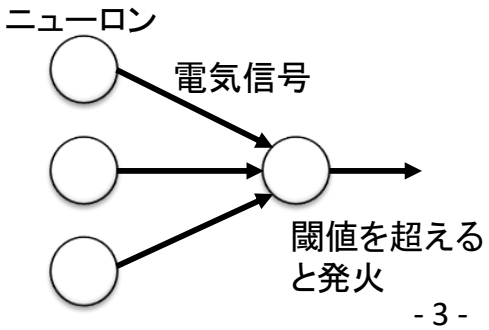
第7、8回
ニューラルネットワークの基礎
ニューラルネットワークを用いた分類

目次

ニューラルネットワークとは

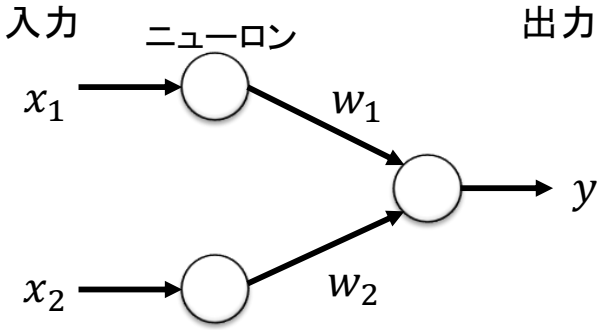
- 人工知能分野におけるアルゴリズムのひとつ
- 人間の脳の構造を模しているという特徴を持つ

- ニューロン
  - 人間の脳を構成する神経細胞
  - 人間の脳では、約140億個のニューロンが巨大な網目のようなネットワークを形成している
    →ニューラルネットワーク
  - ニューロンの間では電気信号による情報伝達が行われる
  - ニューロンが他のニューロンから入力を受け取り、ある閾値を超えると、次のニューロンに電気信号を送る(発火)
  - 各ニューロン間の結合の強さには違いがあり、電気信号の受け取り具合の違いによって人間は異なるパターンを認識している



単純なモデル化(1)

- あるニューロンが2つのニューロンから電気信号を受け取った場合



- 2つのニューロンから伝わる電気信号の総量は

$$w_1x_1 + w_2x_2$$

- w_1, w_2をネットワークの重みと呼ぶ
- ニューロンが発火するかどうかは、受け取った電気信号の総量がある閾値を超えるかどうかによって決まりました。

単純なモデル化(2)

- 発火するかを決める閾値の値をθとすると出力で得られる結果は次の式で表すことができる

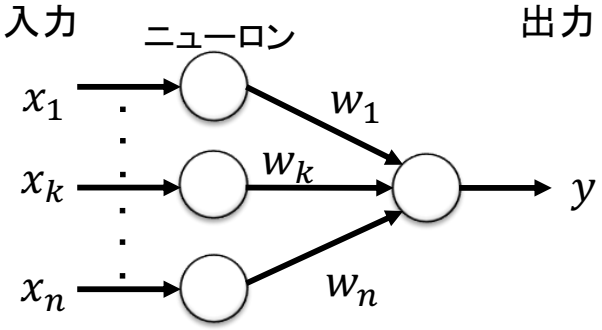
$$y = \begin{cases} 1 & (w_1x_1 + w_2x_2 \geq \theta) \\ 0 & (w_1x_1 + w_2x_2 < \theta) \end{cases}$$

- 発火については、1が発火、0が発火しなかったとする

- ネットワークの重みと、閾値を適切に設定すれば、入力に対応する出力の値が、実際の脳内で伝播する電気信号量と同様になるはずです。
- ニューラルネットワークの形がどれほど複雑になっても、このアプローチを応用すれば実現できる

単純パーセプトロン(1)

- 単純なモデルでは入力は2つでしたが下の図のように、入力をn個に増やした場合について考えていきます。
- パーセプトロンは、複数の信号を入力として受け取り、1つの信号を出力します。



- 入力の数が増えても「受け取った電気の量が閾値を超えると発火する」というニューロンの特徴は変わらないので式は以下のようになる。

$$y = \begin{cases} 1 & (w_1x_1 + w_2x_2 + \cdots + w_nx_n \geq \theta) \\ 0 & (w_1x_1 + w_2x_2 + \cdots + w_nx_n < \theta) \end{cases}$$

単純パーセプトロン(2)

$$y = \begin{cases} 1 & (w_1x_1 + w_2x_2 + \cdots + w_nx_n \geq \theta) \\ 0 & (w_1x_1 + w_2x_2 + \cdots + w_nx_n < \theta) \end{cases}$$

