情報とデータサイエンス

ニューラルネットワークによる分類

より複雑な画像をコンピュータに認識させよう

機械学習での分類の特徴

データからルールやパターンを見つけ出して、
モデルを作りカテゴリーを推測する。

k近傍法 ▶ 近くにあるデータと同じカテゴリーと推測

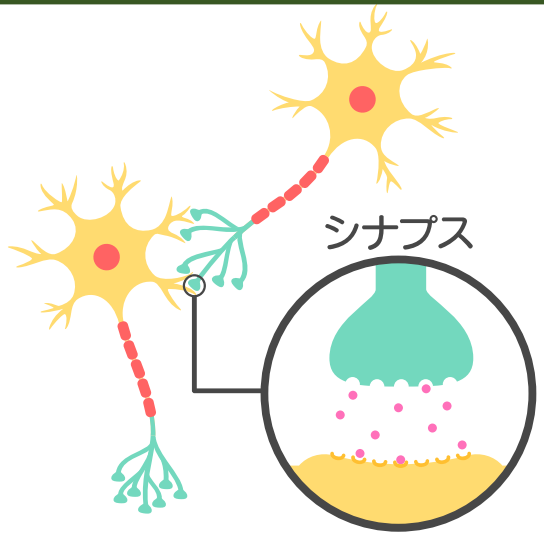
SVM ▶ カテゴリーを分離する直線（平面）を見つけて、
どの領域に含まれるかにより推測

