授業準備：Webclassからコードをダウンロードし、Google colaboratoryで開いておいてください

演習授業中の質問対応について

医療とAI･ビッグデータ入門
演習16

深層学習

*本日演習16の授業後に複合領域コースの説明があります

深層学習(乳がんデータの分類)コードまとめ

STEP0： ライブラリの読み込み

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

STEP1：データの準備

```
from sklearn.datasets import load_brest_cancer
bc = load_breast_cancer(as_frame = False)

from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(bc.data, bc.target, test_size = 0.3,
random_state = 0)
x_train3 = x_train[:, 0:3]
x_test3 = x_test[:, 0:3]
```

STEP2：学習モデルの選択

```
from keras.models import Sequential
from keras.layers import Dense
model_3 = Sequential()
model_3.add(Dense(2, input_shape=(3,), activation = 'relu'))
model_3.add(Dense(1, activation = 'sigmoid'))
model_3.compile(loss = 'binary_crossentropy', optimizer = 'Adam', metrics = ['accuracy'])
model_3.summary()
```

STEP3：データを入れて学習

```
result = model_3.fit(x_train3, y_train, batch_size = 32, epochs = 300 )
```

STEP4：図示

```
plt.plot(result.history['loss'])
plt.title('loss')

plt.plot(result.history['accuracy'])
plt.title('accuracy')
```

STEP5：モデルの評価

```
evaluate_loss, evaluate_accuracy = model_3.evaluate(x_test3, y_test)
print(evaluate_loss)
print(evaluate_accuracy)
```

深層学習(乳がんデータの分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP2：学習モデルの選択

コード15-6 学習モデルを設計

```
from keras.models import Sequential
from keras.layers import Dense
```

kerasは深層学習に特化した便利なライブラリ ← sklearnに代わってこちらを使っていく

クラスをインポート　`from keras.models import Sequential`

- keras：ライブラリ名
- models：モジュール名
- Sequential：クラス名

- 深層学習ではモデルを作っていく
- 中間層のニューロンが2つ、出力層のニューロンが1つのニューラルネットワークを作る

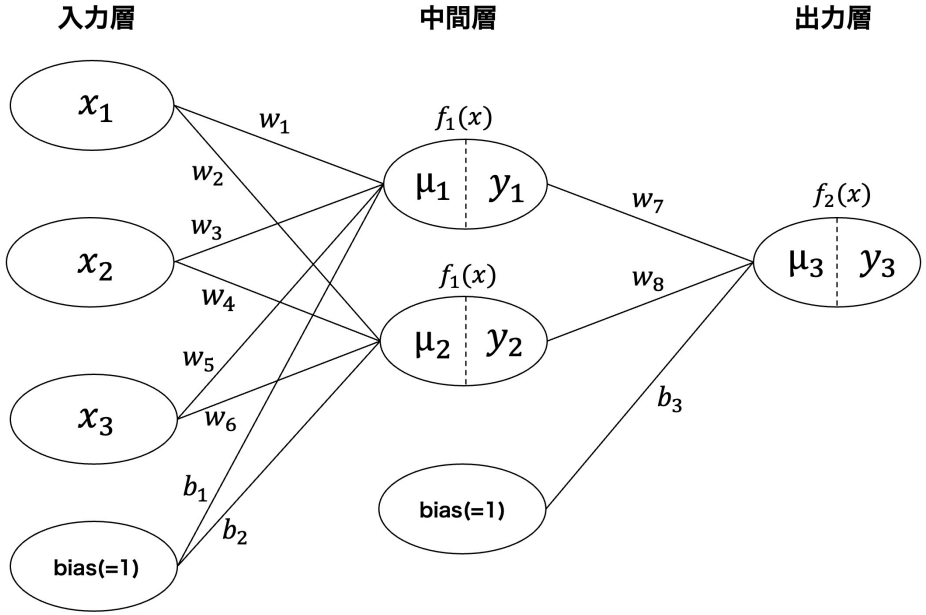
深層学習(乳がんデータの分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP2：学習モデルの選択

コード15-6 学習モデルを設計

中間層のニューロンが2つ、出力層のニューロンが1つのニューラルネットワークを作る



深層学習(乳がんデータの分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP2：学習モデルの選択

コード15-6 学習モデルを設計

```
model_3 = Sequential()
model_3.add(Dense(2, input_shape=(3,), activation = 'relu'))
model_3.add(Dense(1, activation = 'sigmoid'))
model_3.compile(loss = 'binary_crossentropy',
                optimizer = 'Adam',
                metrics = ['accuracy'])
model_3.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 2)                 8

 dense_1 (Dense)             (None, 1)                 3

=================================================================
Total params: 11 (44.00 Byte)
Trainable params: 11 (44.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

この7行(５行)でモデルの設計

深層学習(乳がんデータの分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP2：学習モデルの選択

```
model_3 = Sequential()
model_3.add(Dense(2, input_shape=(3,), activation = 'relu'))
model_3.add(Dense(1, activation = 'sigmoid'))
model_3.compile(loss = 'binary_crossentropy',
                optimizer = 'Adam',
                metrics = ['accuracy'])
model_3.summary()
```

- **最初に Sequentialクラスでmodel_3インスタンスを作成する**

  (*LinearRegressionやRandomForestClassifierなどのモデルと同じ)

この後ニューラルネットワークを入力層から順番に設計できるようになる

深層学習(乳がんデータの分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP2：学習モデルの選択

```
model_3 = Sequential()
model_3.add(Dense(2, input_shape=(3,), activation = 'relu'))
model_3.add(Dense(1, activation = 'sigmoid'))
model_3.compile(loss = 'binary_crossentropy',
                optimizer = 'Adam',
                metrics = ['accuracy'])
model_3.summary()
```

- 次に(モデル名).add()で中間層の設定を行う

  Dense(次の層のニューロンの数, input_shape=(入力するニューロンの数,),
  activation=活性化関数)

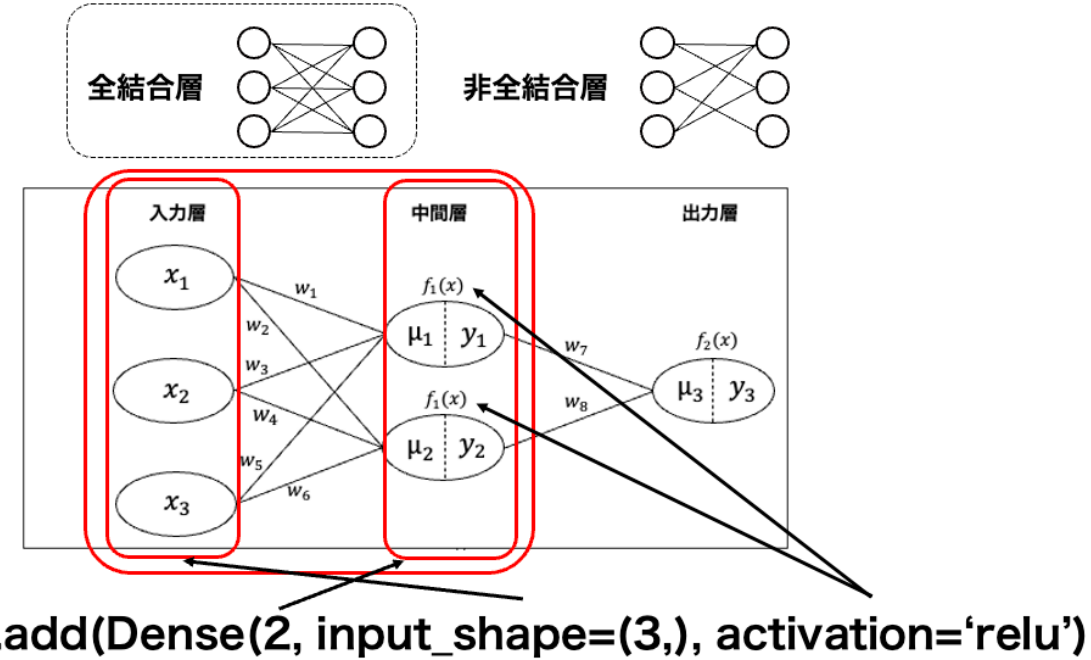
  *Denseは「全結合」（前のニューロンと後ろのニューロンを全て接続する）

深層学習(乳がんデータの分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP2：学習モデルの選択

```
model_3 = Sequential()
model_3.add(Dense(2, input_shape=(3,), activation = 'relu'))
```



深層学習(乳がんデータの分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP2：学習モデルの選択

```
model_3 = Sequential()
model_3.add(Dense(2, input_shape=(3,), activation = 'relu'))
model_3.add(Dense(1, activation = 'sigmoid'))
model_3.compile(loss = 'binary_crossentropy',
                optimizer = 'Adam',
                metrics = ['accuracy'])
model_3.summary()
```

- 次に（モデル名）.add()で出力層の設定を行う

  Dense(次の層のニューロンの数, input_shape=(入力するニューロンの数,),
  activation=活性化関数)

  *Denseは「全結合」（前のニューロンと後ろのニューロンを全て接続する）

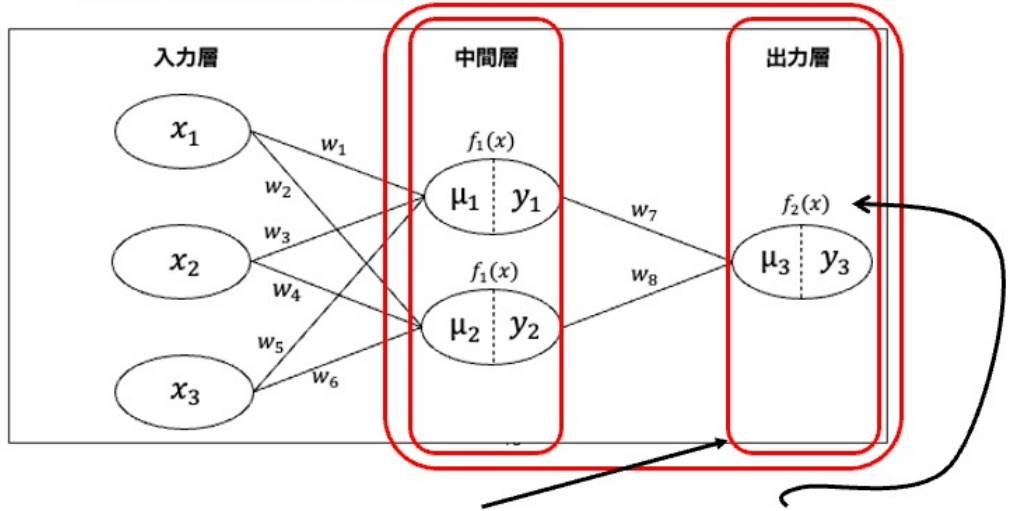
*前の層が指定されている場合は、自動で認識されるので入力が必要ない

深層学習(乳がんデータの分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP2：学習モデルの選択

```
model_3 = Sequential()
model_3.add(Dense(2, input_shape=(3,), activation = 'relu'))
model_3.add(Dense(1, activation = 'sigmoid'))
```



model.add(Dense(1, activation='sigmoid')

深層学習(乳がんデータの分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP2：学習モデルの選択

```
model_3 = Sequential()
model_3.add(Dense(2, input_shape=(3,), activation = 'relu'))
model_3.add(Dense(1, activation = 'sigmoid'))
model_3.compile(loss = 'binary_crossentropy',
                optimizer = 'Adam',
                metrics = ['accuracy'])
model_3.summary()
```

- この3行でニューラルネットワークの設定が完了

  *バイアスは`Sequential()`では自動で作成される

深層学習(乳がんデータの分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP2：学習モデルの選択

```
model_3 = Sequential()
model_3.add(Dense(2, input_shape=(3,), activation = 'relu'))
model_3.add(Dense(1, activation = 'sigmoid'))
model_3.compile(loss = 'binary_crossentropy',
                optimizer = 'Adam',
                metrics = ['accuracy'])
model_3.summary()
```

- **学習の仕方を指定**

  引数loss = では損失関数を「２値交差エントロピー」に指定 (2値分類はこれ)

  引数optimizer = では重みとバイアスを更新するアルゴリズムを'Adam'に指定

  引数metrics=では、学習過程で表示されるものをaccuracyに指定（後から説明）

深層学習(乳がんデータの分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP2：学習モデルの選択

```
model_3 = Sequential()
model_3.add(Dense(2, input_shape=(3,), activation = 'relu'))
model_3.add(Dense(1, activation = 'sigmoid'))
model_3.compile(loss = 'binary_crossentropy',
                optimizer = 'Adam',
                metrics = ['accuracy'])
model_3.summary()
```

- 構築したモデルのまとめが出力される

深層学習(乳がんデータの分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP2：学習モデルの選択

```
model_3 = Sequential()
model_3.add(Dense(2, input_shape=(3,), activation = 'relu'))
model_3.add(Dense(1, activation = 'sigmoid'))
model_3.compile(loss = 'binary_crossentropy',
                optimizer = 'Adam'
                metric
model_3.summary()
```

- 構築したモデルのまとめが

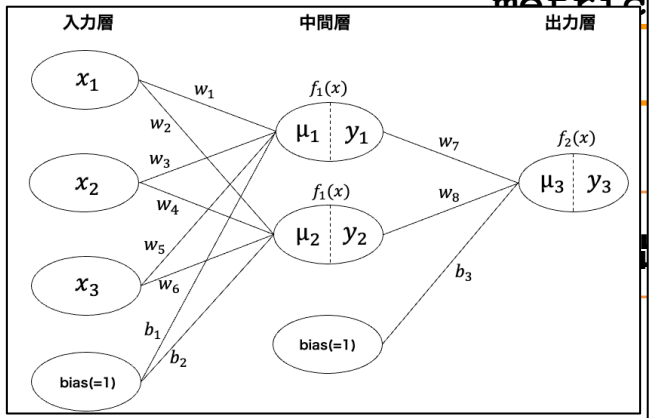
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 2)                 8

 dense_1 (Dense)             (None, 1)                 3

=================================================================
Total params: 11 (44.00 Byte)
Trainable params: 11 (44.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

深層学習(乳がんデータの分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP2：学習モデルの選択

```
model_3 = Sequential()
model_3.add(Dense(2, input_shape=(3,), activation = 'relu'))
model_3.add(Dense(1, activation = 'sigmoid'))
model_3.compile(loss = 'binary_crossentropy',
                optimizer = 'Adam'
                metric
```



↑8個　　↑3個

中間層の設定：重み6個、バイアス2個の計8パラメータが存在

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 2)	8
dense_1 (Dense)	(None, 1)	3

出力層の設定：重み2個、バイアス1個の計3パラメータが存在

```
Trainable params: 11 (44.00 Byte)
Non-trainable params: 0 (0.00 Byte)
```

深層学習(乳がんデータの分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP3：データを入れて学習

コード15-7 学習用データで学習させる

```
result = model_3.fit(x_train3, y_train,
                     batch_size = 32,
                     epochs = 300 )
```

- **これまで通り、(モデル名).fit(x, y)で学習させる**
- **resultに学習結果を入れる**
- **引数batch_size=32で「32組ずつデータを取り出して損失を計算し、重みとバイアスを更新しなさい」という指示**

  *学習用のデータは398組あり、32組ずつデータを取り出すと13回ですべて取り出せる
  全てのデータをひと通り使い尽くすことを1エポックという
- **引数epochs=でエポック数を指定する**

colab

深層学習でよく出てくる言葉たち

エポック (epoch)

訓練データを何回学習したか
(問題集を何周学習したか)

バッチサイズ (batch size)

1 回にどれくらいのデータを学習するか
(問題集を 1 日何問解くか)

例えば訓練データ (問題集) が 6000 問あって、1 日に 600 問解く (バッチサイズ =600) ならば、10 日で問題集が 1 周終わる。
これが 1 エポック学習した状態である。実際には 1 周して完璧になるはずがないのと同じく、AI もたくさんのエポック訓練する。

なお、問題集を理解せずに丸暗記してしまい、初見の問題に手も足も出ない状態 (過学習) になってはいけないというのは
機械学習概論 1 で勉強した通りだ。

オプティマイザー (optimizer)

基本的にこの後出てくる勾配降下法で学習するが、いろいろな細かい改良があり、
学習法のことを optimizer という。よく使われるのは Adam だが、たくさんある。

深層学習(乳がんデータの分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP3：データを入れて学習

```
result = model_3.fit(x_train3, y_train,
                     batch_size = 32,
                     epochs = 300 )
```

*数値は人によって異なる
誤差(epochが進むと誤差が小さくなる)

```
Epoch 1/300
13/13 [==============================] - 1s 2ms/step - loss: 27.1328 - accuracy: 0.3693
Epoch 2/300
13/13 [==============================] - 0s 2ms/step - loss: 25.3038 - accuracy: 0.3693
Epoch 3/300
13/13 [==============================] - 0s 2ms/step - loss: 23.5756 - accuracy: 0.3693
Epoch 4/300
13/13 [==============================] - 0s 2ms/step - loss: 21.9148 - accuracy: 0.3693
                                                    .
                                                    .
                                                    .
Epoch 297/300
13/13 [==============================] - 0s 2ms/step - loss: 0.4363 - accuracy: 0.8342
Epoch 298/300
13/13 [==============================] - 0s 2ms/step - loss: 0.4359 - accuracy: 0.8241
Epoch 299/300
13/13 [==============================] - 0s 2ms/step - loss: 0.4339 - accuracy: 0.8367
Epoch 300/300
13/13 [==============================] - 0s 2ms/step - loss: 0.4328 - accuracy: 0.8342
```

13/13は試行回数
(学習用のデータは398組あり、32組ずつデータを取り出すと13回ですべて取り出せる)

metrics=で指定したので正解率も表示
(epochが進むと正解率は概ね向上)

Epochの回数(1~300回)分表示

colab

深層学習(乳がんデータの分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP4：学習結果の図示

コード15-8　学習結果の表示

```
result.history
```

```
{'loss':  [27.132789611816406,  25.30379867553711,  23.575641632080078,
21.914821624755586, 20.267982482910156, …],
'accuracy': [0.3693467378616333, 0.3693467378616333, 0.3693467378616333,
0.3693467378616333, 0.3693467378616333, …]}
```

辞書型で出力　{key：value, key, value, ...}

{'loss': [1回目の誤差, 2回目の誤差, … , 300回目の誤差],
'accuracy': [1回目の正解率,　2回目の正解率,　… ,　300回目の正解率]}

colab

深層学習(乳がんデータの分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP4：学習結果の図示

コード15-8　学習結果の表示

```
result.history
```

```
{'loss': [27.132789611816406, 25.30379867553711, 23.575641632080078,
21.91482162475586, 20.267982482910156, …],
'accuracy': [0.3693467378616333, 0.3693467378616333, 0.3693467378616333,
0.3693467378616333, 0.3693467378616333, …]}
```

```
result.history['loss']
```

```
[27.132789611816406, 25.30379867553711, 23.575641632080078,
21.91482162475586, 20.267982482910156, …]
```

(変数名)[key]でvalueを取り出せる

colab

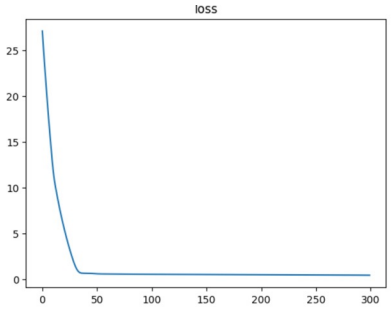
深層学習(乳がんデータの分類)

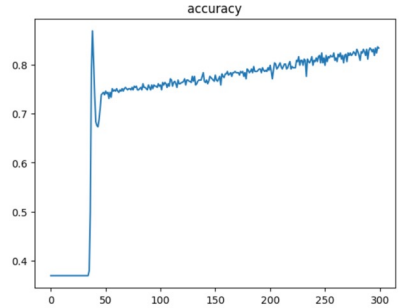
STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP4：学習結果の図示

コード15-9　学習結果の図示

```
plt.plot(result.history['loss'])
plt.title('loss')
plt.show()

plt.plot(result.history['accuracy'])
plt.title('accuracy')
plt.show()
```





- `plt.plot(x,y)`で各点をつなぐ線を描ける
- `y`は結果の`loss/accuracy`を選択
- `x`は指定していないとデータ数(300回)
- `plt.title()`でタイトルをつける
- `plt.show()`で図を表示する

colab

深層学習(乳がんデータの分類)

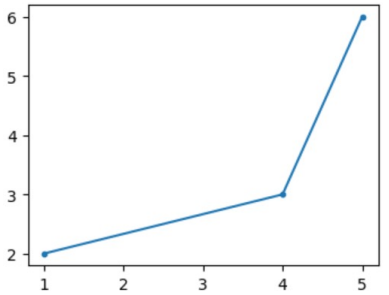
STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP4：学習結果の図示

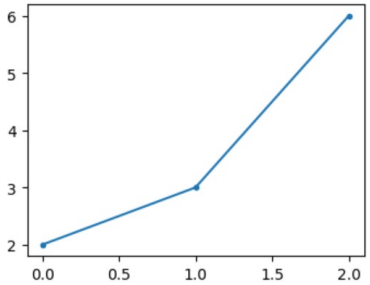
コード15-9　学習結果の図示

```
plt.plot(result.history['loss'])
plt.title('loss')
plt.show()
```

x = [1,4,5]　y = [2,3,6]



y = [2,3,6]



`plt.plot(x,y)`　←x軸にx、y軸にy　で折れ線

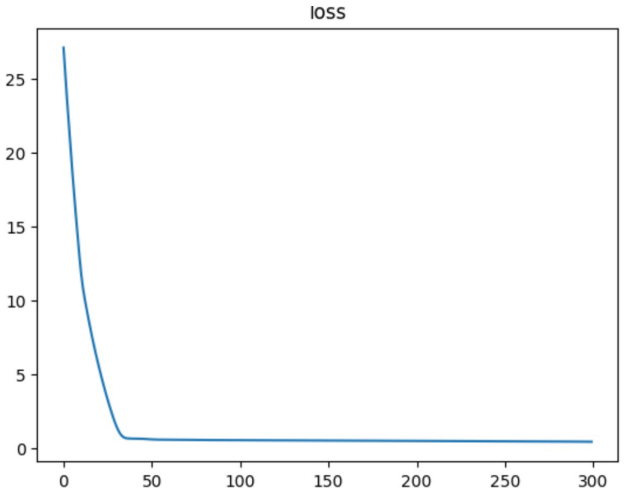
`plt.plot(y)`　←x軸にyのデータの数だけ[0,1,2..]、
y軸にy　で折れ線

`plt.plot(result.history['loss'])`
←x軸に[0,1,2...,299]、
y軸に[1回目の誤差, 2回目の誤差, … , 300回目の誤差],で折れ線

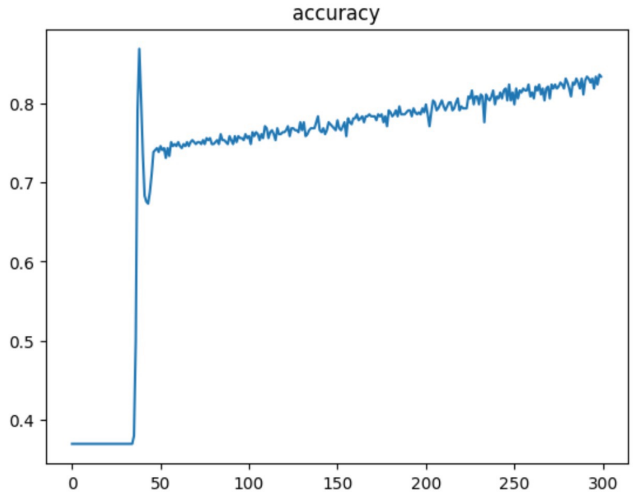
深層学習(乳がんデータの分類)

STEP4：学習結果の図示

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価



エポック回数



エポック回数

30~40回ぐらいでlossが小さくなり、そのあとはある程度一定

60~70回ぐらいでaccuracyが安定して、その後も300回まで微増

深層学習(乳がんデータの分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP5：モデルの評価

コード15-10　検証用データでモデルの評価

```
evaluate_loss, evaluate_accuracy = model_3.evaluate(x_test3, y_test)
print(evaluate_loss)
print(evaluate_accuracy)
```

```
6/6 [==============================] - 0s 5ms/step - loss: 0.4682 - accuracy: 0.8070
0.46817949414253235
0.8070175647735596
```

正解率は80.7%なのであまり高くない

- (モデル名).evaluate(x,y)でlossとaccuracy(モデルで指定したため)を計算

colab

精度を上げられるか検討

今までが中間層1つ、特徴量が3個だったため精度が低い

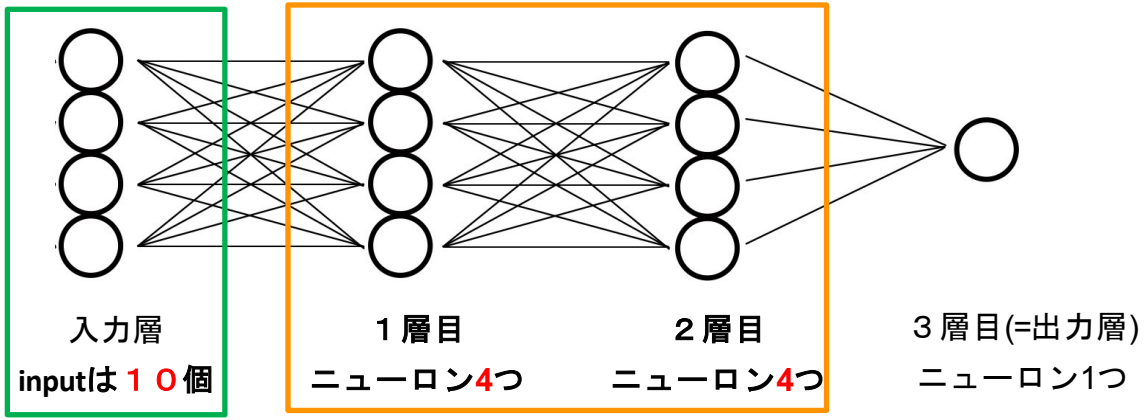
中間層と特徴量を増やしてモデルを複雑にして精度が上がるか検討する

深層学習(精度上げるための調整)

精度が上がるのか検討するため、作成する学習モデル

中間層は2つ

入力する特徴量
は10個にする



二値分類の場合、最後に出る値はY＝1になる確率p

深層学習(精度上げるための調整)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP1：データの用意

コード15-11 10個の特徴量を抽出する

```
x_train10 = x_train[:, 0:10]
x_test10 = x_test[:, 0:10]
print(x_train10.shape)
print(x_test10.shape)
```

```
(398, 10)
(171, 10)
```

- 特徴量を1~10番目(インデックス番号0~10)の特徴量だけを選択
- (データ名)[行番号,列番号]でnp配列の時は抽出できる

深層学習(精度上げるための調整)

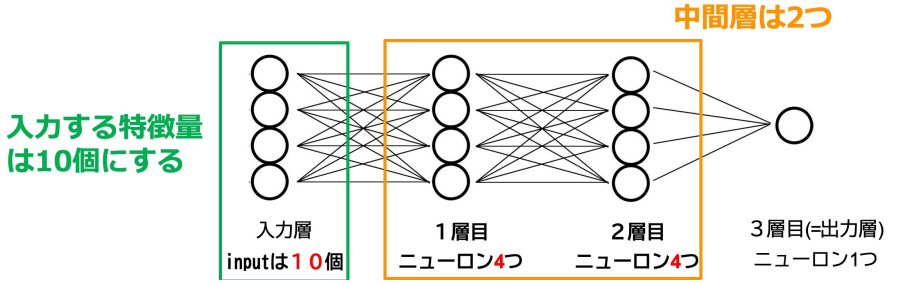
STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP2：学習モデルの選択

コード15-12 学習モデルを設計

