

授業準備 : Webclassからコードをダウンロードし、 Google colaboryatory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

The screenshot shows a Zoom meeting window. The main display area has a dark background with white text that reads: "演習授業中の質問をチューターの先生が対応させていただきます。" (During the practice lesson, the tutor will respond to your questions.) To the right, a "ミーティングチャット" (Meeting Chat) window is open. Two red boxes with arrows provide instructions:

- Left Box:** "演習にエラーが出たなど問題があったらリアクションの**挙手**を押してください。" (If there is an error during the practice, please press the **raise hand** reaction.) An arrow points from this box to the "リアクション" (Reaction) button in the bottom toolbar, which is also circled in red.
- Right Box:** "質問内容を入力して、「**全員**」宛てに送信してください。" (Enter the question content and send it to **everyone**.) An arrow points from this box to the "宛先" (To) dropdown menu in the chat input area, which is also circled in red and currently shows "全員" (Everyone).

At the bottom of the Zoom window, the "リアクション" (Reaction) button is highlighted with a red box. The chat input area shows a dropdown menu with "宛先" (To) and "全員" (Everyone) selected.

医療とAI・ビッグデータ入門

演習16

深層学習

本スライドは、自由にお使いください。
使用した場合は、このQRコードからアンケート
に回答をお願いします。



*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

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

STEP0 : ライブラリの読み込み

```
from sklearn.datasets import load_breast_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]
```

STEP1 : データの準備

```
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()
```

STEP2 : 学習モデルの選択

STEP3 : データを入れて学習

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

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

STEP4 : 図示

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

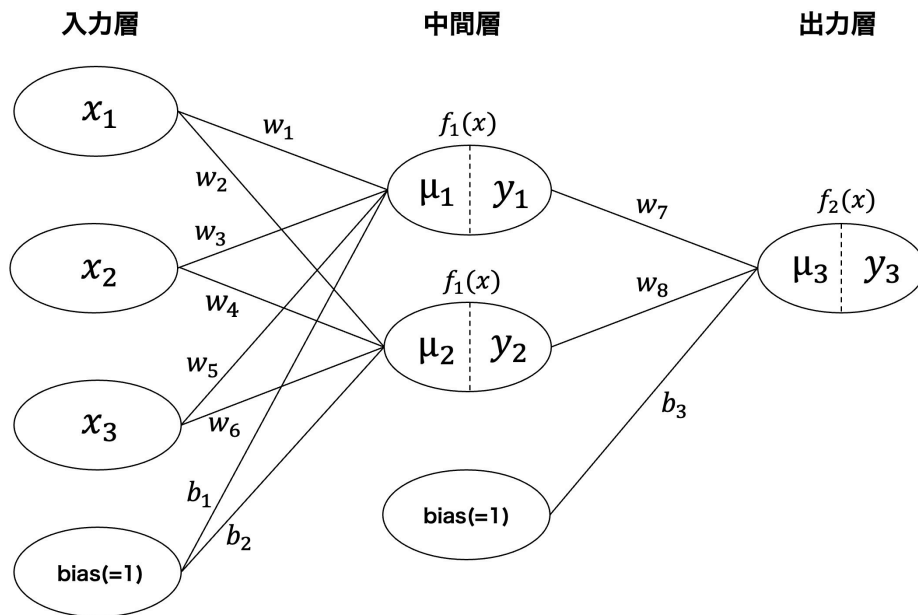
STEP5 : モデルの評価

深層学習(乳がんデータの分類)

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



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

深層学習(乳がんデータの分類)

STEP2 : 学習モデルの選択

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

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)

STEP2 : 学習モデルの選択

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

```
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)

STEP2 : 学習モデルの選択

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

```
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)

