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

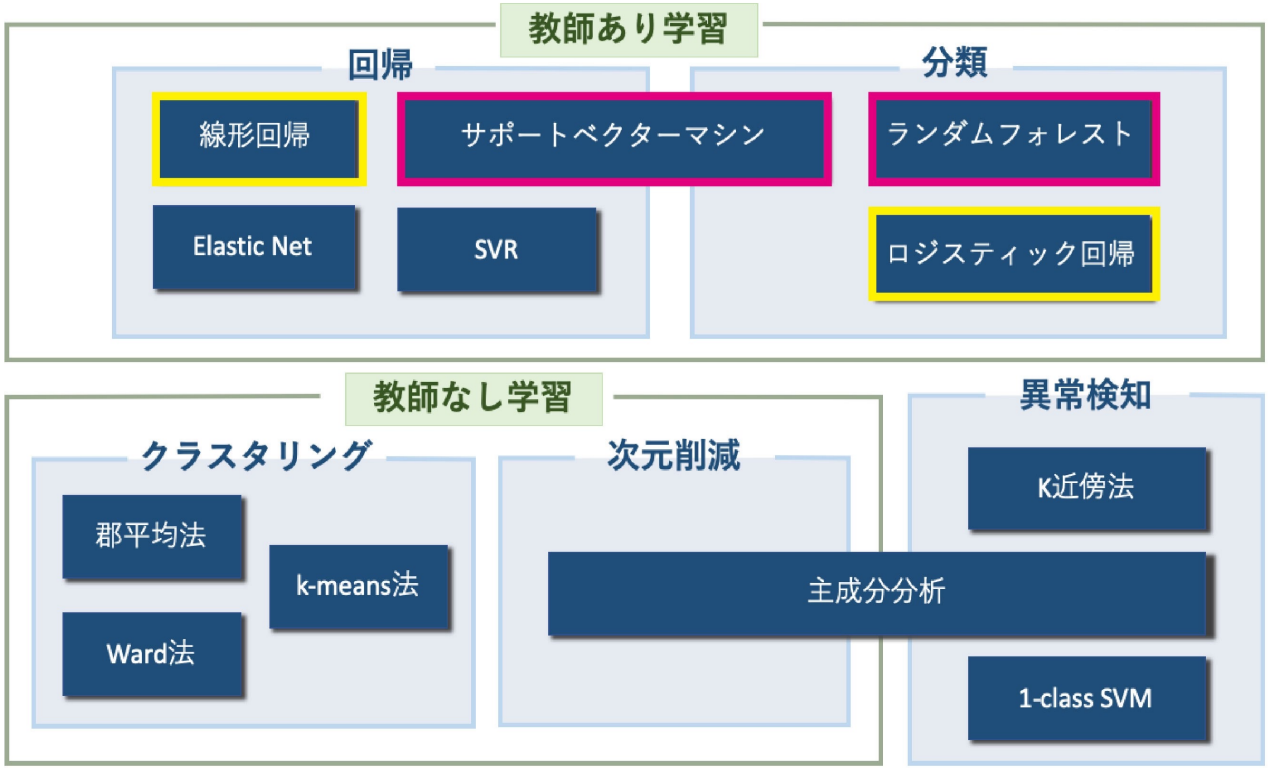
演習授業中の質問対応について

2024/01/24　11:35-12:20

医療とAI・ビッグデータ入門
演習13
教師あり機械学習-決定木-

統合教育機構
石丸美穂

関数 y=f(x) の f の設定次第で、機械学習にはさまざまなアルゴリズムがある



前回学んだ手法

今回学ぶ手法

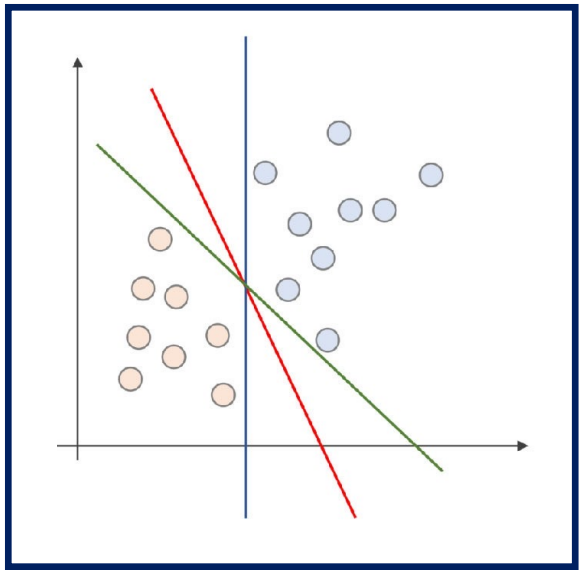
ここに挙げた代表的な手法のうち、
今回はランダムフォレストに代表
される決定木ベースの手法と、

2010 年くらいまでの最強の手法、
サポートベクターマシンを紹介する。

https://ainow.ai/2019/11/26/180809/

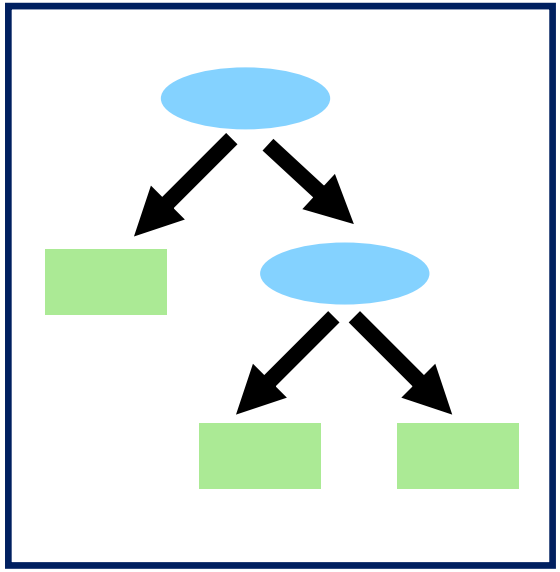
教師あり学習の種類

1.サポートベクターマシン



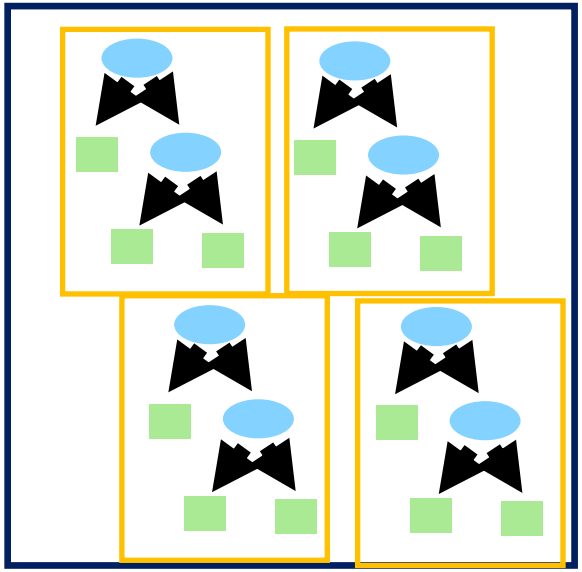
演習12

2.決定木



演習13

3. ランダムフォレスト

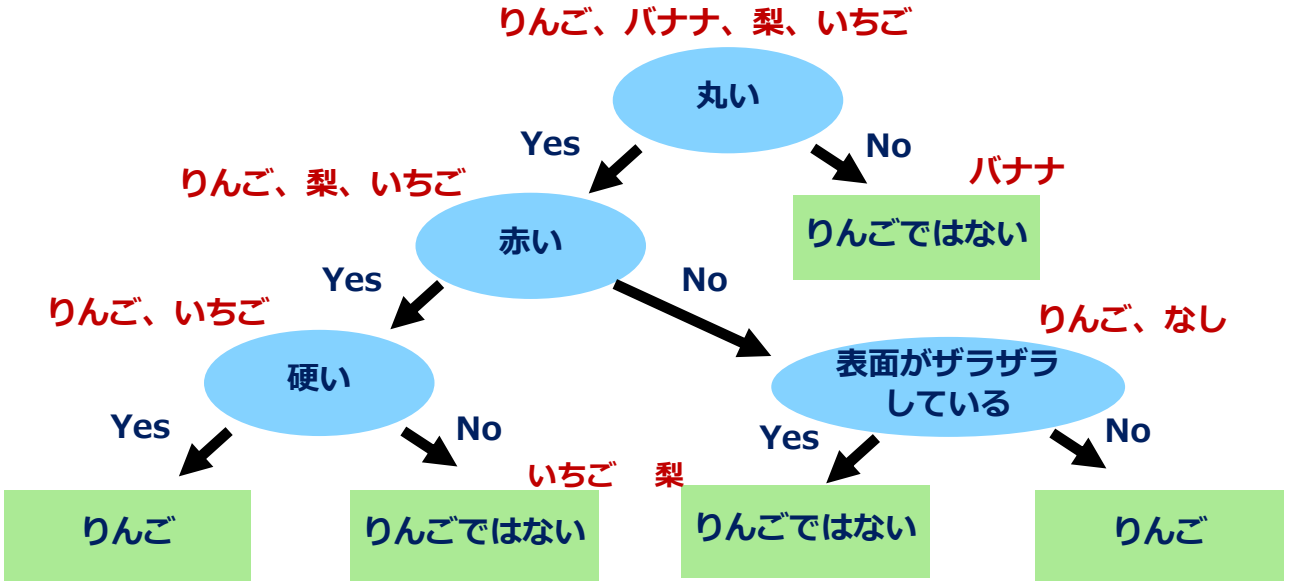


演習14

決定木(decision tree)

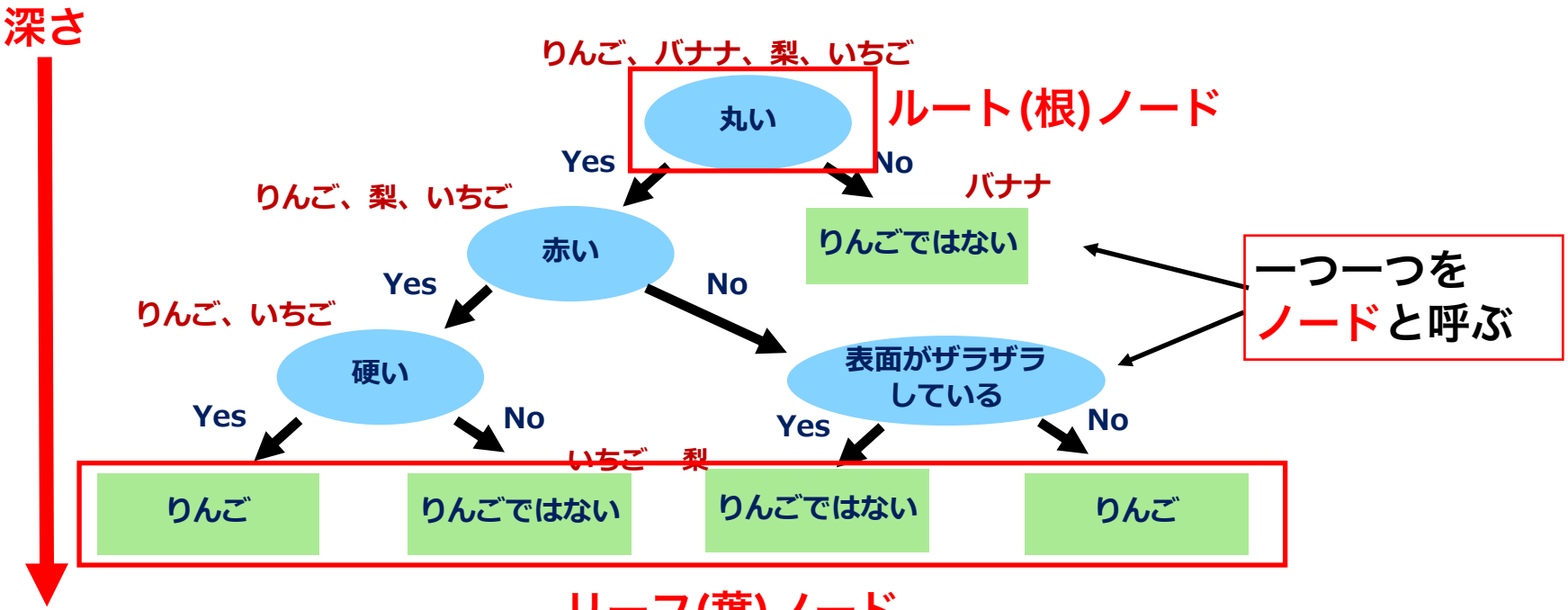
決定木(decision tree)は特徴量をもとに条件分岐をすることで分類する手法

例：果物100個のデータがある（りんご、バナナ、梨、いちご）
正解値：りんごかどうか
特徴量：赤色、硬いか、丸いか、表面がザラザラしている



決定木(decision tree)

決定木の構成要素



ルートノードからリーフノードまでの分岐の回数

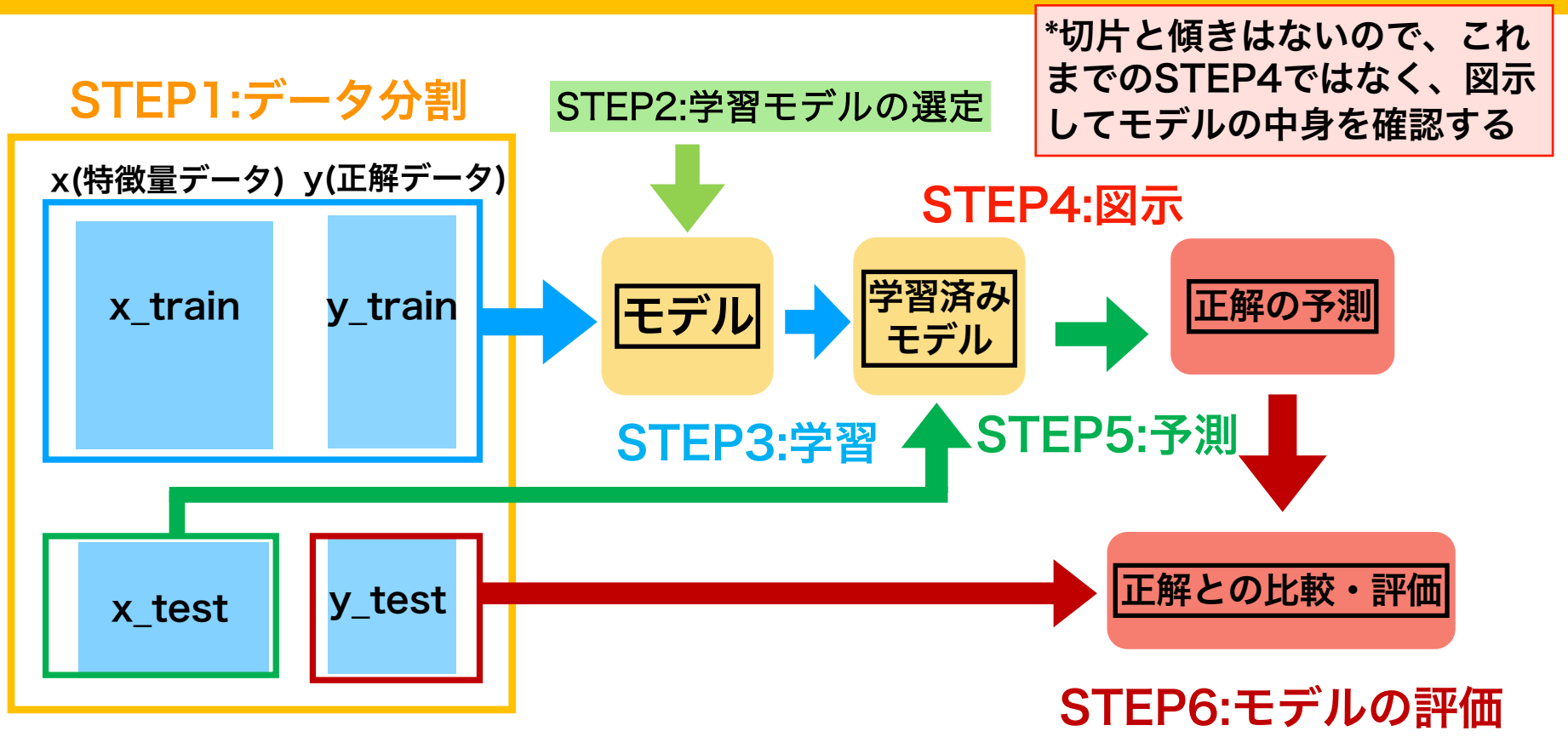
これ以上分岐しないノードをリーフノード

演習

決定木

乳がんのデータを用いて、分類を行う

機械学習の流れのまとめ



決定木のコードのまとめ

STEP1　データの用意

STEP2　学習モデルの選択(今回は決定木)

(モデル名) = DecisionTreeClassifier()

STEP3　データを入れて学習させる

モデルはDecisionTreeClassifier()

(モデル名).fit(特徴量,予測値)

STEP4　決定木の図示

決定木の図示を行う

STEP5　予測を行う：(モデル名).predict()