```
model_10 = Sequential()
model_10.add(Dense(4, input_shape=(10,), activation = 'relu'))
model_10.add(Dense(4, activation = 'relu'))
model_10.add(Dense(1, activation = 'sigmoid'))
model_10.compile(loss = 'binary_crossentropy',
                 optimizer = 'Adam',
                 metrics = ['accuracy'])
model_10.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_6 (Dense)             (None, 4)                 44

 dense_7 (Dense)             (None, 4)                 20

 dense_8 (Dense)             (None, 1)                 5

=================================================================
Total params: 69 (276.00 Byte)
Trainable params: 69 (276.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

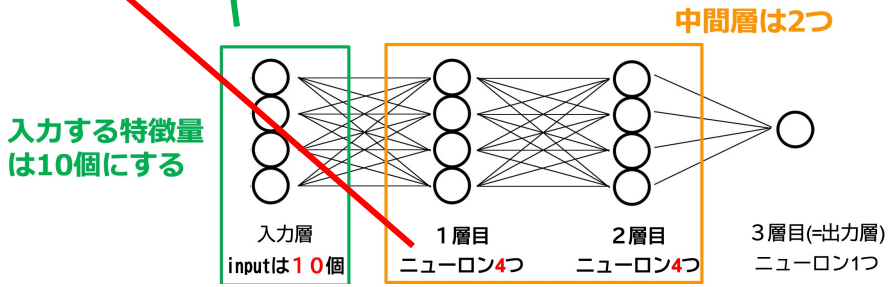


深層学習(精度上げるための調整)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP2：学習モデルの選択

```
model_10 = Sequential()
model_10.add(Dense(4, input_shape=(10,), activation = 'relu'))
model_10.add(Dense(4, activation = 'relu'))
model_10.add(Dense(1, activation = 'sigmoid'))
model_10.compile(loss = 'binary_crossentropy',
                 optimizer = 'Adam',
                 metrics = ['accuracy'])
model_10.summary()
```



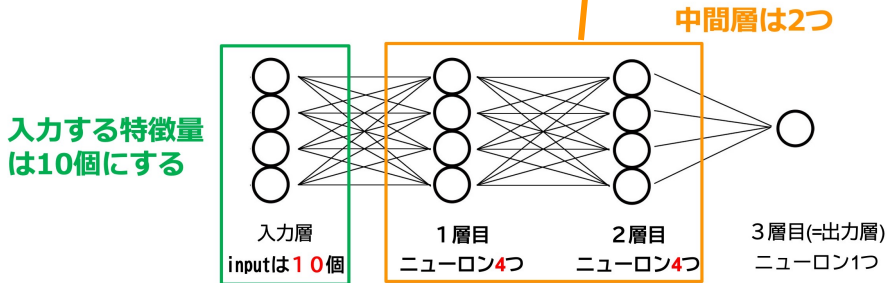
深層学習(精度上げるための調整)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP2：学習モデルの選択

```
model_10 = Sequential()
model_10.add(Dense(4, input_shape=(10,), activation = 'relu'))
model_10.add(Dense(4, activation = 'relu'))
model_10.add(Dense(1, activation = 'sigmoid'))
model_10.compile(loss = 'binary_crossentropy',
                 optimizer = 'Adam',
                 metrics = ['accuracy'])
model_10.summary()
```

入力層と中間層2つの設定



深層学習(精度上げるための調整)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP2：学習モデルの選択

```
model_10 = Sequential()
model_10.add(Dense(4, input_shape=(10,), activation = 'relu'))
model_10.add(Dense(4, activation = 'relu'))
model_10.add(Dense(1, activation = 'sigmoid'))
model_10.compile(loss = 'binary_crossentropy',
                 optimizer = 'Adam',
                 metrics = ['accuracy'])
model_10.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_6 (Dense)             (None, 4)                 44

 dense_7 (Dense)             (None, 4)                 20

 dense_8 (Dense)             (None, 1)                 5

=================================================================
Total params: 69 (276.00 Byte)
Trainable params: 69 (276.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

重み：特徴量10個×4ニューロン
=40個
バイアス 4個

パラメータは計69個

深層学習(精度上げるための調整)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP3：データを入れて学習

コード15-13 学習用データで学習させる

```
result10 = model_10.fit(x_train10, y_train,
                        batch_size = 32,
                        epochs = 300 )
```

- モデルと学習用データを変更し、結果を`result10`に入れる

colab

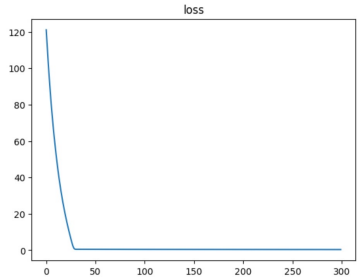
深層学習(精度上げるための調整)

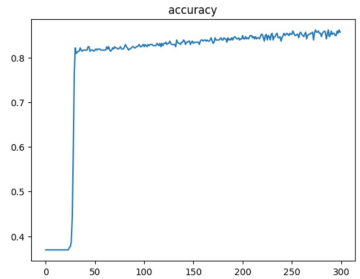
STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP4：学習結果の図示

コード15-14　学習結果の図示

```
plt.plot(result10.history['loss'])
plt.title('loss')
plt.show()

plt.plot(result10.history['accuracy'])
plt.title('accuracy')
plt.show()
```





- plt.plot(x,y)で各点をつなぐ線を描ける
- yは結果のloss/accuracyを選択
- xは指定していないとデータ数(300回)
- plt.title()でタイトルをつける
- plt.show()で図を表示する

colab

深層学習(乳がんデータの分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP5：モデルの評価

コード15-15　検証用データでモデルの評価

```
evaluate_loss, evaluate_accuracy = model_10.evaluate(x_test10, y_test)
print(evaluate_loss)
print(evaluate_accuracy)
```

```
6/6 [==============================] - 0s 5ms/step - loss: 0.4091 - accuracy: 0.8421
0.4091162383556366
0.8421052694320679
```

正解率は84.2%なので、model_3よりは正解率上がっている

- （モデル名）.evaluate(x,y)でlossとaccuracy(モデルで指定したため)を計算

colab

演習16 課題

Webclassで課題を提出してください。締め切りは2024/02/14 23:59まで

breast_cancerデータのデータセットで特徴量を1~20個目(インデックス番号0~19)の特徴量データ(x_train20, x_test20)で深層学習を行なってください

1)作成したx_train20, x_test20の配列の形状を回答してください
2)中間層1つ目を5つのニューロン(ノード)、中間層2つ目を3つのニューロン(ノード)としてモデルを作成し、(モデル名).summary()の結果の図を提出してください
3)epoch数200で学習し、学習過程のaccuracyの結果の折れ線グラフを提出してください(バッチサイズは好きなサイズでいいです)
4)x_test20とy_testでの正解率を回答してください(0~1)

授業準備：Webclassからコードをダウンロードし、Google colaboratoryで開いておいてください

演習授業中の質問対応について

医療とAI・ビッグデータ入門
演習17

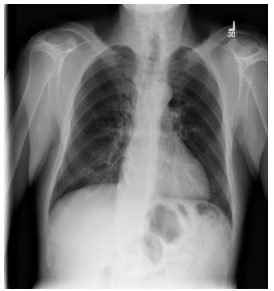
深層学習

演習15-20の概要

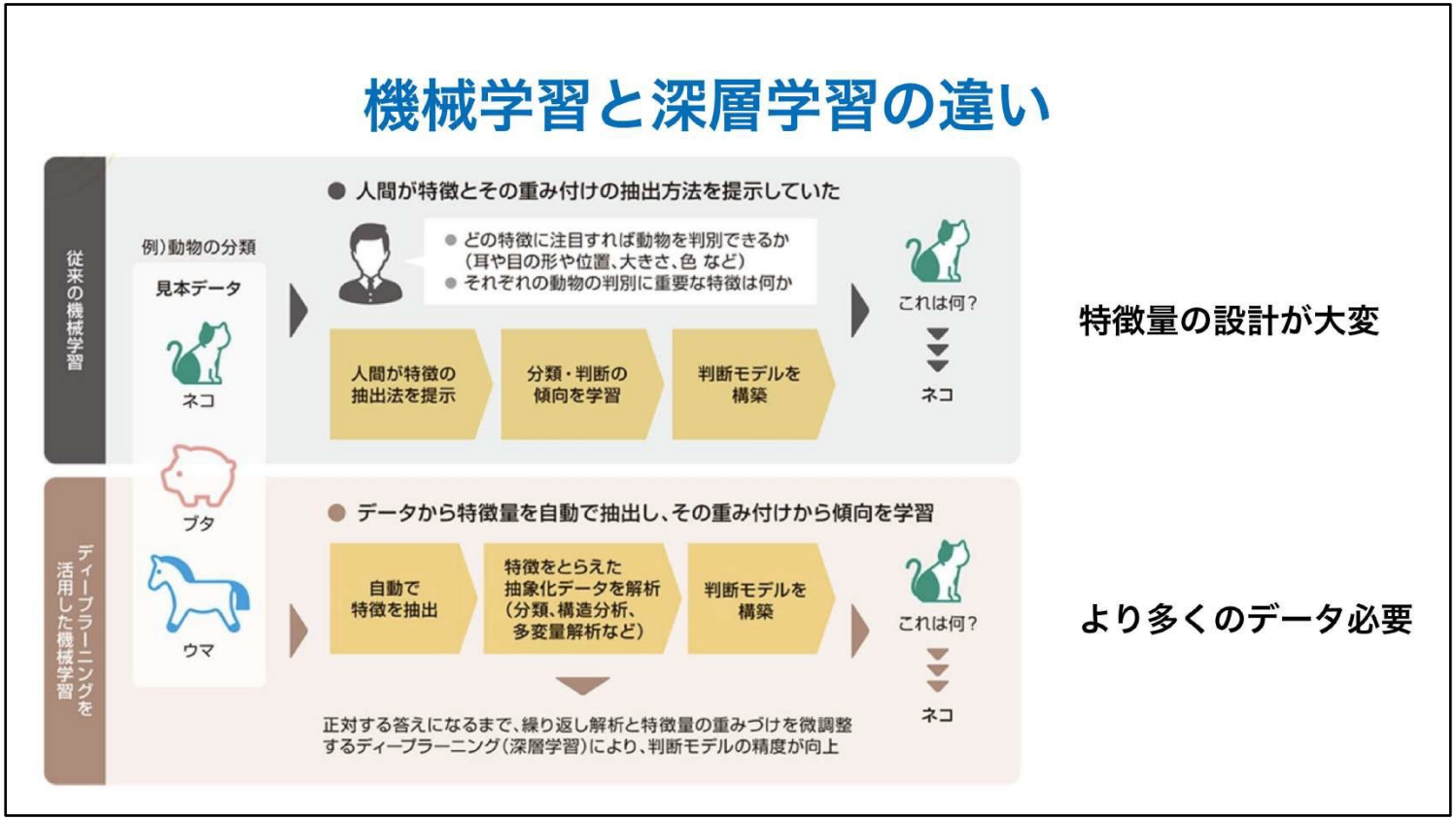
- 今までは教師あり機械学習の基礎を実行してきた
- 演習15-20では深層学習を実行する

15-16で深層学習の基礎と乳がんデータの分類
17-19で画像の分類を深層学習で行う
20　機械学習・深層学習の演習

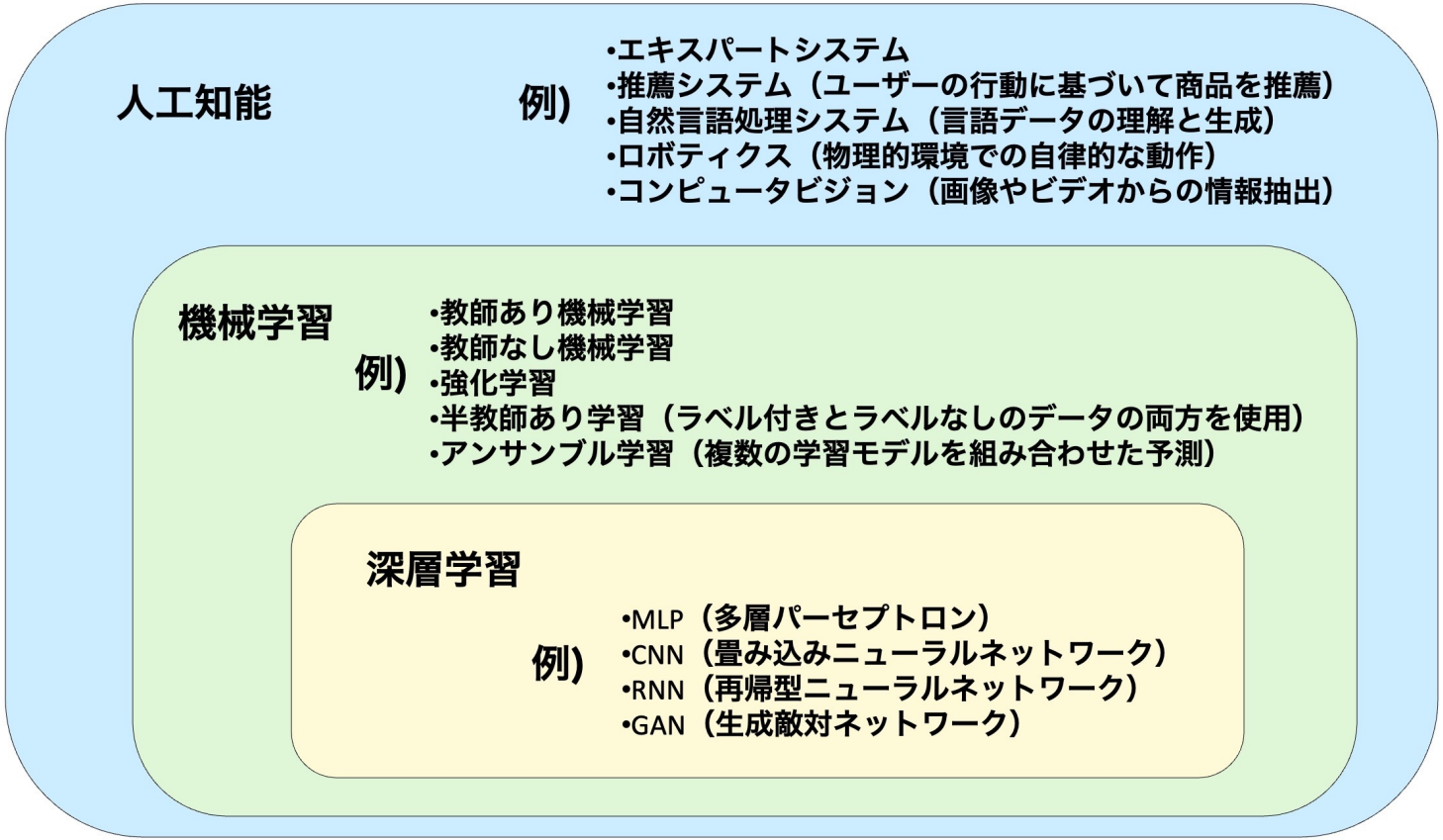
医療分野のAIとして、画像診断支援が非常に重要なトピック
今回から3回で肺のレントゲン画像を用いて、covid19肺炎かどうかを分類する深層学習を行う



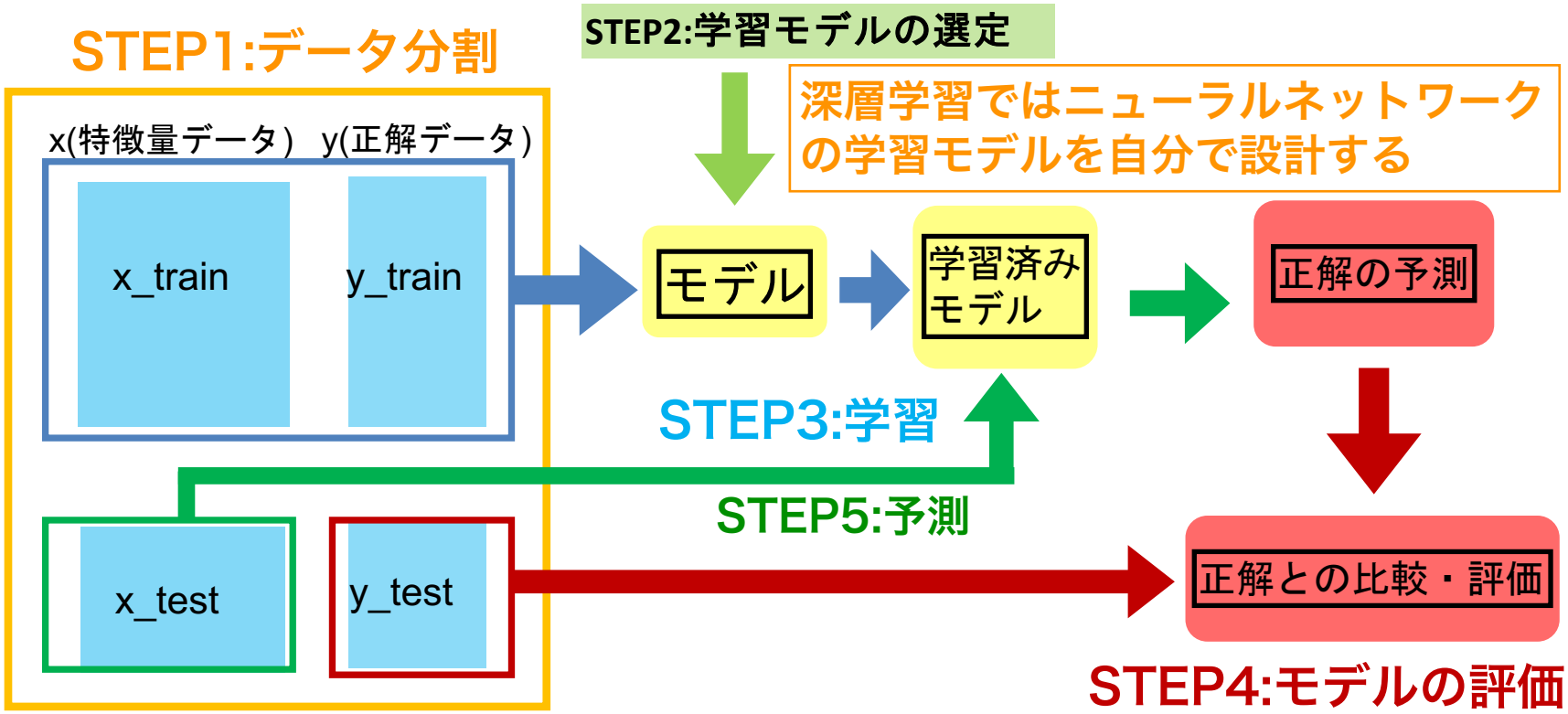
機械学習と深層学習の違い



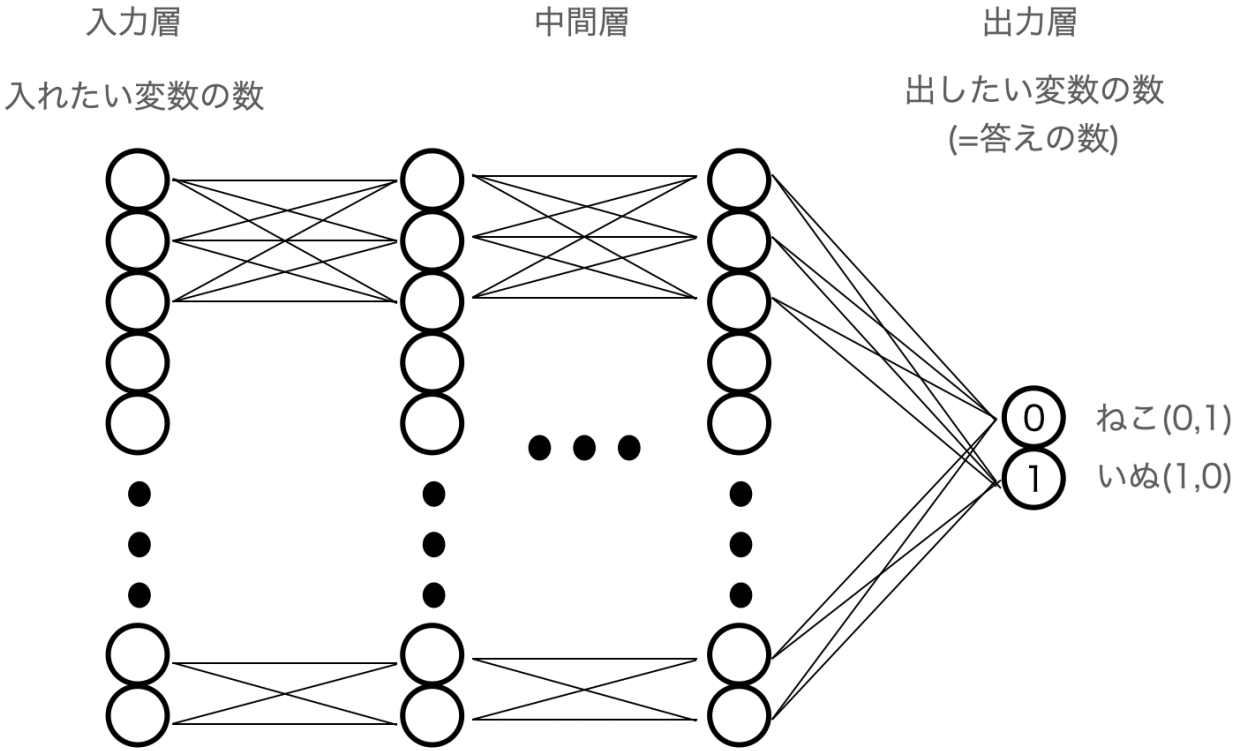
機械学習と深層学習の違い



深層学習のコードの流れ



ニューラルネットワークとは

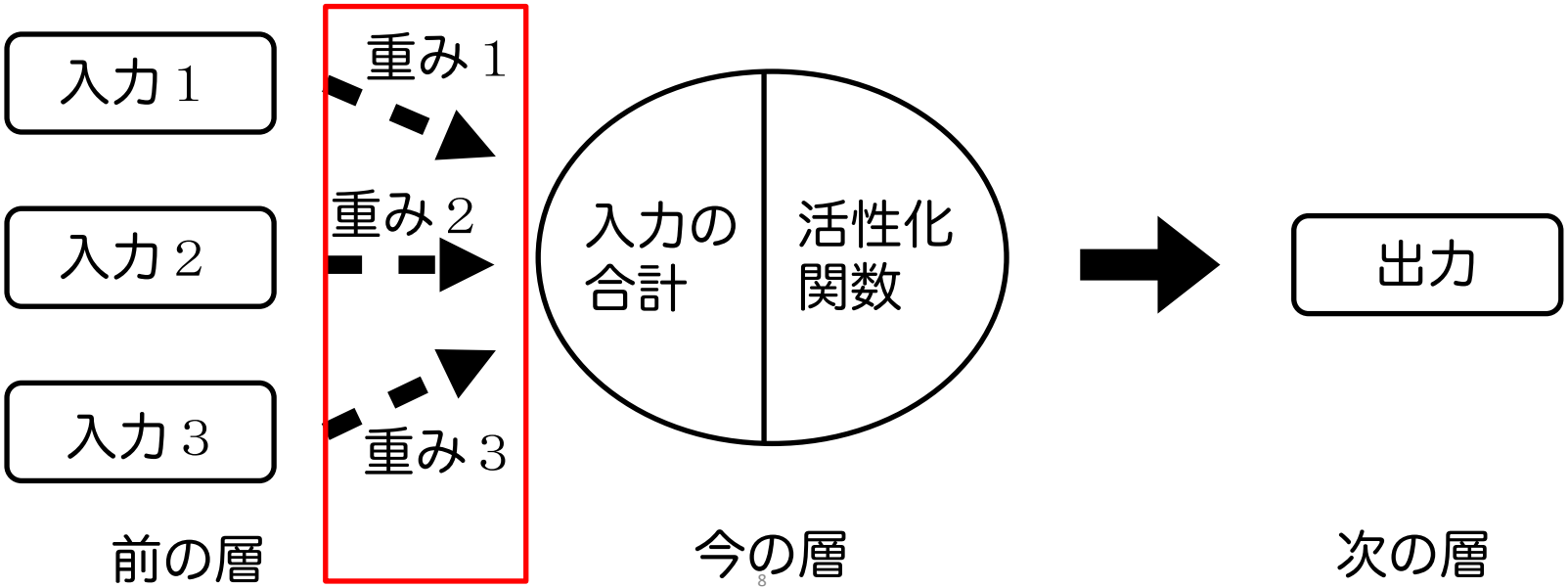


◯が全てニューロン、繋がった線の数だけ数式(関数)が存在する。

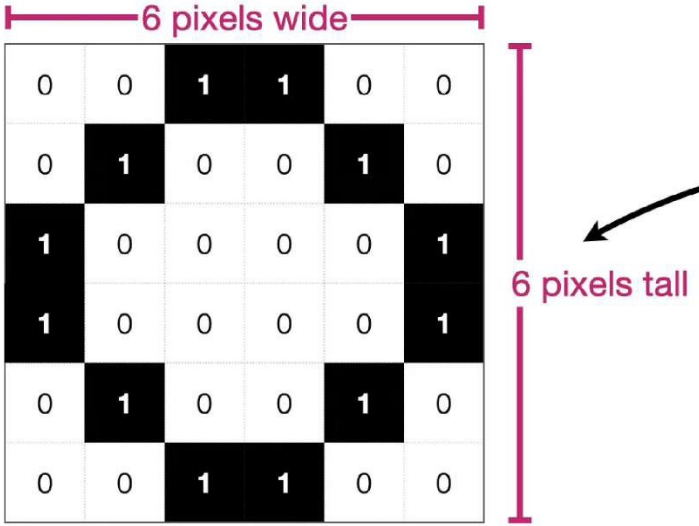
ニューラルネットワークとは

深層学習は、

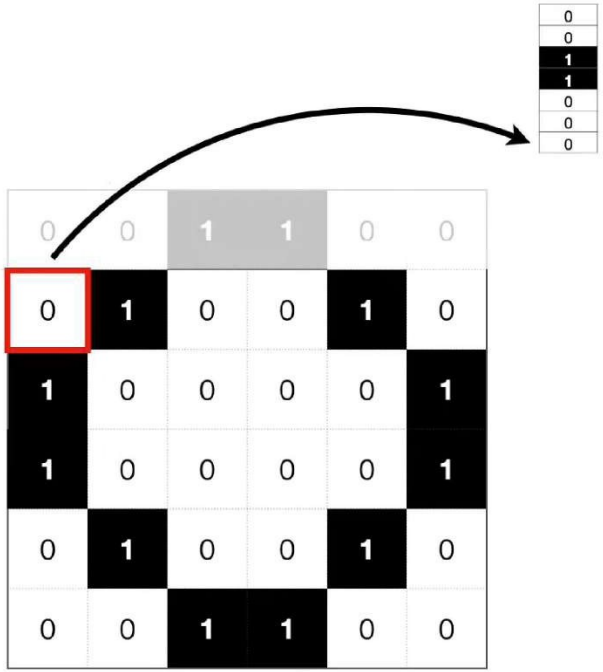
①一定量のデータの予測結果を算出する
②正解と予測結果がどれくらい異なっているかという誤差を計算する
③誤差が小さくなるように重みとバイアスを変える
を何度も繰り返すことで、誤差を減らしていき精度を高める



画像へ AI を使うには

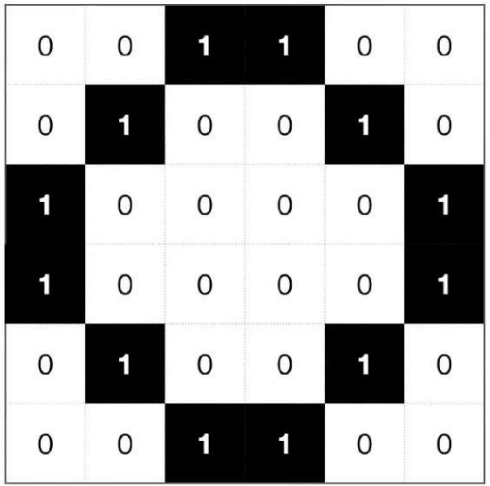


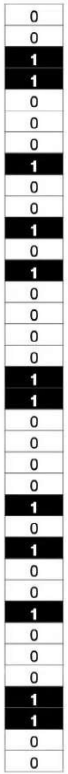
画像はピクセルの集まり。
そしてピクセルは数値で
表現できる。



画像へ AI を使うには

だから全てのピクセルを
順番に並べていけば ....

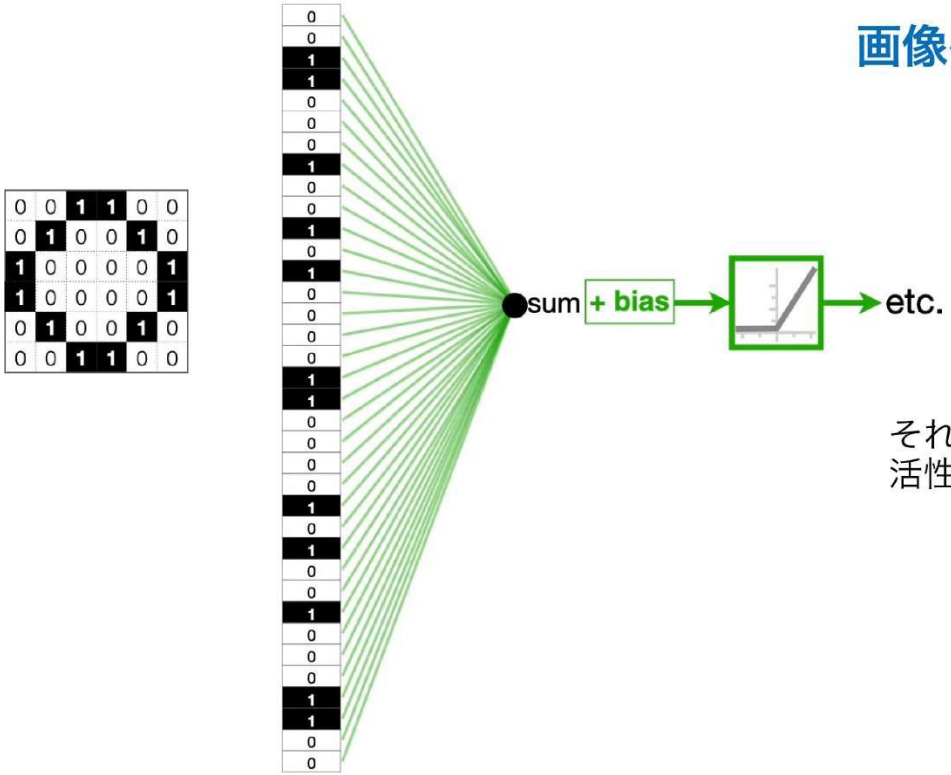




画像へ AI を使うには

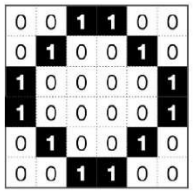
このように数値を並べた
ベクトルとして表現できる

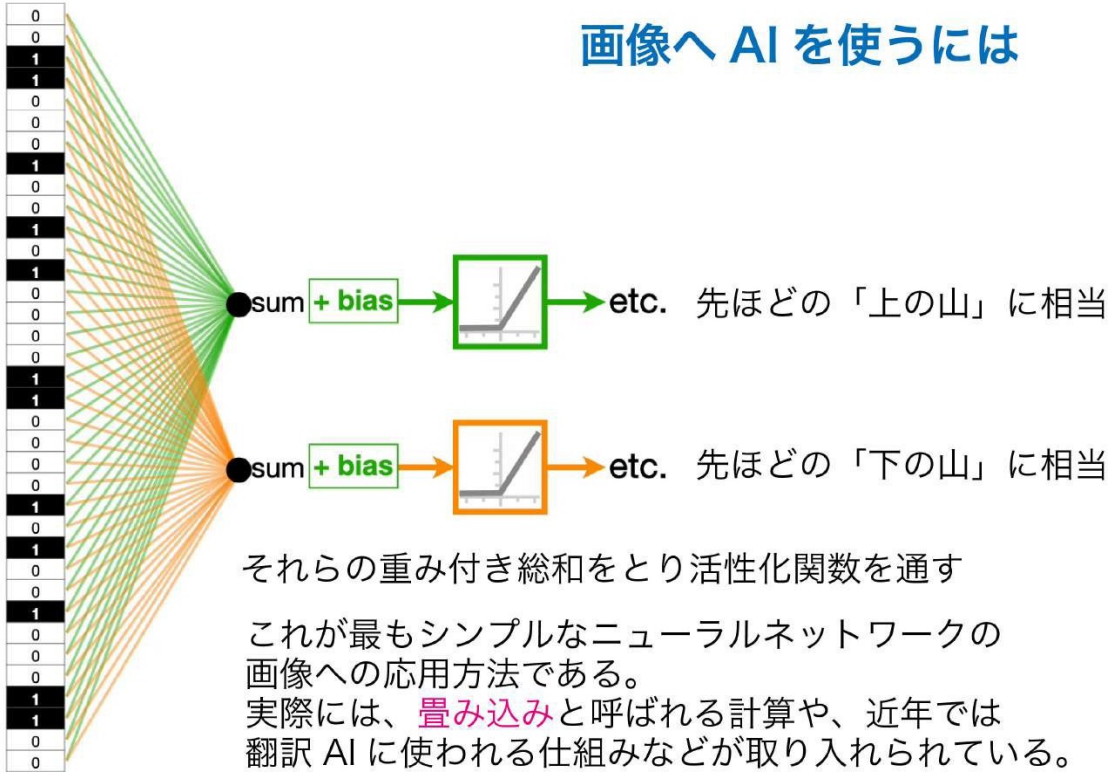
画像へ AI を使うには



それらの重み付き総和をとり
活性化関数を通す

画像へ AI を使うには



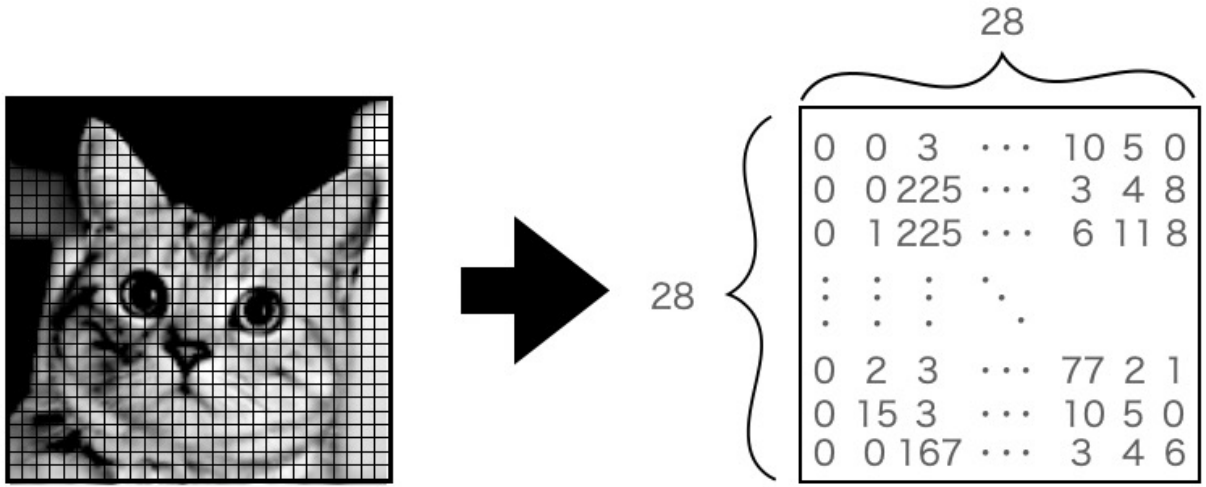


それらの重み付き総和をとり活性化関数を通す

これが最もシンプルなニューラルネットワークの
画像への応用方法である。
実際には、畳み込みと呼ばれる計算や、近年では
翻訳 AI に使われる仕組みなどが取り入れられている。

画像データは何が特徴量(説明変数)なのか

デジタル画像は数字で置き換えることが可能



白黒の濃さを0 ～255で表示

28✖28のマス(ピクセルと言います)に色の濃さの数字を当てはめて画像になっている。

入力層には各マスの色情報の数値(＝変数)を入力する

入力層

変数の数が入力数になる

中間層

出力層

出したい変数の数

64×64
= 4096

X:0, Y:0 の値

X:1, Y:0 の値

X:2, Y:0 の値

X:3, Y:0 の値

X:64, Y:63 の値

X:64, Y:64 の値

0

1

healthy(0,1)

covid19(1,0)

深層学習
演習

画像を使って深層学習を行う

深層学習(画像の分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP0：画像データのアップロード

今回から画像データを使うため、webclassから画像ファイル(image_TMDU.zip)をダウンロードしてください。

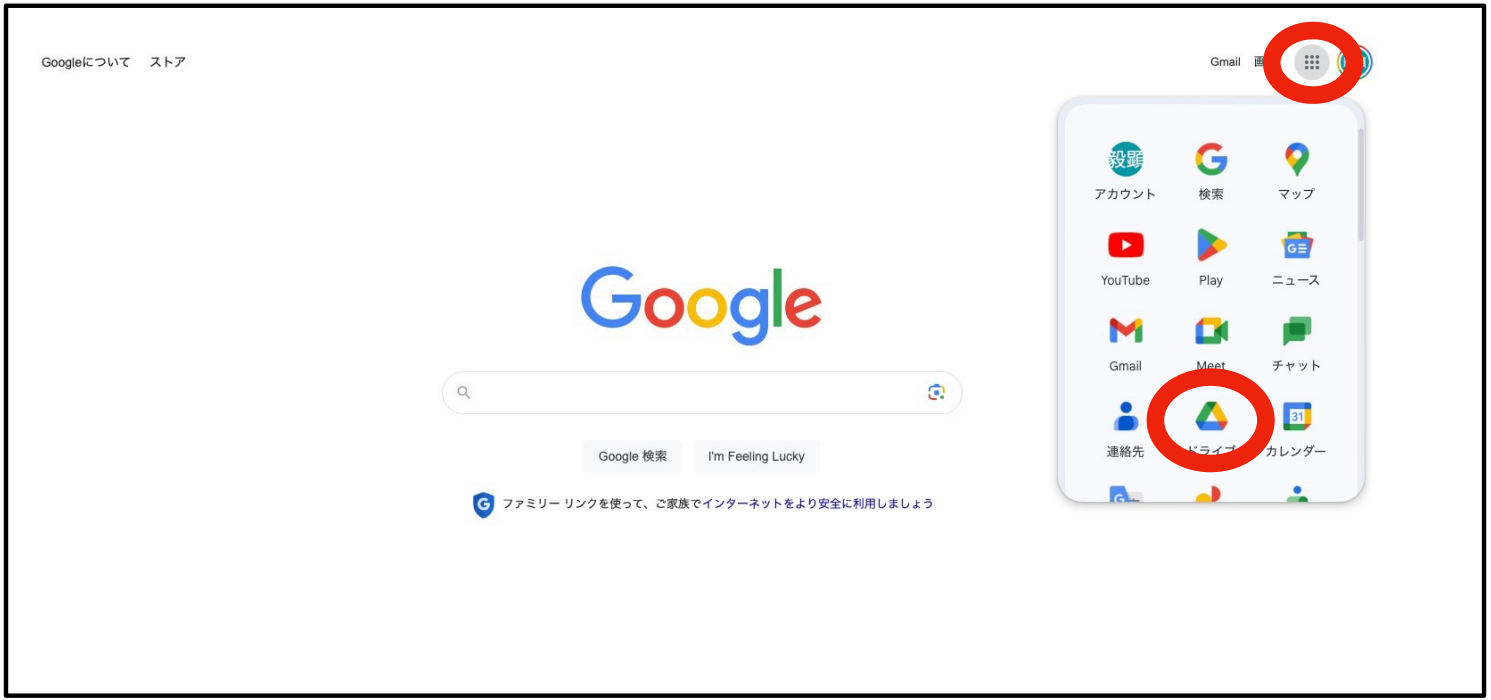
ダウンロードしたzipファイルを展開(解凍)せずに、そのまま自分のGoogle Driveにアップロードしてください。

深層学習(画像の分類)

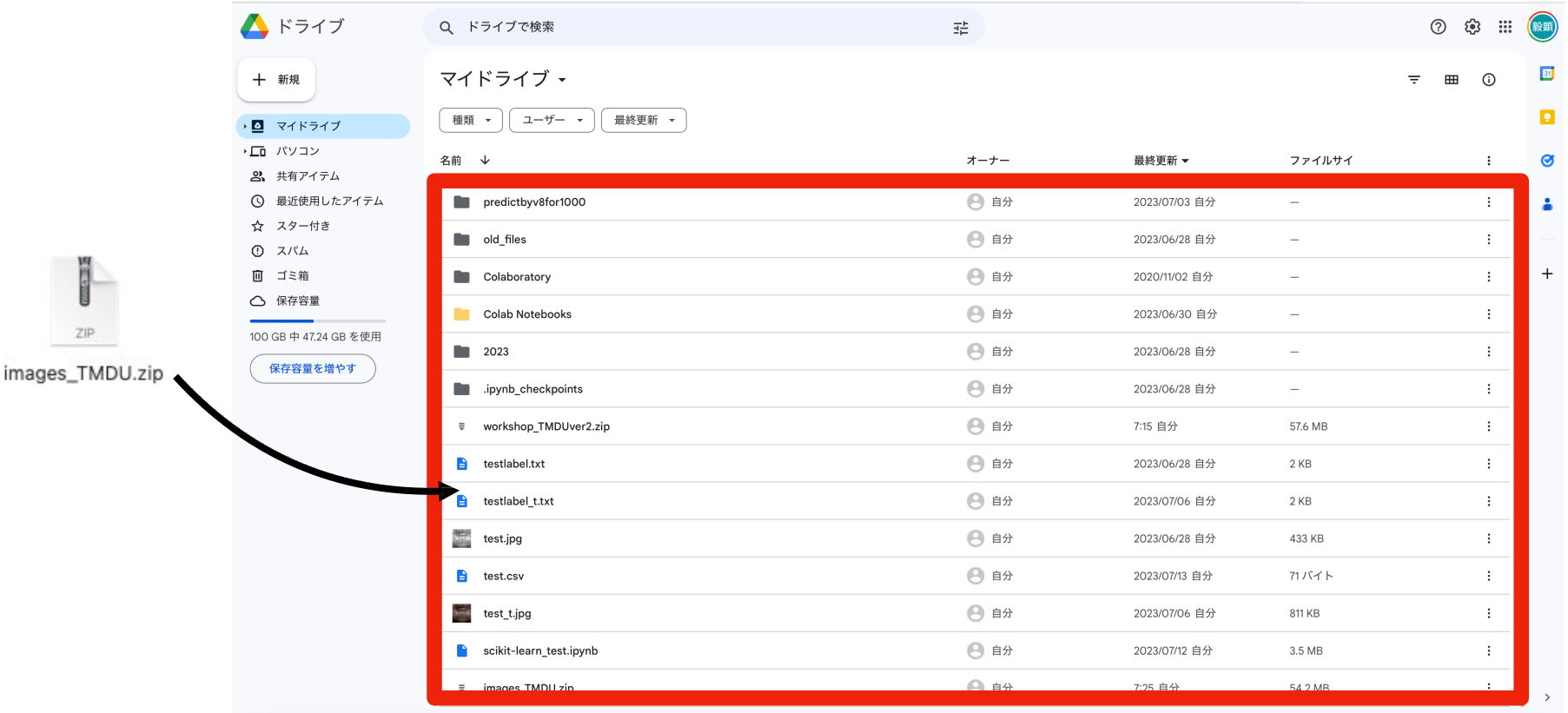
STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP0：画像データのアップロード

①GoogleのブラウザもしくはGmailなどの自分のアカウント画面からGoogle Driveへ移動します



②Google Driveのマイドライブの中にzipファイルのままアップロード(ドラッグアンドドロップ)して下さい



深層学習(画像の分類)

STEP0：Google Colaboratoryの立ち上げ

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

Python基礎　プログラミング基礎

検索google colab　Colaboratory へようこそ - Colaboratory - Google

演習17コード.ipynb

深層学習(乳がんデータの分類)

検索google colab　　Colaboratory へようこそ - Colaboratory - Google

演習17コード.ipynb

深層学習(画像の分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

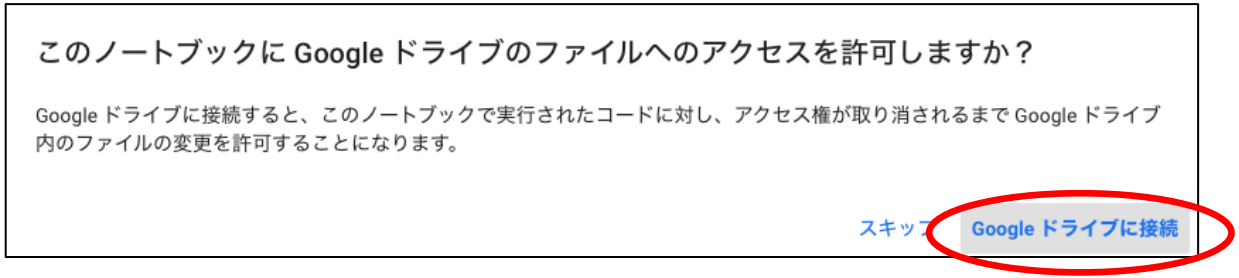
STEP0：Google Driveのマウント

コード17-1 Google Driveのマウント

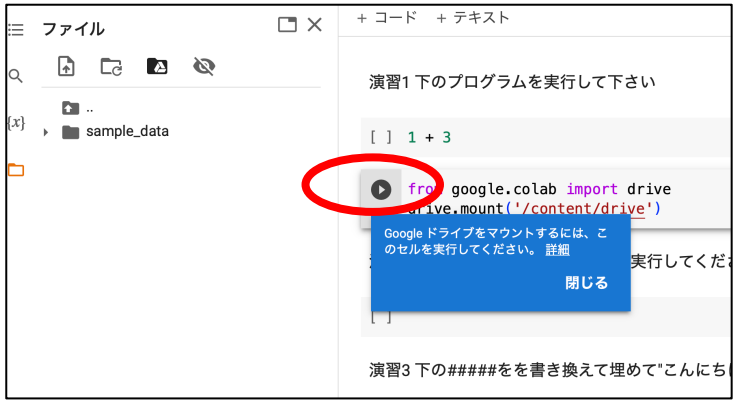
```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