決定木 ▶ 分類するルールをデータから見つけて、
そのルールをもとに推測

ニューラルネットワークとは

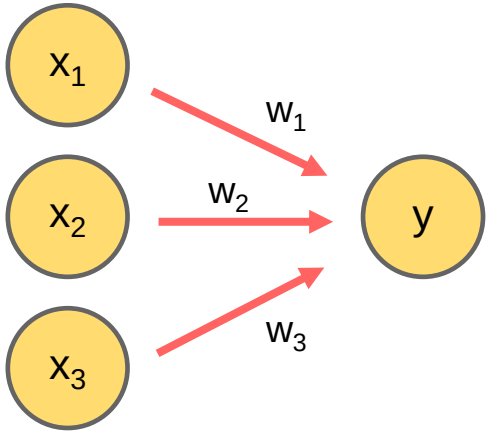
人間の脳内の神経細胞どうしのつながりを模した数理モデル

人間の脳内の神経細胞どうしのつながり



電気信号はシナプスで重みづけされて
一方向に伝播する

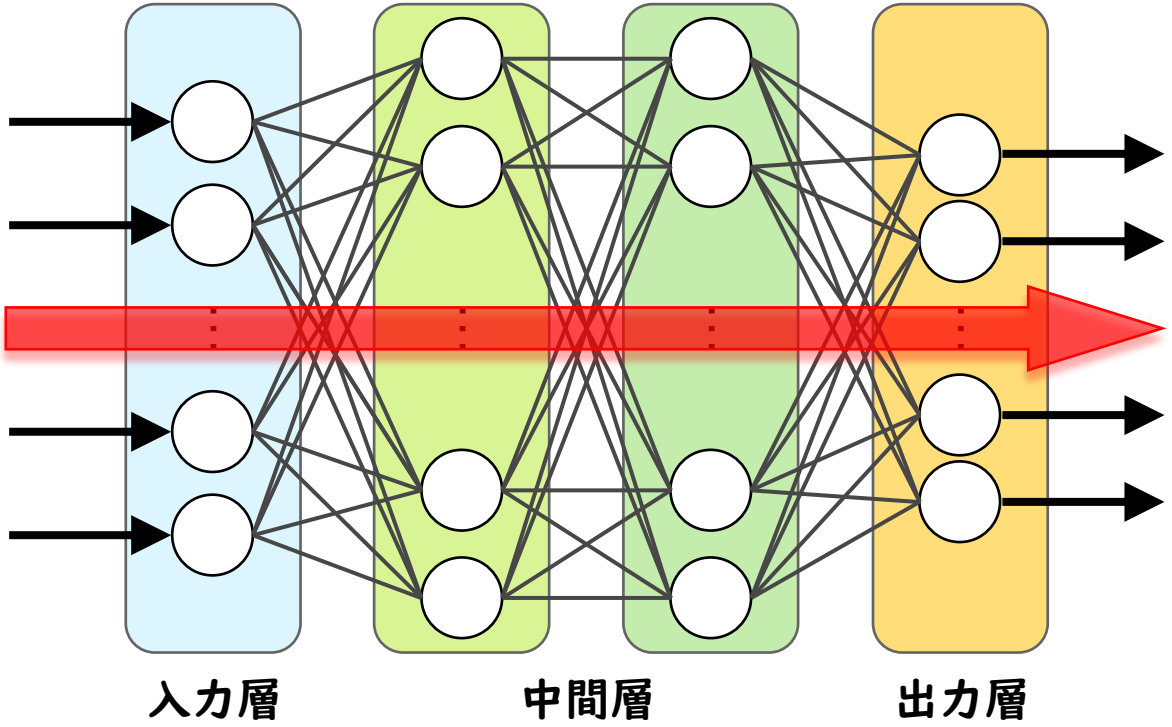
ニューラルネットワーク



信号が伝播されるときに、
乗算によりwiの重みが付けられる

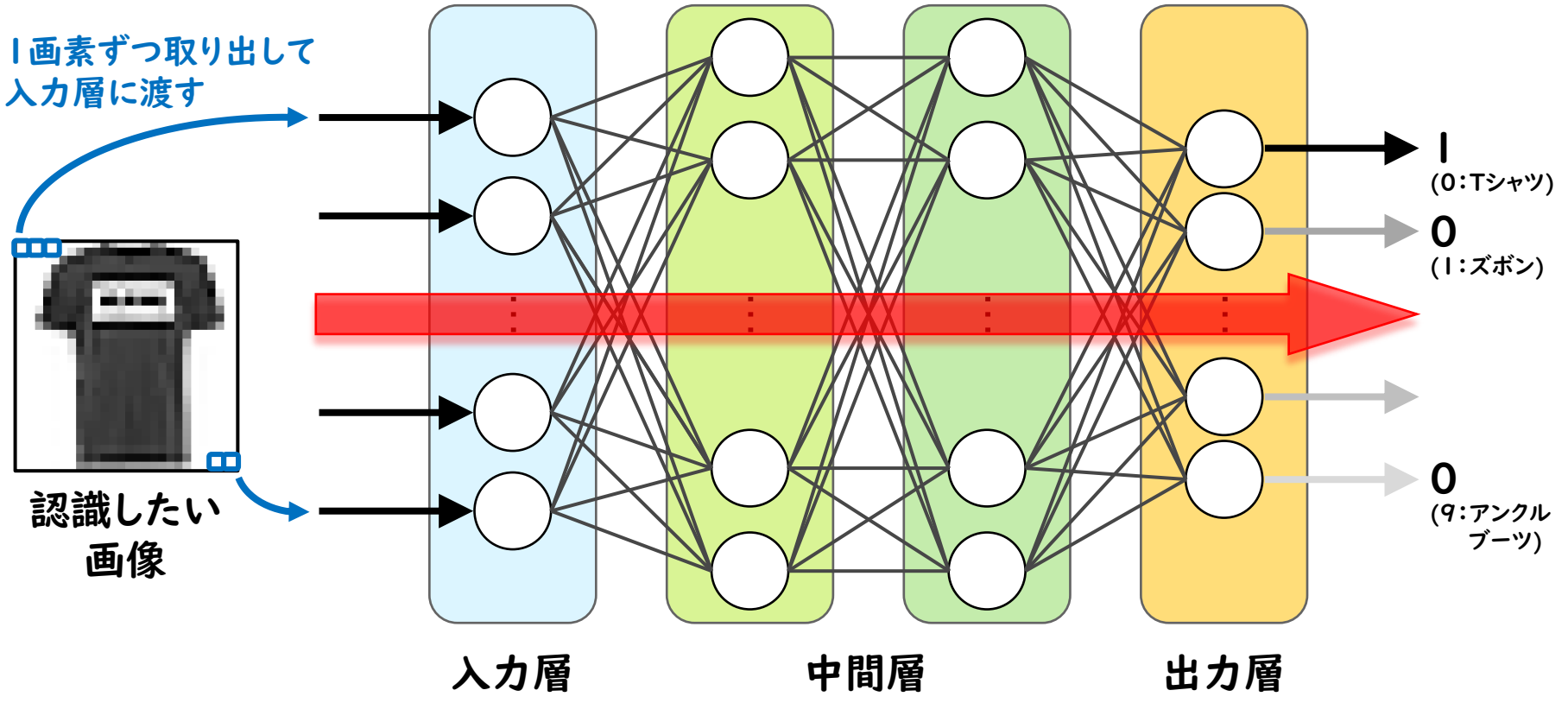
$y = f(w1 \times x1 + w2 \times x2 + w3 \times x3)$