STEP6　モデルの評価：(モデル名).score(特徴量,予測値)

：roc_auc_score(実測値,Y=1の確率)

決定木のコードのまとめ

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 = True)

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)
```

STEP2：学習モデルの選択

```
from sklearn.tree import DecisionTreeClassifier
model_tree_c = DecisionTreeClassifier(criterion = 'gini', random_state = 0)
```

STEP3：データを入れて学習

```
model_tree_c.fit(x_train, y_train)
```

STEP4：図示

```
from sklearn.tree import plot_tree
plt.figure(figsize = (25,15))
plot_tree(model_tree_c, feature_names = bc.feature_names, class_names = bc.target_names,
filled = True)
```

STEP5： 予測

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, model_tree_c.predict(x_test), labels = [1,0])
```

STEP6：モデルの評価

```
print(model_tree_c.score(x_test, y_test))
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test, model_tree_c.predict_proba(x_test)[:,1])
```

決定木分析

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：決定木の図示
STEP5：予測を行う
STEP6：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13コード.ipynb

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

決定木分析

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

STEP0：ライブラリのインポート

コード13-1 ライブラリとモジュールをインポート

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

- いつも使うnumpyとpandasライブラリ、pyplotモジュールをインポートする

ライブラリをインポート：import （ライブラリ名） as 省略形

numpyとpandas

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

決定木分析

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

STEP1：データの用意

コード13-2 乳がんデータの読み込み

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

今回は分類を行うため、breast_cancer(乳がん)のデータを読み込む

関数をインポート　ライブラリ：sklearn　モジュール：datasets　関数：load_breast_cancer

```
from sklearn.datasets import load_breast_cancer
```

- load_breast_cancer(as_frame = True/False) の引数で、True(真)かFalse(偽)を選択する

読み込むデータの形式を指定
True → pdのデータフレーム型
False→ numpy配列

決定木分析

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

STEP1：データの用意

コード13-3 学習データと検証データに分割

*最初から学習データセットと検証データセットに分けてモデルの構築と評価を行う

```
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)
```

- trainデータ70%とtestデータ30%にデータを分割
- 分割時の乱数シード値を0に指定
- x_bcとy_bcを作成せず、bc.data, bc.targetをtrain_test_splitに直接入れてデータ分割を行う
- SVMと同様に、特徴量は30個全て使うことができる

colab

決定木分析

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

STEP2：学習モデルの選択

コード13-4 学習モデルに決定木を選択

```
from sklearn.tree import DecisionTreeClassifier
```

クラスだけをインポート　`from sklearn.tree import DesicionTreeClasifier`

- 今回のモデルは決定木分析で分類を行うので、`DecisionTreeClassifier()`を選択する
- `DecisionTreeClassifier()`クラスから`model_tree_c`を新しいインスタンスとして作成する

決定木分析

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

STEP2：学習モデルの選択

コード13-4 学習モデルに決定木を選択

```
model_tree_c = DecisionTreeClassifier(criterion = 'gini',
                                      random_state = 0)
```

- 引数criterion= で分類のアルゴリズムを指定する。今回は'gini'でジニ不純度が最小になるように分類するCARTアルゴリズム（必ず二つに分岐する構造がシンプルなモデル）を使う
- 引数random_state=では乱数シード値を固定する。決定木の分割時に、特徴量がランダムに並び替えられたり、分割時にタイブレークの時にどれか一つの分割をランダムに選択する必要があるため。

colab

分類決定木の学習方法

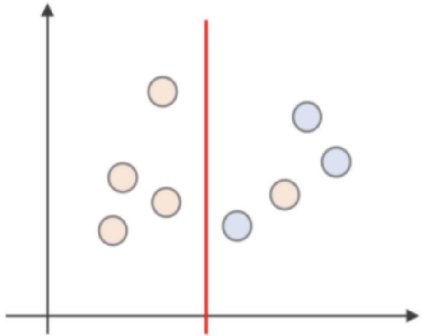
回帰の決定木の場合は RSS を指標にして分割していくが、分類の決定木の場合にはジニ不純度 (gini impurity) をもとに分割方法を決定。
これはクラス分けがどれだけ綺麗かの指標であり、分類した時にどれだけ「不純なもの」が含まれるかを表す。

ジニ不純度 G は $G = \sum_{k=1}^{K} p(k)(1 - p(k))$　　K はクラス数，p(k) はその領域でのクラス k の割合

例えば以下のようにデータを分割すると (赤と青の 2 クラスの例)

左の領域のジニ不純度 G は

$$G = \frac{4}{4}\left(1 - \frac{4}{4}\right) + \frac{0}{4}\left(1 - \frac{0}{4}\right) = 0$$



右の領域のジニ不純度 G は

$$G = \frac{1}{4}\left(1 - \frac{1}{4}\right) + \frac{3}{4}\left(1 - \frac{3}{4}\right) = 0.375$$

このように、混じり物があるとジニ不純度は大きくなる。
したがって、ジニ不純度が小さくなるように木の分割を考えていけばよい。

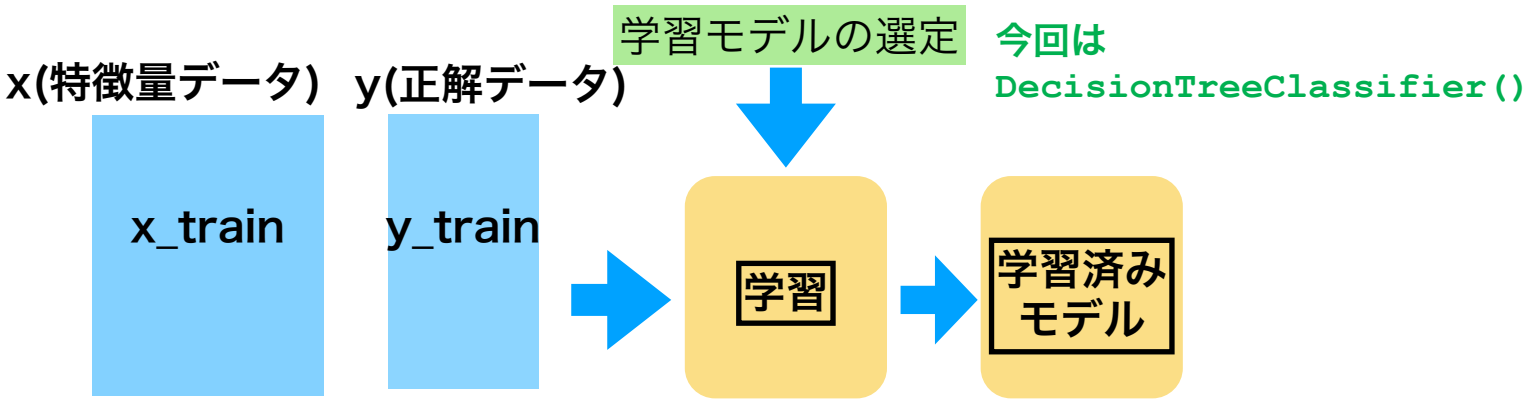
決定木分析

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

STEP3：データを入れて学習させる

コード13-5 学習用データで学習させる

```
model_tree_c.fit(x_train, y_train)
```



colab

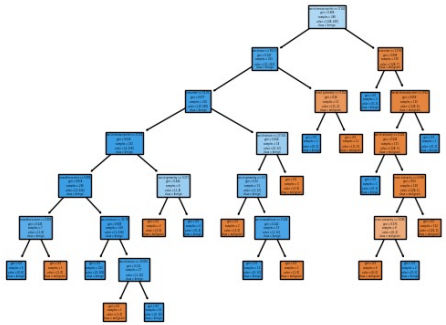
決定木分析

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

STEP4：決定木の図示

コード13-6 決定木の図示

```
from sklearn.tree import plot_tree
plt.figure(figsize = (25,15))
plot_tree(model_tree_c, feature_names = bc.feature_names,
   class_names = bc.target_names, filled = True)
plt.show()
```



- 決定木を図示するために`plot_tree()`関数をインポート
- `plt.figure(figsize =(XX,XX))`で図の大きさを指定できる。指定しないと文字が潰れて読めないので、大きめの`(25,15)`で指定する
- `plot_tree(決定木モデル, feature_names=, class_names=, filled = )`で引数を指定する

決定木分析

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

STEP4：決定木の図示

コード13-7 bcのfeature_namesの内容確認

```
bc.feature_names
```

```
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

```
bc.feature_names.shape
```

```
(30, )
```

```
type(bc.feature_names)
```

```
<class 'numpy.ndarray'>
```

決定木分析

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

STEP4：決定木の図示

コード13-8 bcのtarget_namesの内容確認

```
bc.target_names
```

➡ array(['malignant', 'benign'], dtype='<U9')

```
bc.target_names.shape
```

➡ (2,)

```
type(bc.target_names)
```

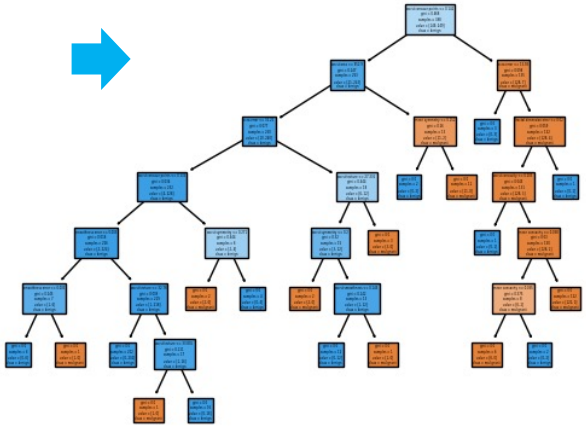
➡ <class 'numpy.ndarray'>

決定木分析

STEP4：決定木の図示

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

```
plot_tree(model_tree_c, feature_names = bc.feature_names,
          class_names = bc.target_names, filled = True)
```



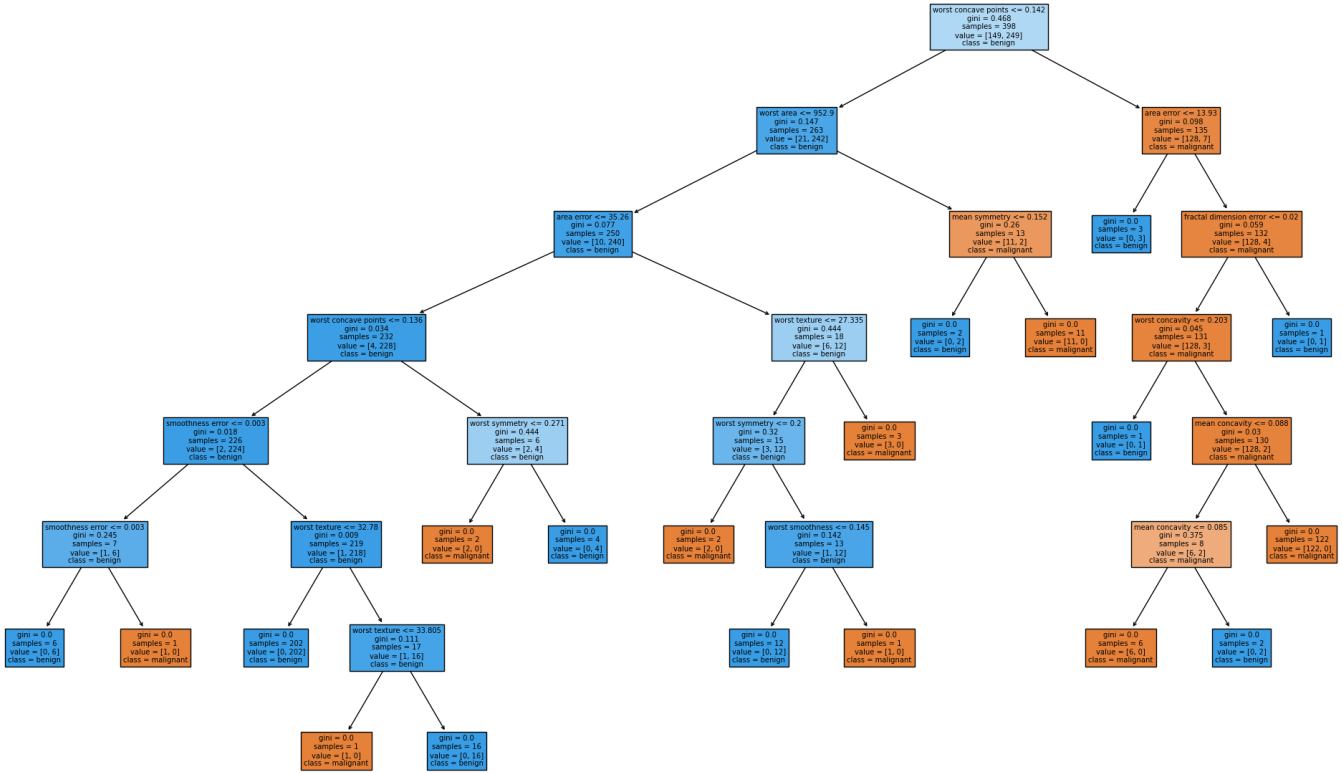
- plot_tree(決定木モデル, feature_names=, class_names=, filled =)で引数を指定
- feature_namesはbc.feature_names（特徴量の名前のリスト), class_namesはbc.target_names(targetのクラス(良性腫瘍/悪性腫瘍))を指定
- 引数filled = Trueを指定すると、カラーになりノードのクラスが良性腫瘍(青)/悪性腫瘍(橙)と異なる色で描出

colab

決定木分析

STEP4：決定木の図示

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価



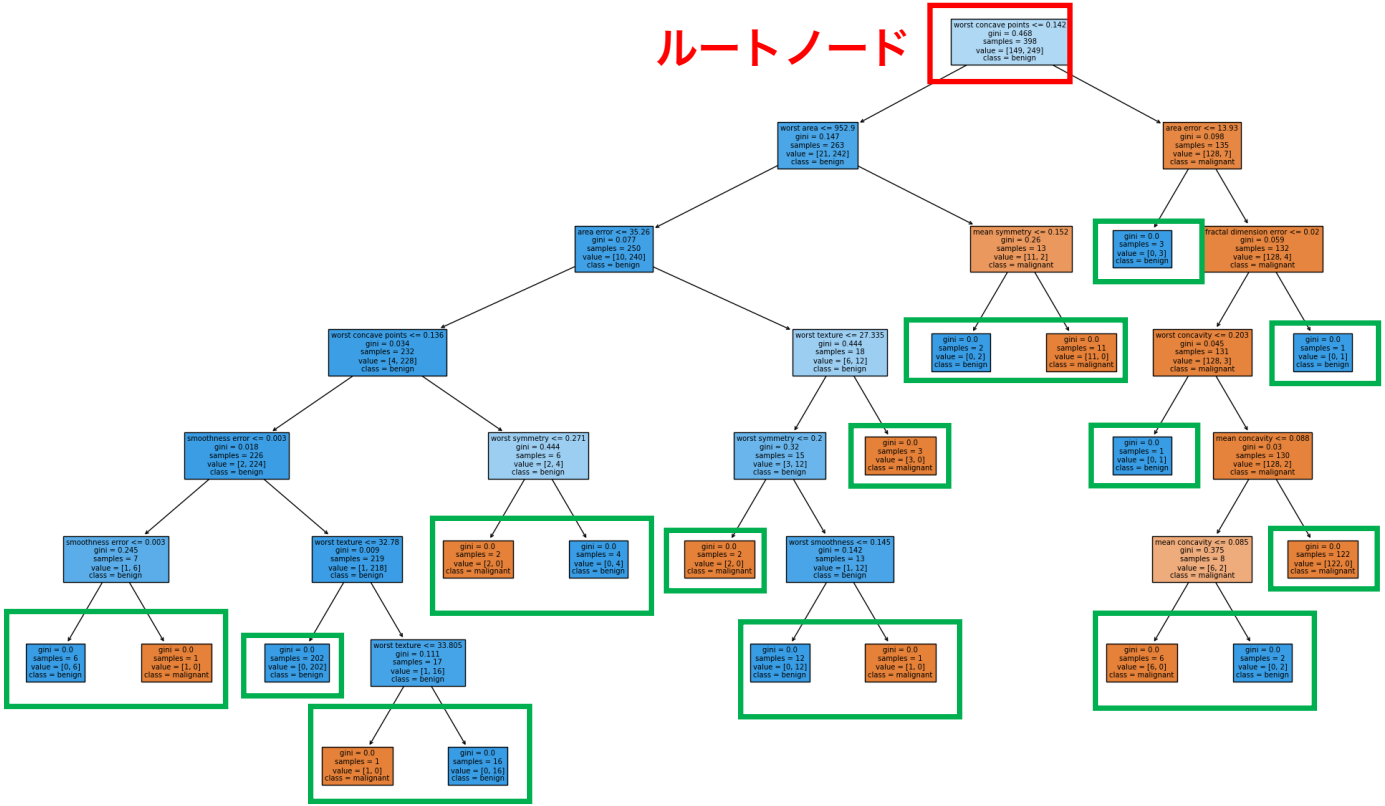
リーフノードが1つのクラス（正解値1/0)になるまで分岐される

決定木分析

STEP4：決定木の図示

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

ルートノード



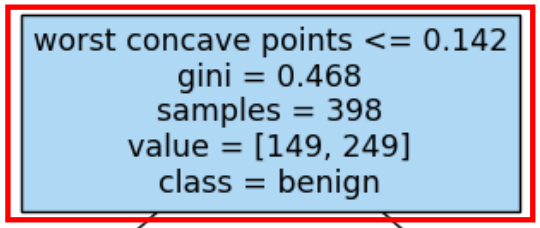
リーフノードが1つのクラス（正解値1/0)になるまで分岐される

決定木分析

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

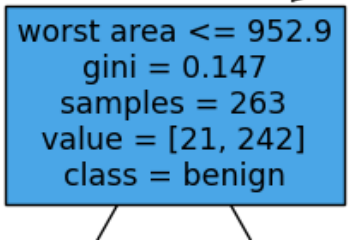
STEP4：決定木の図示

ルートノード



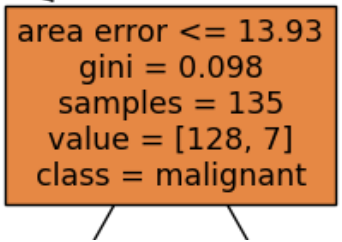
←次の分岐の条件
←このノードのgini不純度
←このノードのサンプルサイズ
←[悪性腫瘍の人数、良性腫瘍の人数]
←良性腫瘍の方が多いのでこのノードのクラスはbenign(良性腫瘍)

分岐条件が’yes’



←良性腫瘍の方が多いのでこのノードのクラスはbenign(良性腫瘍)

分岐条件が’no’

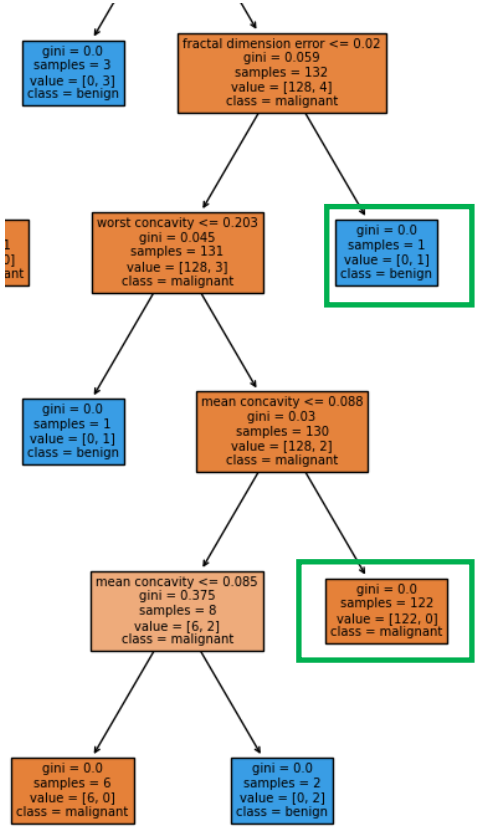


←悪性腫瘍の方が多いのでこのノードのクラスはmalignant(悪性腫瘍)

決定木分析

STEP4：決定木の図示

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価



リーフノード

←ノードのgini不純度は0になる
このノードのサンプルサイズ
[悪性腫瘍の人数、良性腫瘍の人数]
良性腫瘍しかいないのでこのノードのクラスはbenign

←ノードのgini不純度は0になる
122人が全員悪性腫瘍のため、このノードのクラスはmalignant (悪性腫瘍)

決定木分析

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

STEP5：予測を行う

コード13-9　検証用データを用いて予測する