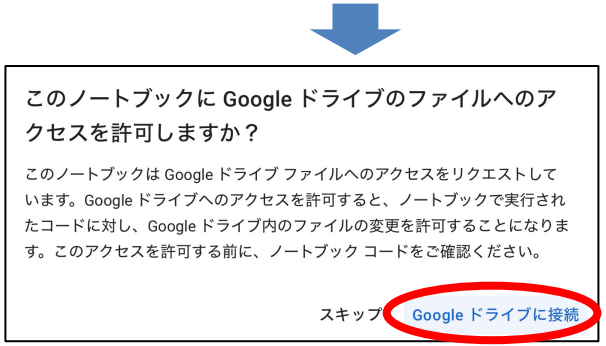
Mounted at /content/drive

上のコードを実行すると下の画面が出てくるので、「Googleドライブに接続」をクリック



最初に別の確認画面が表示された場合





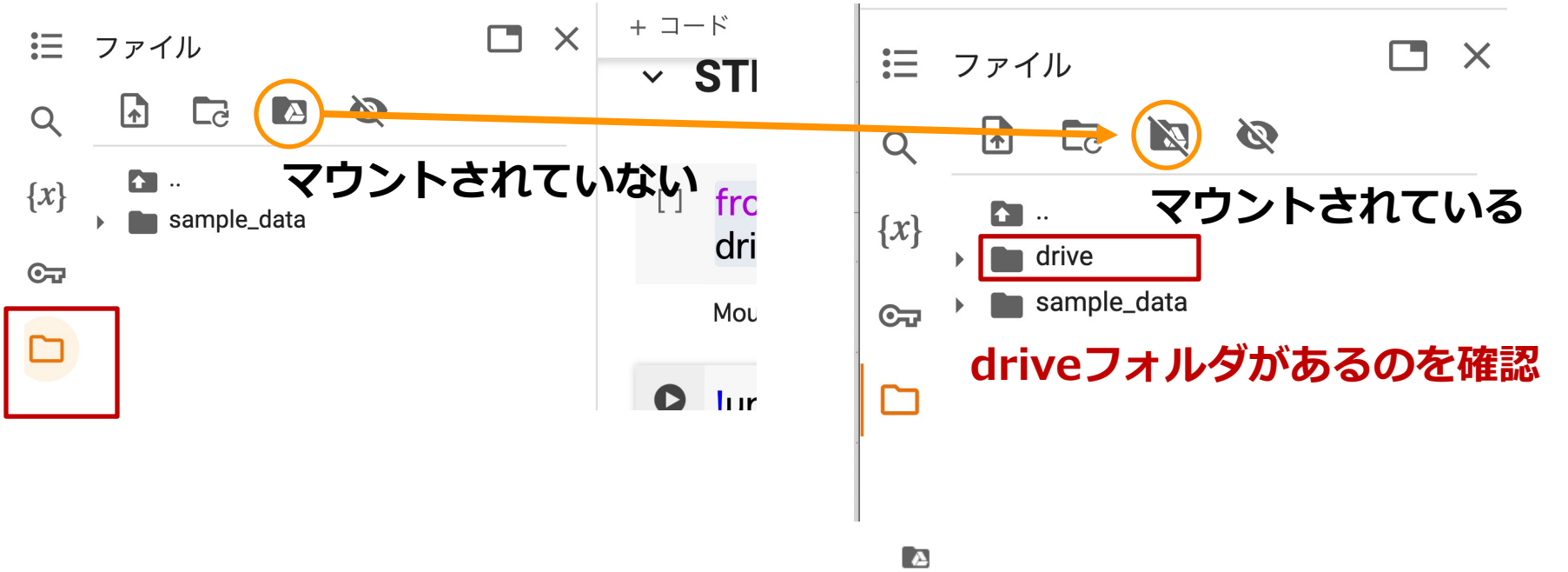
colab

深層学習(画像の分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP0：Google Driveのマウント

Driveをマウントするとアイコンが＼が入る

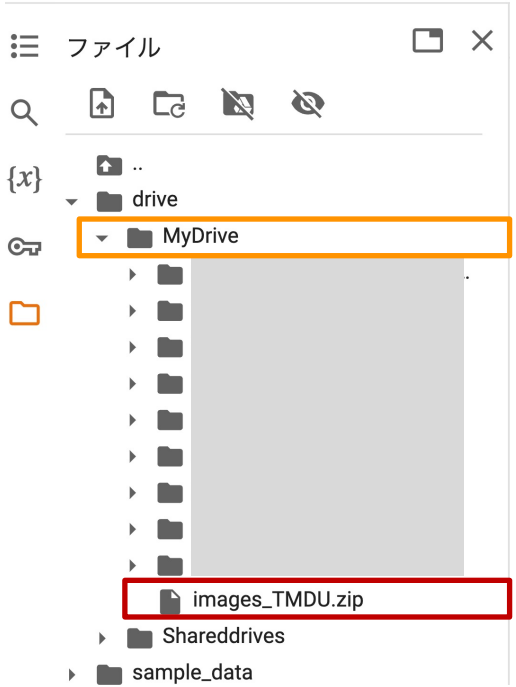


深層学習(画像の分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP0：Google Driveのマウント

images_TMDU.zipがgoogle driveにアップロードがされているかを確認



drive > MyDrive

images_TMDU.zipが存在すれば
きちんとアップロードされている

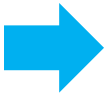
深層学習(画像の分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP0：ライブラリのインポート

コード17-2 zipファイルを解凍する

```
!unzip '/content/drive/MyDrive/images_TMDU.zip'
```



```
Archive:  /content/drive/MyDrive/images_TMDU.zip
  creating: images/
 inflating: images/.DS_Store
 inflating: __MACOSX/images/._.DS_Store
  creating: images/COVID-NORMAL/
 inflating: __MACOSX/images/._COVID-NORMAL
 inflating: images/test.jpg
 inflating: __MACOSX/images/._test.jpg
 inflating: images/covid.jpg
```

2回目以降に実行した場合↓

```
Archive:  /content/drive/MyDrive/images_TMDU.zip
replace images/.DS_Store? [y]es, [n]o, [A]ll, [N]one, [r]ename: N
```

ここにNを入れてEnter

!unzip 'zipファイル名' で指定したファイルを解凍する

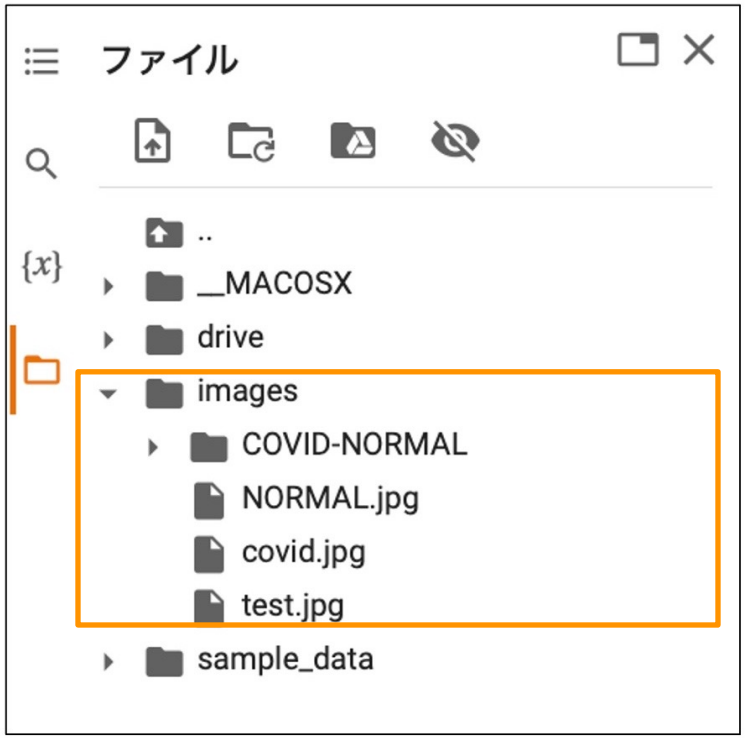
- セルの先頭に「!」をつけると、pythonではなくシェルというコマンドを実行できる
- zipファイルにはファイル名だけでなく、ファイルの場所（パス path）も指定する必要があるので、「'/content/drive/MyDrive/images_TMDU.zip'」となる

＊images_TMDU.zipをMy driveではないフォルダにアップしている人はパスが異なる

深層学習(画像の分類)

STEP0：ライブラリのインポート

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価



正常に動作すると、「images」フォルダの中に「COVID-NORMAL」というフォルダと3つのjpgファイル（「NORMAL.jpg」，「covid.jpg」，「test.jpg」）が入っていることが確認できる

zipファイルを入れている場所が間違えているなど、エラーが起こる人がいるかもしれないので、左のが確認できない人は教えてください
ここができないとこの後3回の演習できないので、**必ず解決してください。**

colab

深層学習(画像の分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP0：ライブラリのインポート

コード17-3 利用するモジュールのインポート

```
import matplotlib.pyplot as plt

from keras.preprocessing.image import load_img, img_to_array
```