- ここで、下の式で表される関数$f(x)$を考える

$$f(x) = \begin{cases} 1 & (x \geq 0) \\ 0 & (x < 0) \end{cases}$$

- このときネットワークの出力は以下のように書き直すことができる

$$y = f(w_1x_1 + w_2x_2 + \cdots + w_nx_n - \theta)$$

- この$f(x)$を**ステップ関数**と呼びます。
- 式を統一し、扱いやすくするために、$\mathrm{b} = -\theta$とおき、重みw_kと入力$x_k(k = 1,2,\ldots,n)$をベクトルの内積の形に置き換えます。

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \qquad w = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix}$$

単純パーセプトロン(3)

- 最終的に出力の式のは以下のようになる

$$y = f(w^T x + b)$$

- これでニューラルネットワークの出力を一般形の式で書くことができました。

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \qquad w = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix}$$

- 式変形した際に利用した上のベクトルを、それぞれwを**重みベクトル**、bを**バイアス**と呼びます。
- 入力に対する出力が誤っていたら、w、bを修正することで徐々に正しい状態に近づけていく方法を誤り訂正学習法と呼びます。
- このように正しい出力が得られるまでパラメータを更新していく過程を、ネットワークを学習させるといいます。

単純パーセプトロン(4)

- 重みを訂正していく際の式を定式化する必要がある。
- 正解の出力をt、モデルで得られた出力をyとする。
- 訂正する際の動作は以下のように定義される
  - $t = y$のときは出力が正しいので修正を行わない
  - $t = 0, y = 1$ のときは出力が大きすぎるので、入力が正ならば重みを小さく、出力が負ならば重みを大きくする修正を行う。また、バイアスを大きくする。
  - $t = 1, y = 0$ のときは出力が小さすぎるので、入力が正ならば重みを大きく、入力が負ならば重みを小さくする修正を行う。また、バイアスを小さくする。

- 訂正する重みとバイアスを$\Delta w, \Delta v$ とし、k回目の試行における重みとバイアスをそれぞれw^k, b^kとすると次のように定式化できる

$$\Delta w = (t - y)x$$
$$\Delta b = t - y$$

$$w^{(k+1)} = w^{(k)} + \Delta w$$
$$b^{(k+1)} = b^{(k)} + \Delta b$$

単純パーセプトロンの実装(1)

- 2種類の正規分布に従うデータ分類について考えます。
- 分類結果を可視化できるように、今回は入力のニューロン数を2とします。
- ニューロンが発火しないデータは平均値が0、発火するデータは平均値が5とし、それぞれのデータが10個ずつあるとします。

```
 8 import numpy as np
 9
10 rng = np.random.RandomState(123)
11
12 d = 2
13 N = 10
14 mean = 5
15
16 x1 = rng.randn(N, d) + np.array([0, 0])
17 x2 = rng.randn(N, d) + np.array([mean, mean])
18
19 x = np.concatenate((x1, x2), axis=0)
```

