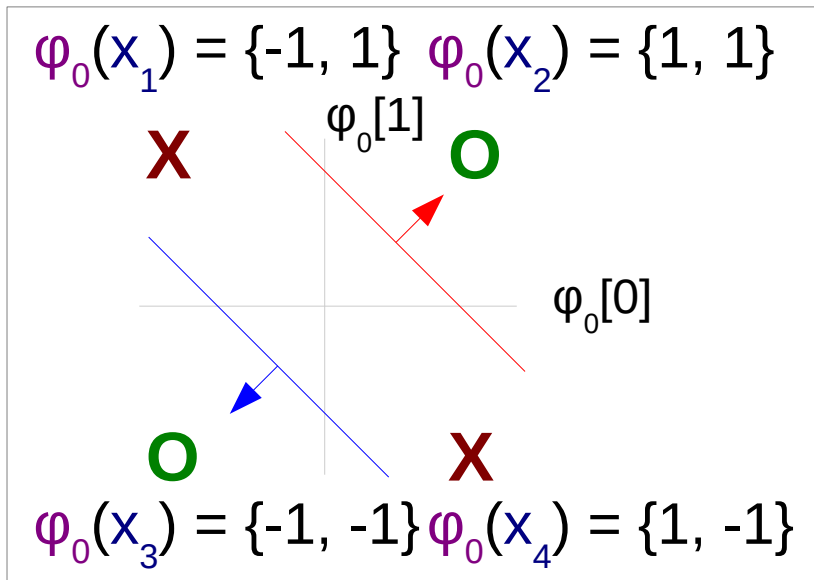
 $\varphi_1[0]$

-1
-1
-1

 $\varphi_1[1]$

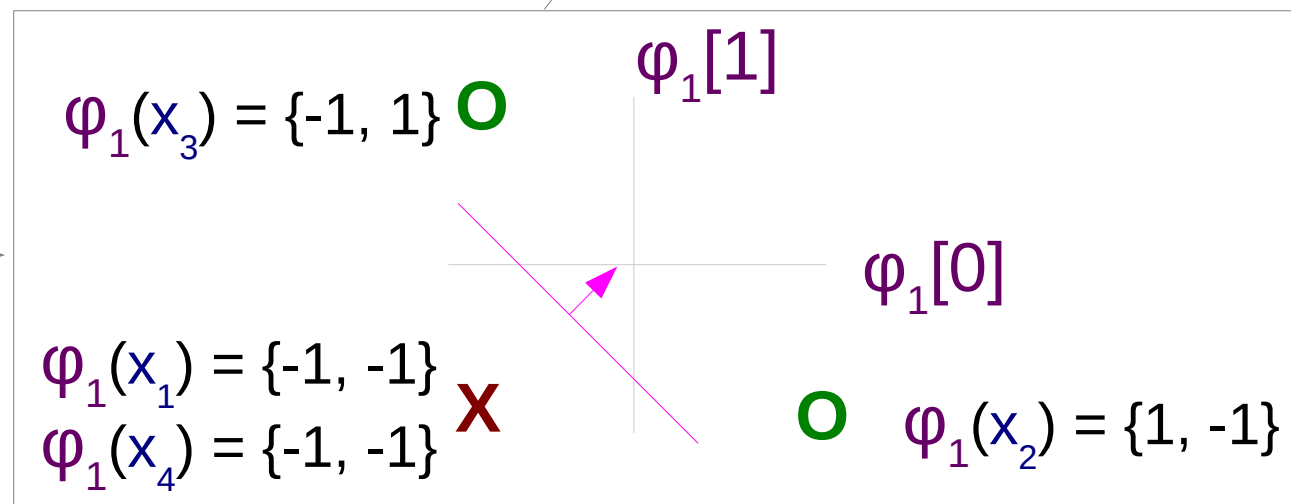
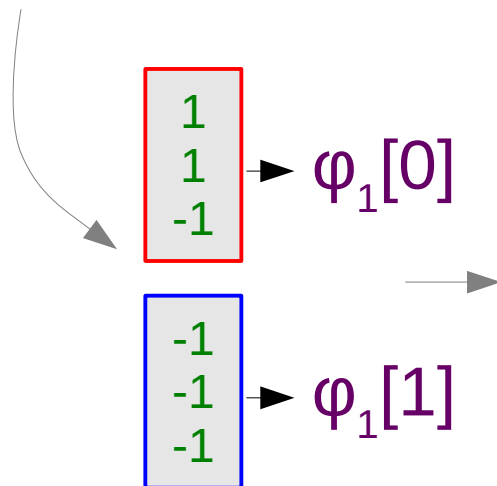
例：