関数をインポート：from import （ライブラリ名）.（モジュール名） as 省略形

matplotlib　pyplot

関数をインポート：from keras.preprocessing.image import load_img

ライブラリ　モジュール　モジュール

img_to_array

関数

colab

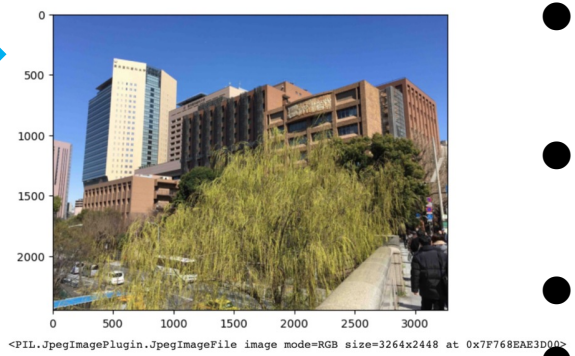
深層学習(画像の分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP0：ライブラリのインポート

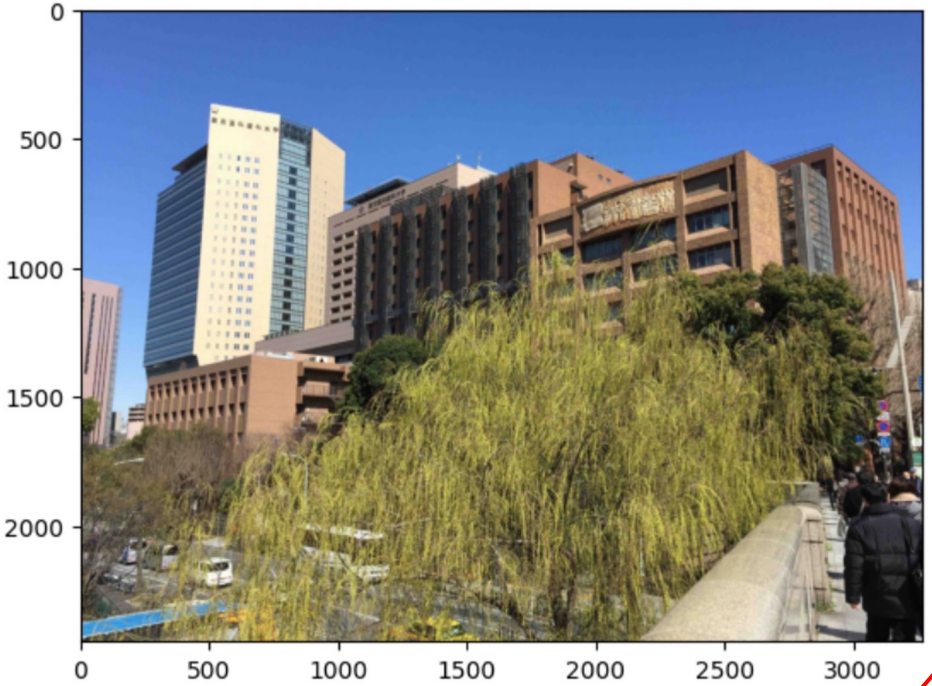
コード17-4 「test.jpg」の読み込みと図示

```
test = load_img('/content/images/test.jpg', color_mode='rgb')
plt.imshow(test)
plt.show()
print(test)
```



- load_img(ファイル名, color_mode =)で画像を読み込む
- color_mode = 'rgb'はRed, Green, Blueのことでカラーで読み込む 白黒は'grayscale'
- plt.imshow(test)： 画像を表示
- print(test)：画像データの情報を表示

画像を読み込んで表示する




```
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=3264x2448 at 0x7F768EAE3D00>
```

最後の行「**image mode=RGB size=3264x2448**」は、色の形式がRGBで表現されていて、画像のサイズが**横3264ピクセル、縦2448ピクセル**ということを示している

colab

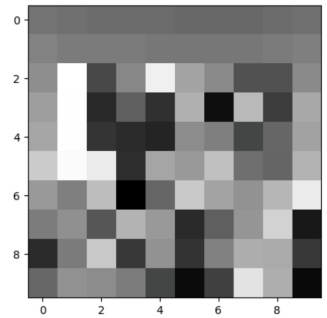
深層学習(画像の分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP0：ライブラリのインポート

コード17-5 画像サイズとカラーモードの変換

```
test2 = load_img('/content/images/test.jpg',
                 color_mode='grayscale', target_size=(10,10))
plt.imshow(test2, 'gray')
plt.show()
print(test2)
```



<PIL.Image.Image image mode=L size=10x10 at 0x7F771A6C2110>

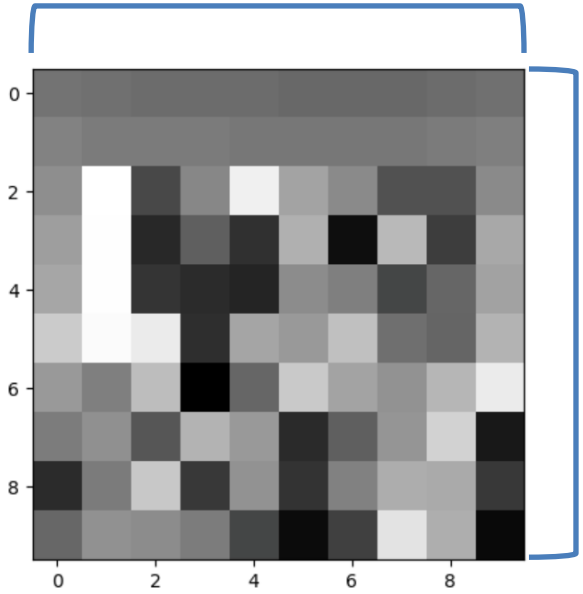
- target_size=で画像ファイルを読み込むときのピクセル数（サイズ）を指定します
- color_mode ='grayscale'で白黒で読み込む
- plt.imshow()の2つ目の引数に'gray'を指定

colab

画像を荒くして白黒で表示する

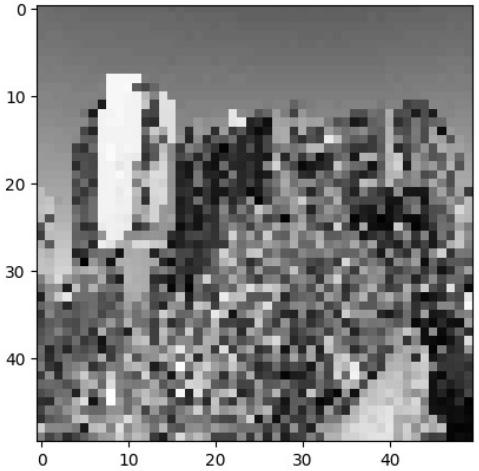
10×10の白黒で読み込む

表示結果がモザイクの様になる

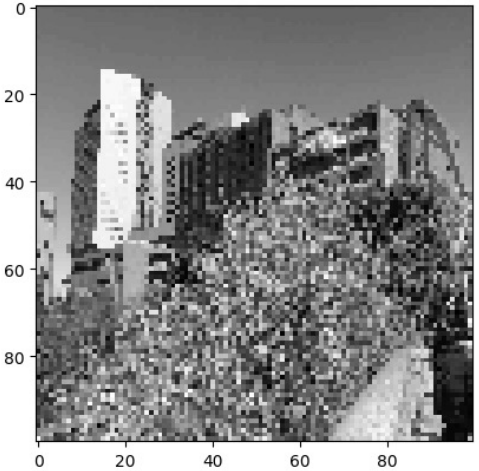



```
<PIL.Image.Image image mode=L size=10x10 at 0x7F771A6C2110>
```

50×50の
白黒で読み込むと↓



100×100の
白黒で読み込むと↓



深層学習(画像の分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP0：ライブラリのインポート

コード17-6 画像データを配列データに変換

```
test2_img = img_to_array(test2)
print(test2_img.shape)
print(test2_img)
```

```
(10,10,1)
[[[114.]
  [111.]
  [107.]
  [107.]
  [107.]
  [104.]
  [104.]
  [104.]
  [107.]
  [111.]]
  :
```

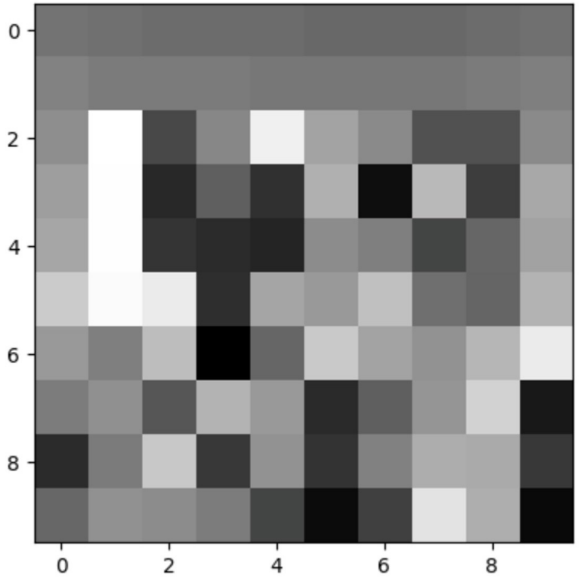
- `img_to_array()`は画像データを配列に変換する
- 配列構造は `(10,10,1)`の三次元配列

  (横ピクセル数，縦ピクセル数，色数)の配列構造
- `print(test2_img)`で、数字の羅列が表示される

colab

画像データを配列データに変換

test2



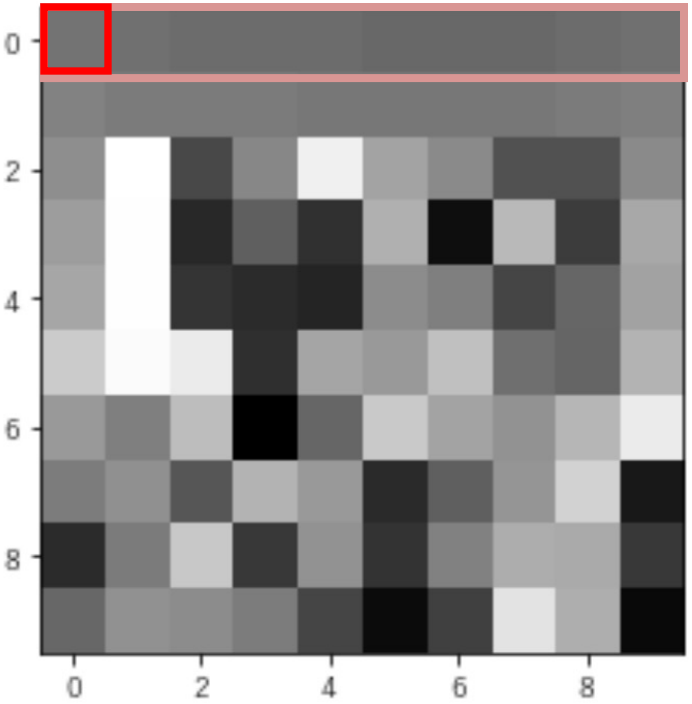
test2_img

```
[[[114.][111.][107.][107.][107.][104.][104.][104.][107.][111.]]
 [[128.][121.][121.][121.][118.][118.][118.][118.][121.][125.]]
 [[139.][244.][ 74.][132.][230.][159.][135.][ 82.][ 83.][135.]]
 [[153.][243.][ 45.][ 95.][ 53.][170.][ 20.][179.][ 63.][163.]]
 [[161.][243.][ 56.][ 47.][ 42.][137.][125.][ 71.][102.][157.]]
 [[195.][240.][225.][ 51.][160.][149.][185.][110.][101.][174.]]
 [[149.][125.][182.][  7.][102.][194.][158.][143.][176.][225.]]
 [[122.][141.][ 87.][174.][149.][ 46.][ 95.][145.][202.][ 30.]]
 [[ 47.][121.][193.][ 60.][143.][ 55.][127.][168.][165.][ 60.]]
 [[103.][142.][137.][122.][ 71.][ 18.][ 67.][219.][169.][ 16.]]]
```

それぞれに色のレベル（輝度）を0から255の大きさで表した数値が割り当てられる

黒 :0 → **白 :255**

画像データを配列データに変換




```
[[[114.]
  [111.]
  [107.]
  [107.]
  [107.]
  [104.]
  [104.]
  [104.]
  [107.]
  [111.]]
 :
```

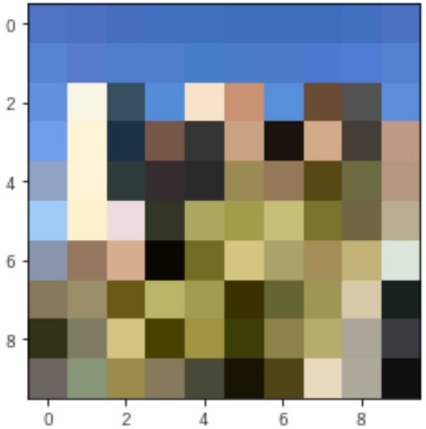
深層学習(画像の分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP0：ライブラリのインポート

コード17-7 画像サイズ(10,10)とカラーでの図示

```
test3 = load_img('/content/images/test.jpg',
                 color_mode='rgb', target_size=(10,10))
plt.imshow(test3)
plt.show()
```



- target_size=で画像ファイルを読み込むときのピクセル数（サイズ）を指定します

colab

深層学習(画像の分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP0：ライブラリのインポート

コード17-8 画像サイズ(10,10)とカラーでの配列

```
test3_img = img_to_array(test3)
print(test3_img)
print(test3_img.shape)
```

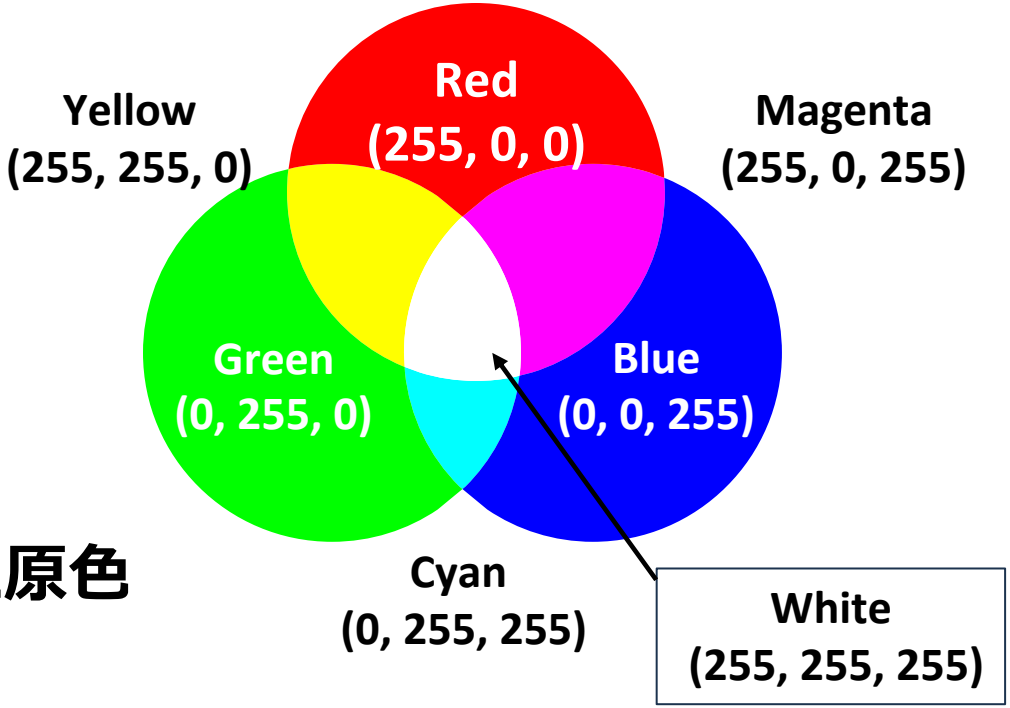
```
[[[ 79. 116. 197.]
  [ 76. 113. 194.]
  [ 71. 109. 190.]
  [ 65. 112. 190.]
  [ 65. 112. 190.]
  [ 62. 109. 187.]
  [ 62. 109. 187.]
  [ 62. 109. 187.]
  [ 65. 112. 190.]
  [ 76. 113. 194.]]
      :
(10, 10, 3)
```

- 3次元配列のデータ
- 1番内側の大括弧内の3つの数字は色を表し、red（R）、green（G）、blue（B）の3つの色のレベル（輝度）をそれぞれ0から255の値で表現している

colab

RGBとは

コンピュータで画像を扱う際の標準的な色の表現方法の一つ
Red, Green, Blueの数値(0~255)の組み合わせで表現する



光の三原色

画像データを配列データに変換

[[[79. 116. 197.]	[76. 113. 194.]	[71. 109. 190.]	[65. 112. 190.]	[65. 112. 190.]	[62. 109. 187.]	[62. 109. 187.]	[62. 109. 187.]	[65. 112. 190.]	[76. 113. 194.]]
[[86. 133. 213.]	[86. 123. 204.]	[79. 126. 204.]	[79. 126. 204.]	[70. 126. 201.]	[75. 124. 201.]	[76. 123. 201.]	[76. 122. 208.]	[79. 125. 211.]	[83. 130. 208.]]
[[97. 144. 222.]	[248. 245. 226.]	[54. 80. 97.]	[84. 140. 215.]	[248. 227. 200.]	[201. 146. 115.]	[87. 143. 218.]	[106. 76. 52.]	[83. 83. 83.]	[93. 140. 218.]]
[[111. 158. 236.]	[255. 243. 214.]	[27. 50. 68.]	[119. 87. 72.]	[53. 53. 53.]	[201. 162. 131.]	[26. 19. 13.]	[210. 171. 138.]	[69. 62. 56.]	[192. 154. 133.]]
[[144. 163. 196.]	[255. 243. 214.]	[46. 60. 61.]	[53. 44. 47.]	[42. 42. 42.]	[155. 138. 84.]	[149. 120. 88.]	[88. 72. 20.]	[108. 106. 65.]	[181. 151. 127.]]
[[160. 203. 248.]	[253. 241. 203.]	[236. 220. 223.]	[51. 54. 37.]	[173. 166. 94.]	[162. 157. 75.]	[198. 191. 119.]	[123. 116. 46.]	[114. 101. 67.]	[184. 175. 144.]]
[[138. 150. 172.]	[149. 119. 95.]	[213. 174. 143.]	[10. 6. 0.]	[115. 109. 35.]	[212. 197. 128.]	[171. 162. 105.]	[164. 143. 88.]	[194. 178. 119.]	[219. 229. 218.]]
[[135. 121. 92.]	[154. 142. 104.]	[105. 90. 23.]	[187. 181. 103.]	[162. 156. 80.]	[58. 49. 0.]	[101. 101. 51.]	[158. 150. 87.]	[215. 202. 168.]	[23. 34. 30.]]
[[48. 51. 22.]	[127. 123. 98.]	[211. 196. 129.]	[72. 65. 0.]	[161. 149. 65.]	[61. 61. 7.]	[141. 130. 74.]	[181. 173. 108.]	[171. 165. 151.]	[60. 59. 65.]]
[[109. 102. 96.]	[135. 150. 119.]	[155. 139. 77.]	[135. 121. 92.]	[71. 74. 57.]	[24. 19. 0.]	[80. 69. 23.]	[232. 218. 189.]	[175. 169. 155.]	[16. 16. 16.]]]

10 × 10 × 3 (3色)

深層学習(画像の分類)

STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP0：ライブラリのインポート

コード17-9 インデックスで三次元配列へのアクセス

```
#0行目
print(test3_img[0])
```

```
[[79. 116. 197.]
 [76. 113. 194.]
 [71. 109. 190.]
 [ 65. 112. 190.]
 [ 65. 112. 190.]
 [ 62. 109. 187.]
 [ 62. 109. 187.]
 [ 62. 109. 187.]
 [ 65. 112. 190.]
 [ 76. 113. 194.]]
```

```
#0行目の0列目
print(test3_img[0][0])
```

```
[ 79. 116. 197.]
```

```
#0行目0列目の0要素目(Redの値)
print(test3_img[0][0][0])
```

```
79.0
```

colab

```
print(test3_img[0][0][0])
```

```
print(test3_img[0][0])
```

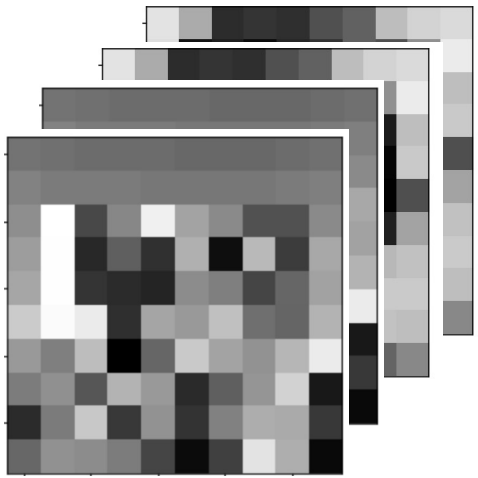
```
print(test3_img[0])
```

```
[[[79. 116. 197.] [76. 113. 194.] [71. 109. 190.] [65. 112. 190.] [65. 112. 190.] [62. 109. 187.] [62. 109. 187.] [62. 109. 187.] [65. 112. 190.] [76. 113. 194.]]
 [[86. 133. 213.] [86. 123. 204.] [79. 126. 204.] [79. 126. 204.] [70. 126. 201.] [75. 124. 201.] [76. 123. 201.] [76. 122. 208.] [79. 125. 211.] [83. 130. 208.]]
 [[97. 144. 222.] [248. 245. 226.] [54. 80. 97.] [84. 140. 215.] [248. 227. 200.] [201. 146. 115.] [87. 143. 218.] [106. 76. 52.] [83. 83. 83.] [93. 140. 218.]]
 [[111. 158. 236.] [255. 243. 214.] [27. 50. 68.] [119. 87. 72.] [53. 53. 53.] [201. 162. 131.] [26. 19. 13.] [210. 171. 138.] [69. 62. 56.] [192. 154. 133.]]
 [[144. 163. 196.] [255. 243. 214.] [46. 60. 61.] [53. 44. 47.] [42. 42. 42.] [155. 138. 84.] [149. 120. 88.] [88. 72. 20.] [108. 106. 65.] [181. 151. 127.]]
 [[160. 203. 248.] [253. 241. 203.] [236. 220. 223.] [51. 54. 37.] [173. 166. 94.] [162. 157. 75.] [198. 191. 119.] [123. 116. 46.] [114. 101. 67.] [184. 175. 144.]]
 [[138. 150. 172.] [149. 119. 95.] [213. 174. 143.] [10. 6. 0.] [115. 109. 35.] [212. 197. 128.] [171. 162. 105.] [164. 143. 88.] [194. 178. 119.] [219. 229. 218.]]
 [[135. 121. 92.] [154. 142. 104.] [105. 90. 23.] [187. 181. 103.] [162. 156. 80.] [58. 49. 0.] [101. 101. 51.] [158. 150. 87.] [215. 202. 168.] [23. 34. 30.]]
 [[48. 51. 22.] [127. 123. 98.] [211. 196. 129.] [72. 65. 0.] [161. 149. 65.] [61. 61. 7.] [141. 130. 74.] [181. 173. 108.] [171. 165. 151.] [60. 59. 65.]]
 [[109. 102. 96.] [135. 150. 119.] [155. 139. 77.] [135. 121. 92.] [71. 74. 57.] [24. 19. 0.] [80. 69. 23.] [232. 218. 189.] [175. 169. 155.] [16. 16. 16.]]]
```

10×10×3(3色)

画像を複数読み込んだ場合は？

画像を4枚読み込んだ時は (4, 10, 10, 1)の4次元配列になる



```
[[[[114.]  [[[114.]  [[[224.]  [[[224.]
  [111.]    [111.]    [169.]    [169.]
  [107.]    [107.]    [ 48.]    [ 48.]
  [107.]    [107.]    [ 55.]    [ 55.]
  [107.]    [107.]    [ 50.]    [ 50.]
  [104.]    [104.]    [ 82.]    [ 82.]
  [104.]    [104.]    [ 98.]    [ 98.]
  [104.]    [104.]    [186.]    [186.]
  [107.]    [107.]    [207.]    [207.]
  [111.]]   [111.]]   [214.]]   [214.]]
    .         .         .         .
  [222. ]]] [222. ]]] [222. ]]] [222. ]]] ]
```

画像を複数読み込んだ配列は(枚数、横のセル、縦のセル、色の数)になる

演習17：課題

Webclassで課題を提出してください。締め切りは2024/02/15 23:59まで

img_TMDU.zipに入っているkadai.jpegの画像データを読み込んで、配列を確認してください。

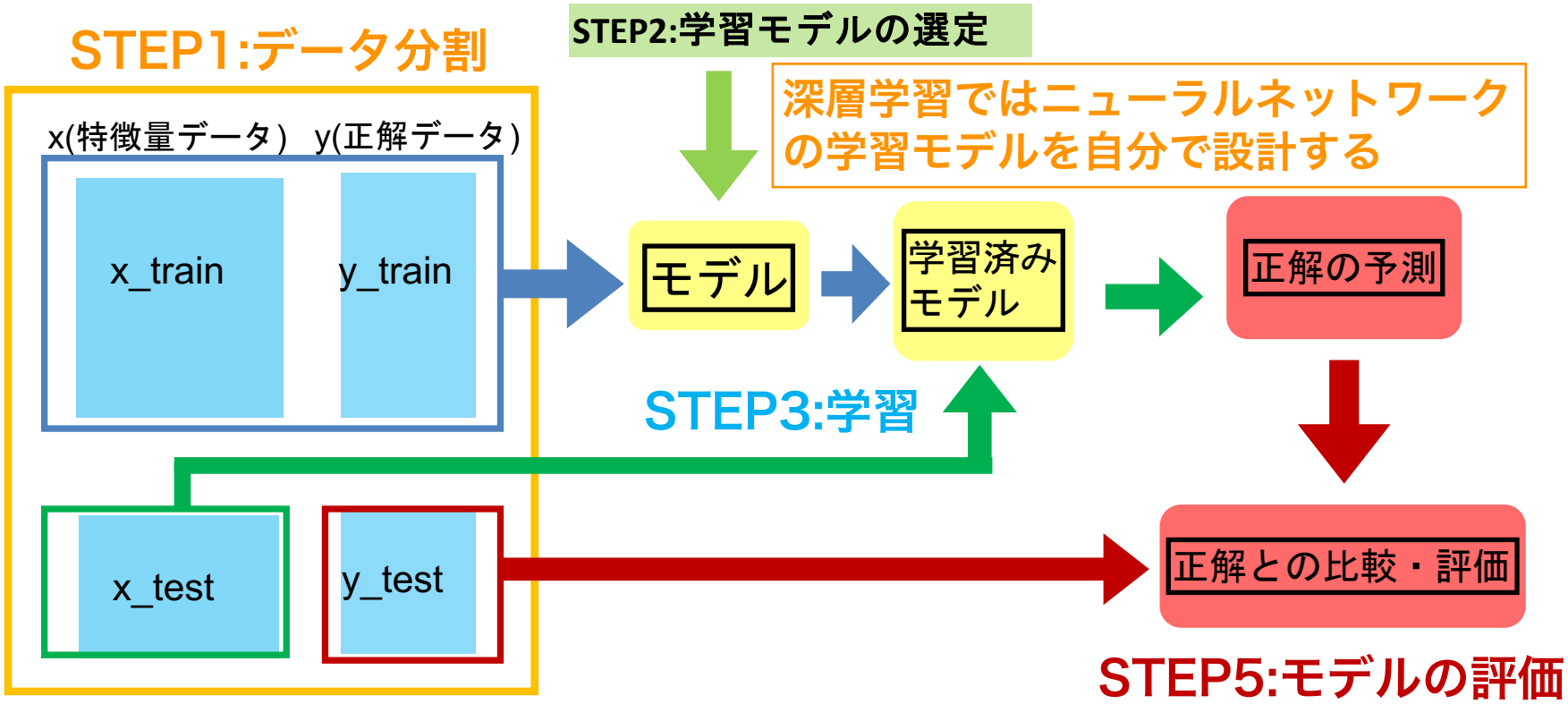
- **load_img()の引数でcolor_modeはカラー(rgb) 、target_sizeは30×30ピクセルに指定してください。**
- **あなたの学生番号の下一桁の番号の行と列のRGBの値を入力してください。**