ニューラルネットワークの構成

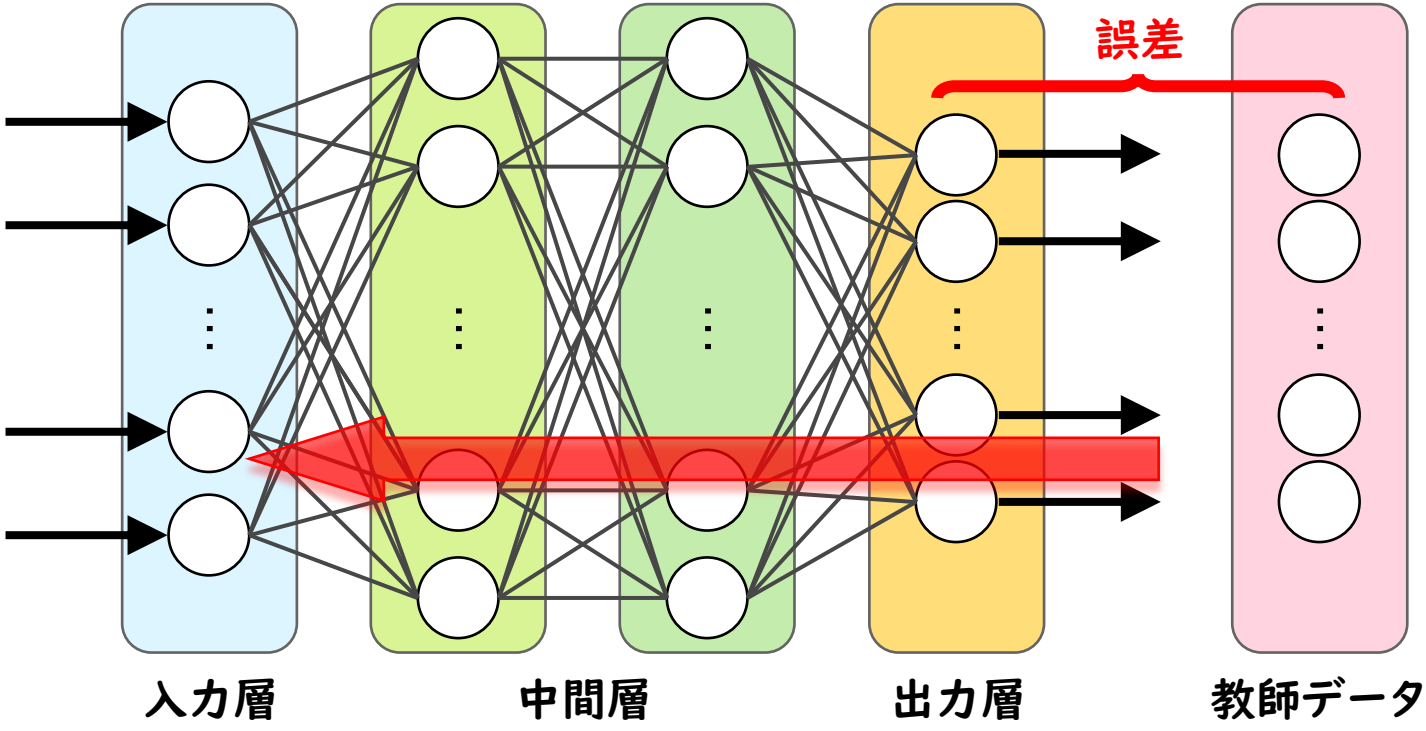


順伝播（フォワードプロパゲーション）で計算して分類する

画像が認識される仕組み



ニューラルネットワークでの学習の仕組み



誤差逆伝播（バックプロパゲーション）により重みを調整

ファッションアイテムを分類してみよう

【ニューラルネットワークによる分類の流れ】

1. 学習に必要なデータを読み込む

2. 学習モデルを作成し、学習を行う

3. テストデータを使って、モデルを検証する

4. モデルを使って、実際のデータを分類する
   （ファッションアイテムを写真に撮って分類する）

プログラムの解説

```
from keras.datasets import fashion_mnist
(X_train, y_train), (X_test, y_test) =
fashion_mnist.load_data()
```

fashion_mnistという
ファッションアイテムの画像データを
読み込む

【訓練データ：60000枚】
【テストデータ：10000枚】

プログラムの解説

```
X_train  = X_train[:3000]
y_train  = y_train[:3000]
X_test = X_test[:300]
y_test = y_test[:300]
```