- 新しい空間で、事例が分類可能に！



$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

$\varphi_2[0] = y$



2 層ニューラルネットの実装 (行列編)

入力

$\varphi_0 = \text{np.array}([1, -1])$

一層目の計算

$w_0 = \text{np.array}([[1, 1], [-1, -1]])$

$b_0 = \text{np.array}([-1, -1])$

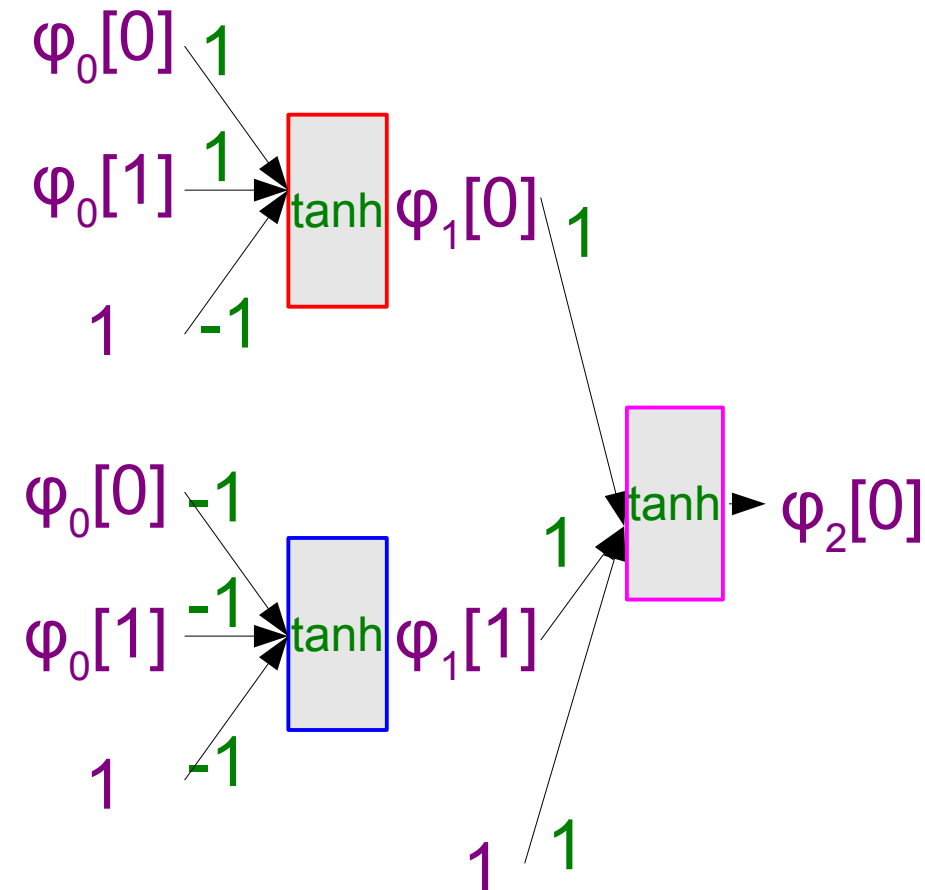
$\varphi_1 = \varphi_0 * w_0 + b_0$

2 層目の計算

$w_1 = \text{np.array}([[1, 1]])$

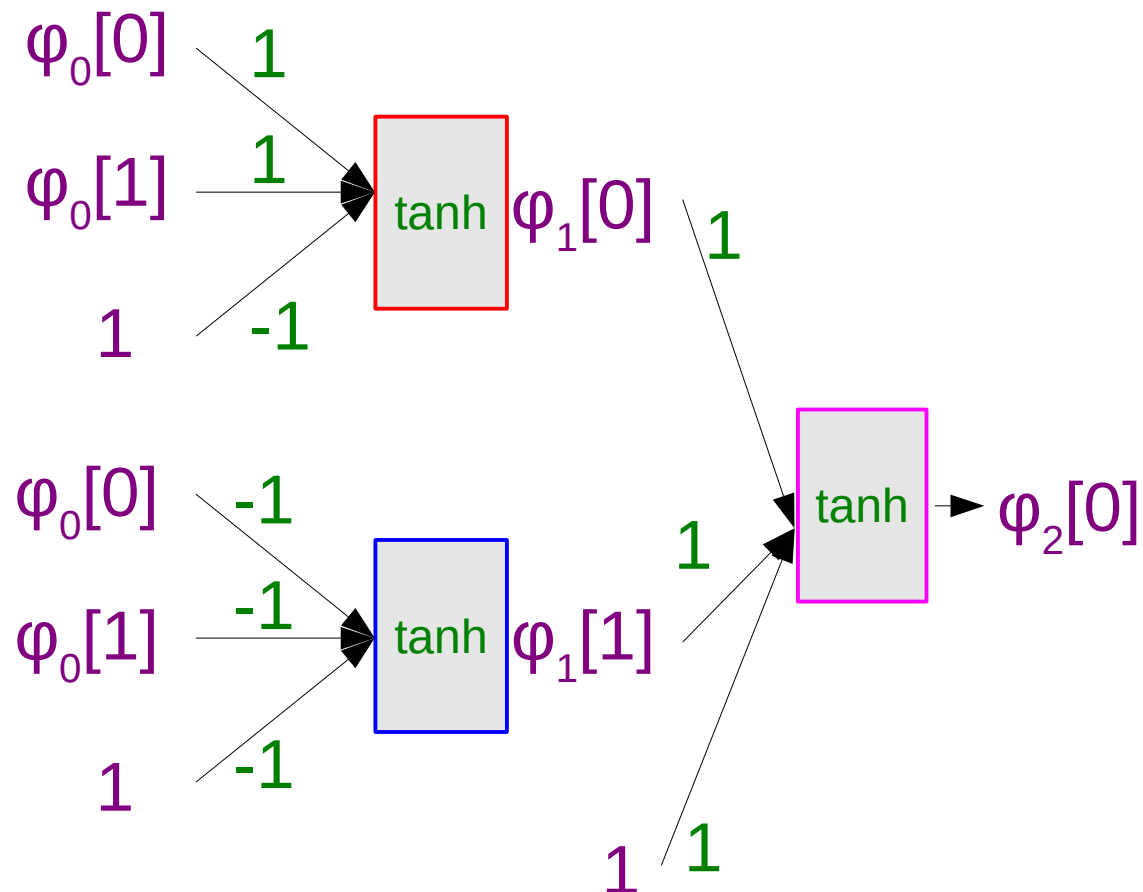
$b_1 = \text{np.array}([-1])$