(0001なら1なので、読み込んだ画像データの配列の1行目1列目(インデックス番号が1)のデータのRGBの値を入力する)

医療とAI・ビッグデータ入門
演習18

深層学習（画像データ）

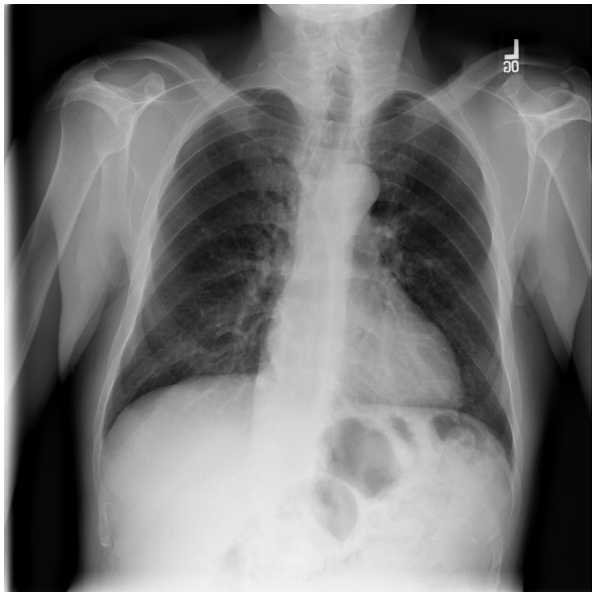
深層学習のコードの流れ



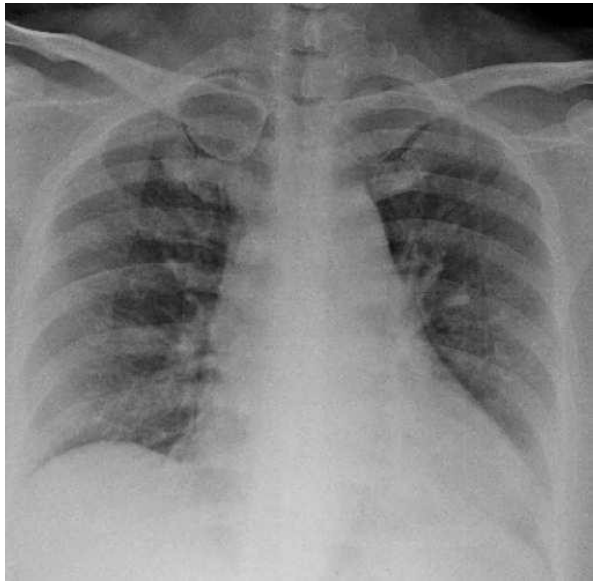
深層学習 (画像の分類)

肺のレントゲン画像で深層学習を行いCovid19肺炎かどうかを予測する

健康



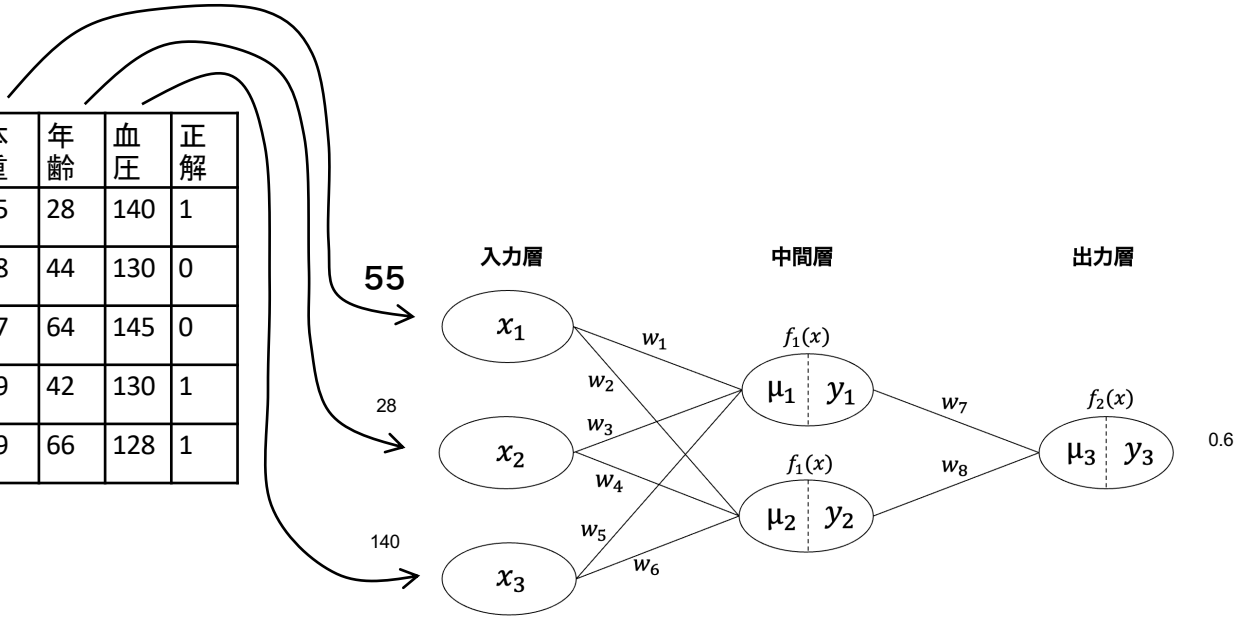
Covid19肺炎



深層学習 (画像の分類)

(復習) ニューラルネットワークとは

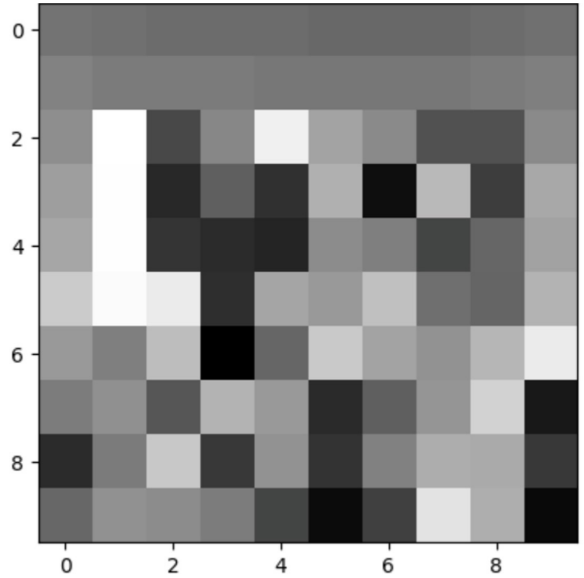
	体重	年齢	血圧	正解
1	55	28	140	1
2	48	44	130	0
3	77	64	145	0
4	49	42	130	1
5	59	66	128	1



	体重	年齢	血圧	正解	予測
1	55	28	140	1	0.6
2	48	44	130	0	0.3
3	77	64	145	0	0.4
4	49	42	130	1	0.9
5	59	66	128	1	0.6

深層学習（画像の分類）

画像データを配列データに変換



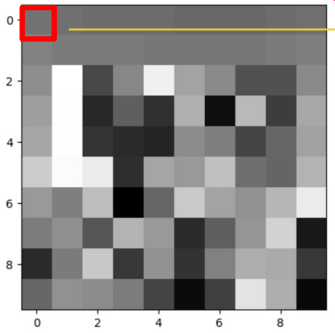

```
[[[114.][111.][107.][107.][107.][104.][104.][104.][107.][111.]]
 [[128.][121.][121.][121.][118.][118.][118.][118.][121.][125.]]
 [[139.][244.][ 74.][132.][230.][159.][135.][ 82.][ 83.][135.]]
 [[153.][243.][ 45.][ 95.][ 53.][170.][ 20.][179.][ 63.][163.]]
 [[161.][243.][ 56.][ 47.][ 42.][137.][125.][ 71.][102.][157.]]
 [[195.][240.][225.][ 51.][160.][149.][185.][110.][101.][174.]]
 [[149.][125.][182.][  7.][102.][194.][158.][143.][176.][225.]]
 [[122.][141.][ 87.][174.][149.][ 46.][ 95.][145.][202.][ 30.]]
 [[ 47.][121.][193.][ 60.][143.][ 55.][127.][168.][165.][ 60.]]
 [[103.][142.][137.][122.][ 71.][ 18.][ 67.][219.][169.][ 16.]]]
```

それぞれに色のレベル（輝度）を0から255の大きさで表した数値が割り当てられる

黒 :0 → 白 :255

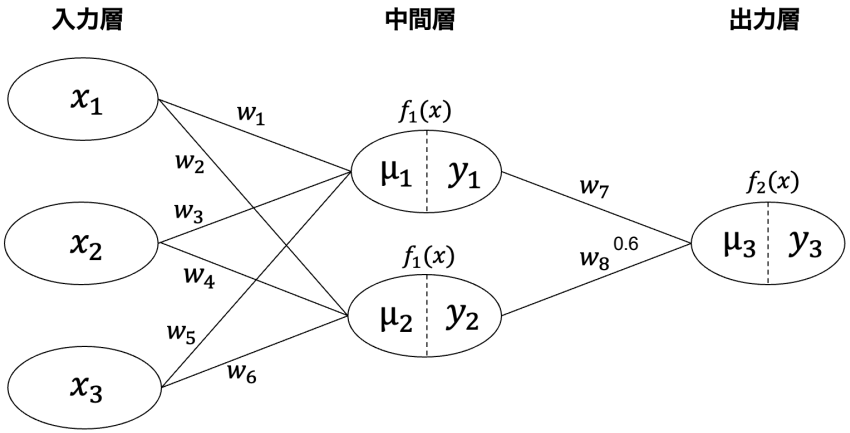
深層学習（画像の分類）

	[ピクセル1の輝度]	[ピクセル2の輝度]	[ピクセル3の輝度]
1枚目	114	111	107
2枚目	224	169	48
3枚目	107	103	107
4枚目	112	157	94
5枚目	・・・	・・・	・・・



test2_img

```
[[[114.][111.][107.][107.][107.][104.][104.][104.][1
 [[128.][121.][121.][121.][118.][118.][118.][118.][1
 [[139.][244.][ 74.][132.][230.][159.][135.][ 82.][ 83.][135.]]
 [[153.][243.][ 45.][ 95.][ 53.][170.][ 20.][179.][ 63.][163.]]
 [[161.][243.][ 56.][ 47.][ 42.][137.][125.][ 71.][102.][157.]]
 [[195.][240.][225.][ 51.][160.][149.][185.][110.][101.][174.]]
 [[149.][125.][182.][ 7.][ 102.][194.][158.][143.][176.][225.]]
 [[122.][141.][ 87.][174.][149.][ 46.][ 95.][145.][202.][ 30.]]
 [[ 47.][121.][193.][ 60.][143.][ 55.][127.][168.][165.][ 60.]]
 [[103.][142.][137.][122.][ 71.][ 18.][ 67.][219.][169.][ 16.]]]
```



	予測
1	0.6
2	0.3
3	0.4
4	0.9
5	0.6

画像データのそれぞれのピクセルの輝度を

特徴量として予測を行う

深層学習 (画像の分類)

今回は画像を
64×64
= 4096ピク
セルで読み込む

入力層
変数の数が入力数

X:0, Y:0 の値
X:1, Y:0 の値
X:2, Y:0 の値
X:3, Y:0 の値

X:64, Y:63 の値
X:64, Y:64 の値

中間層

中間層

出力層

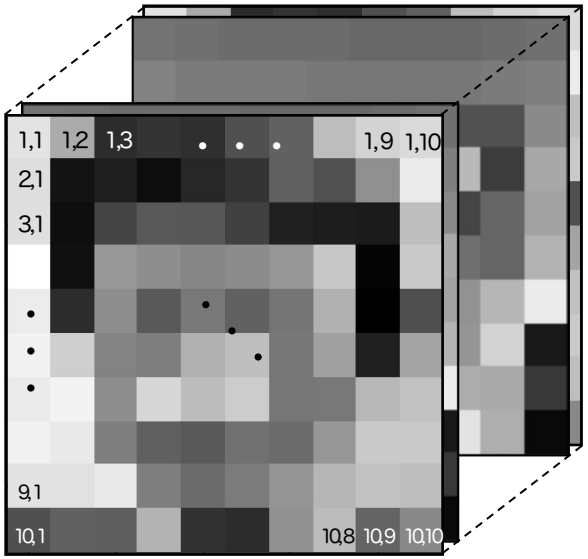
確率

covid19(1)
healthy(0)

入力層には各マスの色情報の数値(＝変数)を入力する

深層学習（画像の分類）

画像データにおける特徴量



(x,y)	(1,1)	(1,2)	(1,3)	・・・	(10,8)	(10,9)	(10,10)
1枚目	36	45	45		68	114	7
2枚目	43	27	78		44	57	80
3枚目	17	98	55		44	53	109
⋮				⋱			⋮
230枚目	36	45	45		68	114	7
231枚目	43	27	78		44	57	80
232枚目	17	98	55		44	53	109

各ピクセルの色を変換した値が特徴量になる

深層学習 (画像の分類)

配布したCOVID-NORMALの中身の構成

深層学習 (画像の分類) のコードまとめ

STEP1　データの用意と前処理

STEP2　学習モデルの選択

　　(モデル名) = Sequential()

モデルはSequential()と
(モデル名).addで自分で設計する

STEP3　データを入れて学習させる

　　(モデル名).fit()

STEP4　学習結果の図示

STEP5　モデルの評価

STEP6　予測 (モデル名).predict()

深層学習 (画像の分類) のコードまとめ

STEP1　データの用意と前処理

STEP2　学習モデルの選択

(モデル名) = Sequential()

STEP3　データを入れて学習させる

(モデル名).fit()

STEP4　学習結果の図示

STEP5　モデルの評価

STEP6　予測 (モデル名).predict()

STEP1-1
画像ファイル名を取得
STEP1-2
(232,64,64,1)の配列をあらかじめ作成
STEP1-3
画像ファイルの読み込みと配列への代入
STEP1-4
x_train(特徴量)とy_train(正解)を作成

深層学習 (画像の分類)

前処理の手順

STEP1-1　画像ファイル名を取得

- HealthyとCovid19のフォルダからファイル名を取得して配列にして変数に代入
- 画像ファイル数が116枚ずつ(HealthyとCovid19)で計232枚であることの確認

STEP1-2　(232,64,64,1)の配列をあらかじめ作成

- 中身が0の配列を作成する (232枚画像データを64×64の白黒で読み込むため)

STEP1-3 画像ファイルの読み込みと配列への代入

- 画像ファイルを順に読み込み**STEP1-2**で作成した配列に代入(Covid19=1, Healthy=0)

STEP1-4 x_train(特徴量)とy_train(正解)を作成

- **STEP1-3**の配列をランダムにシャッフルして、x_train(特徴量データ)とy_train(正解データ)を作成
- x_trainは(232,64,64,1)、y_trainは(232,1)の配列構造をとる

深層学習（画像の分類）

STEP0：Google Colaboratoryの立ち上げ

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

Python基礎　プログラミング基礎

検索google colab　Colaboratory へようこそ - Colaboratory - Google

演習18コード.ipynb

検索google colab　　Colaboratory へようこそ - Colaboratory - Google

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP0：ライブラリのインポート

コード18-1 ライブラリ・モジュール・関数のインポート

```
import numpy as np
import os
import matplotlib.pyplot as plt
from keras.preprocessing.image import load_img, img_to_array
from keras.models import Sequential
from keras.layers import Dense, Flatten, Dropout
```

ライブラリをインポート：import （ライブラリ名） as 省略形

numpyとos

モジュールをインポート：import （ライブラリ名）.（モジュール名） as 省略形

matplotlib　pyplot　colab

深層学習 (画像の分類)

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP0：ライブラリのインポート

```
import numpy as np
import os
import matplotlib.pyplot as plt
from keras.preprocessing.image import load_img, img_to_array
from keras.models import Sequential
from keras.layers import Dense, Flatten, Dropout
```

画像で深層学習を行うときにkerasの関数、クラスを使うのでインポートする

関数をインポート　`from keras.preprocessing.image import load_img`

ライブラリ　モジュール　モジュール　関数

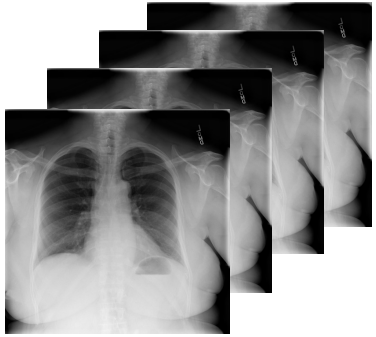
colab

深層学習 (画像の分類)

画像データを深層学習に利用するには、データの前処理が必要

STEP1-1
画像ファイル名を取得
STEP1-2
(232,64,64,1)の配列をあらかじめ作成
STEP1-3
画像ファイルの読み込みと配列への代入
STEP1-4
x_train(特徴量)とy_train(正解)を作成

たくさんの画像データ



x_train
(232, 64, 64, 1)

[[[[0.03137255], [0.00784314], … [[0.16862746], [0.10588235],
…, …,
[0.00784314], [0.00392157]], [0.08627451], [0.14117648]]]]

y_train
(232, 1)

[[1], … [1]]

深層学習 (画像の分類)

前処理の手順

STEP1-1　画像ファイル名を取得

- HealthyとCovid19のフォルダからファイル名を取得して配列にして変数に代入
- 画像ファイル数が116枚ずつ(HealthyとCovid19)で計232枚であることの確認

STEP1-2　(232,64,64,1)の配列をあらかじめ作成

- 中身が0の配列を作成する (232枚画像データを64×64の白黒で読み込むため)

STEP1-3 画像ファイルの読み込みと配列への代入

- 画像ファイルを順に読み込み**STEP1-2**で作成した配列に代入(Covid19=1, Healthy=0)

STEP1-4 x_train(特徴量)とy_train(正解)を作成

- **STEP1-3**の配列をランダムにシャッフルして、x_train(特徴量データ)とy_train(正解データ)を作成
- x_trainは(232,64,64,1)、y_trainは(232,1)の配列構造をとる

深層学習（画像の分類）

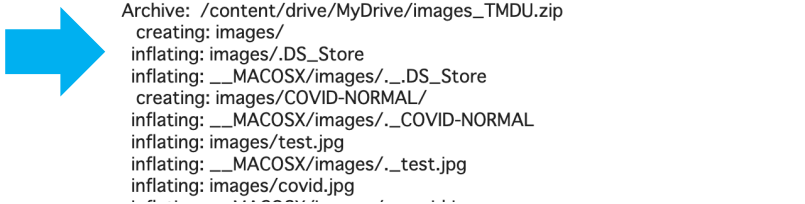
STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-1 画像ファイル名を取得

演習17で行っている人はやらなくて良い

コード18-2 zipファイルを解凍する

```
#!unzip '/content/drive/MyDrive/images_TMDU.zip'
```



```
Archive:  /content/drive/MyDrive/images_TMDU.zip
  creating: images/
 inflating: images/.DS_Store
 inflating: __MACOSX/images/._.DS_Store
  creating: images/COVID-NORMAL/
 inflating: __MACOSX/images/._COVID-NORMAL
 inflating: images/test.jpg
 inflating: __MACOSX/images/._test.jpg
 inflating: images/covid.jpg
```

!unzip 'zipファイル名' で指定したファイルを解凍する

- zipファイルにはファイル名だけでなく、ファイルの場所（パス；path）も指定する必要があるので、「'/content/drive/MyDrive/images_TMDU.zip'」となる

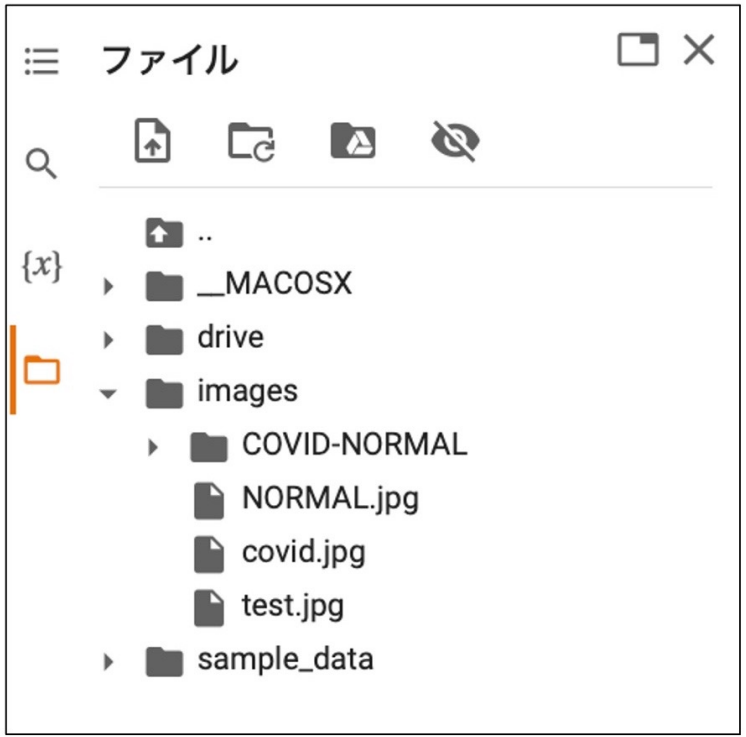
＊images_TMDU.zipをMy driveではないフォルダにアップしている人はパスが異なる

colab

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-1 画像ファイル名を取得



正常に動作すると、「images」フォルダの中に「COVID-NORMAL」というフォルダと3つのjpgファイル（「NORMAL.jpg」，「covid.jpg」，「test.jpg」）が入っていることが確認できる

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-1 画像ファイル名を取得

コード18-3 ファイル名のリストを作成

```
list_healthy = [i for i in os.listdir('/content/images/COVID-
               NORMAL/healthy') if not i.startswith('.')]

list_covid19 = [i for i in os.listdir('/content/images/COVID-
               NORMAL/covid19') if not i.startswith('.')]
```

colab

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-1 画像ファイル名を取得

```
list_healthy = [i for i in os.listdir('/content/images/COVID-NORMAL/healthy') if not i.startswith('.')]

list_covid19 = [i for i in os.listdir('/content/images/COVID-NORMAL/covid19') if not i.startswith('.')]

print(len(list_healthy))
print(len(list_covid19))
```

リスト内包表記という書き方

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-1 画像ファイル名を取得

コード18-4 リスト内包表記の例1

```
example1 = [i for i in range(5)]
print(example1)
```

➡ [0, 1, 2, 3, 4]

range()関数：range(a)で0以上a未満の連続した数値のリストを作成

➡ [0,1,2,3,4]

[i for i in range(5)]で0から4までの数字のデータを順にiに代入して[0,1,2,3,4]というリストを作成

colab

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-1 画像ファイル名を取得

コード18-5 リスト内包表記の例2

```
temp = [1,3,5,7,9]
example2 = [i*2 for i in temp]
print(example2)
```

```
[2, 6, 10, 14, 18]
```

- tempというリストから、リスト内包表記[i×2 for i in temp]でexample2作成
- 「tempを順にiに代入してi×2のリストを作成せよ」という指示
- tempが[1,3,5,7,9]なので順に1×2、3×2、5×2、7×2、9×2が要素のリストが作られる

colab

STEP1-1 画像ファイル名を取得

コード18-6 リスト内包表記の例3

```
temp = [1,3,5,7,9]
example3 = [i*2 for i in temp if i > 4]
print(example3)
```

➡ [10, 14, 18]

- if i > 4 で 「iが4より大きい時」と条件指定
- 「iが4より大きい時、tempを順にiに代入してi×2のリストを作成せよ」という指示
- tempが[1,3,5,7,9]で4より大きい[5,7,9]で順に5×2、7×2、9×2が要素のリストが作られる

colab

STEP1-1 画像ファイル名を取得

コード18-7 リスト内包表記の例4 (文字列)

```
temp2 = ['須藤', '佐藤', '加藤', '佐川']
example4 = [name for name in temp2 if name.startswith('佐')]
print(example4)
```

➡ ['佐藤', '佐川']

- if name.startswith('佐')で、「文字列が'佐'から始まる時」という意味
- 「nameの文字列が'佐'から始まる時、temp2を順にnameに代入してリストを作成せよ」という指示

colab

深層学習（画像の分類）

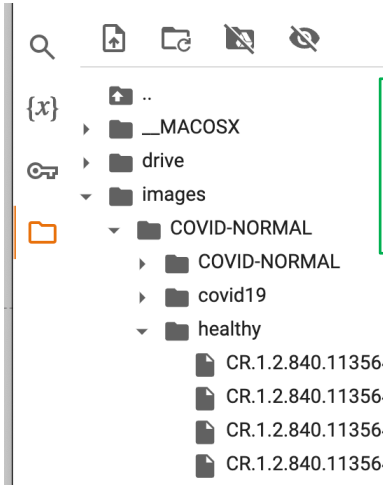
STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-1 画像ファイル名を取得

コード18-3の説明に戻ると…

```
list_healthy = [i for i in os.listdir('/content/images/COVID-
                NORMAL/healthy') if not i.startswith('.')]
```

os.listdir(*path*)：pathに存在するディレクトリ・ファイル名の一覧をリストで取得




```
['DX.1.2.840.113564.1722810162.20200403113227763540.1203801020003.jpg',
 'DX.1.2.840.113564.1722810162.20200414101717389720.1203801020003.jpg',
 'CR.1.2.840.113564.1722810170.20200321174111187570.1003000225002.jpg',
 'DX.1.2.840.113564.1722810162.20200403105318208480.1203801020003.jpg',…
```

このファル名をリストで取得する

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-1 画像ファイル名を取得

コード18-3の説明に戻ると…

```
list_healthy = [i for i in os.listdir('/content/images/COVID-
                NORMAL/healthy') if not i.startswith('.')]
```

os.listdir(*path*)：pathに存在するディレクトリ・ファイル名の一覧をリストで取得

if not（条件式）：条件式**ではない**時にその前の指示を実行する
条件式：iが'.'から始まる文字列の時

colab

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-1 画像ファイル名を取得

コード18-3の説明に戻ると…

```
list_healthy = [i for i in os.listdir('/content/images/COVID-
                NORMAL/healthy') if not i.startswith('.')]
```

os.listdir(*path*)：pathに存在するディレクトリ・ファイル名の一覧をリストで取得

if not（条件式）：条件式**ではない**時にその前の指示を実行する
条件式：iが'.'から始まる文字列の時

「ファイル名がドット「.」から始まるファイル（隠しファイル）を除外して（'/content/images/COVID-NORMAL/healthy'）**ディレクトリ**にあるファイル名を取得し、list_healthyに代入する」という指示

深層学習 (画像の分類)

前処理の手順

STEP1-1　画像ファイル名を取得

- HealthyとCovid19のフォルダからファイル名を取得して配列にして変数に代入
- **画像ファイル数が116枚ずつ(HealthyとCovid19)で計232枚であることの確認**

STEP1-2　(232,64,64,1)の配列をあらかじめ作成

- 中身が0の配列を作成する (232枚画像データを64×64の白黒で読み込むため)

STEP1-3 画像ファイルの読み込みと配列への代入

- 画像ファイルを順に読み込み**STEP1-2**で作成した配列に代入(Covid19=1, Healthy=0)

STEP1-4 x_train(特徴量)とy_train(正解)を作成

- **STEP1-3**の配列をランダムにシャッフルして、x_train(特徴量データ)とy_train(正解データ)を作成
- x_trainは(232,64,64,1)、y_trainは(232,1)の配列構造をとる

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-1 画像ファイル名を取得

コード18-8 ファイル名リストの要素数確認

```
print(len(list_healthy))
print(len(list_covid19))
```