毎回「同じランダム状態」を作るために乱数を定める

ニューロンの数

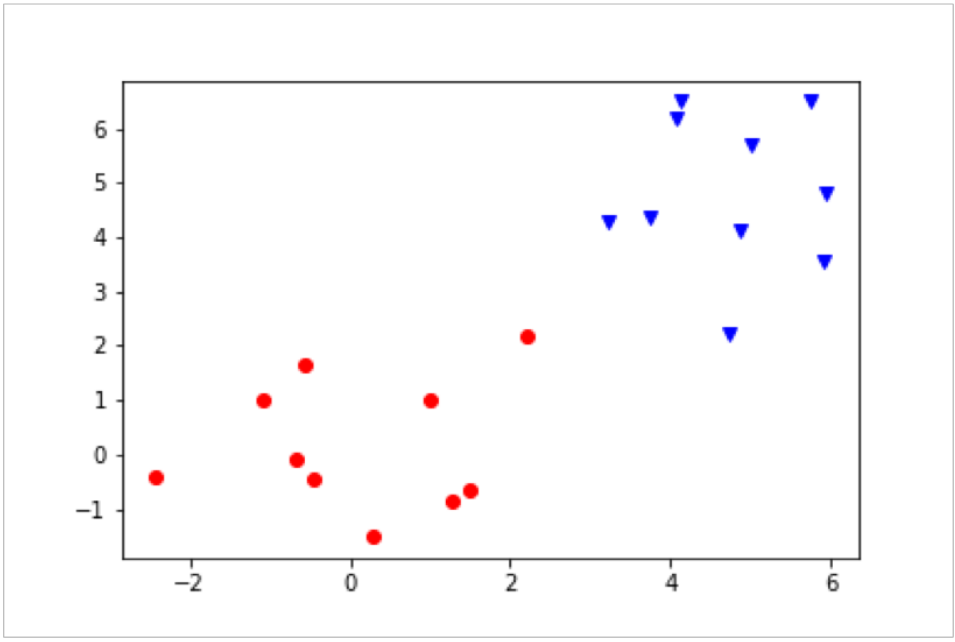
各パターンのデータ数

ニューロンが発火するデータの平均値

まとめてデータを処理するためにひとまとまりにする

単純パーセプトロンの実装(2)

- 生成したデータをmatplotlibを用いて可視化すると、下のような分布のデータが得られます。
- これが今回分類していくデータになります。



単純パーセプトロンの実装(3)

- 単純パーセプトロンにおけるパラメータは、重みwとバイアスbなので、それぞれを初期化します。

```
25 w = np.zeros(d)
26 b = 0
```

- 出力は先ほど定義したステップ関数になるので、以下のように定義できます。

$y = f(w^T x + b)$を表す関数

```
34 def y(x):
35     return step(np.dot(w, x) + b)
36
37
38 def step(x):
39     return 1 * (x > 0)
```

xが0より大きかったら1を返す関数

単純パーセプトロンの実装(4)

- パラメータは、重みwとバイアスbを更新するには正しい出力が必要となるので、これを以下のように定めます。

```
37 def t(i):
38     if i < N:
39         return 0
40     else:
41         return 1
```

- 先ほど定義した式でパラメータの更新を行うため、パラメータの更新処理は以下のようになります。

$$\Delta w = (t - y)x$$
$$\Delta b = t - y$$

$$w^{(k+1)} = w^{(k)} + \Delta w$$
$$b^{(k+1)} = b^{(k)} + \Delta b$$

```
44 while True:
45     classified = True
46     for i in range(N * 2):
47         delta_w = (t(i) - y(x[i])) * x[i]
48         delta_b = (t(i) - y(x[i]))
49         w += delta_w
50         b += delta_b
51         classified *= all(delta_w == 0) * (delta_b == 0)
52     if classified:
53         break
```

データが正しく分類されているかどうかを判別

単純パーセプトロンの実装(5)

- 学習した重みとバイアスはそれぞれ

```
58 print('w:', w)
59 print('b:', b)
```

```
w: [2.14037745 1.2763927 ]
b: -9
```

- 得られた重みとバイアスがわかったので、今回学習したデータの各クラスの境界線は次のような式で表すことができる

$$2.14037745x_1 + 1.2763927x_2 - 9 = 0$$

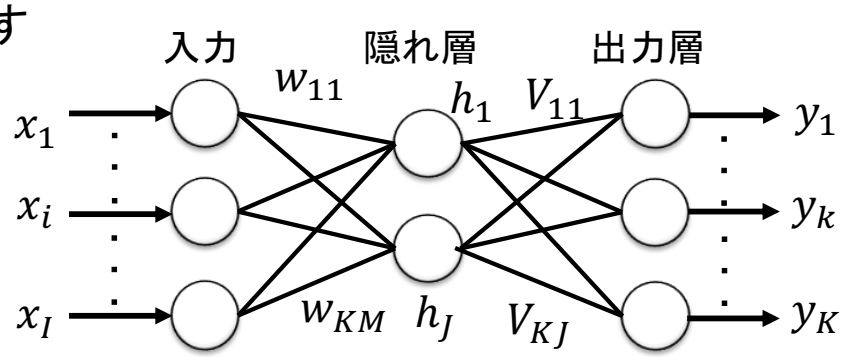
- ニューロンが正しく発火するかどうかを確かめるには、任意の値を与えたときに、出力がどのようになるかを見ればよい

```
61 print('\nTest:')
62 print(y([3, 0]))
63 print(y([6, 5]))
```

```
Test:
0
1
```

まとめ

- より複雑な分類を行うためには、より多数のニューロンを階層的に接続します



- 学習データxを入力し、予測結果yを出力します。yと教師データとの誤差を計算し、これを元に各層の重みを入力層に向かって段階的に修正していきます。これをバックプロパゲーション（誤差逆伝播法）といいます。
- 深層学習では、数十から数百もの隠れ層を使うことがあります。

- より発展的な内容については、付録をご活用ください。