STEP2 : 学習モデルの選択

```
model_3 = Sequential()
```

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

STEP0 : 事前準備

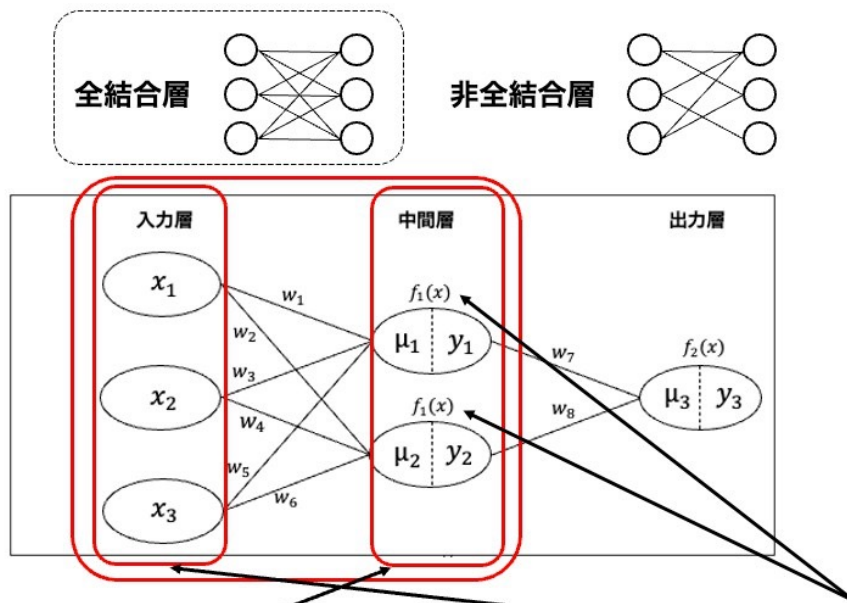
STEP1 : データの用意

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



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

深層学習(乳がんデータの分類)

STEP2 : 学習モデルの選択

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

```
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)

STEP2 : 学習モデルの選択

STEP0 : 事前準備

STEP1 : データの用意

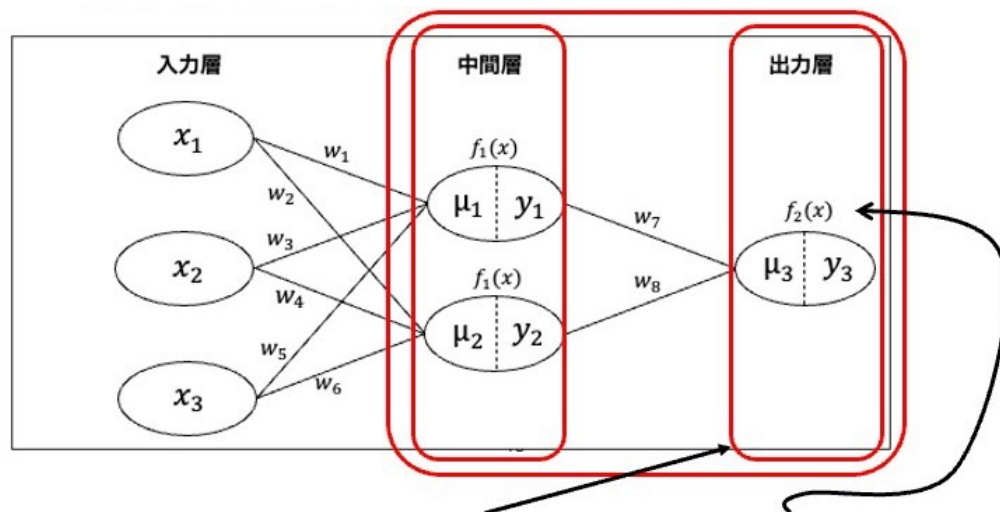
STEP2 : 学習モデルの選択

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

```
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)

STEP2 : 学習モデルの選択

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

```
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)

STEP2 : 学習モデルの選択

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

```
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)

STEP2 : 学習モデルの選択

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

```
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)

STEP2 : 学習モデルの選択

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

```
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

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)