116
116

- len()関数で要素の数を取得

116のファイル名があるということ
→画像ファイル116個ずつ

colab

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-1 画像ファイル名を取得

コード18-9 画像ファイル数の集計

```
num_healthy = len(list_healthy)
num_covid19 = len(list_covid19)
num_all = num_healthy + num_covid19
print(num_all)
```

➡ 232

- num_healthyとnum_covid19はそれぞれlist_healthyとlist_covid19に含まれる画像ファイル名の数
- num_allに合計値（それぞれ116であるため合計232になる）を代入する

colab

深層学習 (画像の分類)

前処理の手順

STEP1-1　画像ファイル名を取得

- HealthyとCovid19のフォルダからファイル名を取得して配列にして変数に代入
- 画像ファイル数が116枚ずつ(HealthyとCovid19)で計232枚であることの確認

STEP1-2　(232,64,64,1)の配列をあらかじめ作成

- **中身が0の配列を作成する (232枚画像データを64×64の白黒で読み込むため)**

STEP1-3 画像ファイルの読み込みと配列への代入

- 画像ファイルを順に読み込み**STEP1-2**で作成した配列に代入(Covid19=1, Healthy=0)

STEP1-4 x_train(特徴量)とy_train(正解)を作成

- **STEP1-3**の配列をランダムにシャッフルして、x_train(特徴量データ)とy_train(正解データ)を作成
- x_trainは(232,64,64,1)、y_trainは(232,1)の配列構造をとる

深層学習 (画像の分類)

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-2 (232,64,64,1)の配列をあらかじめ作成

コード18-10 画像データ格納用numpy配列の作成

```
images_temp = np.zeros((num_all,64,64,1),dtype = float)
labels_temp = np.zeros((num_all, 1), dtype = int)
print(images_temp.shape)
print(labels_temp.shape)
```

```
(232, 64, 64, 1)
(232, 1)
```

- np.zeros()：括弧の中に配列の形状を指定して、要素0からなる配列を作る関数
  1つ目の引数で作成する配列を指定
  2つ目の引数でデータタイプを指定する （float：浮動小数[この後正規化するため]、int：整数）
- image_tempは(232,64,64,1)、labels_tempは(232,1)という配列構造を作成

colab

深層学習 (画像の分類)

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-2 (232,64,64,1)の配列をあらかじめ作成

コード18-11 画像データ格納用numpy配列の値の確認

```
print(labels_temp)
```

```
[[0]
 [0]
 [0]
 [0]
 [0]
```

```
print(images_temp[0])
```

```
[[[0.]
  [0.]
  [0.]
  ...
  [0.]
  [0.]
  [0.]]

 [[0.]
  [0.]
  [0.]
  ...
  [0.]
  [0.]
  [0.]]

 [[0.]
  [0.]
  [0.]
  ...
  [0.]
  [0.]
  [0.]]
```

全ての要素が0となっていることが確認できる

colab

深層学習 (画像の分類)

前処理の手順

STEP1-1　画像ファイル名を取得

- HealthyとCovid19のフォルダからファイル名を取得して配列にして変数に代入
- 画像ファイル数が116枚ずつ(HealthyとCovid19)で計232枚であることの確認

STEP1-2　(232,64,64,1)の配列をあらかじめ作成

- 中身が0の配列を作成する (232枚画像データを64×64の白黒で読み込むため)

STEP1-3 画像ファイルの読み込みと配列への代入

- **画像ファイルを順に読み込みSTEP1-2で作成した配列に代入(Covid19=1, Healthy=0)**

STEP1-4 x_train(特徴量)とy_train(正解)を作成

- **STEP1-3**の配列をランダムにシャッフルして、x_train(特徴量データ)とy_train(正解データ)を作成
- x_trainは(232,64,64,1)、y_trainは(232,1)の配列構造をとる

深層学習 (画像の分類)

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-3 画像ファイルの読み込みと配列への代入

コード18-12 画像データの読み込み

```
path = '/content/images/COVID-NORMAL'
for i in range(num_healthy):
  file = f'{path}/healthy/{list_healthy[i]}'
  file_img = load_img(file, color_mode = 'grayscale',
             target_size = (64,64), interpolation = 'lanczos')
  images_temp[i] = img_to_array(file_img)/255
```

colab

深層学習 (画像の分類)

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-3 画像ファイルの読み込みと配列への代入

```
path = '/content/images/COVID-NORMAL'
for i in range(num_healthy):
  file = f'{path}/healthy/{list_healthy[i]}'
  file_img = load_img(file, color_mode = 'grayscale',
                      target_size = (64,64), interpolation = 'lanczos')
  images_temp[i] = img_to_array(file_img)/255
```

COVID-NORMALまでのフォルダの場所（パス）が長いので、
「path」という変数に文字列データとして代入

深層学習 (画像の分類)

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-3 画像ファイルの読み込みと配列への代入

```
path = '/content/images/COVID-NORMAL'
for i in range(num_healthy):
  file = f'{path}/healthy/{list_healthy[i]}'
  file_img = load_img(file, color_mode = 'grayscale',
                      target_size = (64,64), interpolation = 'lanczos')
  images_temp[i] = img_to_array(file_img)/255
```

健康な肺のX線画像(Healthy)を読み込む

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-3 画像ファイルの読み込みと配列への代入

```
path = '/content/images/COVID-NORMAL'
for i in range(num_healthy):
    file = f'{path}/healthy/{list_healthy[i]}'
    file_img = load_img(file, color_mode = 'grayscale',
                        target_size = (64,64), interpolation = 'lanczos')
    images_temp[i] = img_to_array(file_img)/255
```

0~115（num_healthyが116なので）までを順にiという変数に代入して
forの処理内容を実行

➡ 3～6行目に書かれた内容を116回繰り返す

深層学習 (画像の分類)

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-3 画像ファイルの読み込みと配列への代入

```
path = '/content/images/COVID-NORMAL'
for i in range(num_healthy):
  file = f'{path}/healthy/{list_healthy[i]}'
  file_img = load_img(file, color_mode = 'grayscale',
                      target_size = (64,64), interpolation = 'lanczos')
  images_temp[i] = img_to_array(file_img)/255
```

フォーマット文字リテラル　f'{変数名}'　文字を変数として認識させる

```
f'{path}/healthy/{list_healthy[i]}'
```

深層学習 (画像の分類)

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-3 画像ファイルの読み込みと配列への代入

コード18-13 フォーマット文字リテラルの例

```
name = '石丸'
print(name)
print('name')
print(f'{name}')
print('私の名前はnameです')
print(f'私の名前は{name}です')
```

→ 石丸
→ name
→ 石丸
→ 私の名前はnameです
→ 私の名前は石丸です

文字列として扱うための「' '」の中に変数名を入れて、変数として認識させるためには、
3行目のように「' '」の前にfを入れて、変数を「{ }」で囲う必要があります。

colab

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-3 画像ファイルの読み込みと配列への代入

```
path = '/content/images/COVID-NORMAL'
for i in range(num_healthy):
  file = f'{path}/healthy/{list_healthy[i]}'
  file_img = load_img(file, color_mode = 'grayscale',
                      target_size = (64,64), interpolation = 'lanczos')
  images_temp[i] = img_to_array(file_img)/255
```

フォーマット文字リテラル　f'{変数名}'　文字を変数として認識させる

```
f'{path}/healthy/{list_healthy[i]}'
```

```
list_healthy = [i for i in os.listdir('/content/images/COVID-NORMAL/healthy')
                if not i.startswith('.')]
```

i=0の時：file = /content/images/COVID-NORMAL/healthy/
DX.1.2.840.113564.1722810162.20200403113227763540.1203801020003.jpg

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-3 画像ファイルの読み込みと配列への代入

```
path = '/content/images/COVID-NORMAL'
for i in range(num_healthy):
  file = f'{path}/healthy/{list_healthy[i]}'
  file_img = load_img(file, color_mode = 'grayscale',
                      target_size = (64,64), interpolation = 'lanczos')
  images_temp[i] = img_to_array(file_img)/255
```

load_img()関数：画像ファイルを「file_img」に読み込んでいる

引数で、ファイル名（file)、カラーモード(白黒: grayscale)、画像サイズ(64, 64)、画像サイズを変換する際の補完方法(lanczos)を指定

深層学習 (画像の分類)

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-3 画像ファイルの読み込みと配列への代入

```
path = '/content/images/COVID-NORMAL'
for i in range(num_healthy):
  file = f'{path}/healthy/{list_healthy[i]}'
  file_img = load_img(file, color_mode = 'grayscale',
                      target_size = (64,64), interpolation = 'lanczos')
  images_temp[i] = img_to_array(file_img)/255
```

コード18-9で作った `images_temp`

コード18-10で作ったimages_tempの[i]番目に、file_imgを配列データ(64×64×1)に変換した値/255を代入する

255で割るのは正規化の処理(全ての数値を0~1に変換すること)

(232, 64, 64, 1)の配列をあらかじめ作って画像データの配列を代入していく

images_temp[0] images_temp[1] images_temp[231]

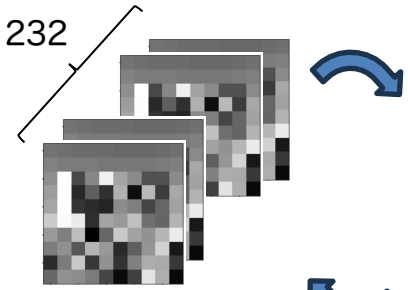
images_temp
(232, 64, 64, 1)

[[[[0][0]...[0]] [[[0][0]...[0]] ... [[[0][0]...[0]]
[[0][0]...[0]] [[0][0]...[0]] [[0][0]...[0]]
... ...
[[0][0]...[0]]] [[0][0]...[0]]] [[0][0]...[0]]]]

labels_temp[0] labels_temp[1] labels_temp[231]

labels_temp
(232, 1)

[[0] [0] ... [0]]



0 から231まで
232枚分繰り返す

画像を読み込み配列に変換

[[[202][103]...[10]]
[[56][46]...[156]]
...
[[5][26]...[234]]]

file_array (64,64,1)

images_temp[0] = file_array
で画像の配列データをimages_tempに代入

[[[0][0]...[0]]
[[0][0]...[0]]
...
[[0][0]...[0]]]

[[[202][103]...[10]]
[[56][46]...[156]]
...
[[5][26]...[234]]]

images_temp[0]

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-3 画像ファイルの読み込みと配列への代入

コード18-14 images_tempの変換後の値

```
images_temp[0]
```

```
array([[[0.01960784],
        [0.02745098],
        [0.03137255],
        ...,
        [0.10588235],
        [0.06666667],
        [0.01960784]],

       [[0.0627451 ],
        [0.10196079],
        [0.17254902],
         ...
        [0.23529412],
        [0.19607843],
        [0.23529412]]])
```

colab

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-3 画像ファイルの読み込みと配列への代入

コード18-15 画像の読み込み 「list_covid19」

```
path = '/content/images/COVID-NORMAL/covid19'
for i in range(num_covid19):
  file = f'{path}/{list_covid19[i]}'
  file_img = load_img(file, color_mode='grayscale',
              target_size = (64,64), interpolation='lanczos')
images_temp[i + num_healthy] = img_to_array(file_img)/255
labels_temp[i + num_healthy] = 1
```

colab

深層学習 (画像の分類)

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-3 画像ファイルの読み込みと配列への代入

```
for i in range(num_covid19):
  file = f'{path}/covid19/{list_covid19[i]}'
  file_img = load_img(file, color_mode='grayscale',
                      target_size = (64,64), interpolation='lanczos')
  images_temp[i + num_healthy] = img_to_array(file_img)/255
  labels_temp[i + num_healthy] = 1
```

Covid19肺炎の画像(covid19)を読み込む

深層学習 (画像の分類)

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-3 画像ファイルの読み込みと配列への代入

```
for i in range(num_covid19):
    file = f'{path}/covid19/{list_covid19[i]}'
    file_img = load_img(file, color_mode='grayscale',
                        target_size = (64,64), interpolation='lanczos')
    images_temp[i + num_healthy] = img_to_array(file_img)/255
    labels_temp[i + num_healthy] = 1
```

0~115(num_covidが116なので)までを順にiという変数に代入してforの処理内容を実行

2～6行目に書かれた内容を116回繰り返す

深層学習 (画像の分類)

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-3 画像ファイルの読み込みと配列への代入

```
for i in range(num_covid19):
  file = f'{path}/covid19/{list_covid19[i]}'
  file_img = load_img(file, color_mode='grayscale',
                      target_size = (64,64), interpolation='lanczos')
  images_temp[i + num_healthy] = img_to_array(file_img)/255
  labels_temp[i + num_healthy] = 1
```

load_img()関数：画像ファイルを「file_img」に読み込んでいる

引数で、ファイル名（file)、カラーモード(白黒: grayscale)、画像サイズ(64, 64)、画像サイズを変換する際の補完方法(lanczos)を指定

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-3 画像ファイルの読み込みと配列への代入

```
for i in range(num_covid19):
  file = f'{path}/covid19/{list_covid19[i]}'
  file_img = load_img(file, color_mode='grayscale',
                      target_size = (64,64), interpolation='lanczos')
  images_temp[i + num_healthy] = img_to_array(file_img)/255
  labels_temp[i + num_healthy] = 1
```

コード18-10で作ったimages_tempの[i+ num_healthy]番目に、file_imgを配列データ(64×64×1)に変換した値/255を代入する

healthyではimages_temp[0]からimages_temp[115]までの配列を置き換えた

covid19ではimages_temp[116]からimages_temp[231] の116枚分の配列データを置き換えるので、image_tempのインデックス番号は「i + num_healthy」

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-3 画像ファイルの読み込みと配列への代入

```
for i in range(num_covid19):
  file = f'{path}/covid19/{list_covid19[i]}'
  file_img = load_img(file, color_mode='grayscale',
                      target_size = (64,64), interpolation='lanczos')
  images_temp[i + num_healthy] = img_to_array(file_img)/255
  labels_temp[i + num_healthy] = 1
```

コード18-10で作ったlabels_tempの[i+ num_healthy]番目に、1を代入する

ラベル（正解値：健康な肺のx線画像を0、Covid19の肺炎のx線画像を1とする）を作成する

- labels_tempは全て0の要素からなる(232,1)形状の配列
- healthyの方は最初から0が入っている
- covid19の方はlabels_temp[i + num_healthy] = 1で、labels_temp[116]からlabel_temp[231]までの0を全て1に置き換える

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-3 画像ファイルの読み込みと配列への代入

コード18-16 images_tempとlabels_tempの変換後の値

```
print(images_temp[116])

print(labels_temp[116])
```

```
[[[0.18431373]
  [0.20784314]
  [0.19215687]
  ...
  [0.09411765]
  [0.09411765]]
 [1]
```

images_temp[116]は、covid19の1番目の画像データ

0から1までの要素からなる(64, 64, 1)形状の配列

labels_temp[116]は[1]となっている

colab

深層学習 (画像の分類)

前処理の手順

STEP1-1　画像ファイル名を取得

- HealthyとCovid19のフォルダからファイル名を取得して配列にして変数に代入
- 画像ファイル数が116枚ずつ(HealthyとCovid19)で計232枚であることの確認

STEP1-2　(232,64,64,1)の配列をあらかじめ作成

- 中身が0の配列を作成する (232枚画像データを64×64の白黒で読み込むため)

STEP1-3 画像ファイルの読み込みと配列への代入

- 画像ファイルを順に読み込みSTEP1-2で作成した配列に代入(Covid19=1, Healthy=0)

STEP1-4 x_train(特徴量)とy_train(正解)を作成

- **STEP1-3の配列をランダムにシャッフルして、x_train(特徴量データ)とy_train(正解データ)を作成**
- **x_trainは(232,64,64,1)、y_trainは(232,1)の配列構造をとる**

深層学習 (画像の分類)

前処理の手順

STEP1-1

- Health
- 画像フ

STEP1-2

- 中身が

STEP1-3

- 画像フ

- この後の学習で配列データを使うが、今は前半が正解０の健康な肺のx線写真、後半が正解1のcovid19肺炎のx線写真の並びになっている
- この後モデルの中でvalidation_splitをすることになるが、train_test_splitと違い、自動ではシャッフルされないため、後ろ側のデータがvalidationデータになる
- validationデータが全て正解値１のデータになると予測ができないので、あらかじめ並び替えておく必要がある

STEP1-4 x_train(特徴量)とy_train(正解)を作成

- STEP1-3の配列をランダムにシャッフルして、x_train(特徴量データ)とy_train(正解データ)を作成
- x_trainは(232,64,64,1)、y_trainは(232,1)の配列構造をとる

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-4 x_train(特徴量)とy_train(正解)を作成

コード18-17 0から231までの数字の配列を作成

```
num_list = np.arange(num_all)
print(num_list)
```

```
[  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19
 . . .
 219 220 221 222 223 224 225 226 227 228 229 230 231]
```

- np.arange()関数：「0~引数に入れた数字」の1次元の配列を作る

「num_all」は232なので、「num_list = np.arange(num_all)」を出力した結果は、0から231までの連続した整数の要素からなる配列になる

colab

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-4 x_train(特徴量)とy_train(正解)を作成

コード18-18 numpy配列をシャッフルする

```
np.random.seed(0)
np.random.shuffle(num_list)
print(num_list)
```

```
[155 159 132 206 145 18 73 55 205 11 222 202 38 83 56 65
 . . .
  10 12 7 185 199 40 210 118 66 169 19 86 229 212 108 84]
```

`np.random.seed(数字)`：ランダムシード値を設定し、毎回同じ乱数発生させる

`np.random.shuffle()`：引数に指定した配列の要素をランダムにシャッフルする
元の配列「`num_list`」自体が変更される

colab

深層学習 (画像の分類)

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価
STEP6：予測

STEP1-4 x_train(特徴量)とy_train(正解)を作成

コード18-19
images_tempとlabels_tempの配列をシャッフル

```
x_train = images_temp[num_list]
y_train = labels_temp[num_list]
print(labels_temp[0:10])
print(y_train[0:10])
print(x_train.shape)
print(y_train.shape)
```

```
[[0]   [[1]
 [0]    [1]
 [0]    [1]
 [0]    [1]
 [0]    [1]
 [0]    [0]
 [0]    [0]
 [0]    [0]
 [0]    [1]
 [0]]   [0]]
(232, 64, 64, 1)
(232, 1)
```

numpy配列では、インデックスにリスト形式で順番を指定すると、その順番で値を並べ替える

コード18-17で作成した0から231までのランダムな数字の列であるnum_listをインデックスにして、 x_trainとy_trainのデータをシャッフルしている

colab

深層学習 (画像の分類)

前処理の手順

STEP1-1　画像ファイル名を取得

- HealthyとCovid19のフォルダからファイル名を取得して配列にして変数に代入
- 画像ファイル数が116枚ずつ(HealthyとCovid19)で計232枚であることの確認

STEP1-2　(232,64,64,1)の配列をあらかじめ作成

- 中身が0の配列を作成する (232枚画像データを64×64の白黒で読み込むため)

STEP1-3 画像ファイルの読み込みと配列への代入

- 画像ファイルを順に読み込みSTEP1-2で作成した配列に代入(Covid19=1, Healthy=0)

STEP1-4 x_train(特徴量)とy_train(正解)を作成

- STEP1-3の配列をランダムにシャッフルして、x_train(特徴量データ)とy_train(正解データ)を作成
- **x_trainは(232,64,64,1)、y_trainは(232,1)の配列構造をとる**

深層学習 (画像の分類)

データの確認

x_train

```
[[[[0.16862746],      [[0.02745098],
  [0.1882353 ],       [0.18431373],
  [0.2 ],             [0.47843137],
  ...,                ...,
  [0.03529412],       [0.81960785],
  [0.01568628],       [0.80000001],
  [0.01176471]]]      [0.78431374]]]]
```

(232, 64, 64, 1)

y_train

```
array([[1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [1],
       [0],
       [1],
       [1],
       [1],
       [0],
```

(232, 1)

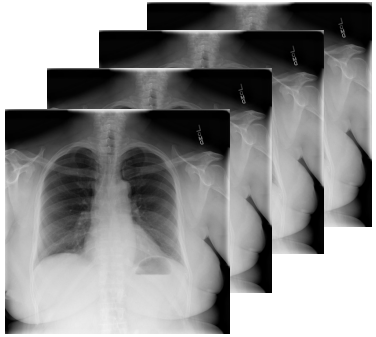
肺炎が1、健康が0

深層学習 (画像の分類)

画像データを深層学習に利用するには、データの前処理が必要

STEP1-1
画像ファイル名を取得
STEP1-2
(232,64,64,1)の配列をあらかじめ作成
STEP1-3
画像ファイルの読み込みと配列への代入
STEP1-4
x_train(特徴量)とy_train(正解)を作成

たくさんの画像データ



x_train
(232, 64, 64, 1)

[[[[0.03137255], [0.00784314], … [[0.16862746], [0.10588235],
…, …,
[0.00784314], [0.00392157]], [0.08627451], [0.14117648]]]]

y_train
(232, 1)

[[1], … [1]]

深層学習 (画像の分類) のコードまとめ

STEP1　データの用意と前処理

STEP2　学習モデルの選択

```
(モデル名) = Sequential()
```

STEP3　データを入れて学習させる

```
(モデル名).fit()
```

STEP4　学習結果の図示

STEP5　モデルの評価

STEP1-1
画像ファイル名を取得
STEP1-2
(232,64,64,1)の配列をあらかじめ作成
STEP1-3
画像ファイルの読み込みと配列への代入
STEP1-4
x_train(特徴量)とy_train(正解)を作成

モデルはSequential()と
(モデル名).addで自分で設計する

演習18：課題

Webclassで課題を提出してください。締め切りは2024/02/15 23:59まで

画像データの前処理について正しい選択肢を選んでください

問1：STEP1-3で `img_to_array(file_img)/255` で255で割る作業を何ですか？

1. 正規化
2. 標準化
3. 量子化
4. 二値化

問2：STEP1-4で、STEP1-3で作成した配列をシャッフルしてから`x_train`と`y_train`を作成する理由はなんですか？

1. 過学習を抑制するため
2. 学習の速度を上げるため
3. データセットの偏りを防ぐため
4. 計算負荷を軽減させるため

授業準備：Webclassからコードをダウンロードし、Google colaboratoryで開いておいてください

演習授業中の質問対応について

医療とAI・ビッグデータ入門
演習19

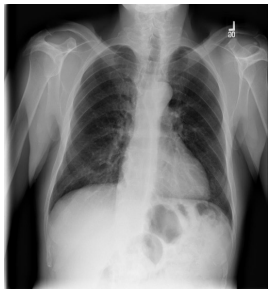
深層学習（画像データ）3

演習15-20の概要

- 今までは教師あり機械学習の基礎を実行してきた
- 演習15-20では深層学習を実行する

15-16で深層学習の基礎と乳がんデータの分類
17-19で画像の分類を深層学習で行う
20　機械学習・深層学習の演習

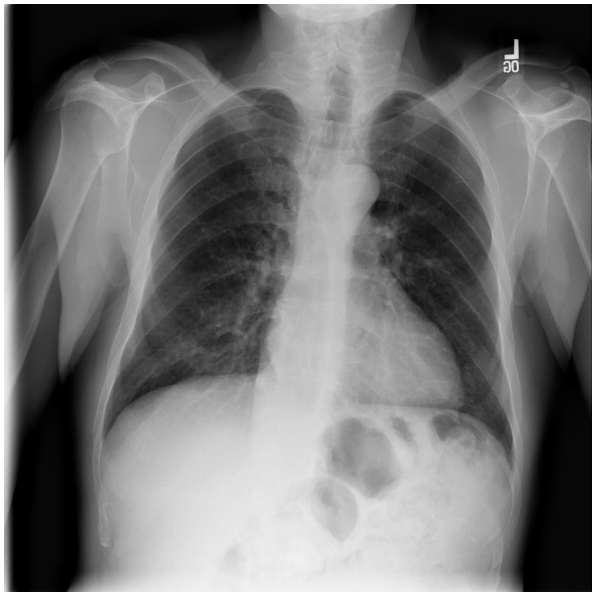
医療分野のAIとして、画像診断支援が非常に重要なトピック
今回から3回で肺のレントゲン画像を用いて、covid19肺炎かどうかを分類する深層学習を行う



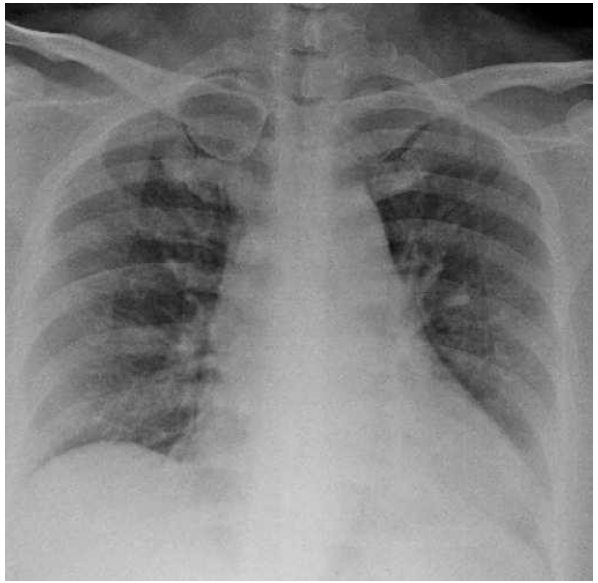
深層学習 (画像の分類)

肺のレントゲン画像で深層学習を行いCovid19肺炎かどうかを予測する

健康

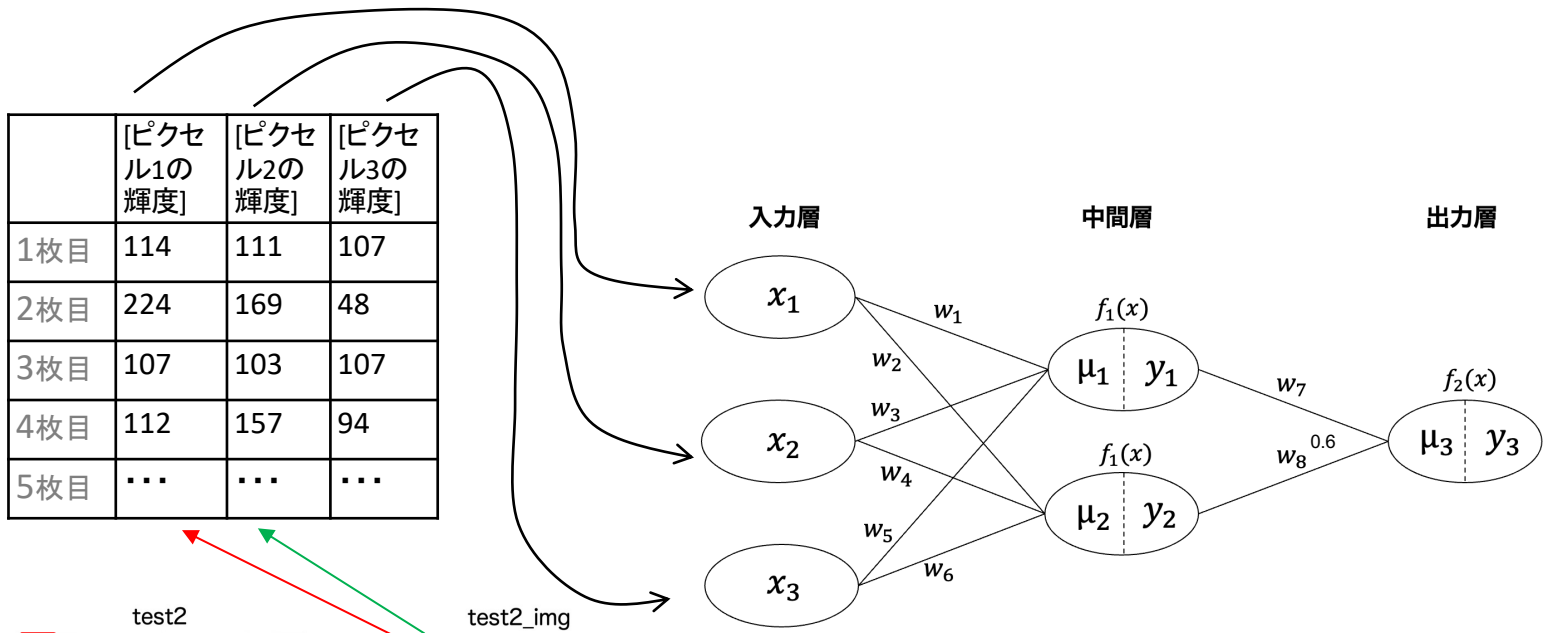


Covid19肺炎

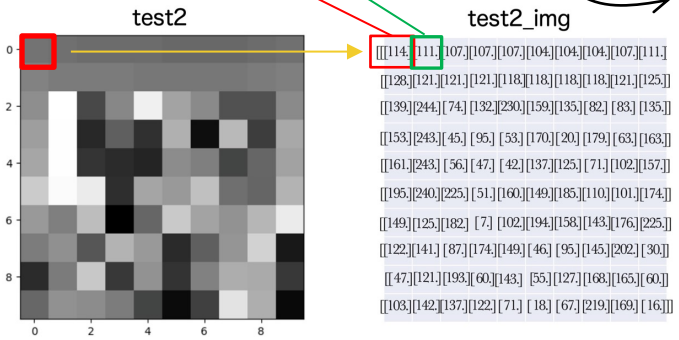


深層学習（画像の分類）

	[ピクセル1の輝度]	[ピクセル2の輝度]	[ピクセル3の輝度]
1枚目	114	111	107
2枚目	224	169	48
3枚目	107	103	107
4枚目	112	157	94
5枚目	・・・	・・・	・・・