$\varphi_2 = \varphi_1 * w_1 + b_1$



例：

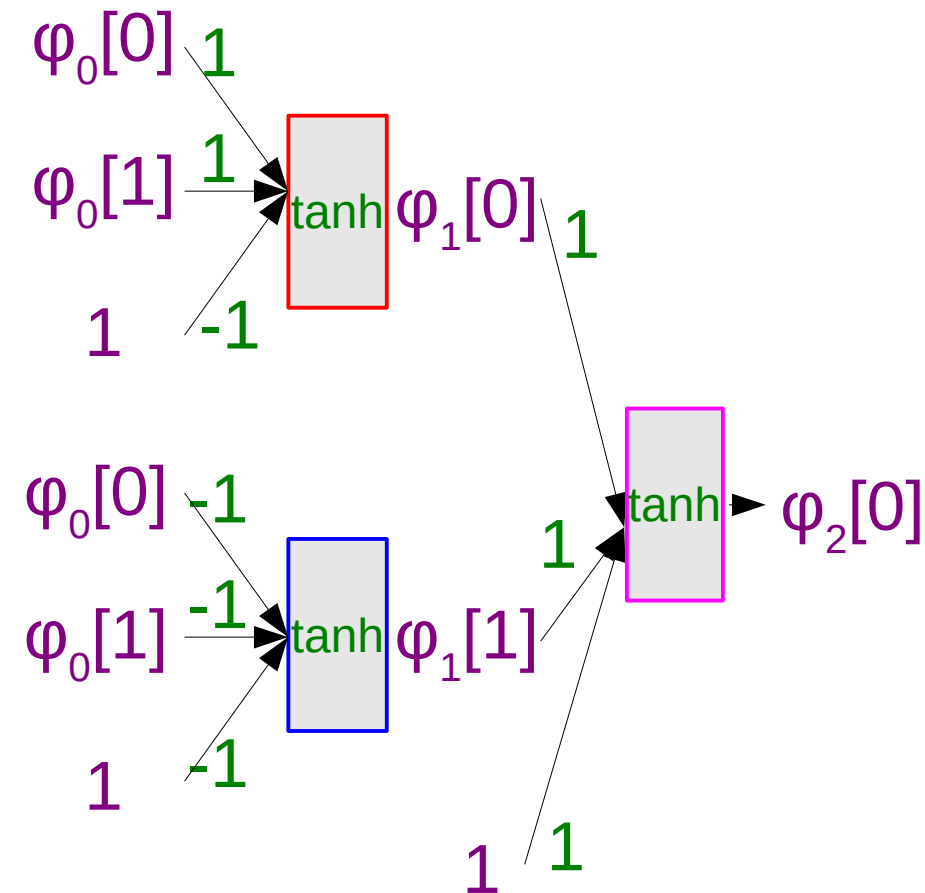
- 最終的なニューラルネット



2 層ニューラルネットの実装 (ベクトル編)

入力

$\varphi_0 = \text{np.array}([1, -1])$



一層目の計算

$w_{0,0} = \text{np.array}([1, 1])$

$b_{0,0} = \text{np.array}([-1])$

$w_{0,1} = \text{np.array}([-1, -1])$

$b_{0,1} = \text{np.array}([-1])$

$\varphi_1 = \text{np.zeros}(2)$

$\varphi_1[0] = \text{np.tanh}(\varphi_0 * w_{0,0} + b_{0,0})[0]$

$\varphi_1[1] = \text{np.tanh}(\varphi_0 * w_{0,1} + b_{0,1})[0]$

2 層目の計算

$w_{1,0} = \text{np.array}([1, 1])$

$b_{1,0} = \text{np.array}([-1])$

$\varphi_2 = \text{np.zeros}(1)$

$\varphi_2[0] = \text{np.tanh}(\varphi_1 * w_{1,0} + b_{1,0})[0]$

ニューラルネットの予測コード

```
predict_nn(network, phi_0)
phi = [phi_0, {}, {} ...] # 各層の値
for each layer i in 1 .. len(phi):
    w, b = network[i]
    # 前の層の値に基づいて値を計算
    phi[i] = phi[i-1] * w + b
return phi[-1][0] # 最後の層のパーセプトロンの値
```