STEP2 : 学習モデルの選択

STEP0 : 事前準備

STEP1 : データの用意

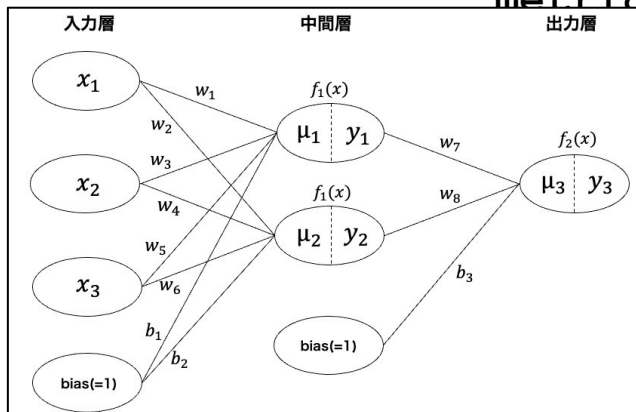
STEP2 : 学習モデルの選択

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

```
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 = 'accuracy')
```



↑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)

STEP3 : データを入れて学習

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

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

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)

STEP3 : データを入れて学習

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

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

***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)



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)

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

Epochの回数(1~300回)分表示

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)

colab

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

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, ...]}
```

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)

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


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

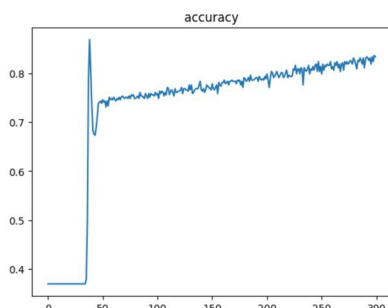
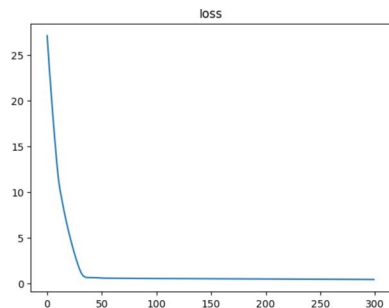
colab

深層学習(乳がんデータの分類)

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

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

colab

深層学習(乳がんデータの分類)

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(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

STEP0 : 事前準備