	予測
1	0.6
2	0.3
3	0.4
4	0.9
5	0.6



画像データのそれぞれのピクセルの輝度を

特徴量として予測を行う

深層学習 (画像の分類)のコードまとめ

STEP1　データの用意と前処理

STEP2　学習モデルの選択

```
(モデル名) = Sequential()
```

STEP3　データを入れて学習させる

```
(モデル名).fit()
```

STEP4　モデルの評価

STEP5　予測 (モデル名).predict()

モデルはSequential()と
(モデル名).addで自分で設計する

深層学習のコードの流れ

STEP1:データ分割

x(特徴量データ)　y(正解データ)

x_train　y_train

x_test　y_test

STEP2:学習モデルの選定

深層学習ではニューラルネットワークを学習モデルを自分で設計する

モデル

学習済みモデル

STEP3:学習

正解の予測

1：covid19肺炎
0：健康

正解との比較・評価

STEP4:モデルの評価

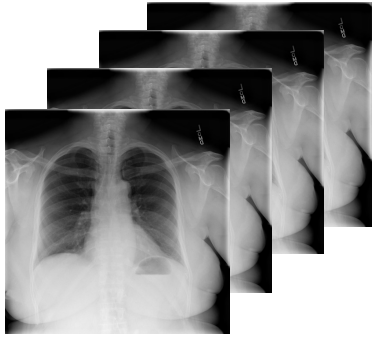
STEP5:予測

深層学習 (画像の分類)

画像データを深層学習に利用するには、データの前処理が必要

STEP1-1
画像ファイル名を取得
STEP1-2
(232,64,64,1)の配列をあらかじめ作成
STEP1-3
画像ファイルの読み込みと配列への代入
STEP1-4
x_train(特徴量)とy_train(正解)を作成

たくさんの画像データ



x_train
(232, 64, 64, 1)

[[[[0.03137255], [0.00784314], … [[0.16862746], [0.10588235],
…, …,
[0.00784314], [0.00392157]], [0.08627451], [0.14117648]]]]

y_train
(232, 1)

[[1], … [1]]

深層学習 (画像の分類)

今回は画像を
64×64
= 4096ピク
セルで読み込む

入力層
変数の数が入力数

X:0, Y:0 の値
X:1, Y:0 の値
X:2, Y:0 の値
X:3, Y:0 の値

X:64, Y:63 の値
X:64, Y:64 の値

中間層

中間層

出力層

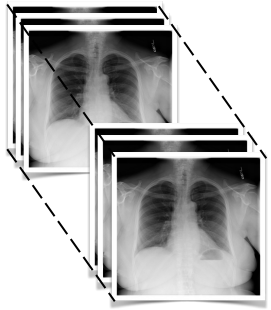
確率

covid19(1)
healthy(0)

入力層には各マスの色情報の数値(＝変数)を入力する

演習19で作成する学習モデル

中間層は3層



(232, 64, 64, 1)

(X:0, Y:0)
(X:1, Y:0)
(X:2, Y:0)
(X:3, Y:0)
⋮
(X:64, Y:63)
(X:64, Y:64)

covid19(1)
healthy(0)

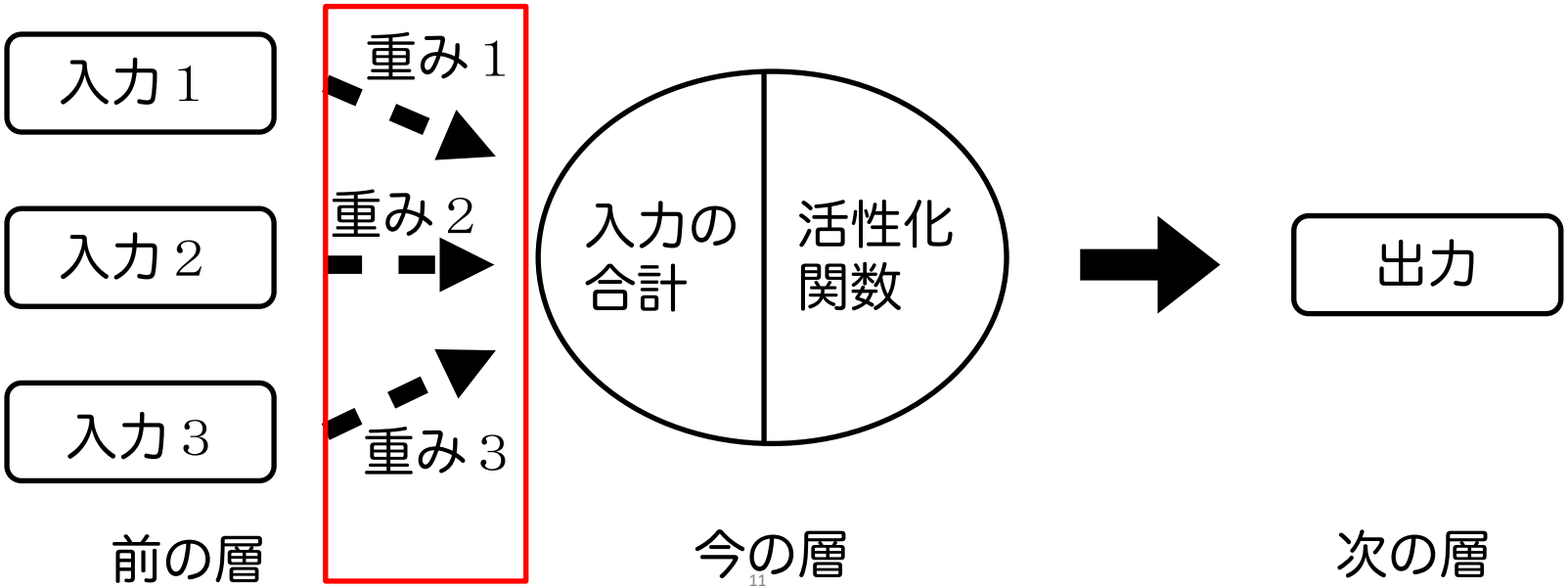
	入力データ	1層目	2層目	3層目	出力層
(ニューロンの数)	4096個	512個	256個	128個	1個

それぞれの層のニューロン数を指定

ニューラルネットワークとは

深層学習は、

①一定量のデータの予測結果を算出する
②正解と予測結果がどれくらい異なっているかという誤差を計算する
③誤差が小さくなるように重みとバイアスを変える
を何度も繰り返すことで、誤差を減らしていき精度を高める



深層学習 (画像の分類)

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP1：前回からの続き

前回の続きのため、
コード19-1~コード19-10を実行してください

colab

深層学習 (画像の分類)

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP1：前回からの続き

コード19-11 ランダムシード値を設定

```
set_random_seed(0)

tf.config.experimental.enable_op_determinism()
```

- この後の機械学習で使うkerasモデルのランダムシード値を設定する
- モデルを決定的（同じように動くように規定）にする

colab

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP2：学習モデルの選択

コード19-12 モデルの設計

```
model = Sequential()
model.add(Flatten(input_shape=(64,64,1)))
model.add(Dense(512, activation ='relu'))
model.add(Dense(256, activation ='relu'))
model.add(Dense(128, activation ='relu'))
model.add(Dense(1, activation ='sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer='Adam',
              metrics =['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 4096)              0
dense (Dense)                (None, 512)               2097664
dense_1 (Dense)              (None, 256)               131328
dense_2 (Dense)              (None, 128)               32896
dense_3 (Dense)              (None, 1)                 129
=================================================================
Total params: 2262017 (8.63 MB)
Trainable params: 2262017 (8.63 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP2：学習モデルの選択

結果は後で説明

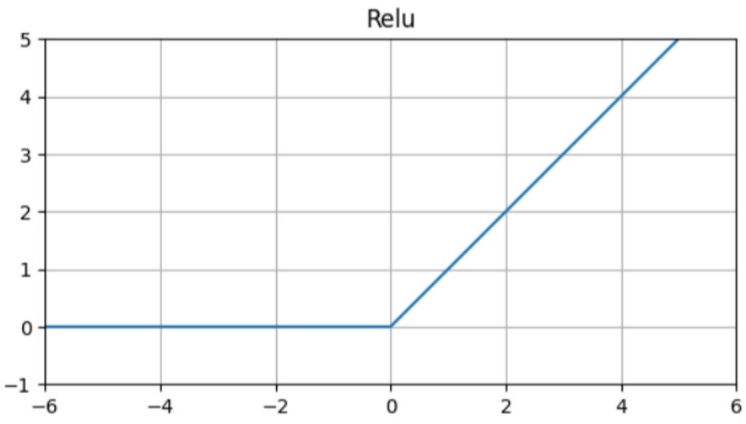
Model: "sequential"

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 4096)	0
dense (Dense)	(None, 512)	2097664
dense_1 (Dense)	(None, 256)	131328
dense_2 (Dense)	(None, 128)	32896
dense_3 (Dense)	(None, 1)	129

Total params: 2262017 (8.63 MB)
Trainable params: 2262017 (8.63 MB)
Non-trainable params: 0 (0.00 Byte)

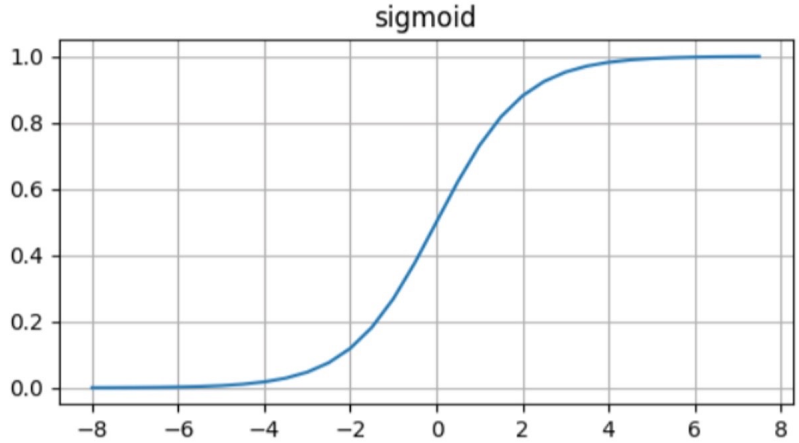
活性化関数

(a) ReLU関数



$y = x\ (x > 0)$
$y = 0\ (x \leqq 0)$

(b) sigmoid関数



$$y = \frac{1}{1 + e^{-x}}$$

深層学習 (画像の分類)

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP2：学習モデルの選択

```
model = Sequential()
model.add(Flatten(input_shape=(64,64,1)))
model.add(Dense(512, activation ='relu'))
model.add(Dense(256, activation ='relu'))
model.add(Dense(128, activation ='relu'))
model.add(Dense(1, activation =sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer='Adam',
              metrics =['accuracy'])
model.summary()
```

深層学習 (画像の分類)

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP2：学習モデルの選択

```
model = Sequential()
model.add(Flatten(input_shape=(64,64,1)))
model.add(Dense(512, activation ='relu'))
model.add(Dense(256, activation ='relu'))
model.add(Dense(128, activation ='relu'))
model.add(Dense(1, activation =sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer='Adam',
              metrics =['accuracy'])
model.summary()
```

- **最初に Sequentialクラスでmodelインスタンスを作成する**

  (*LinearRegressionやRandomForestClassifierなどのモデルと同じ)

この後ニューラルネットワークを入力層から順番に設計できるようになる

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP2：学習モデルの選択

```
model = Sequential()
model.add(Flatten(input_shape=(64,64,1)))
model.add(Dense(512, activation ='relu'))
model.add(Dense(256, activation ='relu'))
model.add(Dense(128, activation ='relu'))
model.add(Dense(1, activation =sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer='Adam',
              metrics =['accuracy'])
model.summary()
```

- （モデル名）.add()で中間層の設定を行う

(64,64,1)→(4096,)

Flatten(input_shape=(多次元配列))

*Flatten()：多次元の配列を1次元に変換する関数

今回は64×64×1=4096の1次元配列に変換

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP2：学習モデルの選択

```
model = Sequential()
model.add(Flatten(input_shape=(64,64,1)))
model.add(Dense(512, activation ='relu'))
model.add(Dense(256, activation ='relu'))
model.add(Dense(128, activation ='relu'))
model.add(Dense(1, activation =sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer='Adam',
              metrics =['accuracy'])
model.summary()
```

- （モデル名）.add()で1層目の中間層の設定を行う

  Dense（次の層のニューロンの数，input_shape=（入力するニューロンの数，），
  activation = 活性化関数）

  前の行で指定しているので省略可能

  *Denceは「全結合」（前のニューロンと後ろのニューロンを全て接続する）

深層学習（画像の分類）

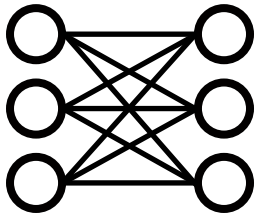
STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP2：学習モデルの選択

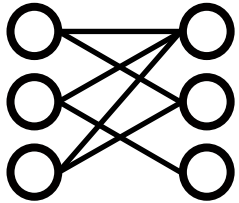
```
model = Sequential()
model.add(Flatten(input_shape=(64,64,1)))
model.add(Dense(512, activation ='relu'))
model.add(Dense(256, activation ='relu'))
model.add(Dense(128, activation ='relu'))
model.add(Dense(1, activation =sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer='Adam',
              metrics =['accuracy'])
model.summary()
```

Denseは全ての入力が全てのニューロンと結合している状態(=全結合層)

全結合層



非全結合層



- **（モデル名）.add()で1層目の中間層の設定を行う**

  Dense(次の層のニューロンの数, input_shape=(入力するニューロンの数,),

  activation = 活性化関数)　　**前の行で指定しているので省略可能**

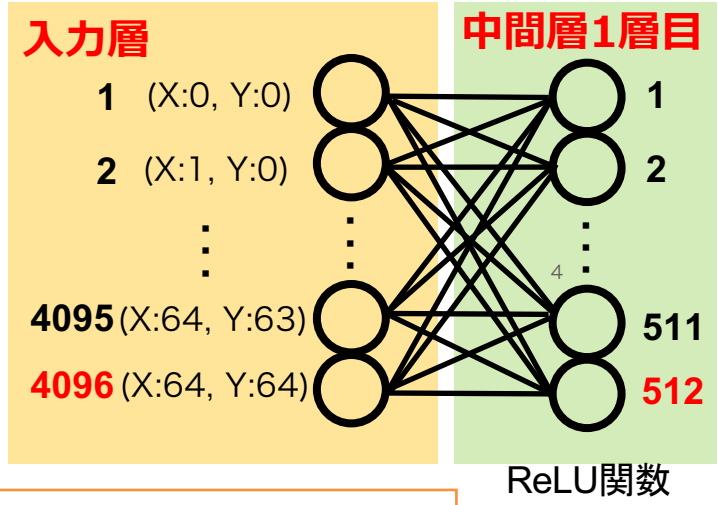
  ***Denceは「全結合」（前のニューロンと後ろのニューロンを全て接続する）**

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP2：学習モデルの選択

```
model = Sequential()
model.add(Flatten(input_shape=(64,64,1)))
model.add(Dense(512, activation ='relu'))
model.add(Dense(256, activation ='relu'))
model.add(Dense(128, activation ='relu'))
model.add(Dense(1, activation =sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer='Adam',
              metrics =['accuracy'])
model.summary()
```



ReLU関数

- （モデル名）.add()で1層目の中間層の設定を行う

  Dense(次の層のニューロン数, input_shape=(入力するニューロン数,),
  activation = 活性化関数)　　前の行で指定しているので省略可能

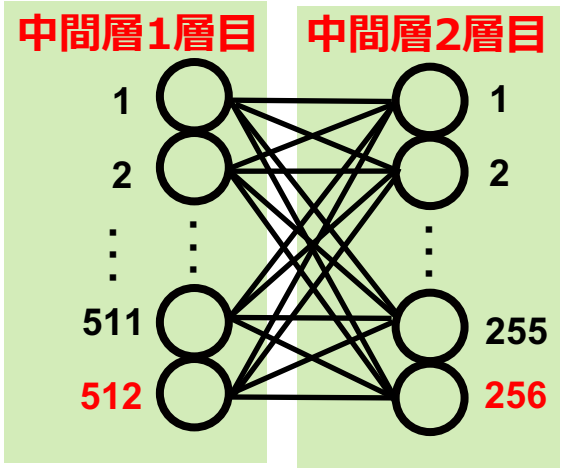
  *Denceは「全結合」（前のニューロンと後ろのニューロンを全て接続する）

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP2：学習モデルの選択

```
model = Sequential()
model.add(Flatten(input_shape=(64,64,1)))
model.add(Dense(512, activation ='relu'))
model.add(Dense(256, activation ='relu'))
model.add(Dense(128, activation ='relu'))
model.add(Dense(1, activation =sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer='Adam',
              metrics =['accuracy'])
model.summary()
```



ReLU関数

- （モデル名）.add()で2層目の中間層の設定を行う

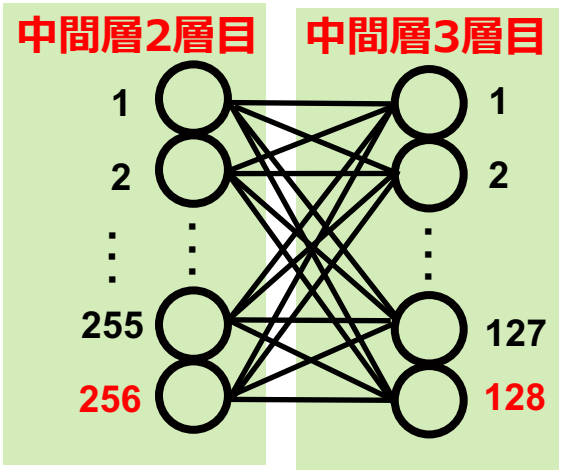
  Dense(次の層のニューロンの数, activation = 活性化関数)

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP2：学習モデルの選択

```
model = Sequential()
model.add(Flatten(input_shape=(64,64,1)))
model.add(Dense(512, activation ='relu'))
model.add(Dense(256, activation ='relu'))
model.add(Dense(128, activation ='relu'))
model.add(Dense(1, activation =sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer='Adam',
              metrics =['accuracy'])
model.summary()
```



ReLU関数

- （モデル名）.add()で3層目の中間層の設定を行う

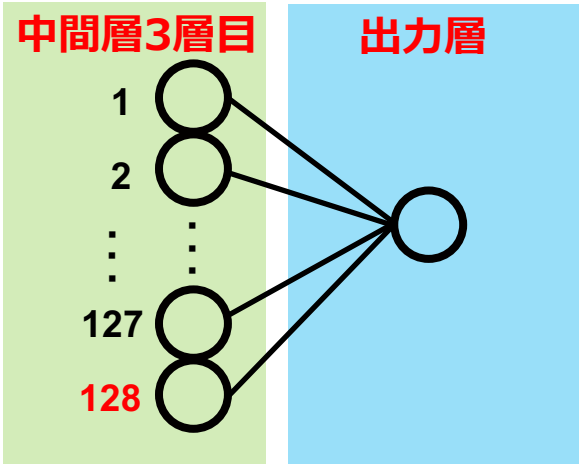
  Dense(次の層のニューロンの数, activation = 活性化関数)

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP2：学習モデルの選択

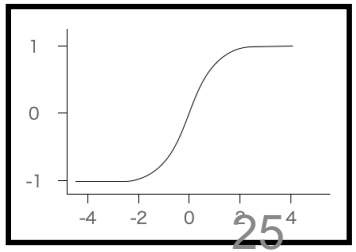
```
model = Sequential()
model.add(Flatten(input_shape=(64,64,1)))
model.add(Dense(512, activation ='relu'))
model.add(Dense(256, activation ='relu'))
model.add(Dense(128, activation ='relu'))
model.add(Dense(1, activation =sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer='Adam',
              metrics =['accuracy'])
model.summary()
```



- 次に（モデル名）.add()で出力層の設定を行う

  Dense(次の層のニューロンの数, activation=活性化関数)

Sigmoid関数

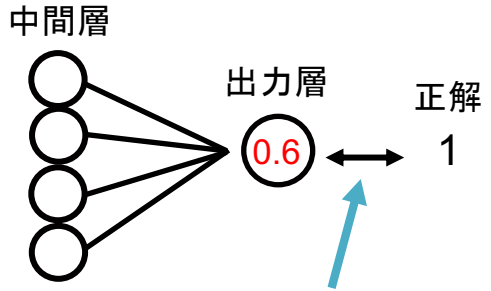


深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP2：学習モデルの選択

```
model = Sequential()
model.add(Flatten(input_shape=(64,64,1)))
model.add(Dense(512, activation ='relu'))
model.add(Dense(256, activation ='relu'))
model.add(Dense(128, activation ='relu'))
model.add(Dense(1, activation =sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer='Adam',
              metrics =['accuracy'])
model.summary()
```



この誤差を0に近づけるように、
誤差逆伝播を行い各パラメーターを調整
(今回は'Adam'を最適化アルゴリズムを使用)

- **学習の仕方を指定**

  引数loss：損失関数を「２値交差エントロピー」に指定
  予測と正解の誤差を計算

  引数optimizer：最適化関数を「Adam」に指定
  重みとバイアスを更新

  引数metrics：評価指標として正解率を示すaccuracyを指定

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP2：学習モデルの選択

```
model = Sequential()
model.add(Flatten(input_shape=(64,64,1)))
model.add(Dense(512, activation ='relu'))
model.add(Dense(256, activation ='relu'))
model.add(Dense(128, activation ='relu'))
model.add(Dense(1, activation =sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer='Adam',
              metrics =['accuracy']
model.summary()
```

- 構築したモデルのまとめが出力される

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 4096)              0

dense (Dense)                (None, 512)               2097664

dense_1 (Dense)              (None, 256)               131328

dense_2 (Dense)              (None, 128)               32896

dense_3 (Dense)              (None, 1)                 129

=================================================================
Total params: 2262017 (8.63 MB)
Trainable params: 2262017 (8.63 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

深層学習（画像の分類）

STEP2：学習モデルの選択

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

Model: "sequential"

Layer (type)	Output Shape	Param #
flatten (Flatten)	(None, 4096)	0
dense (Dense)	(None, 512)	2097664
dense_1 (Dense)	(None, 256)	131328
dense_2 (Dense)	(None, 128)	32896
dense_3 (Dense)	(None, 1)	129

Total params: 2262017 (8.63 MB)
Trainable params: 2262017 (8.63 MB)
Non-trainable params: 0 (0.00 Byte)

64×64×1 = 4096個の1次元配列

中間層の設定：
(input4096+バイアス1)×512個
= 2,097,664個のパラメータ

2,097,664 + 131,328 + 32,896+129
=2,262,017個のパラメータ

colab

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP3：データを入れて学習

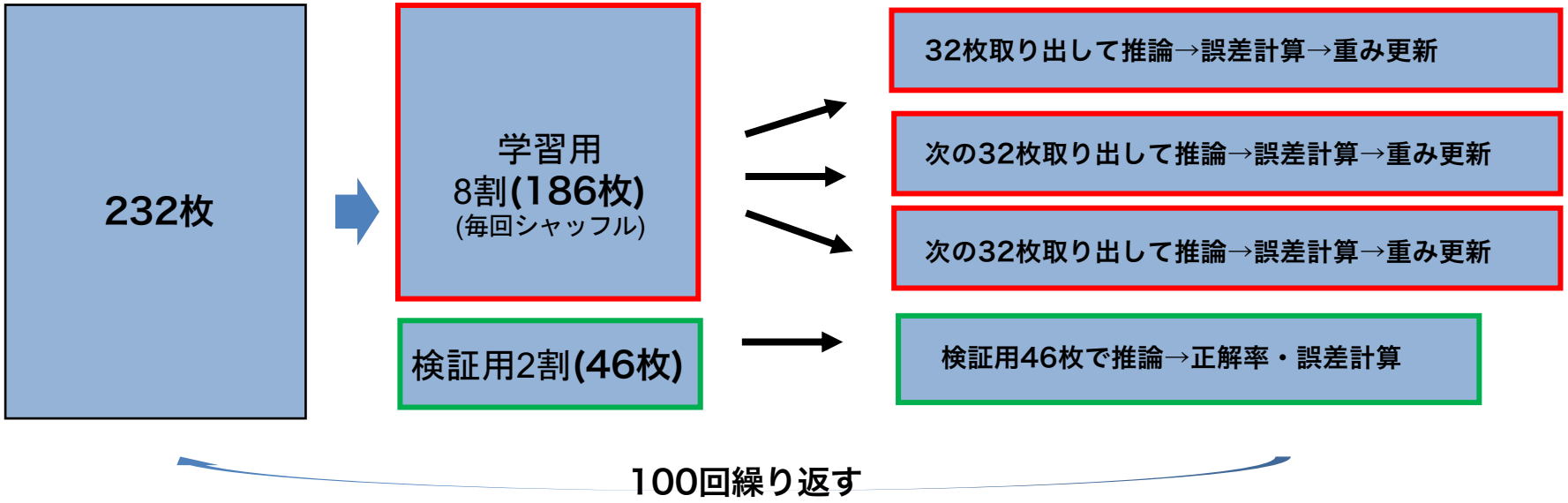
コード19-13 学習させる

```
result = model.fit(x_train, y_train,
                   batch_size = 32,
                   epochs = 100,
                   validation_split = 0.2)
```

- （モデル名）.fit(x, y)で学習させ、resultに学習結果を入れる
- 引数batch_size=32で「32組ずつデータを取り出して損失を計算し、重みとバイアスを更新しなさい」という指示
- 引数epochs=でエポック数を指定する(全てのデータを使い尽くすことを1エポック)
- 引数validatin_split=で任意の割合で分割して学習を行う（*学習用のデータは232組あり186組を学習用、46枚を検証用に分割して学習を行う)

深層学習（画像の分類）

- `batch_size = 32`：学習用のデータを用いて、32枚ずつデータを抽出して学習を行う
- `epochs =100`：全体のデータを100回使って学習を行う
- `validation_split = 0.2`：学習データの2割を検証用に分割して評価する



深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP3：データを入れて学習

```
result = model.fit(x_train, y_train,
                   batch_size = 32,
                   epochs = 100,
                   validation_split = 0.2)
```