```
print(model_tree_c.predict(x_test))

print(np.array(y_test))
```

```
[0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 0 0 0 0 1 1 0 1 1 0 1 0 1 1 1 0 1 0 1
 0 1 0 0 1 0 1 1 0 1 1 1 0 0 0 0 1 1 1 0 1 0 0 0 0 1 1 0 0 0 0 0 1 1 0 1 1
 0 1 1 1 1 1 0 0 0 1 0 1 1 1 0 0 1 0 1 0 1 1 0 0 1 1 1 1 1 0 0 1 0 1 0 1 1
 0 0 1 1 1 1 1 1 1 1 1 0 1 0 0 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 0 1 0
 1 1 1 0 0 0 1 0 1 1 1 0 0 1 0 0 1 0 0 0 1 1 1]
[0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 1 1 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1
 0 1 0 0 1 0 1 1 0 1 1 1 0 0 0 0 1 1 1 1 1 1 1 0 0 0 1 1 0 1 0 0 0 1 1 0 1 0
 0 1 1 1 1 1 0 0 0 1 0 1 1 1 0 0 1 0 1 0 1 1 0 1 1 1 1 1 1 1 1 0 1 0 1 0 0 1
 0 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 0 1 0
 1 1 1 1 1 1 1 0 1 0 1 0 0 1 1 0 1 0 0 0 1 1 1]
```

x_testデータで予測したyが0/1かの値

y_testの実際の値

決定木分析

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

STEP5：予測を行う

コード13-10 混同行列の出力

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, model_tree_c.predict(x_test),
  labels = [1, 0])
```

```
array([[97, 11],
       [ 4, 59]])
```

ライブラリ名　モジュール名　関数名

●関数をインポート `from sklearn.metrics import confusion_matrix`

● `confusion_matrix(実際の分類,予測分類,labels=[])`で混同行列を出力する。
`labels`引数=で出力の順番（1：良性腫瘍、0：悪性腫瘍）を指定する

決定木分析

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

STEP5：予測を行う

```
array([[97, 11],
       [ 4, 59]])
```

		予測結果	
		Positive(正)	Negative(負)
実際の分類結果	Positive (正)	真陽性True Positive **97**	偽陰性False Negative **11**
	Negative (負)	偽陽性False Positive **4**	真陰性True Negative **59**

colab

決定木分析

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

STEP6：モデルの評価

コード13-11 正解率の出力

```
print(model_tree_c.score(x_test, y_test))
```

➡ 0.9122807017543859 ⬅ 検証データの正解率

```
print(model_tree_c.score(x_train y_train))
```

➡ 1.0 ⬅ 学習データではすべて分類できているため、正解率は1(100%)になる

(モデル名).scoreで分類の時は正解率が出力される

colab

決定木分析

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

STEP6：モデルの評価

コード13-12 検証用データでAUCを算出

```
from sklearn.metrics import roc_auc_score

roc_auc_score(y_test, model_tree_c.predict_proba(x_test)[:,1])
```

*roc_auc_score(実際の分類, Y=1になる確率)でAUCを算出できる

0.9173280423280423

決定木の方が少し予測性能が低く出力された

ロジスティック回帰のAUC：0.982363315696649

colab

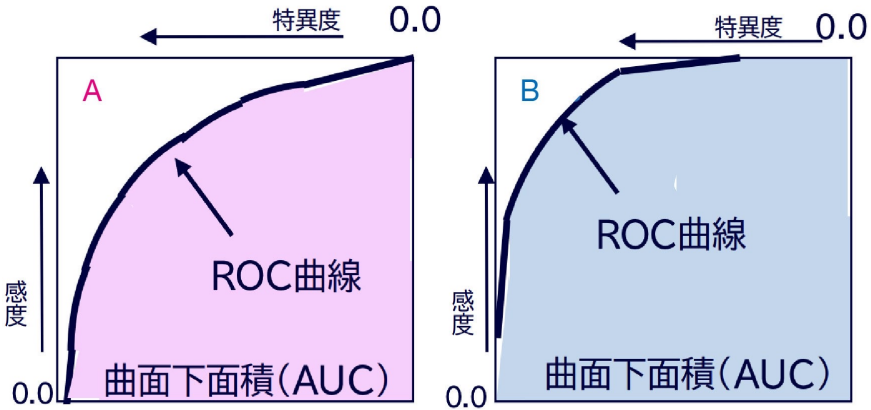
(清水先生の講義引用)

ROC 曲線と AUC による分類モデルの良し悪しの比較

ROC（Receiver operating characteristic）曲線はもともとレーダーの測定能力の評価に関する種々の研究や理論が1970 年以降に医療や機械学習へ応用されたものだ。

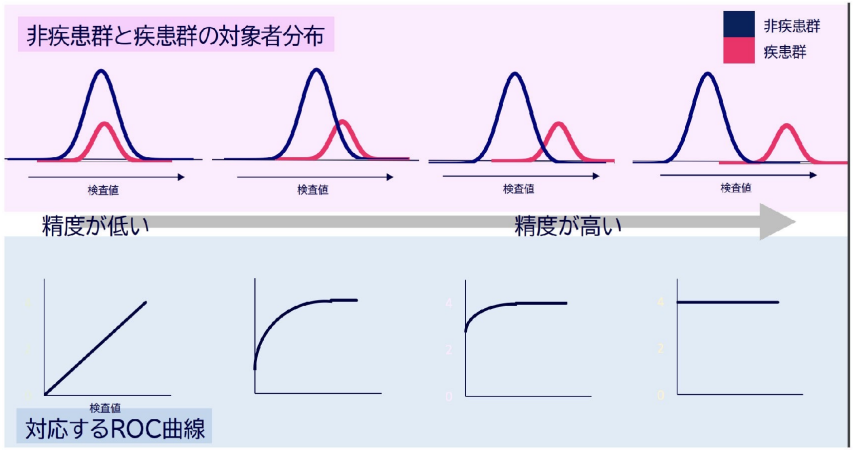
さまざまなカットオフで感度・特異度を計算し、横軸に 1 - 特異度、縦軸に感度をプロットしたものだ。
左上が感度 1、特異度 1 という理想的な分類器を示す。

ROC 曲線の下側の面積が AUC (Area under curve) である。ROC 曲線が左上にあればあるほど AUC は 1 に近い値をとる。
この AUC の値をもとに、どちらの分類器が優れるかを判断できる。



A と B だと、AUC が大きい B の方が優れるといえる。

ROC のイメージはこのようになる。
2 群を全く区別できない場合、ROC は対角線、AUC は 0.5 となる。



https://www.jsph.jp/covid/files/5BAA6E3.pdf より一部改変

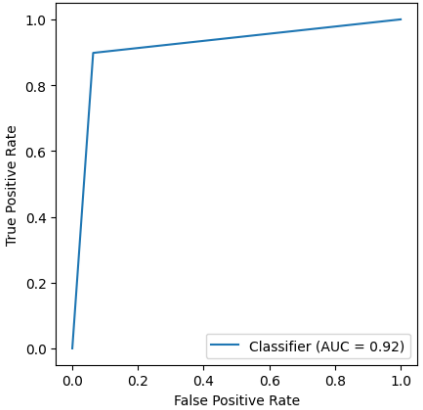
決定木分析

STEP6：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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

コード13-13 ROC曲線の描出

```
from sklearn.metrics import RocCurveDisplay
RocCurveDisplay.from_predictions(y_test,
   model_tree_c.predict_proba(x_test)[:,1])
plt.axis('square')
plt.show()
```



*RocCurveDisplay.from_prediction(実際の分類, Y=1になる確率)でROC曲線を描出できる
*plt.axis('square')で正方形の図に指定
*plt.show()で図を表示

colab

剪定 pruning

剪定 pruning

決定木分析では、全て純粋なノード（良性腫瘍か悪性腫瘍のどちらか一つ)
になるまで分類するため、過学習が起きやすい

= 学習データに対しては100%の正解率になる

木の深さやノードの作り方を制限することで過学習を抑えることができる

決定木分析

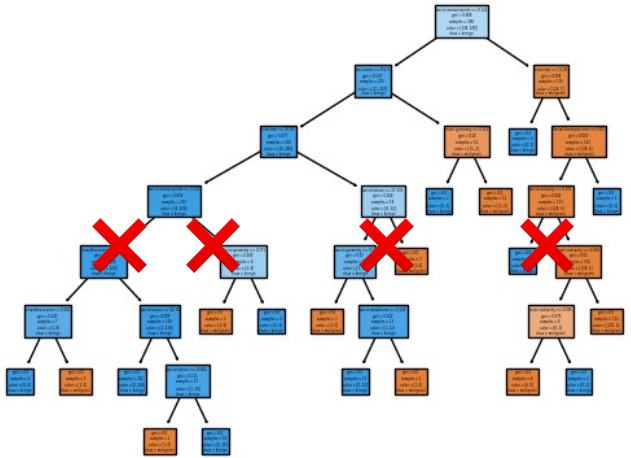
STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

STEP2：学習モデルの選択

DecisionTreeClasifier()の引数

- 引数max_depth=で木の深さ（何回分類を行うか）の最大値を指定する。**深さの最大値を指定しないと、全てのデータが分類されるまで分割を繰り返す。**大きいと過学習の傾向になる

- 引数min_samples_split=では、**分岐を作成するために必要な最低サンプルサイズを指定。**小さいと過学習の傾向になる

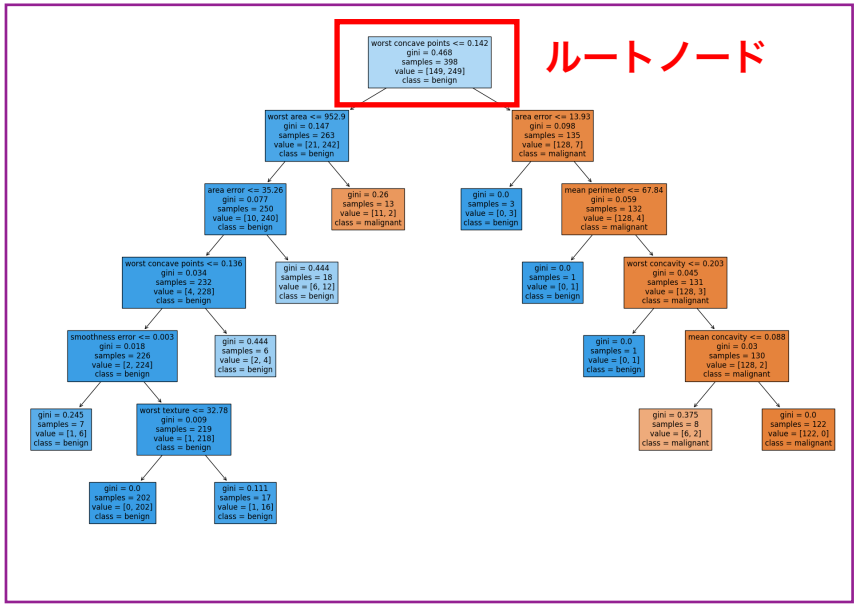


決定木分析 -剪定モデル-

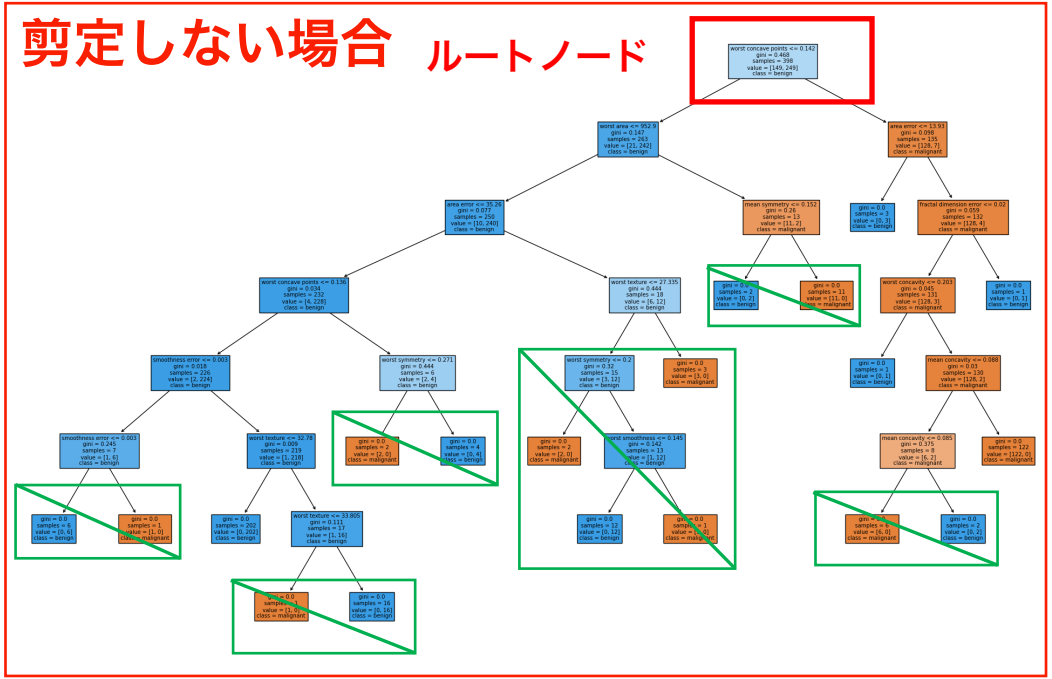
STEP2：学習モデルの選択

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

min_samples_split=20に設定した場合
(ノードのサンプル数が20以下では分岐しない)