STEP1 : データの用意

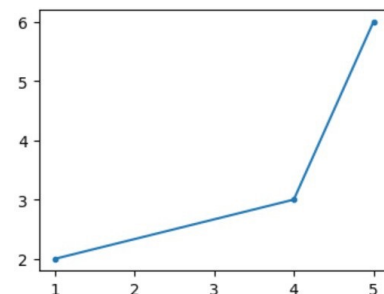
STEP2 : 学習モデルの選択

STEP3 : データを入れて学習

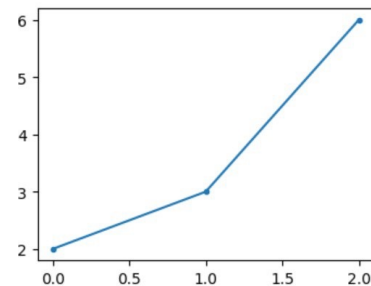
STEP4 : 学習結果の図示

STEP5 : モデルの評価

x = [1,4,5] y = [2,3,6]



y = [2,3,6]



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

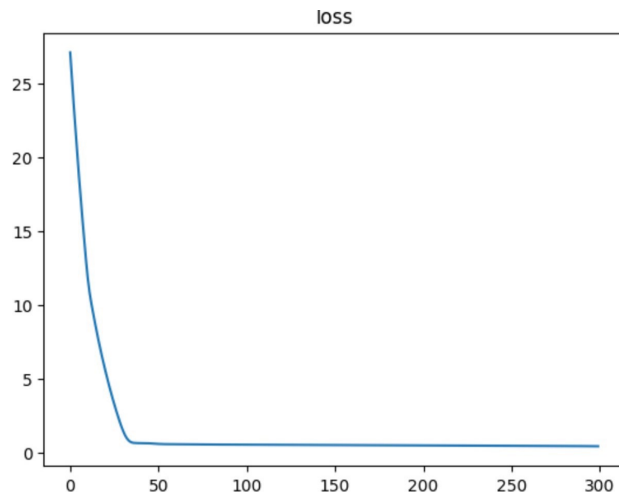
STEP1 : データの用意

STEP2 : 学習モデルの選択

STEP3 : データを入れて学習

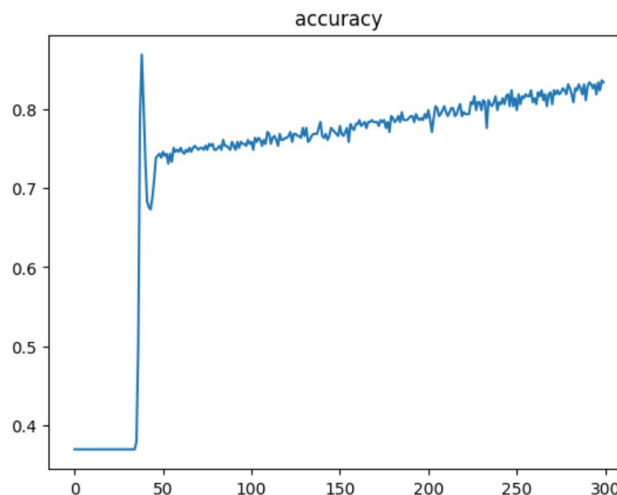
STEP4 : 学習結果の図示

STEP5 : モデルの評価



エポック回数

**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**



エポック回数

**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)

STEP5 : モデルの評価

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

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



精度を上げられるか検討

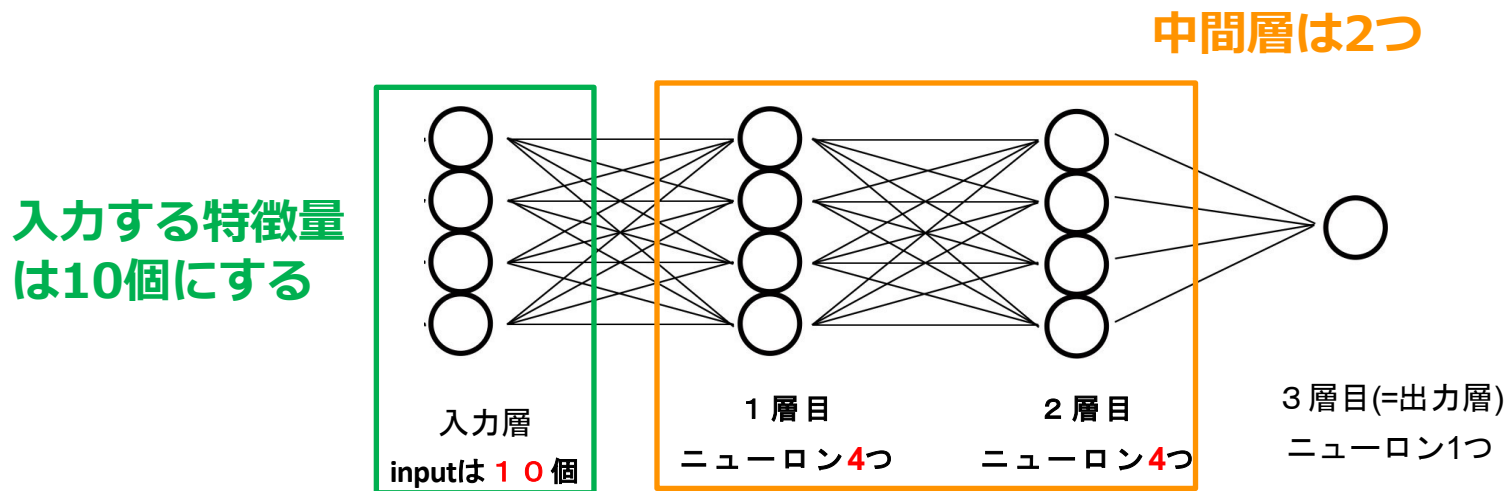
今までの中間層1つ、特徴量が3個だったため精度が低い



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



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)

STEP1 : データの用意

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

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)

STEP2 : 学習モデルの選択

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

コード15-12 学習モデルを設計

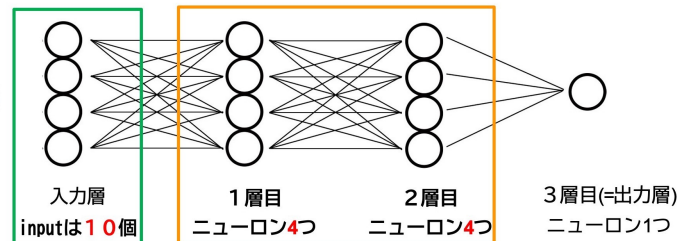
```
model_10 = Sequential()  
model_10.add(Dense(4, input_shape=(10,), activation = 'relu'))  
model_10.add(Dense(4, activation = 'relu'))  
model_10.add(Dense(1, activation = 'sigmoid'))  
model_10.compile(loss = 'binary_crossentropy',  
                  optimizer = 'Adam',  
                  metrics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