```
Epoch 1/100
6/6 [==============================] - 2s 91ms/step - loss: 1.2146 - accuracy: 0.5459 - val_loss: 0.7125 - val_accuracy: 0.4468
Epoch 2/100
6/6 [==============================] - 0s 29ms/step - loss: 0.6743 - accuracy: 0.5459 - val_loss: 0.8089 - val_accuracy: 0.5532
Epoch 3/100
6/6 [==============================] - 0s 21ms/step - loss: 0.7427 - accuracy: 0.6324 - val_loss: 0.9195 - val_accuracy: 0.5532
Epoch 4/100
6/6 [==============================] - 0s 22ms/step - loss: 0.8396 - accuracy: 0.5027 - val_loss: 0.7134 - val_accuracy: 0.5532
Epoch 5/100
6/6 [==============================] - 0s 21ms/step - loss: 0.6899 - accuracy: 0.5730 - val_loss: 0.5800 - val_accuracy: 0.7021
```

186枚から32枚ずつ(6回)

学習用データでの結果

検証用データでの結果

colab

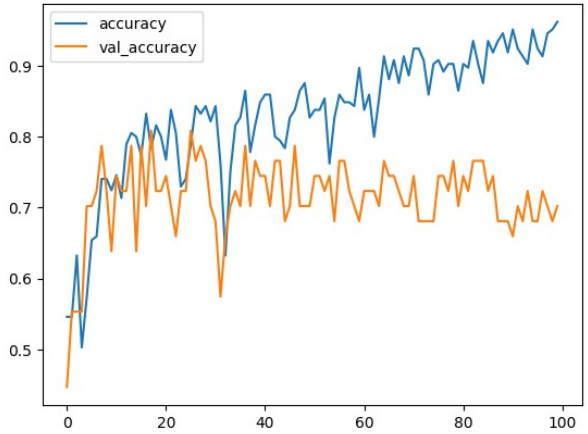
深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP4：モデルの評価

コード19-14　正解率の図示

```
plt.plot(result.history['accuracy'], label = 'accuracy')
plt.plot(result.history['val_accuracy'],label ='val_accuracy')
plt.legend()
plt.show()
```



- plt.plot(x,y)で各点をつなぐ線を描ける
- yは結果の'accuracy'と'val_accuracy'を選択
- xは指定していないとデータ数(100回)
- plt.legend()で凡例を表示する
- plt.show()で図を表示する

colab

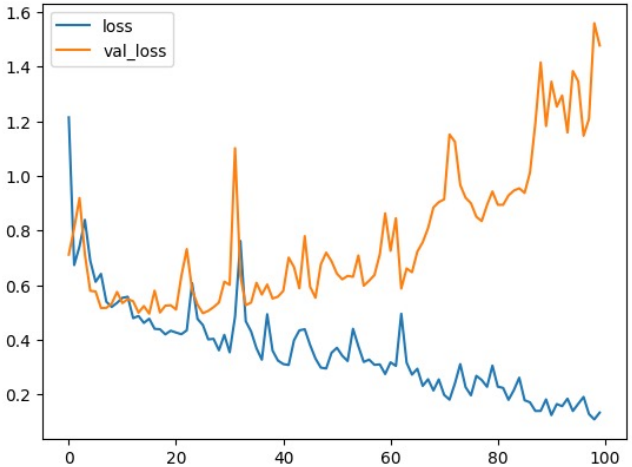
深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP4：モデルの評価

コード19-15　損失の図示

```
plt.plot(result.history['loss'], label='loss')
plt.plot(result.history['val_loss'], label='val_loss')
plt.legend()
plt.show()
```



- plt.plot(x,y)で各点をつなぐ線を描ける
- yは結果の'loss'と'val_loss'を選択
- xは指定していないとデータ数(100回)
- plt.legend()で凡例を表示する
- plt.show()で図を表示する

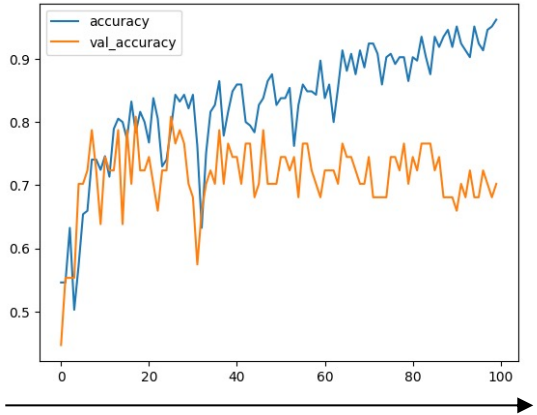
colab

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

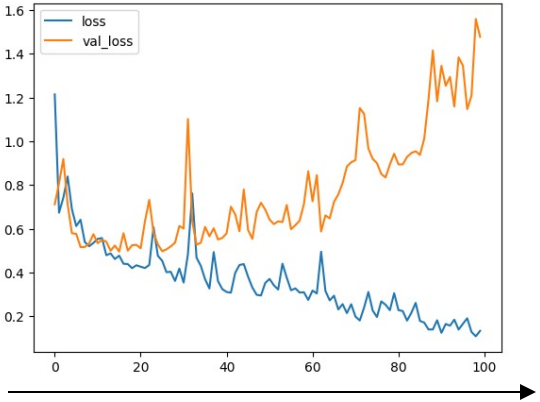
STEP4：モデルの評価

accuracy(正解率)



学習回数を重ねると

loss(損失)



学習回数を重ねると

学習用データの結果(accuracy)：
正解率が増加し、精度が上がっている

検証用データの結果(val_accuracy)：
精度が右肩上がりに上がっていない

学習用データの結果(loss)：
損失が減少し、精度が上がっている

検証用データの結果(val_loss)：
最初は減少しているが、それ以降はむしろ増加しており精度が悪くなっている

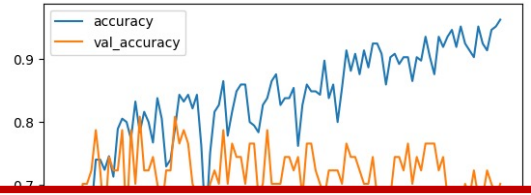
過学習を抑制する

深層学習（画像の分類）

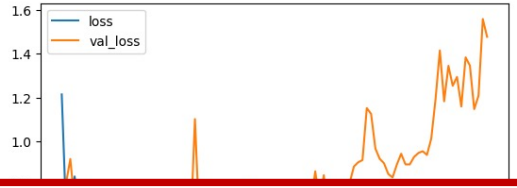
STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP4：モデルの評価

accuracy(正解率)



loss(損失)



このモデルではあまりうまく学習できていない

過学習の可能性があるため何らかの方法で抑制する必要がある

学習回数を重ねると

学習用データの結果(accuracy)：
正解率が増加し、精度が上がっている

検証用データの結果(val_accuracy)：
精度が右肩上がりに上がっていない

学習回数を重ねると

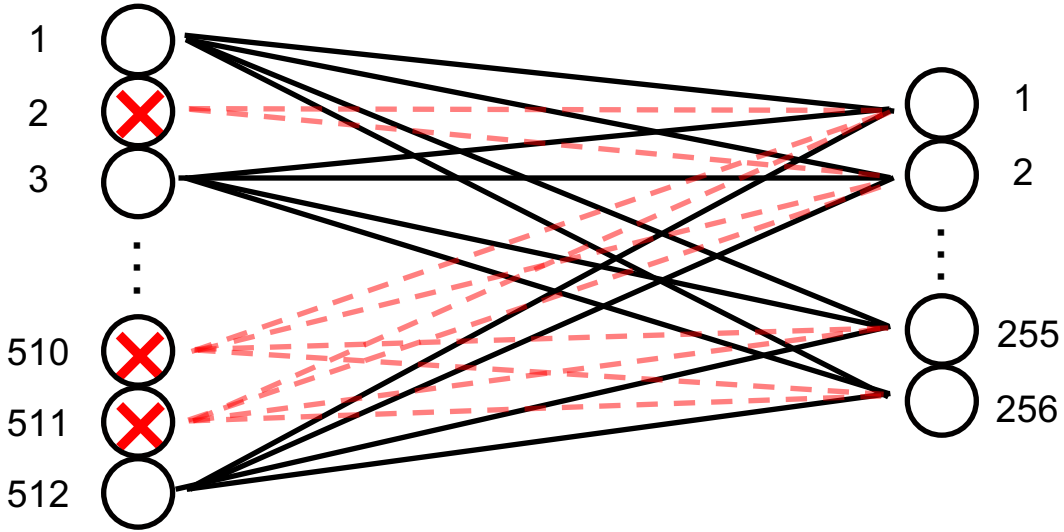
学習用データの結果(loss)：
損失が減少し、精度が上がっている

検証用データの結果(val_loss)：
最初は減少しているが、それ以降はむしろ増加しており精度が悪くなっている

Dropout

過学習を防ぐための対策の1つで、特定の層の出力をランダムに０にする手法

局所特徴が過剰に評価されてしまうのを防ぎ、モデルの精度を向上させる



深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP2：学習モデルの選択

コード19-16 ドロップアウトを加えて過学習を防ぐ

```
model_d = Sequential()
model_d.add(Flatten(input_shape = (64,64,1)))
model_d.add(Dense(512, activation ='relu'))
model_d.add(Dropout(0.5))
model_d.add(Dense(256, activation ='relu'))
model_d.add(Dropout(0.5))
model_d.add(Dense(128, activation ='relu'))
model_d.add(Dense(1, activation ='sigmoid'))
model_d.compile(loss = 'binary_crossentropy'
                optimizer = 'Adam',
                metrics = ['accuracy'])
model_d.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 4096)              0
dense (Dense)                (None, 512)               2097664
dense_1 (Dense)              (None, 256)               131328
dense_2 (Dense)              (None, 128)               32896
dense_3 (Dense)              (None, 1)                 129
=================================================================
Total params: 2262017 (8.63 MB)
Trainable params: 2262017 (8.63 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP2：学習モデルの選択

```
model_d = Sequential()
model_d.add(Flatten(input_shape = (64,64,1)))
model_d.add(Dense(512, activation ='relu'))
model_d.add(Dropout(0.5))
model_d.add(Dense(256, activation ='relu'))
model_d.add(Dropout(0.5))
model_d.add(Dense(128, activation ='relu'))
model_d.add(Dense(1, activation ='sigmoid'))
model_d.compile(loss = 'binary_crossentropy',
                optimizer = 'Adam',
                metrics = ['accuracy'])
model_d.summary()
```

中間層の1層目と2層目の間にDropoutを追加

中間層の2層目と3層目の間にDropoutを追加

Dropout(0.5)：
50%の出力値が0になる

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP2：学習モデルの選択

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten_1 (Flatten)          (None, 4096)              0
dense_4 (Dense)              (None, 512)               2097664
dropout (Dropout)            (None, 512)               0
dense_5 (Dense)              (None, 256)               131328
dropout_1 (Dropout)          (None, 256)               0
dense_6 (Dense)              (None, 128)               32896
dense_7 (Dense)              (None, 1)                 129
=================================================================
Total params: 2262017 (8.63 MB)
Trainable params: 2262017 (8.63 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Dropoutができている

2,097,664 + 131,328 + 32,896+129
=2,262,017個のパラメータ

colab

Dropout以外は同じ

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP3：データを入れて学習

コード19-17 学習させる

```
result_d = model_d.fit(x_train, y_train,
                       batch_size = 32,
                       epochs = 100,
                       validation_split = 0.2)
```

- （モデル名）.fit(x, y)で学習させ、result_dに学習結果を入れる
- 引数batch_size=32で「32組ずつデータを取り出して損失を計算し、重みとバイアスを更新しなさい」という指示
- 引数epochs=でエポック数を指定する(全てのデータを使い尽くすことを1エポック)
- 引数validatin_split=で任意の割合で分割して学習を行う（*学習用のデータは232組あり186組を学習用、46枚を検証用に分割して学習を行う）

colab

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP4：モデルの評価

コード19-18 ドロップアウトを加えたモデルの学習過程の図示

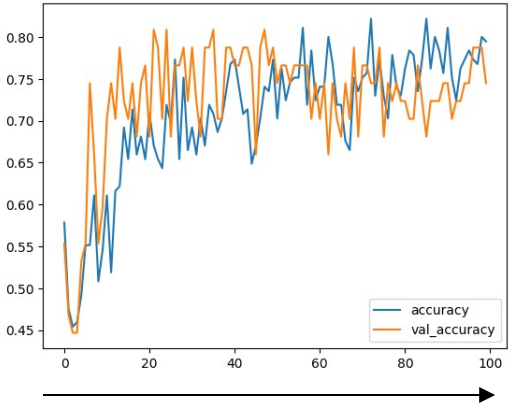
```
plt.plot(result_d.history['accuracy'], label = 'accuracy')
plt.plot(result_d.history['val_accuracy'],
         label = 'val_accuracy')
plt.legend()
plt.show()

plt.plot(result_d.history['loss'], label = 'loss')
plt.plot(result_d.history['val_loss'], label = 'val_loss')
plt.legend()
plt.show()
```

深層学習（画像の分類）

STEP4：モデルの評価

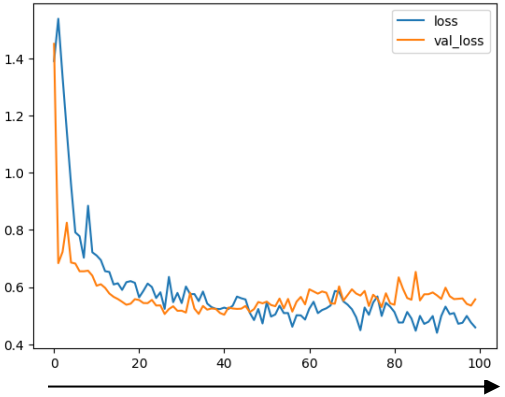
STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

accuracy(正解率)



学習回数を重ねると

loss(損失)



学習回数を重ねると

学習用データの結果(accuracy)：
正解率が増加し、精度が上がっている

検証用データの結果(val_accuracy)：
正解率が増加し、精度が上がっている

学習用データの結果(loss)：
損失が減少し、精度が上がっている

検証用データの結果(val_loss)：
損失が減少し、精度が上がっている

colab

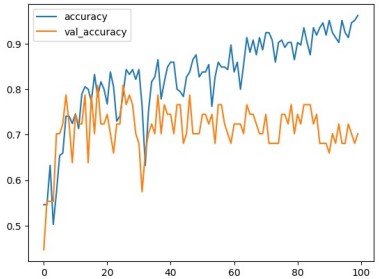
深層学習 (画像の分類)

STEP4：モデルの評価

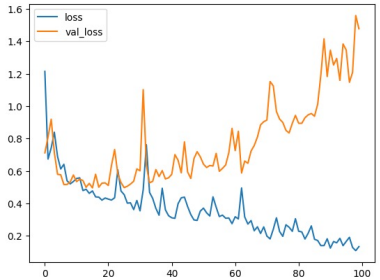
STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

Dropoutがないモデル (model)

accuracy(正解率)



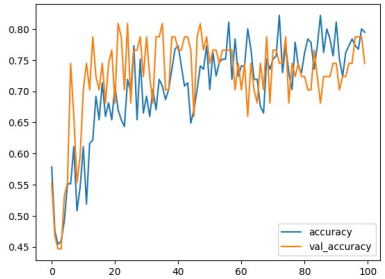
loss(損失)



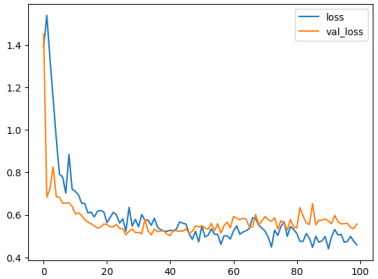
検証用データでは精度が悪い

Dropoutがあるモデル (model_d)

accuracy(正解率)



loss(損失)



検証用データでも学習データと似たような性能になっている

深層学習のコードの流れ

STEP1:データ分割

x(特徴量データ)　y(正解データ)

x_train　y_train

x_test　y_test

STEP2:学習モデルの選定

深層学習ではニューラルネットワークを学習モデルを自分で設計する

モデル

学習済みモデル

STEP3:学習

正解の予測

1：covid19肺炎
0：健康

正解との比較・評価

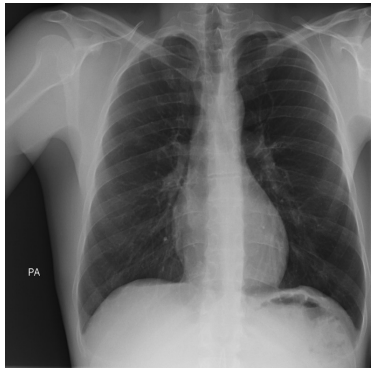
STEP4:モデルの評価

STEP5:予測

深層学習 (画像の分類)

新たな画像で分類を試してみよう

covid.jpg



NORMAL.jpg

学習モデルに入れたデータx_train, y_trainに含まれていない画像データ(covid.jpgとNORMAL.jpg)で分類を予測する

深層学習 (画像の分類)

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP5：予測

コード19-19 新たな画像での分類(covid.jpg)

```
img1 = img_to_array(load_img('/content/images/covid.jpg',
       color_mode = 'grayscale',
       target_size=(64,64)))/255
check = np.zeros((1,64,64,1))
check[0] = img1
```

- covid.jpgの画像データ(Covid19肺炎のx線写真)を読み込み、前処理でしたことと同じデータ構造にする

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP5：予測

```
img1 = img_to_array(load_img('/content/images/covid.jpg',
       color_mode = 'grayscale', target_size=(64,64)))/255
```

```
check = np.zeros((1,64,64,1))
```

```
check[0] = img1
```

画像を白黒で読み込んで(load_img)、numpy配列に変換(img_to_array)

全て0の要素の(1, 64, 64, 1)の4次元配列の行列を作る

画像データの配列をcheck[0]に代入

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP5：予測

コード19-20 新たな画像での分類(covid.jpg)

```
print(model_d.predict(check))
```

機械学習と同様に　(モデル名).predict()で予測値が出力される

```
1/1 [==============================] - 0s 104ms/step
[[0.9668271]]
```

この画像がcovid19肺炎である確率

＊covid19肺炎のデータを読み込んだので、予測としては正解

colab

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP5：予測

コード19-21 新たな画像での分類(NORMAL.jpg)

```
img2 = img_to_array(load_img('/content/images/NORMAL.jpg',
       color_mode = 'grayscale', target_size=(64,64)))/255
check = np.zeros((1,64,64,1))
check[0] = img2
```

- NORMAL.jpgの画像データ（健康な肺のx線写真）を読み込み、前処理でしたことと同じデータ構造にする

深層学習（画像の分類）

STEP0：事前準備
STEP1：データの用意と前処理
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：モデルの評価
STEP5：予測

STEP5：予測

コード19-22 新たな画像での分類 (NORMAL.jpg)

```
print(model_d.predict(check))
```

機械学習と同様に　(モデル名).predict()で予測値が出力される

```
1/1 [==============================] - 0s 26ms/step
[[0.26801327]]
```

この画像がcovid19肺炎である確率

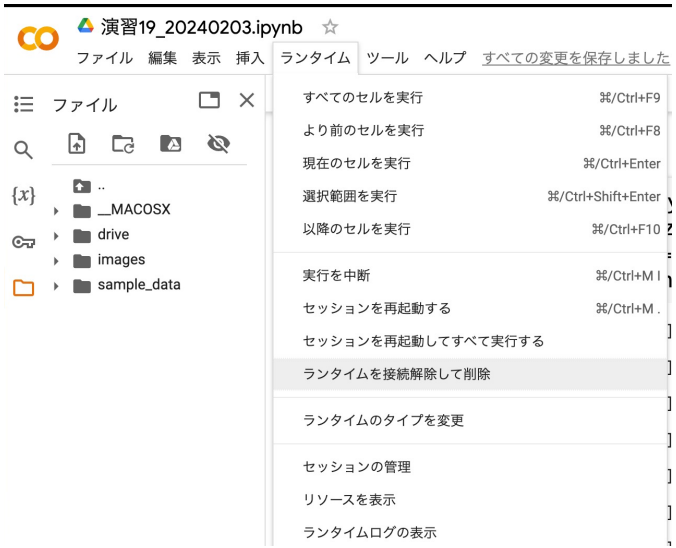
＊健康データを読み込んだので、予測としては正解

colab

演習19 課題

Webclassで課題を提出してください。締め切りは2024/02/15 23:59まで

model_dで学習するときのepochs数を300にした場合の最終的なval_accuracyとval_lossを回答してください。



- コード19-17のmodel_dで学習するときのepochs数を300に変更
- コード19-16～コード19-17を実行する
- 学習の300回目のval_accuracyとval_lossの値を回答

授業準備：Webclassからコードをダウンロードし、Google colaboratoryで開いておいてください

演習授業中の質問対応について

医療とAI・ビッグデータ入門
演習20

まとめの演習

演習20 課題

Webclassで課題を提出してください。締め切りは**2024/02/15 23:59**まで

(必修課題) Heart Attack dataset を使い、分類を行なってください

- **使用するモデルは演習10-演習14で取り扱った、LogisticRegression, SVC, DecisionTreeClasifier, RandomForestClasifierのうち、どれか一つを選択してください。**
- **webclassからkadai_ML.ipynbをダウンロードし、空欄となっているコード5～コード8にコードを書き、実行できることを確認してください。**
- **kadai_ML_(学生番号).ipynbに名前を変更し、webclassで提出してください。**

(発展課題) 全員が提出する必要はありません。提出された場合は成績に加点します。

皮膚がんの画像データを使って、分類を行なってください。

- **ニューラルネットワークで自分でモデルを設計してください。**
- **webclassからkadai_DL.ipynbをダウンロードし、空欄となっているコード6~コード8にコードを書き、実行できることを確認してください。**
- **kadai_DL_(学生番号).ipynbをwebclssで提出してください。**

演習20 課題

Heart Attack dataset を使い、分類を行なってください

コード3
heart.csvをGoogle Driveにアップロードして、pd.read_csv()で読み込む

```
df = pd.read_csv('/content/drive/MyDrive/heart.csv')
df
```

* Google Driveの MyDrive直下にheart.csvをおいている必要があることに注意

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

演習20 課題

Heart Attack dataset を使い、分類を行なってください

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns ➡ 303行 × 14列(13個の特徴量+1個の正解値)

演習20 課題

Heart Attack dataset を使い、分類を行なってください

age sex cp trtbps chol fbs restecg thalachh exng oldpeak slp caa thall output

age：年齢
sex：性別
cp：Chest pain type （0=典型的狭心症、1=非定型狭心症、2=非狭心症性疼痛、3=無症状）
trtbps：安静時血圧
chol：コレステロール値
fbs：空腹時血糖 120mmHg/dl以上の時1、以下の時0
restecg：安静時心電図結果～0＝正常、1＝ST-T波正常、2＝左室肥大
thalachh：最大心拍数
exng：運動誘発狭心症～1＝あり、0＝なし
oldpeak：安静時に比べて運動により誘発されるST低下
slp：運動ピークSTセグメントの傾き： 0：上り勾配、1：平坦、2：下り勾配
caa：主要血管の数(0-3)
thall：タリウム負荷試験： 0：正常0、1：正常1、2：固定欠損、3：可逆欠損
output：ターゲット変数（1=心臓発作の可能性が高い）

演習20 課題

Heart Attack dataset を使い、分類を行なってください

コード4 y_data（正解値データ)、x_data（特徴量データ)を作成する

```
y_data = df.iloc[:, 13]
x_data = df.iloc[:, 0:13]
```

y_data

```
0    1
1    1
2    1
3    1
4    1
    ..
298  0
299  0
300  0
301  0
302  0
Name: output, Length: 303, dtype: int64
```

y_data.shape

```
(303,)
```

x_data

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2

303 rows × 13 columns

x_data.shape

```
(303, 13)
```

演習20 課題

Heart Attack dataset を使い、分類を行なってください

kadai_ML.ipynbのコード5～8 に以下のコードを書いて実行してください。

コード5：train_test_split()で学習データと検証データに分割してください

コード6：学習モデルを選択してください(**LogisticRegression, SVC,**

DecisionTreeClasifier, RandomForestClasifierのうちどれか一つ)

***LogisticRegressionで行う場合には、収束しないエラーが起こることがあり、その場合は、**

(モデル名)= LogisticRegression(max_iter=1000)でインスタンス化を行なってください。

コード7：学習データを入れて、学習を行なってください

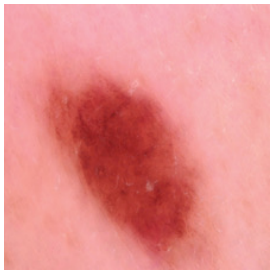
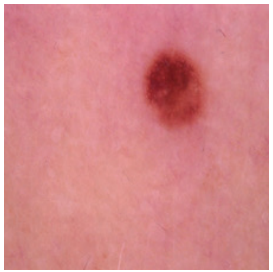
コード8：モデルの評価を行なってください(正解率、AUC、precisionなど好きなもの一つ以上)

演習20 課題

(発展課題) 皮膚がんの画像データを使って、分類を行なってください。

1) images_skin_cancer.zipをGoogle Driveにアップロードしてください。

良性腫瘍(正解値：0)

1440枚のデータ

悪性腫瘍 (正解値：1)


1197枚のデータ

出典(https://www.kaggle.com/datasets/fanconic/skin-cancer-malignant-vs-benign)

演習20 課題

(発展課題) 皮膚がんの画像データを使って、分類を行なってください。

コード1：Google Driveにマウントする

```
from google.colab import drive

drive.mount('/content/drive', force_remount=True)
```

コード2：images_skin_cancer.zip を解凍する

```
!unzip '/content/drive/MyDrive/images_skin_cancer.zip'
```

* Google Driveの MyDrive直下にimages_skin_caner.zipをおいている必要があることに注意

演習20 課題

(発展課題) 皮膚がんの画像データを使って、分類を行なってください。

コード4：前処理 (このまま全部実行してください)

- 画像データを読み込み、x_trainとy_trainを作成する
- 今回はカラーモード、サイズは64×64で読み込む

演習20 課題

(発展課題) 皮膚がんの画像データを使って、分類を行なってください。

コード5 ランダムシード値の設定

```
set_random_seed(0)
```

演習20 課題

(発展課題) 皮膚がんの画像データを使って、分類を行なってください。

kadai_DL.ipynbのコード6～8に以下のコードを書いて実行してください。

コード6 モデルを設計してください

(中間層は2層以上でニューラルネットワークで作成してください)

コード7 学習をしてください(モデル名.fit())

(batch_size =, epochs =, validation_split =の値を自分で決める)

***かなり時間がかかるので、epochs数は10~20ぐらいから始めるのをお勧め**

コード8 accuracyとval_accuracyの学習過程の図示、lossとval_lossの学習過程の図示

演習を始める前に2つアナウンス

医療とAI・ビッグデータ応用のご案内

「医療とAI・ビッグデータ応用」の履修について

■開講時期：前期　2024年度は4月25日～7月18日の予定
■単位数：1単位
■対象学科
　必修科目　　：医学科2年、歯学科2年
　選択必修科目：保健衛生学科・検査技術学専攻2年～
　自由科目　　：口腔保健学科・口腔保健衛生学専攻/口腔保健工学専攻　2年～
　　保健衛生学科・看護学専攻　2年～
■履修方法：
　必修科目以外については、4月初旬に各教務係から案内があります。
　希望者は申し込みをしてください。

演習を始める前に2つアナウンス

授業評価アンケート

授業評価アンケートについて

Webclass　QRコード

学科・専攻	コース番号
医学科	IL2300828
看護学専攻	IL2300325
検査技術学専攻	IL2300332
歯学科	IL2300143
口腔保健衛生学専攻	IL2300597
口腔保健工学専攻	IL2300596
自由科目、複合領域コース	入門コース内のアンケートから 回答してください。