tanh を用いたパーセプトロン学習

- エラーを計算：

$$\delta = y' - y$$

正解 システム出力

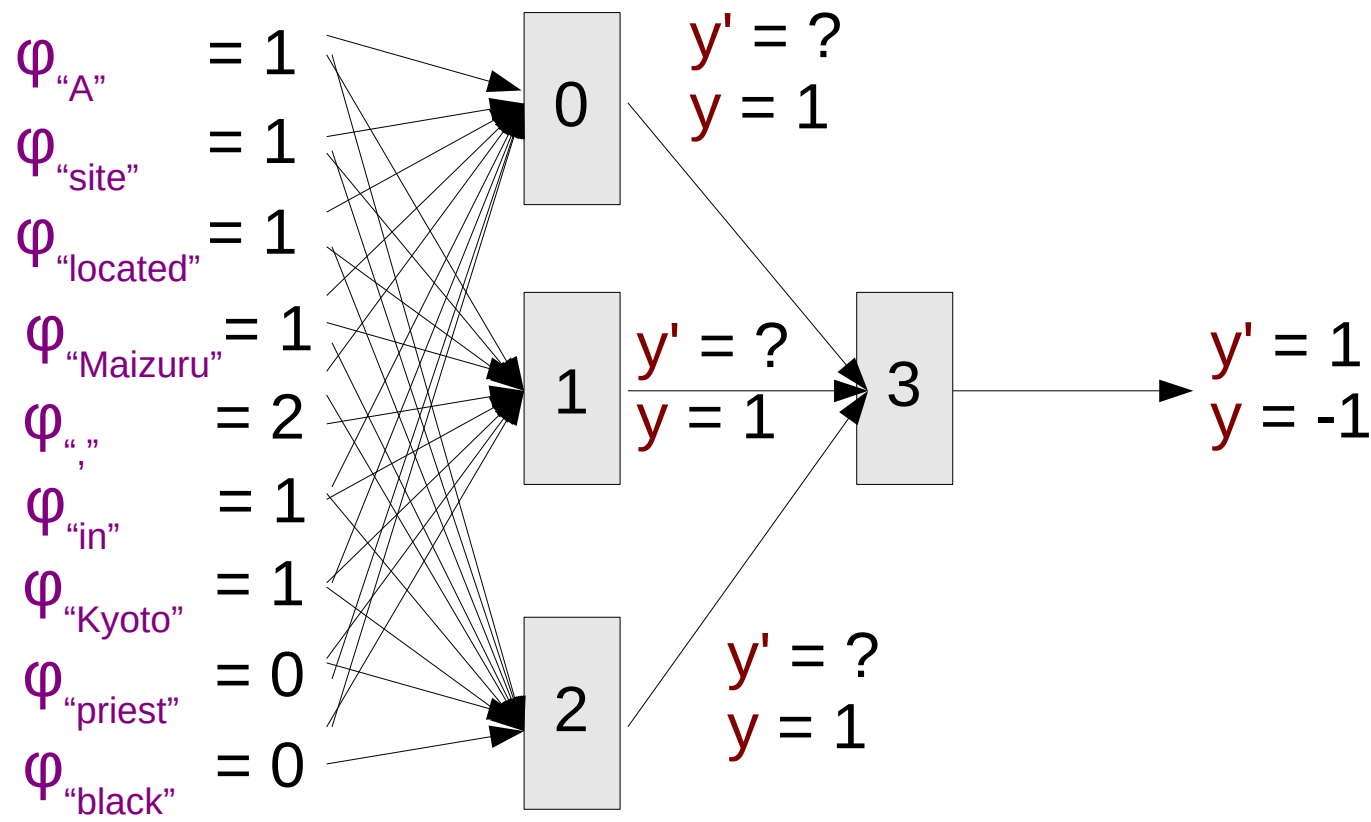
- 各重みを更新：

$$w \leftarrow w + \lambda \cdot \delta \cdot \varphi(x)$$

- λ は学習率
- (step 関数パーセプトロンでは $\delta = -2$ or $+2$, $\lambda = 1/2$)