剪定しない場合　ルートノード



決定木分析 -剪定モデル-

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

STEP2：学習モデルの選択

コード13-14 剪定した学習モデルの選択

```
model_tree_p = DecisionTreeClassifier(
criterion = 'gini', max_depth = 3, random_state = 0)
```

- DecisionTreeClassifier()クラスからmodel_tree_pを新しいインスタンスとして作成する
- 引数max_depth=3で最大3回の深さで分岐するように指定

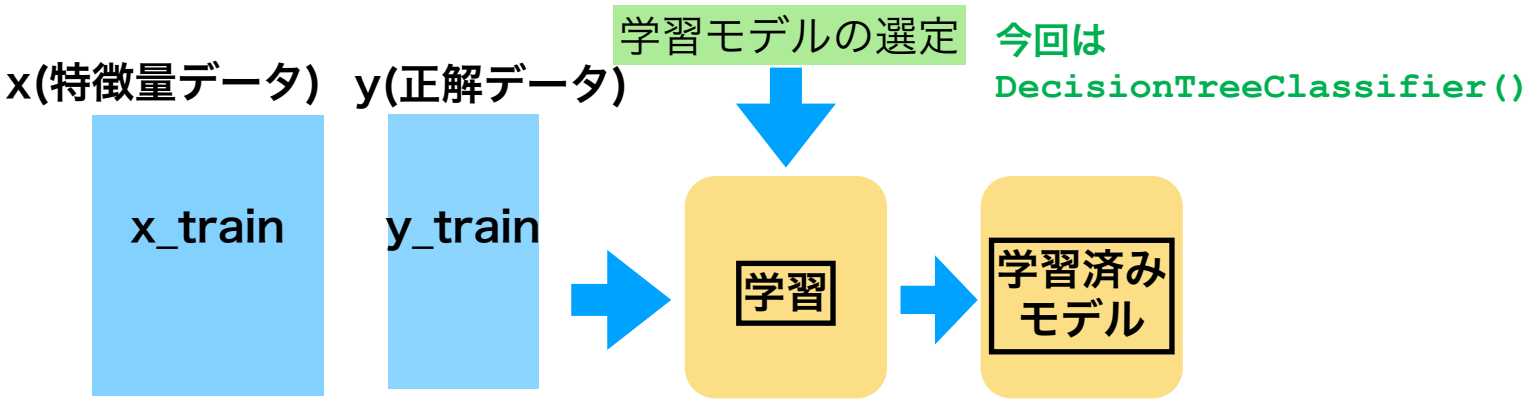
決定木分析 -剪定モデル-

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

STEP3：データを入れて学習させる

コード13-15 学習用データで学習させる

```
model_tree_p.fit(x_train, y_train)
```



colab

決定木分析 -剪定モデル-

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

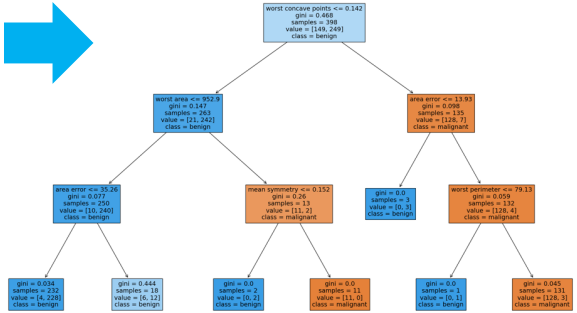
STEP4：決定木の図示

コード13-16　剪定モデルの図示

```
plt.figure(figsize = (25,15))

plot_tree(model_tree_p, feature_names = bc.feature_names,

   class_names = bc.target_names, filled = True)
```

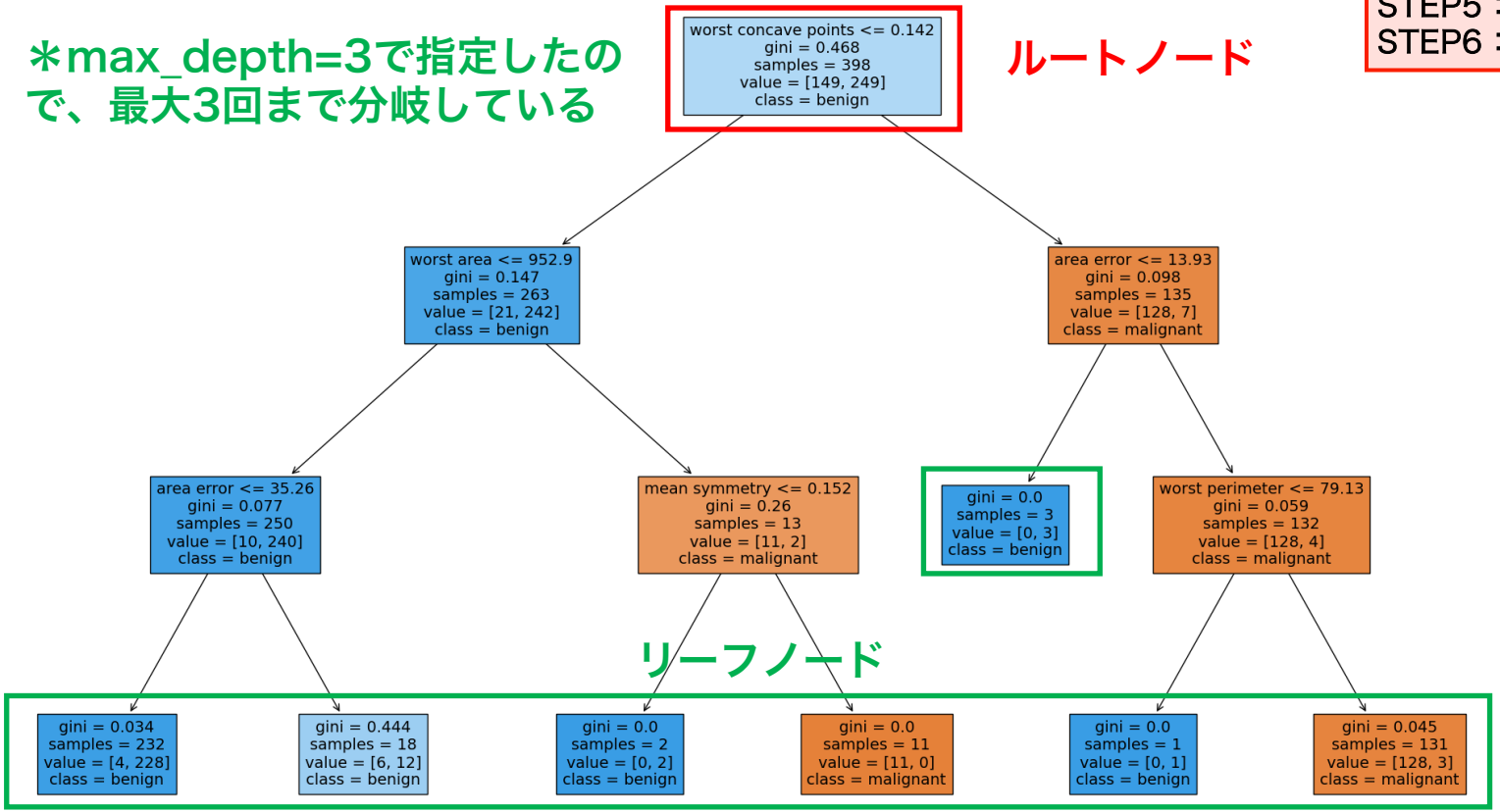


決定木分析 -剪定モデル-

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

STEP4：決定木の図示

＊max_depth=3で指定したので、最大3回まで分岐している



決定木分析 -剪定モデル-

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

STEP5：予測を行う

コード13-17 剪定モデルの混同行列の出力

```
confusion_matrix(y_test, model_tree_p.predict(x_test),
                 labels = [1, 0])
```

➡
```
array([[103,   5],
       [  4,  59]])
```

ライブラリ名　モジュール名　関数名

●関数をインポート `from sklearn.metrics import confusion_matrix`

● `confusion_matrix(実際の分類,予測分類,labels=[])`で混同行列を出力する。
引数`labels=`で出力の順番（1：良性腫瘍、0：悪性腫瘍）を指定する

決定木分析 -剪定モデル-

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

STEP5：予測を行う

```
array([[103,   5],
       [  4,  59]])
```

		予測結果	
		positive(正)	Negative(負)
実際の分類結果	positive (正)	真陽性True Positive **103**	偽陰性False Negative **5**
	Negative (負)	偽陽性False Positive **4**	真陰性True Negative **59**

colab

決定木分析 -剪定モデル-

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

STEP6：モデルの評価

コード13-18 depth=3のモデルの性能評価

（モデル名）.scoreで分類の時は正解率が出力される

```
print(model_tree_p.score(x_test, y_test))
```

0.9473684210526315 ← 剪定前の検証データ0.91から増加

```
print(model_tree_p.score(x_train y_train))
```

0.9673366834170855 ← 剪定前の学習データ1.0から減少

過学習を抑制できている

決定木分析 -剪定モデル-

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

STEP6：モデルの評価

コード13-19 検証用データでAUCを算出

```
roc_auc_score(y_test, model_tree_p.predict_proba(x_test)[:,1])
```

0.9431951793062904

決定木（剪定）の方が剪定前より
予測性能が高く出力された

ロジスティック回帰のAUC：0.982363315696649
剪定していない決定木のAUC：0.9173280423280423

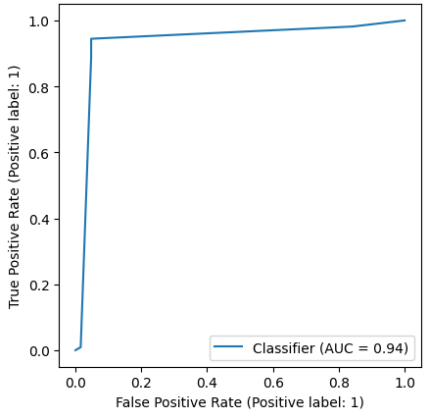
決定木分析 -剪定モデル-

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

STEP6：モデルの評価

コード13-20 ROC曲線の描出

```
from sklearn.metrics import RocCurveDisplay
RocCurveDisplay.from_predictions(y_test,
   model_tree_p.predict_proba(x_test)[:,1])
plt.axis('square')
plt.show()
```



*RocCurveDisplay.from_prediction(実際の分類, Y=1になる確率)でROC曲線を描出できる
*plt.axis('square')で正方形の図に指定
*plt.show()で図を表示

colab

決定木分析 -剪定モデル-

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

STEP6：モデルの評価

コード13-21 検証用データでPrecision(適合率)、Recall（再現率)、F1値を算出

```
from sklearn.metrics import classification_report
pd.DataFrame(classification_report(y_test,
   model_tree_p.predict(x_test), output_dict = True))
```

	0	1	accuracy	macro avg	weighted avg
precision	0.921875	0.962617	0.947368	0.942246	0.947607
recall	0.936508	0.953704	0.947368	0.945106	0.947368
f1-score	0.929134	0.958140	0.947368	0.943637	0.947453
support	63.000000	108.000000	0.947368	171.000000	171.000000

決定木分析 -剪定モデル-

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：決定木の図示
STEP5：予測を行う
STEP6：モデルの評価

STEP6：モデルの評価

検証用データでPrecision(適合率)、Recall（再現率)、F1値を算出

	0	1	accuracy	macro avg	weighted avg
precision	0.921875	0.962617	0.947368	0.942246	0.947607
recall	0.936508	0.953704	0.947368	0.945106	0.947368
f1-score	0.929134	0.958140	0.947368	0.943637	0.947453
support	63.000000	108.000000	0.947368	171.000000	171.000000

SVC

	0	1	accuracy	macro avg	weighted avg
precision	0.980769	0.899160	0.923977	0.939964	0.929226
recall	0.809524	0.990741	0.923977	0.900132	0.923977
f1-score	0.886957	0.942731	0.923977	0.914844	0.922183
support	63.000000	108.000000	0.923977	171.000000	171.000000

演習13：課題

Webclassで課題を提出してください。締め切りは2024/02/07 23:59まで

DecisionTreeClassifier() で剪定を行い、STEP2~STEP6まで実行してください。

max_depth=3で作った決定木（正解率0.94736、AUC：0.94319)より分類性能(検証データにおける正解率かAUCのどちらか）が優れた決定木を作成するように、剪定をしてください。

剪定は max_depth=, と min_samples_split= で自分で自由に数値を指定して行ってください。random_state=0に指定してください。

- webclassにはあなたが作成した優れた決定木を作れる(1)max_depthと
  (2)min_samples_splitで指定した数値を入力してください。
  (3)優れた指標は正解率かAUCのどちらか、(4)(3)で選んだ指標の値を入力してください
  *正解率もAUCも優れたモデルになった場合は、AUCを記載してください

演習13：課題

Webclassで課題を提出してください。締め切りは2024/02/07 23:59まで

- 決定木について理解を問う選択問題が二問あります。正しい選択肢を一つwebclassで選択してください。

(5) 問題1：決定木の分割を行うのに使用される指標はなんですか？

1. AUC
2. ジニ不純度
3. 平均二乗誤差
4. 適合率

(6) 問題2：決定木の剪定の目的は何ですか？

1. 過学習を防ぐため
2. 未学習を防ぐため
3. 学習時間を短縮するため
4. 適切な特徴量を選ぶため

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

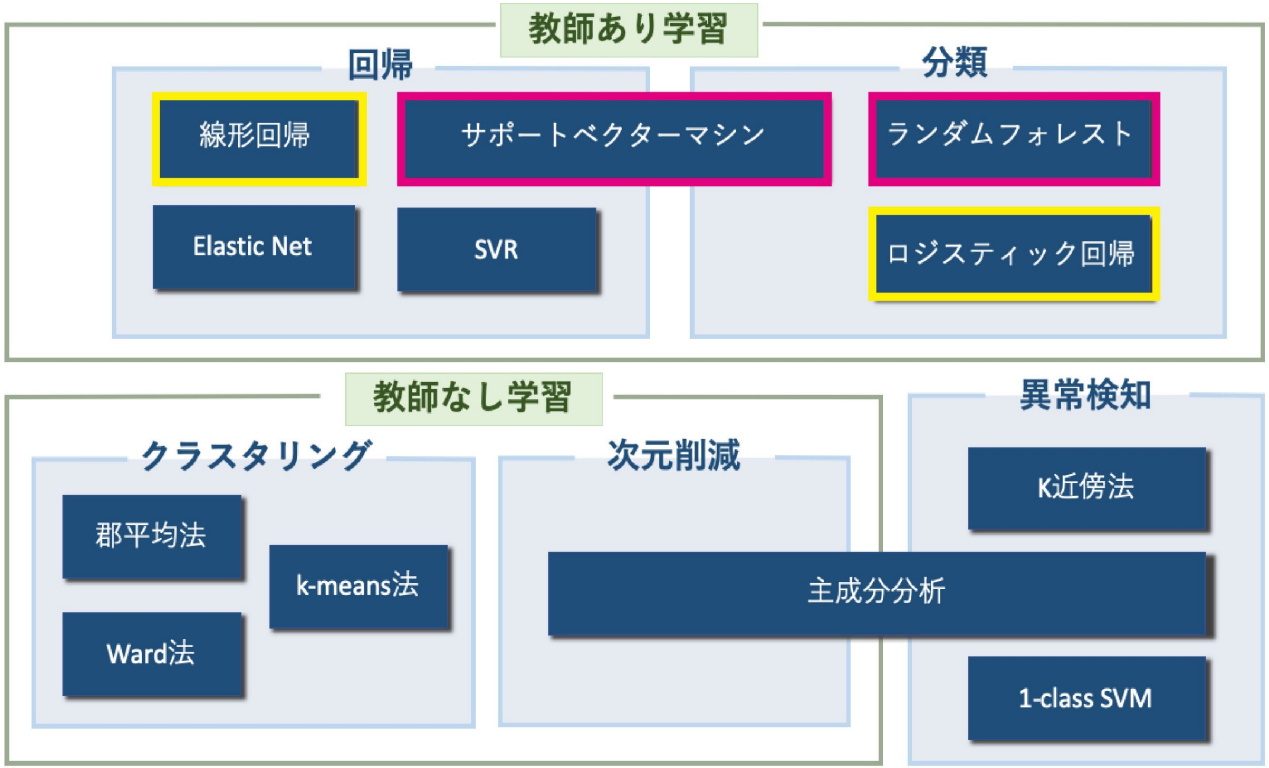
2024/01/25　10:40-11:25

医療とAI・ビッグデータ入門
演習14

教師あり機械学習
-ランダムフォレスト-

統合教育機構
石丸美穂

関数 y=f(x) の f の設定次第で、機械学習にはさまざまなアルゴリズムがある



前回学んだ手法

今回学ぶ手法

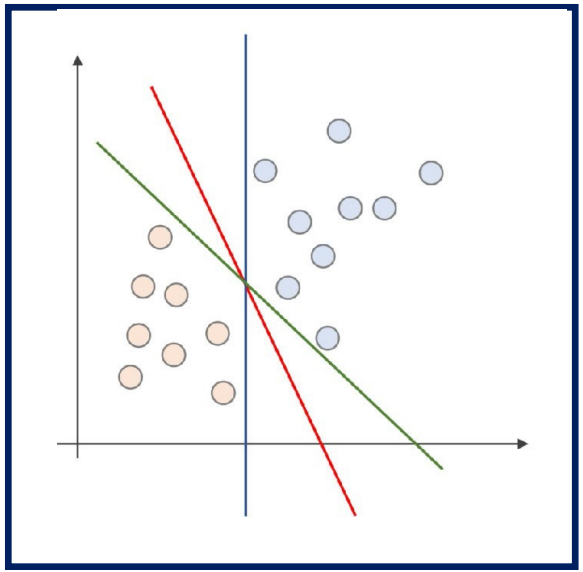
ここに挙げた代表的な手法のうち、
今回はランダムフォレストに代表
される決定木ベースの手法と、