入力する特徴量
は10個にする



深層学習(精度上げるための調整)

STEP2 : 学習モデルの選択

STEP0 : 事前準備

STEP1 : データの用意

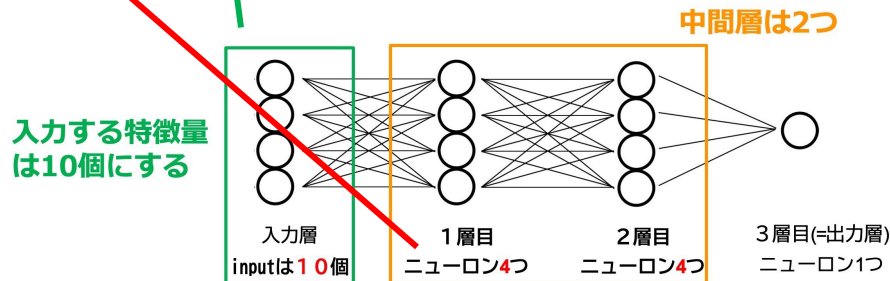
STEP2 : 学習モデルの選択

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

```
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)

STEP2 : 学習モデルの選択

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

```
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'))
```

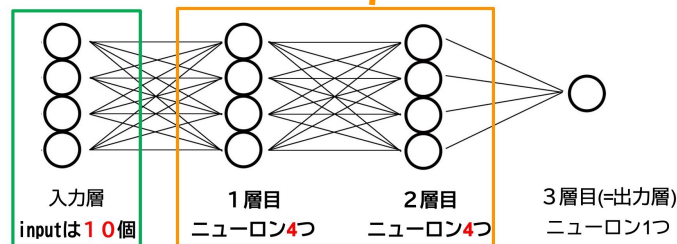
```
model_10.compile(loss = 'binary_crossentropy',  
                  optimizer = 'Adam',  
                  metrics = ['accuracy'])
```

```
model_10.summary()
```

入力層と中間層2つの設定

中間層は2つ

入力する特徴量
は10個にする



深層学習(精度上げるための調整)

STEP2 : 学習モデルの選択

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

```
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)

STEP3 : データを入れて学習

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

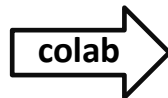
STEP4 : 学習結果の図示

STEP5 : モデルの評価

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



深層学習(精度上げるための調整)

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()
```

STEP0 : 事前準備

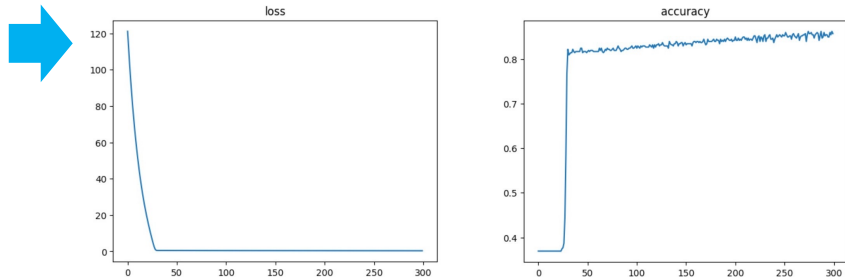
STEP1 : データの用意

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



- `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)

STEP5 : モデルの評価

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

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

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)