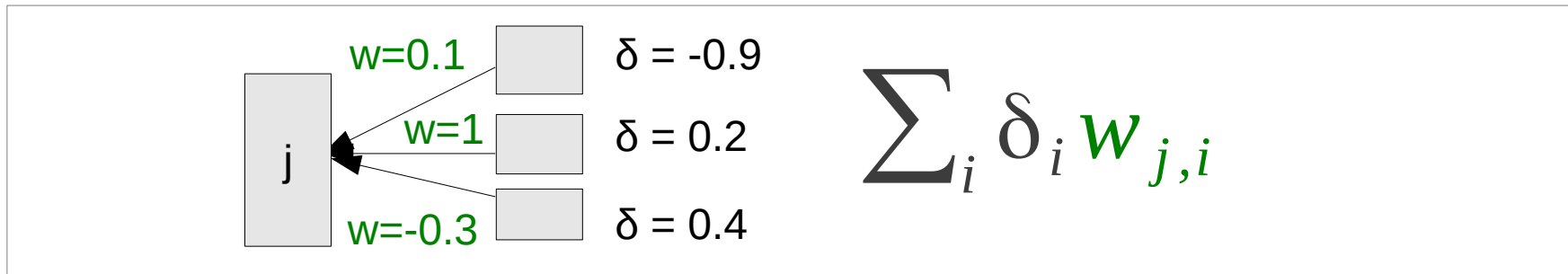
問題：正解は分からない！

- NN では出力層のみで正解が与えられる

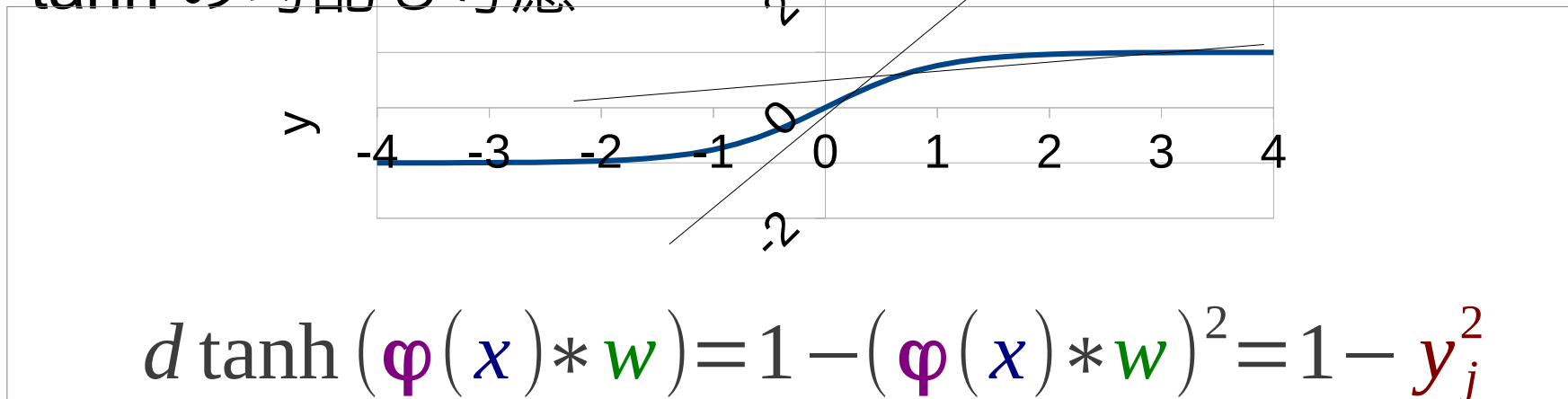


解決策：逆伝搬法

- 出力層からエラーを後ろへ伝搬



- \tanh の勾配も考慮



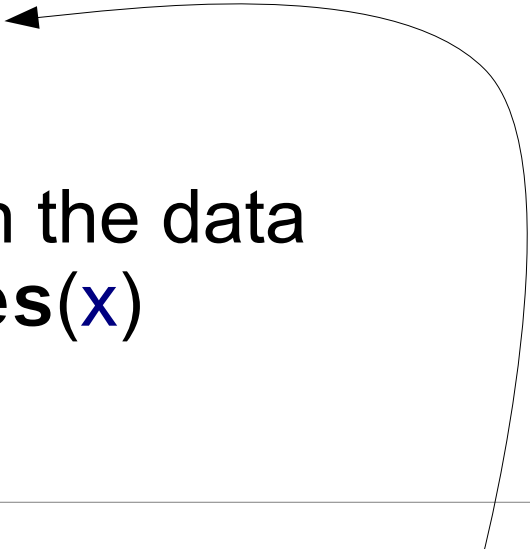
- 合わせて：
$$\delta_j = (1 - y_j^2) \sum_i \delta_i w_{j,i}$$

逆伝搬のコード

```
update_nn(network, phi, y')  
  calculate y using predict_nn  
  create array  $\delta = [0, 0, \dots, \text{np.array}([y' - y_j])]$   
  for i in len( $\delta$ ) - 1 .. 0  
    network[]
```

学習コード

```
create network  
randomize network weights  
for / iterations  
  for each labeled pair x, y in the data  
    phi = create_features(x)  
    update_nn(w, phi, y)
```



- 単純なパーセプトロンで、重みを 0 へ初期化
- NN ではランダム初期化
(全てのパーセプトロンが同一の値にならないよう)

演習課題

演習課題 (1)

- 実装
 - train-nn: NN を学習するプログラム
 - test-nn: NN を用いて予測するプログラム
- テスト
 - 入力 : test/03-train-input.txt
 - 学習 1 回、隠れ層 1 つ, 隠れ層のノード 2 つ
 - 更新を手で確認

演習課題 (2)

- 学習 `data/titles-en-train.labeled`
- 予測 `data/titles-en-test.word`
- 評価
 - `script/grade-prediction.py data-en/titles-en-test.labeled your_answer`
- 比較
 - 単純なパーセプトロン、SVM
 - 様々な隠れ層の数、ノード数

Thank You!