2010 年くらいまでの最強の手法、
サポートベクターマシンを紹介する。

https://ainow.ai/2019/11/26/180809/

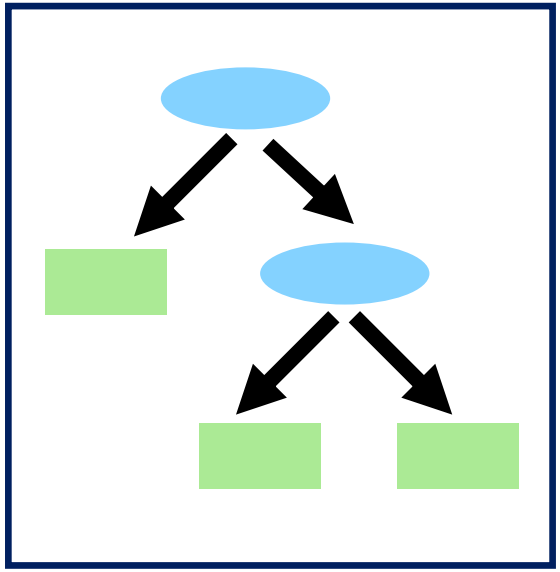
教師あり学習の種類

1.サポートベクターマシン



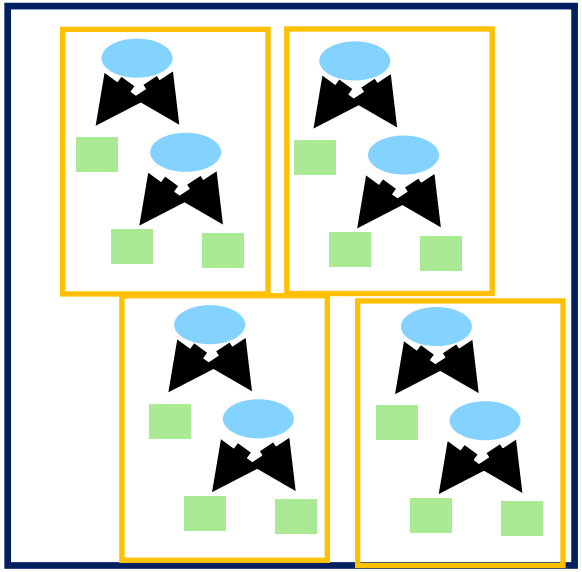
演習12

2.決定木



演習13

3. ランダムフォレスト



演習14

アンサンブル学習：「三人寄れば文殊の知恵」作戦

決定木は、解釈しやすいというメリットがある反面、精度はそれほど高くないというデメリットもあった。

単独で使うと精度が低い学習器のことを弱学習器 (weak learner) というが、だからこそたくさんの弱学習器を集めて「多数決」なり「平均」なり集団として使うことで精度を大きく上げることができる。これをアンサンブル (ensemble) 学習という。
（音楽の用語で二人以上が同時に演奏すること）

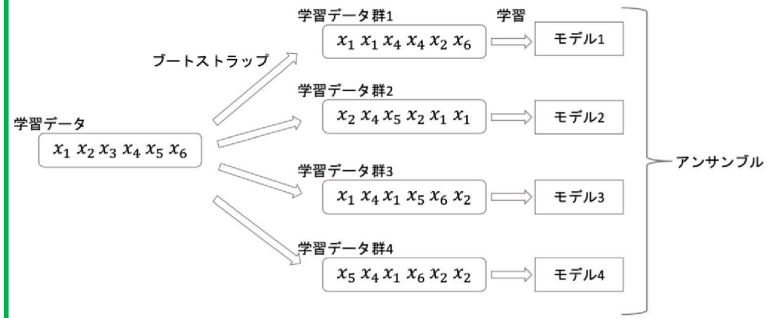
どのようにアンサンブルするか、によって、バギング、ブースティング、スタッキングという 3 つの考え方がある。
そのうち前者 2 つを取り上げる。

アンサンブルのやり方 1: バギング

バギング (bagging) は **b**ootstrap **agg**rega**t**i**ng** の略。

ブートストラップ (bootstrap) というのは，母集団から重複を許してランダムにデータを取って標本にするやり方。

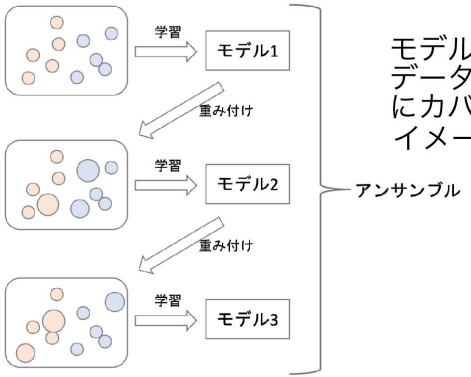
学習データから新たに学習データ群を複数作って
それぞれの学習データ群でモデルを学習させる。



決定木にバギング法を適用したのがランダムフォレスト

アンサンブルのやり方 2: ブースティング

ブースティング (boosting) は，バギングのように並列に複数のモデルを学習するのではなく，直列にモデルを学習していく。学習したモデルがうまく予測できなかった学習データに重みをつけてさらにモデルを学習するというのを繰り返していくイメージ。



モデルがうまく予測できなかったデータは、次のモデルがより重点的にカバーできるよう大きな重みがつくイメージ

決定木にブースティング法を適用したのが勾配ブースティング決定木であり、代表的な機械学習手法として XGBoost や LightGBM がある。

(清水先生の講義スライドより引用)

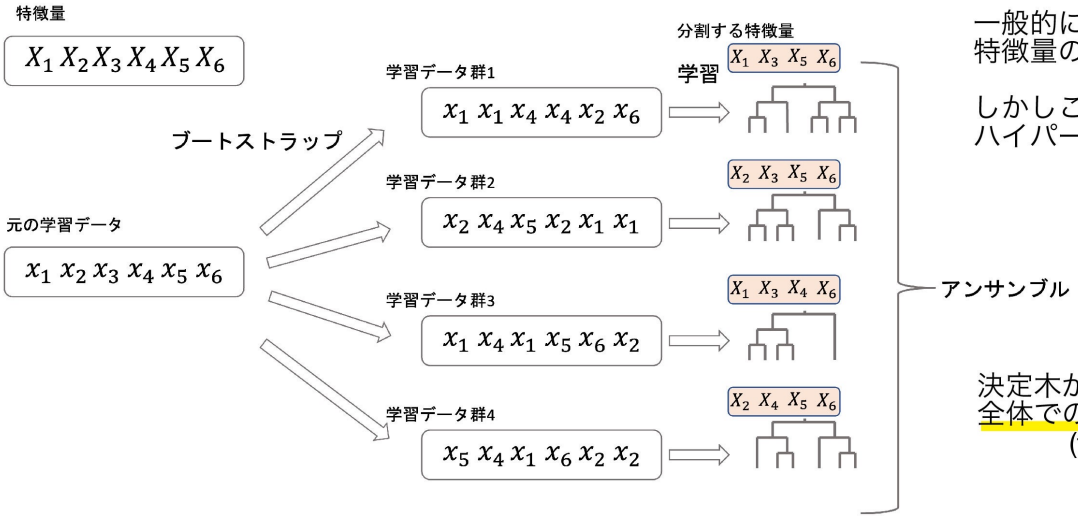
ランダムフォレスト

決定木のアンサンブル (正確にはバギング) を行ったのが、ランダムフォレスト (random forest) と呼ばれる手法である。

バギングは，ブートストラップ法を使ってサンプル抽出した複数のデータ群に対してそれぞれモデルを構築して，最後に平均や多数決を取る方法だが、ランダムフォレストにはさらに工夫がある。

それは、それぞれの決定木において**ランダム**に選んだ一部の特徴量のみを使って分割を行うようにしているのだ (これが名前の由来)。

つまり，それぞれの決定木を構築する際に、「一部の学習データを使わない」と「一部の特徴量でしか分割を行わない」の 2 つを行うことで、少しずつ違う決定木を作ろうとしている。



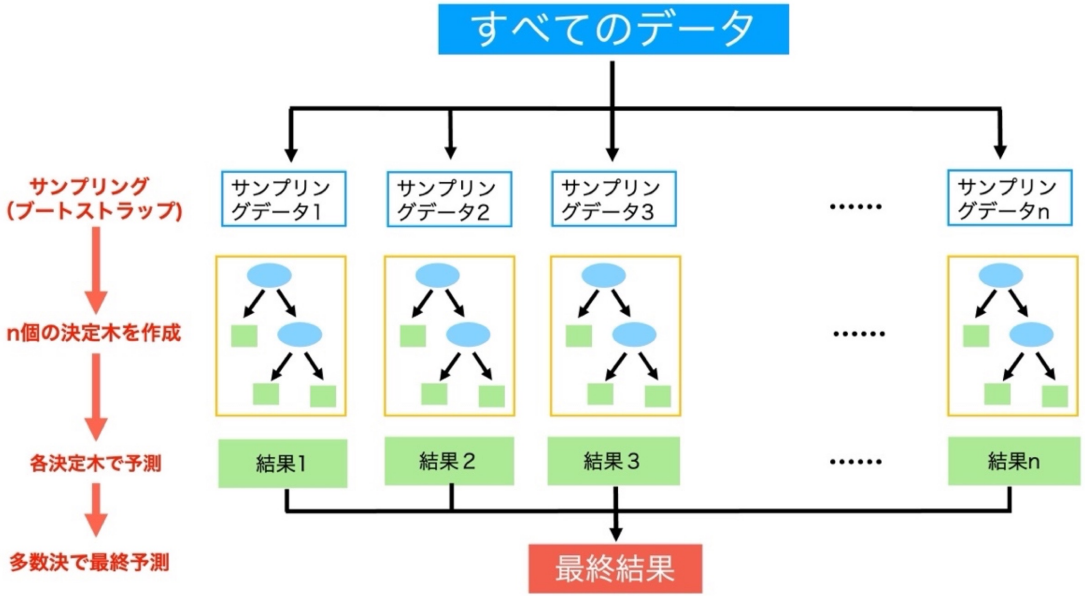
一般的には、決定木の数は 100、分割に使う特徴量の数は利用可能な特徴量数の平方根をとったもの。

しかしこれらの値はモデルを作る人が自分で設定するべきハイパーパラメーター (hyper parameter) である。

決定木がたくさんあるので各決定木の詳細を見るのは難しいが，全体での特徴量の重要度を確認することができる。
(feature importance)

(清水先生の講義スライドより引用)

ランダムフォレスト (random forest)



1. すべてのデータから、n個のサンプリングデータセットを作成する
2. n個の決定木を作成する（この時に特徴量もランダムに選択）
3. 各決定木モデルで予測
4. 多数決で最終予測を行う

ランダムフォレスト (random forest)

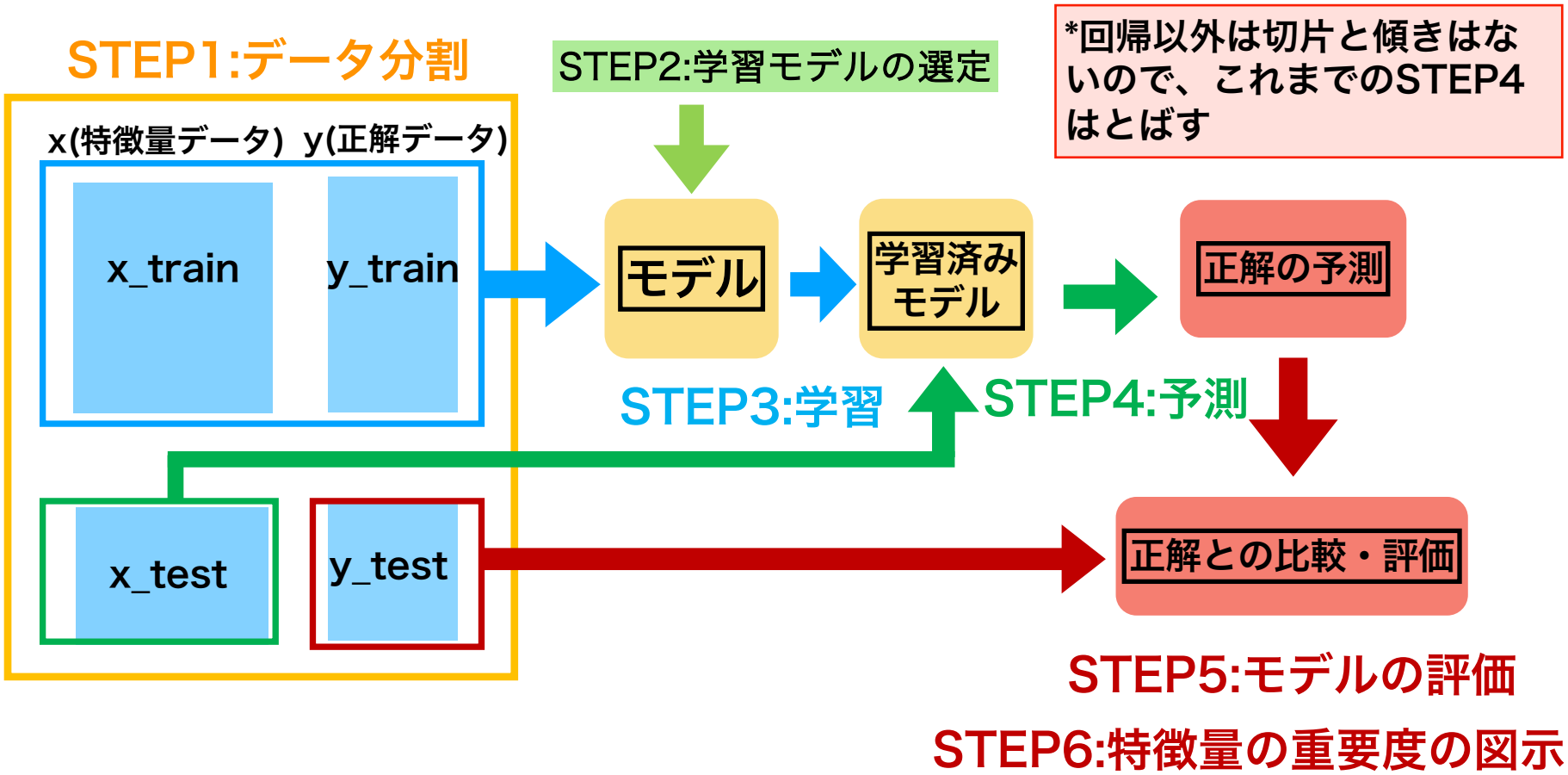
- 回帰・SVM・決定木分析は一つのモデルを作成していた
- 単一のモデルでどれが最適かを選ぶのは難しく、また精度がそれほど高くないこともある
- アンサンブル学習は複数のモデルを作成し、最終予測を行うことで精度を高めることができる

演習

ランダムフォレスト

乳がんのデータを用いて、分類を行う

機械学習の流れのまとめ



ランダムフォレストのコードのまとめ

STEP1　データの用意

STEP2　学習モデルの選択(今回は決定木)

(モデル名) = RandomForestClassifier()

STEP3　データを入れて学習させる

(モデル名).fit(特徴量,予測値)

モデルは
RandomForesteClassifier()

STEP4　予測を行う：(モデル名).predict()

STEP5　モデルの評価：(モデル名).score(特徴量,予測値)

：roc_auc_score(実測値,Y=1になる確率)

STEP6　特徴量の重要度の図示

特徴量重要度の図示を行う

ランダムフォレストのコードまとめ

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 = True)
```

STEP2：学習モデルの選択

```
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)
```

```
from sklearn.ensemble import RandomForestclassifier
model_forest = RandomForestClassifier(n_estimators = 100, max_depth = 3, max_features = 5,
random_state = 0)
```

STEP3：データを入れて学習

```
model_forest.fit(x_train, y_train)
```

STEP4： 予測

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, model_forest.predict(x_test), labels = [1,0])
```

STEP5：モデルの評価

```
print(model_forest.score(x_test, y_test))
```

```
from sklearn.metrics import roc_auc_score
roc_auc_score(y_test, modelforest.predict_proba(x_test)[:,1])
```

STEP6：特徴量重要度

```
forest_importances = pd.DataFrame(model_forest.feature_importances_, index =
bc.feature_names, columns = ['Importance'])
forest_importances.plot.bar()
```