【訓練データ：60000枚 → 3000枚】

【テストデータ：10000枚 → 300枚】

```
X_train = X_train/255.0
X_test = X_test/255.0
```

色の濃淡：0～255を256階調 → 0～1の値に変換

プログラムの解説

```
import matplotlib.pyplot as plt
```

matplotlibのpyplotというグラフを描画するモジュールを読み込む

```
plt.imshow(X_train[1], cmap='Greys')
plt.show()
```

X_train[1]の画素データをグレーの濃淡で表示する

プログラムの解説

```
item_names = ['Tシャツ', 'ズボン', 'プルオーバー', 'ドレス', 'コート',
              'サンダル', 'スカート', 'スニーカー', 'バッグ', 'アンクルブーツ']
print('正解ラベル', item_names[y_train[1]])
```

y_train[1]の正解ラベルを表示する

0：Tシャツ	5：サンダル
1：ズボン	6：シャツ
2：プルオーバー	7：スニーカー
3：ドレス	8：バッグ
4：コート	9：アンクルブーツ

プログラムの解説

```
import keras
```

kerasというニューラルネットワークのライブラリを読み込む

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),  ←入力層
    keras.layers.Dense(10000, activation='relu'),      ←中間層
    keras.layers.Dense(10, activation='softmax') ←出力層
])
```

ニューラルネットワークの入力層・中間層・出力層の構成を定義する
入力層：28×28の形状のデータを1次元配列に変換して受け渡す
中間層：ニューロン数が10000で、全結合の層とする
出力層：ニューロン数が正解のラベル数10で、全結合の層とする

プログラムの解説

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

モデルに学習プロセスを設定する

optimizer 最適化をするアルゴリズム

loss　　損失関数（正解値と予測値の誤差を求める）

metrics　評価関数

```
model.fit(X_train, y_train, epochs=10)
```

学習回数（エポック数）を10回として、訓練データを用いて学習を行う

プログラムの解説

```
score = model.evaluate(X_test, y_test)
print('loss=', score[0])
print('accuracy=', score[1])
```

モデルを評価し、テストデータを用いて損失率・正解率を求める

プログラムの解説

```
import cv2
```
画像処理のライブラリをインポートする

```
im = cv2.imread('pic.jpg')
```
「pic.jpg」というファイルを読み込む

```
im = cv2.resize(im,(28,28))
```
画素数が28×28になるように調整（縮小）する

```
im = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
```
BGRの3色をグレーに変換する

```
im = cv2.bitwise_not(im)
```
色を反転する

```
plt.imshow(im, cmap='Greys')
```
画像をグレーの濃淡で表示する

プログラムの解説

import numpy as np

数値演算のライブラリをインポートする

images = []

認識させる画像の配列（リスト）を初期化する

data_expand = np.expand_dims(im, axis=0)

ニューラルネットワークで分類できるデータ形式に変換する

images.append(data_expand)

配列imagesに変換したデータを追加する

predictions = model.predict(images)

配列imagesに格納された画像のデータをニューラルネットワークを使って分類する

プログラムの解説

```
for i in range(np.size(predictions[0])):
```

i ＝ 0, 1, …, 9 として出力層の個数（10個）だけ繰り返す

```
print(item_names[i], ' : ', predictions[0][i]*100, '%')
```

分類される確率を、アイテムごとに出力する

```
bunrui = np.argmax(predictions[0])
```

アイテム0～アイテム9の中で最も確率が高いものの番号を求める

```
print('分類結果:', bunrui, item_names[bunrui])
```

アイテムの番号とそのアイテム名を出力する

さらに学習を深めるには・・・

ニューラルネットワークについての理解を深める

- 中間層のニューロンの数や層の数を変えてみて
  よりよいモデルを作成する。
- epoch数を変えて、正解率などの評価を確かめてみる。

実際の画像認識に応用してみる

- 食べ物や人の顔の写真を多く集めて、
  認識できるか試してみる。

他のデータを学習させて分類してみよう

他にもこのような学習データがあります

【物体の画像写真】
（CIFAR-10・CIFAR-100）