ランダムフォレスト

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：予測を行う
STEP5：モデルの評価
STEP6：特徴量重要度の図示

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14コード.ipynb

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

ランダムフォレスト

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：予測を行う
STEP5：モデルの評価
STEP6：特徴量重要度の図示

STEP0：ライブラリのインポート

コード14-1 ライブラリとモジュールをインポート

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

- いつも使うnumpyとpandasライブラリ、pyplotモジュールをインポートする

ライブラリをインポート：import （ライブラリ名） as 省略形

numpyとpandas

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

ランダムフォレスト

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：予測を行う
STEP5：モデルの評価
STEP6：特徴量重要度の図示

STEP1：データの用意

コード14-2 乳がんデータの読み込み

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

今回は分類を行うため、breast_cancer(乳がん)のデータを読み込む

ライブラリ　モジュール　関数

関数をインポート　`from sklearn.datasets import load_breast_cancer`

● `load_breast_cancer(as_frame = True/False)` の引数で、True(真)かFalse(偽)を選択する

読み込むデータの形式を指定
True → pdのデータフレーム型
False→ numpy配列

ランダムフォレスト

STEP1：データの用意

コード14-3 学習データと検証データに分割

*最初から学習データセットと検証データセットに分けてモデルの構築と評価を行う

```
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)
```

- trainデータ70%とtestデータ30%にデータを分割
- 分割時の乱数シード値を"0"に指定
- x_bcとy_bcを作成せず、bc.data, bc.targetをtrain_test_splitに直接入れてデータ分割を行う
- SVM・決定木と同様に、特徴量は30個全て使うことができる

ランダムフォレスト

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：予測を行う
STEP5：モデルの評価
STEP6：特徴量重要度の図示

STEP2：学習モデルの選択

コード14-4 学習モデルにランダムフォレストを選択

```
from sklearn.ensemble import RandomForestClassifier
```

クラスをインポート　`from sklearn.ensemble import RandomForestClasifier`

ライブラリ名　モジュール名　クラス名

- 今回のモデルはランダムフォレストで分類を行うので、
  `RandomForestClassifier()`を選択する
- `RandomForestClassifier()`クラスから`model_forest`を新しいインスタンスとして作成する

ランダムフォレスト

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：予測を行う
STEP5：モデルの評価
STEP6：特徴量重要度の図示

STEP2：学習モデルの選択

コード14-5 ランダムフォレストの引数指定

```
model_forest = RandomForestClassifier(n_estimators = 100,
 max_depth = 4, max_features = 5,  random_state = 0)
```

- 引数 n_estimators = 100 で作成する決定木の数を指定
- 引数 max_depth = 4 で決定木の最大深さを指定
- 引数 max_features = で一つのモデルに入れる特徴量の最大値を指定

一般的には、決定木の数は 100、分割に使う
特徴量の数は利用可能な特徴量数の平方根をとったもの。

しかしこれらの値はモデルを作る人が自分で設定するべき
ハイパーパラメーター (hyper parameter) である。

(清水先生の講義資料より引用)

$\sqrt{30} = 5.48 \rightarrow 5$ に指定

ランダムフォレスト

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：予測を行う
STEP5：モデルの評価
STEP6：特徴量重要度の図示

```
model_forest = RandomForestClassifier(n_estimators = 100,
    max_depth = 4, max_features = 5, random_state = 0)
```

- random_state = 0 で乱数シード値を固定

木を構築するときに使用されるサンプルのブートストラップのランダム性と、各ノードで最良の分割を探すときに考慮する特徴のサンプリングの両方に関係

＊指定していないハイパーパラメータはデフォルト設定のものを使う

- criterion = ”gini”
- min_samples_split = 2

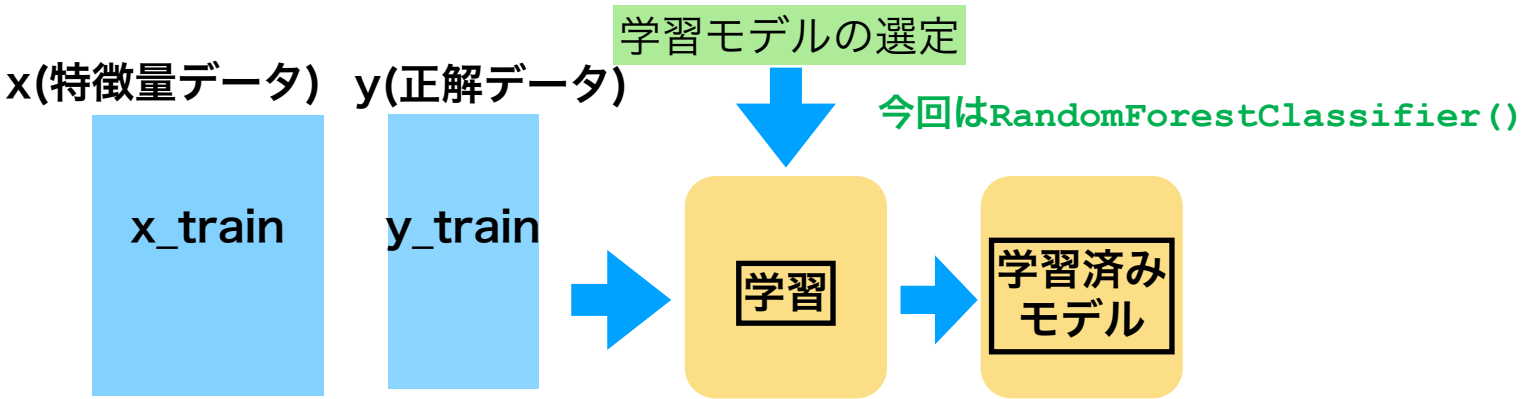
ランダムフォレスト

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：予測を行う
STEP5：モデルの評価
STEP6：特徴量重要度の図示

STEP3：データを入れて学習させる

コード14-6 学習用データで学習させる

```
model_forest.fit(x_train, y_train)
```



ランダムフォレスト

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：予測を行う
STEP5：モデルの評価
STEP6：特徴量重要度の図示

STEP4：予測を行う

コード14-7　検証用データを用いて予測する

```
print(model_forest.predict(x_test))

print(np.array(y_test))
```

```
[0 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 0 1 0 0 0 0 0 0 1 1 0 1 1 0 1 0 1 0 1 0 1 0 1
 0 1 0 1 1 0 1 1 0 1 1 1 0 0 0 0 1 1 1 1 1 1 0 0 0 1 1 0 1 0 0 0 1 1 0 1 1 1
 0 1 1 1 1 1 0 0 0 1 0 1 1 1 0 0 1 0 1 0 1 1 0 1 1 1 1 1 1 1 1 0 1 0 1 1 0 1
 0 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 0 1 0
 1 1 1 0 1 1 1 0 1 1 1 0 0 1 1 0 1 0 0 0 1 1 1]
[0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 1 1 0 1 1 0 1 0 1 0 1 0 1 0 1
 0 1 0 0 1 0 1 1 0 1 1 1 0 0 0 0 1 1 1 1 1 1 0 0 0 1 1 0 1 0 0 0 1 1 0 1 0 1 0
 0 1 1 1 1 1 0 0 0 1 0 1 1 1 0 0 1 0 1 0 1 1 0 1 1 1 1 1 1 1 1 0 1 0 1 0 0 1
 0 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 0 1 0
 1 1 1 1 1 1 1 0 1 0 1 0 0 1 1 0 1 0 0 0 1 1 1]
```

x_testデータで予測した
yが0/1かの値

y_testの実際の値

ランダムフォレスト

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：予測を行う
STEP5：モデルの評価
STEP6：特徴量重要度の図示

STEP4：予測を行う

コード14-8 混同行列の出力

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, model_forest.predict(x_test),
  labels = [1, 0])
```

➡ array([[104, 4],
[3, 60]])

ライブラリ名　モジュール名　クラス名

●関数をインポート `from sklearn.metrics import confusion_matrix`

● `confusion_matrix(実際の分類,予測分類,labels=[])`で混同行列を出力する。
`labels`引数=で出力の順番(1:良性腫瘍、0:悪性腫瘍）を指定する

ランダムフォレスト

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：予測を行う
STEP5：モデルの評価
STEP6：特徴量重要度の図示

STEP4：予測を行う

```
array([[104,   4],
       [  3,  60]])
```

		予測結果	
		Positive (1:良性腫瘍)	Negative (0:悪性腫瘍)
実際の分類結果	Positive (1:良性腫瘍)	真陽性True Positive **104**	偽陰性False Negative **4**
	Negative (0:悪性腫瘍)	偽陽性False Positive **3**	真陰性True Negative **60**

colab

ランダムフォレスト

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：予測を行う
STEP5：モデルの評価
STEP6：特徴量重要度の図示

STEP5：モデルの評価

コード14-9 正解率の出力

```
print(model_forest.score(x_test, y_test))
```

➡ 0.9590643274853801 ⬅ 検証データの正解率

```
print(model_tree_c.score(x_train y_train))
```

➡ 0.992462311557789 ⬅ 学習データの正解率

（モデル名）.scoreで分類の時は正解率が出力される

ランダムフォレスト

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：予測を行う
STEP5：モデルの評価
STEP6：特徴量重要度の図示

STEP5：モデルの評価

コード14-10 検証用データでAUCを算出

```
from sklearn.metrics import roc_auc_score

roc_auc_score(y_test, model_forest.predict_proba(x_test)[:,1])
```

*roc_auc_score(実際の分類, Y=1になる確率)でAUCを算出できる

0.9951499118165785

ランダムフォレストが一番高い

ロジスティック回帰のAUC：0.982363315696649
剪定していない決定木のAUC：0.9173280423280423
剪定した決定木のAUC：0.9431951793062904

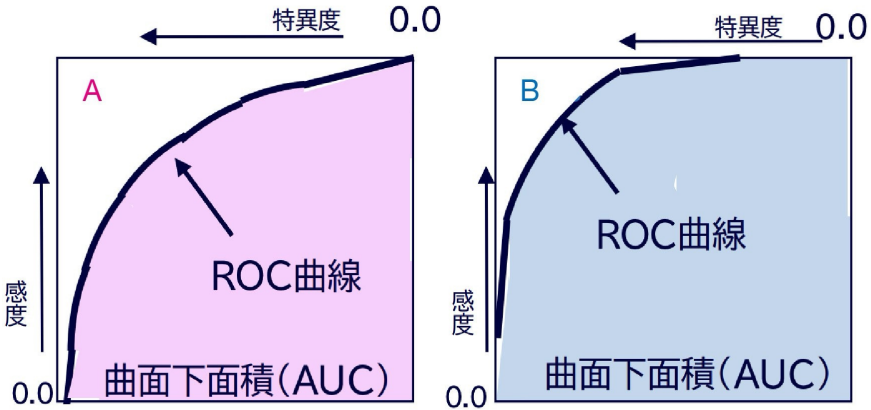
(清水先生の講義引用)

ROC 曲線と AUC による分類モデルの良し悪しの比較

ROC（Receiver operating characteristic）曲線はもともとレーダーの測定能力の評価に関する種々の研究や理論が1970 年以降に医療や機械学習へ応用されたものだ。

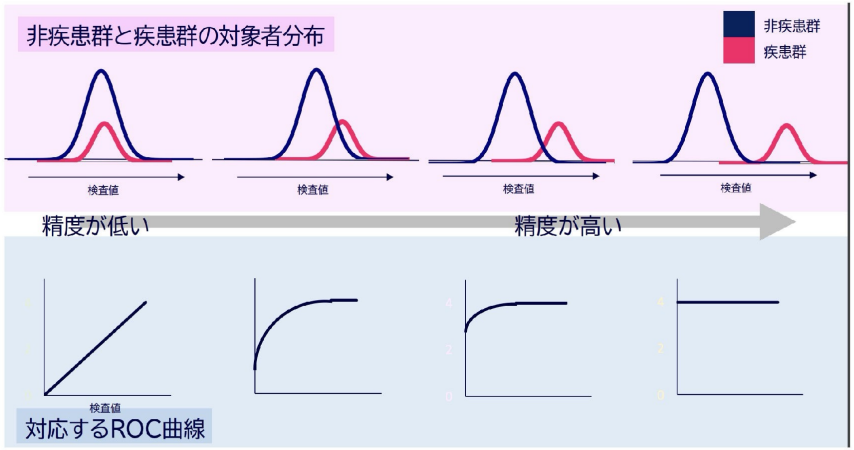
さまざまなカットオフで感度・特異度を計算し、横軸に 1 - 特異度、縦軸に感度をプロットしたものだ。
左上が感度 1、特異度 1 という理想的な分類器を示す。

ROC 曲線の下側の面積が AUC (Area under curve) である。ROC 曲線が左上にあればあるほど AUC は 1 に近い値をとる。
この AUC の値をもとに、どちらの分類器が優れるかを判断できる。



A と B だと、AUC が大きい B の方が優れるといえる。

ROC のイメージはこのようになる。
2 群を全く区別できない場合、ROC は対角線、AUC は 0.5 となる。



https://www.jsph.jp/covid/files/5BAA6E3.pdf より一部改変

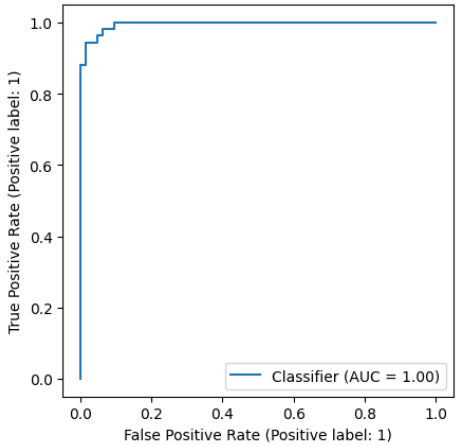
ランダムフォレスト

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：予測を行う
STEP5：モデルの評価
STEP6：特徴量重要度の図示

コード14-11 ROC曲線の描出

```
from sklearn.metrics import RocCurveDisplay
RocCurveDisplay.from_predictions(y_test,
   model_forest.predict_proba(x_test)[:,1])
plt.axis('square')
plt.show()
```



*RocCurveDisplay.from_prediction(実際の分類, Y=1になる確率)でROC曲線を描出できる
*plt.axis('square')で正方形の図に指定
*plt.show()で図を表示

colab

ランダムフォレスト

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：予測を行う
STEP5：モデルの評価
STEP6：特徴量重要度の図示

STEP5：モデルの評価

コード14-12 検証用データでPrecision(適合率)、Recall（再現率)、F1値を算出

```
from sklearn.metrics import classification_report
pd.DataFrame(classification_report(y_test,
   model_forest.predict(x_test), output_dict = True))
```

	0	1	accuracy	macro avg	weighted avg
precision	0.937500	0.971963	0.959064	0.954731	0.959266
recall	0.952381	0.962963	0.959064	0.957672	0.959064
f1-score	0.944882	0.967442	0.959064	0.956162	0.959130
support	63.000000	108.000000	0.959064	171.000000	171.000000

特徴量の重要度
Feature Importance

特徴量の重要度　Feature Importance

ランダムフォレストでは特徴量の重要度を算出できる

特徴量重要度：Mean Decrease Impurity(MDI)

その特徴量を使って分割した時の不純度の減少量の木々の平均

- 綺麗に分割できる特徴量ほど重要度が大きくなる、
- 値は特徴量全体から見た相対的な指標であり、すべての特徴量の重要度を足すと1になる

ランダムフォレスト

STEP6：特徴量重要度の図示

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：予測を行う
STEP5：モデルの評価
STEP6：特徴量重要度の図示

コード14-13 特徴量重要度を取得

```
print(model_forest.feature_importances_)
```

```
[0.0095214  0.01263454 0.01393639 0.06960096 0.00217414 0.
0.12990861 0.1690554  0.00140434 0.         0.05101289 0.         0.00981748
0.1344274  0.00302646 0.         0.00130498 0.00224143 0.00111216
0.00644376 0.11146473 0.00258438 0.08518957 0.00120382 0.
0.00445832 0.01797664 0.14089711 0.         0.01860311]
```

（モデル名）.feature_importances_で特徴量の重要度を取得できる

ランダムフォレスト

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：予測を行う
STEP5：モデルの評価
STEP6：特徴量重要度の図示

STEP6：特徴量重要度の図示

コード14-14　特徴量重要度を見やすくする

```
forest_importances = pd.DataFrame(
   model_forest.feature_importances_,
   index = bc.feature_names,
   columns=['Importance'])
```

- （モデル名）.feature_importances_で特徴量の重要度を取得
- DataFrame型にして、index=の引数で行名をbc.feature_namesで指定
- columns = で列名に名前をつける

ランダムフォレスト

STEP6：特徴量重要度の図示

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：予測を行う
STEP5：モデルの評価
STEP6：特徴量重要度の図示

コード14-15 特徴量重要度を見やすくする

```
forest_importances
```

- 出力すると右のように特徴量とその重要度のデータフレームが作成されている

	Importance
mean radius	0.023322
mean texture	0.009979
mean perimeter	0.066447
mean area	0.039999
mean smoothness	0.003435
mean compactness	0.005039
mean concavity	0.093138
mean concave points	0.136197
mean symmetry	0.001898
mean fractal dimension	0.002101
radius error	0.025128
texture error	0.002757

ランダムフォレスト

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：予測を行う
STEP5：モデルの評価
STEP6：特徴量重要度の図示

STEP6：特徴量重要度の図示

コード14-16 特徴量重要度を降順に表示

```
forest_importances.sort_values('Importance',
   ascending = False)
```

	Importance
worst perimeter	0.163678
worst concave points	0.150038
mean concave points	0.136197
worst radius	0.103167
mean concavity	0.093138
mean perimeter	0.066447
worst area	0.050634

（データフレーム名）.sort_values（変数名，
ascending = True/False）で変数名で並び替えて表示

表示の順序
True → 昇順 （0→大きい数値）
False→ 降順 （大きい数値→0）

ランダムフォレスト

STEP6：特徴量重要度の図示

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：予測を行う
STEP5：モデルの評価
STEP6：特徴量重要度の図示

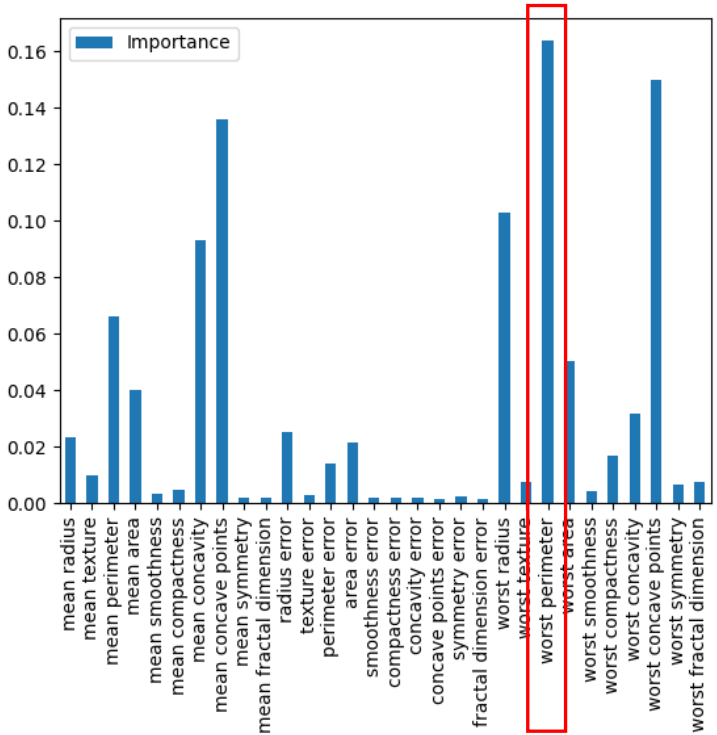
コード14-17 特徴量重要度の図示

```
forest_importances.plot.bar()
```

pandasのDataFrame型では

(データフレーム名).plot.bar()で

棒グラフを描出



まとめ

- 演習8-11では教師あり機械学習の回帰(線形回帰、ロジスティック回帰)
  演習12-14では教師あり機械学習のSVM, 決定木、ランダムフォレストを
  行いました

	回帰	SVM	決定木	ランダムフォレスト
利点	・モデルがわかりやすい ・調整するパラメータが少ない	・未知のデータにも予測性能が高い ・多くの特徴量を一度に調べられる	・分類の仕方がわかりやすい ・調整するパラメータが少ない	・アンサンブル学習であり、たまたまできたモデルに依拠しない ・予測性能が高い ・特徴量重要度を求めることができる
欠点	・データ自体の分布にうまく当てはめられないこともある	・モデルの中身がよくわからない ・特徴量自体の意味がなくなる	・過学習になりやすい	・データ数が少ないときに過学習になる

まとめ

- 同じ**乳がんデータの分類**に対して色々なアルゴリズムを実践してみたけれど、結局どれが予測性能が高いかは、実際のデータとの相性次第(アンサンブル学習は高くなりやすい)

- 今後、医療データで機械学習の分類を行うときには、どのモデルを使って行うのか、特徴を理解するのが大事

- また、Pythonで実践することはそれほど難しくなく、コードをポチポチ押すだけで行うことができる。しかし、ハイパーパラメーターのチューニングによっては全然違う結果になることもあり、内容を理解してチューニングを行う必要がある

次回からは、さらに複雑なモデルである深層学習(deep learning)を行う

演習14：課題

Webclassで課題を提出してください。締め切りは2024/02/08 23:59まで

RandomForestClassifier() でハイパーパラメータの調整を行い、STEP2~STEP6まで実行してください。

- 決定木の作成数を1000、木の最大深さ5、最大特徴量を6に設定してください。
- webclassに

(1)検証データの正解率、(2)学習データの正解率、(3)AUC　を回答してください。

特徴量重要度を求めてください。

(4)最も重要な特徴量、(5)その値　を回答してください。

演習14：課題

Webclassで課題を提出してください。締め切りは2024/02/08 23:59まで

- ランダムフォレストについて理解を問う選択問題が二問あります。正しい選択肢を一つwebclassで選択してください。

(6) 問題1：ランダムフォレストの主な利点は何ですか？

1.計算コストが非常に低い。
2.個々の決定木の性能が低くても、集合としては高い精度を達成できる。
3.トレーニングデータの少ない状況でも、常に高い精度を保証する。
4.ハイパーパラメータのチューニングが不要である。

(7) 問題2：ランダムフォレスト(分類)はどのように最終的な結果を予測を行いますか？

1.すべての決定木の予測を平均化する。
2.最も精度の高い決定木の予測を使用する。
3.すべての決定木の予測の中で最も多数を占めるクラスを最終的な予測とする。
4.それぞれの決定木が重み付けされ、重みが最も高い決定木の予測を使用する。

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15

深層学習

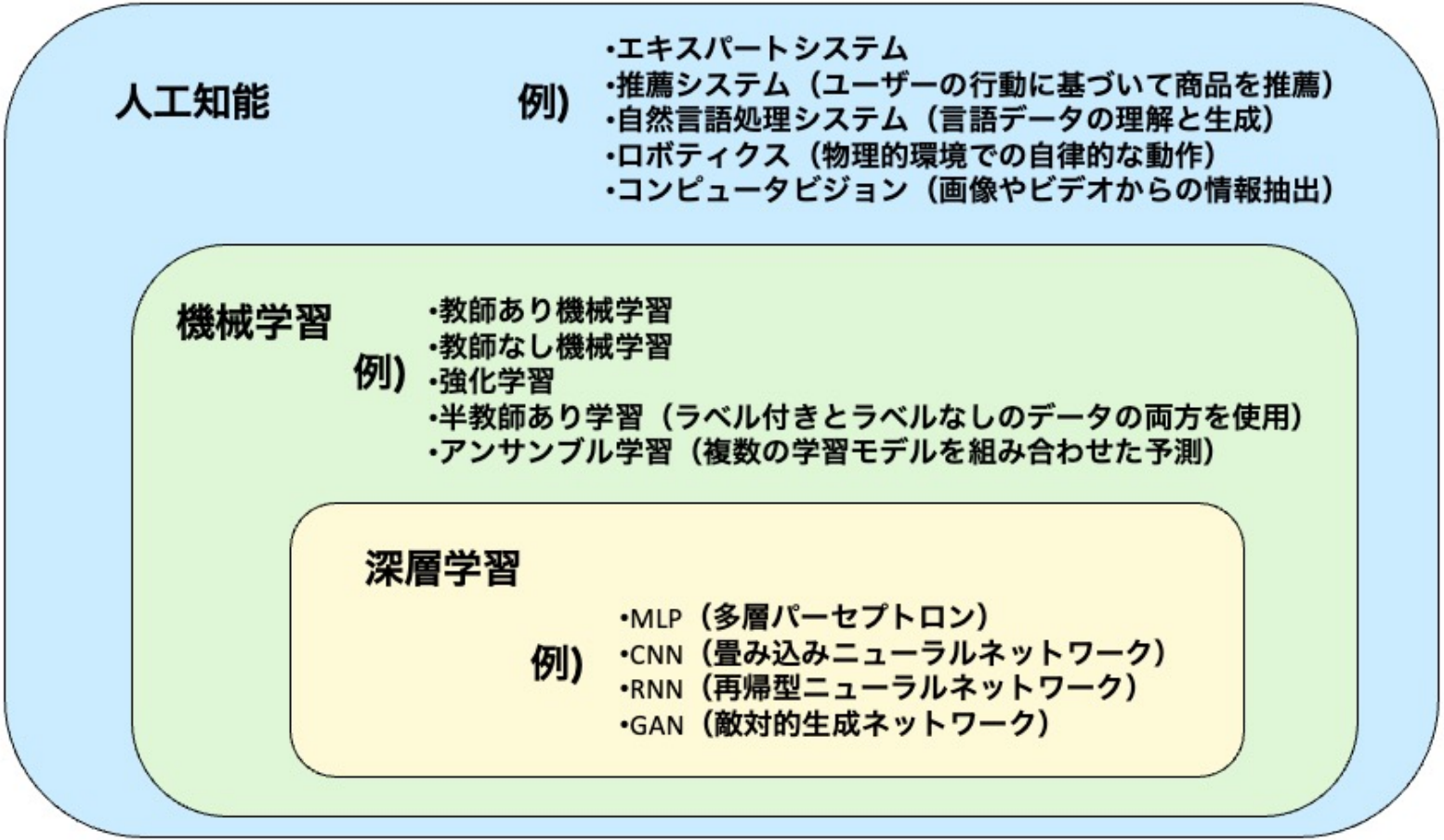
*本日演習16の授業後に複合領域コースの説明があります

演習15-20の概要

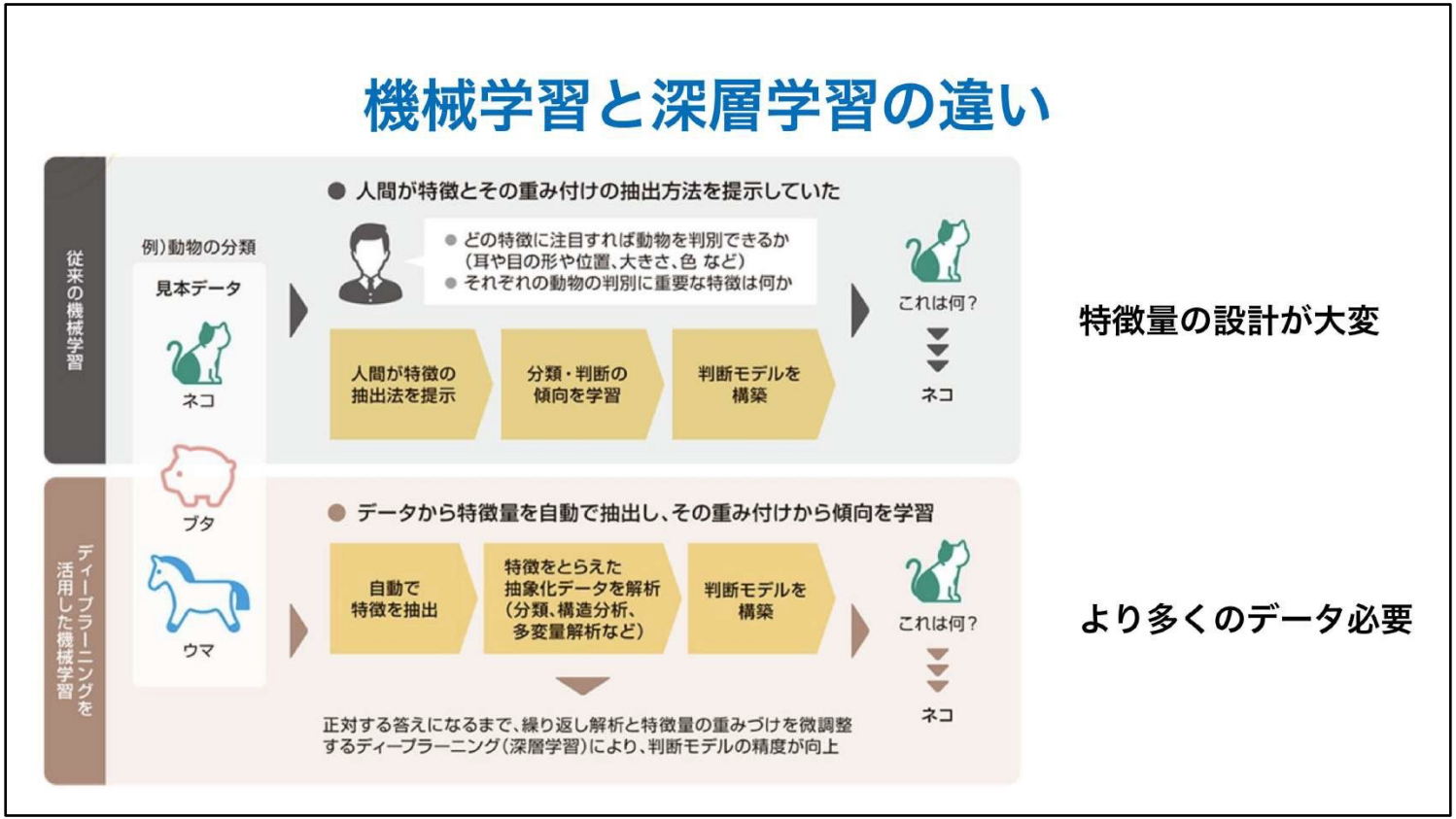
- 今までは教師あり機械学習の基礎を実行してきた
- 演習15-20では深層学習を実行する

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

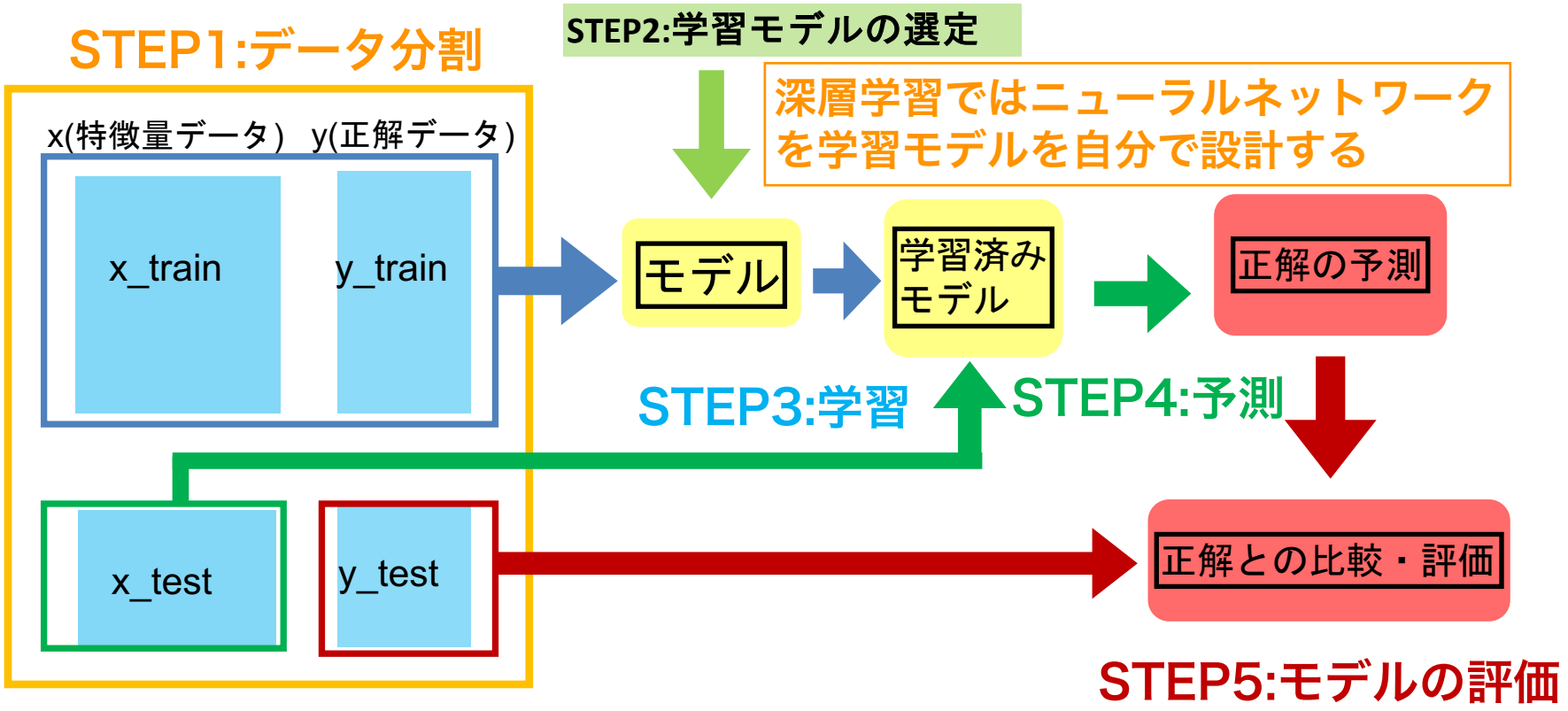
機械学習と深層学習の違い



機械学習と深層学習の違い

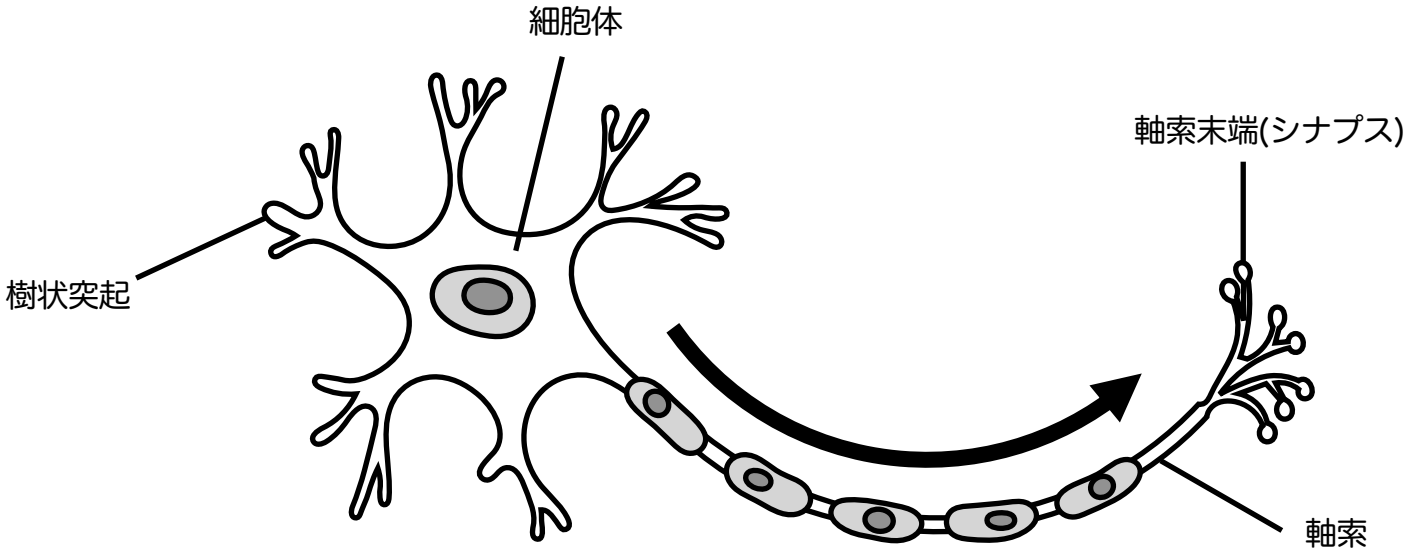


深層学習のコードの流れ



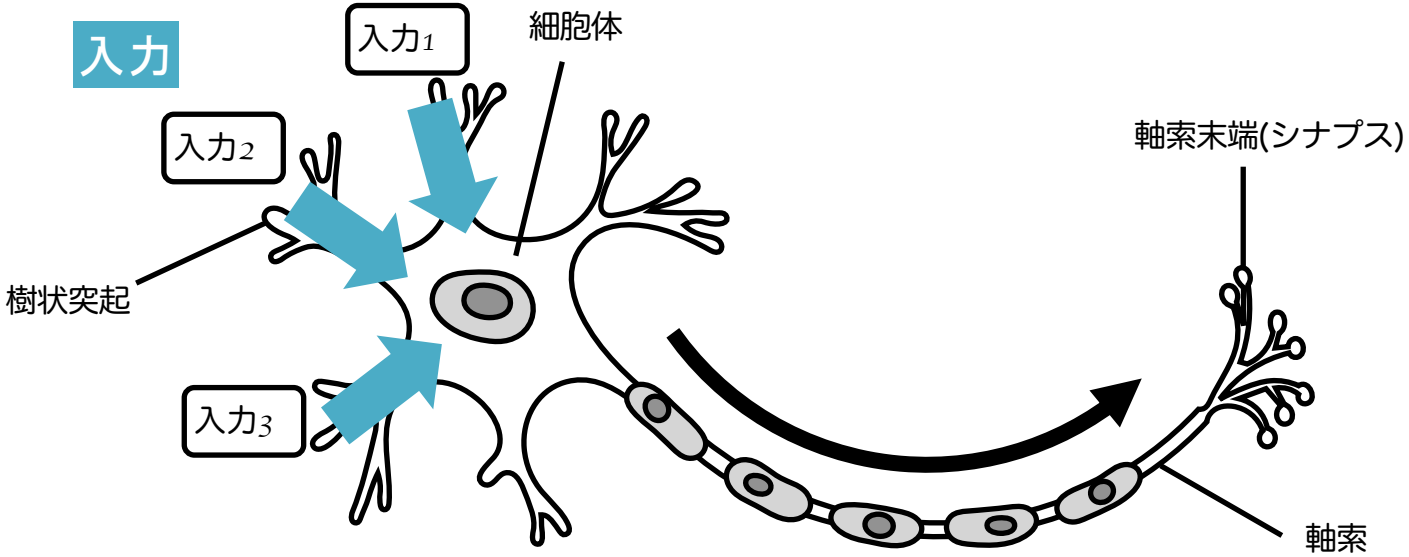
ニューラルネットワークとは

・ニューロンは、樹状突起、細胞体、軸索からなる
・ニューロンは、樹状突起から入力された電気信号が神経細胞内の電位を超えるかどうかの閾値を持っている
・閾値を超えるとニューロンは興奮状態となり、軸索末端から電気信号が出力される



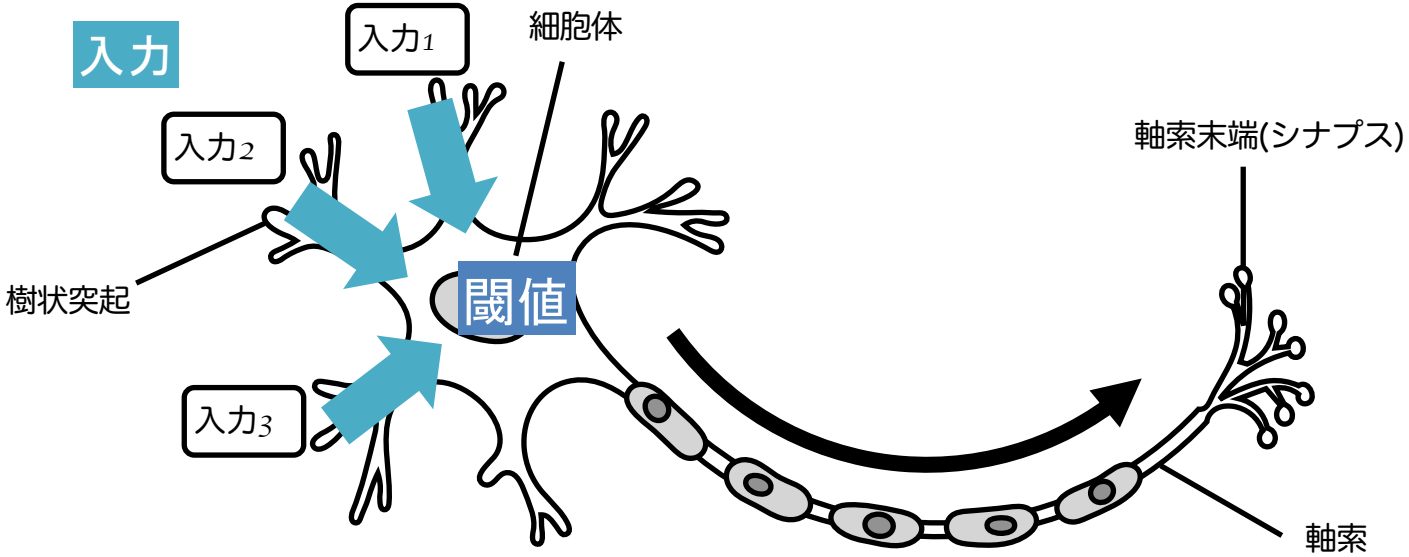
ニューラルネットワークとは

・ニューロンは、樹状突起、細胞体、軸索からなる
・ニューロンは、樹状突起から入力された電気信号が神経細胞内の電位を超えるかどうかの閾値を持っている
・閾値を超えるとニューロンは興奮状態となり、軸索末端から電気信号が出力される



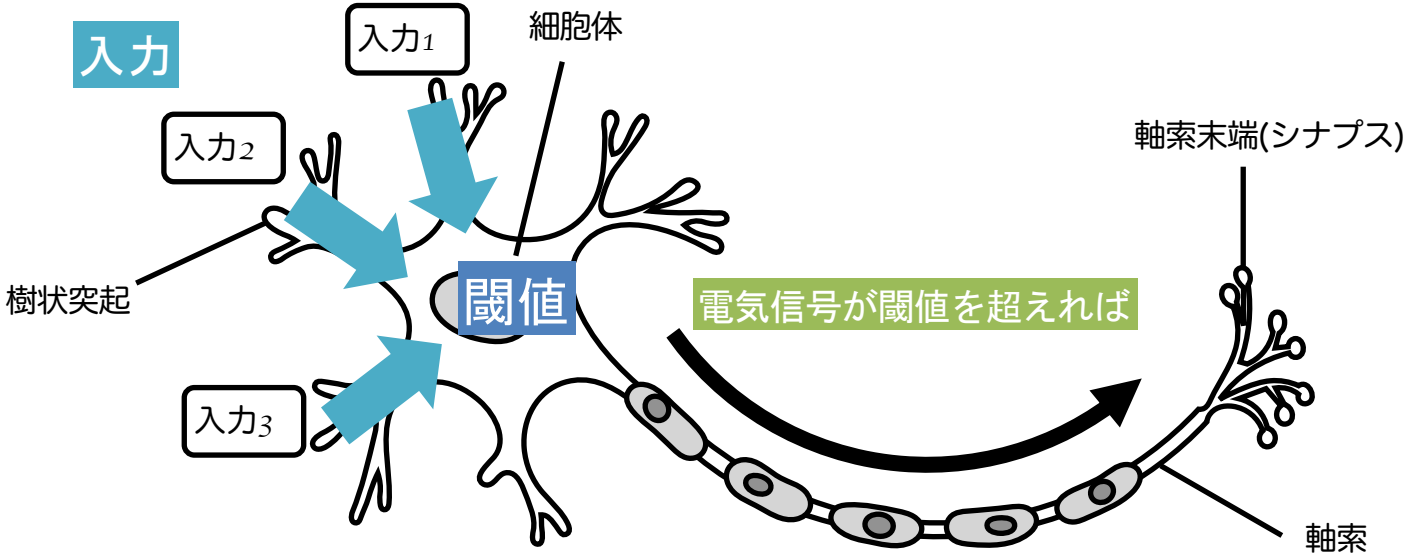
ニューラルネットワークとは

・ニューロンは、樹状突起、細胞体、軸索からなる
・ニューロンは、樹状突起から入力された電気信号が神経細胞内の電位を超えるかどうかの閾値を持っている
・閾値を超えるとニューロンは興奮状態となり、軸索末端から電気信号が出力される



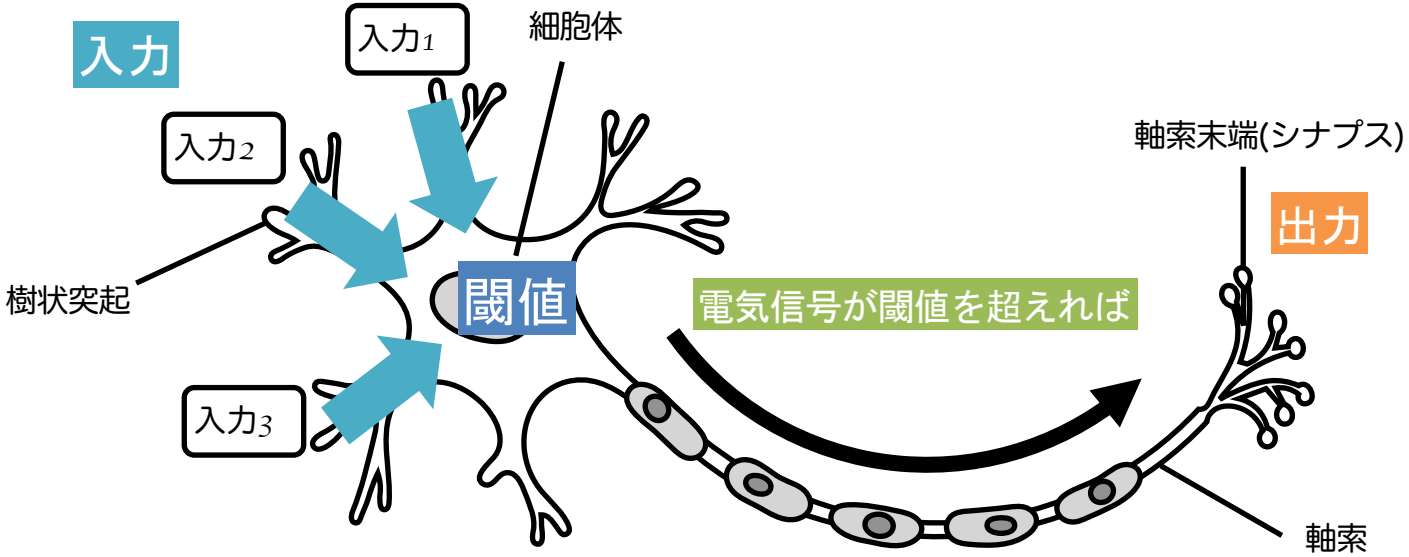
ニューラルネットワークとは

・ニューロンは、樹状突起、細胞体、軸索からなる
・ニューロンは、樹状突起から入力された電気信号が神経細胞内の電位を超えるかどうかの閾値を持っている
・閾値を超えるとニューロンは興奮状態となり、軸索末端から電気信号が出力される



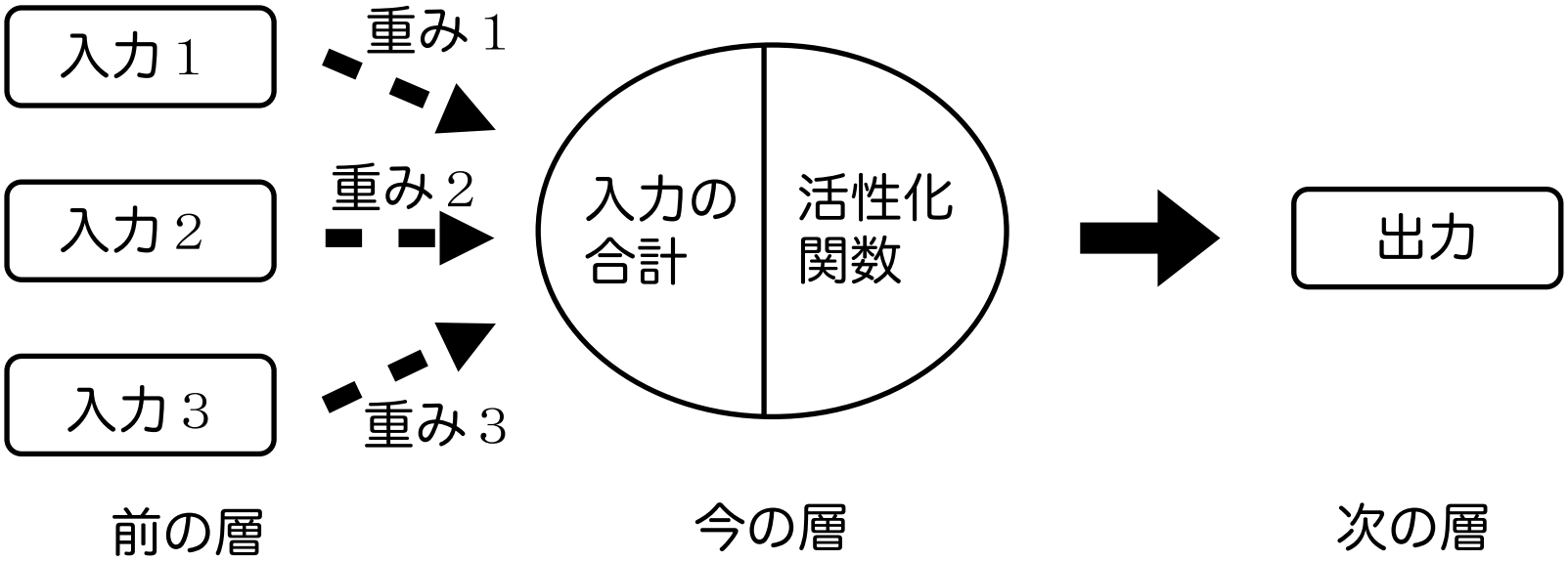
ニューラルネットワークとは

・ニューロンは、樹状突起、細胞体、軸索からなる
・ニューロンは、樹状突起から入力された電気信号が神経細胞内の電位を超えるかどうかの閾値を持っている
・閾値を超えるとニューロンは興奮状態となり、軸索末端から電気信号が出力される



ニューラルネットワークとは

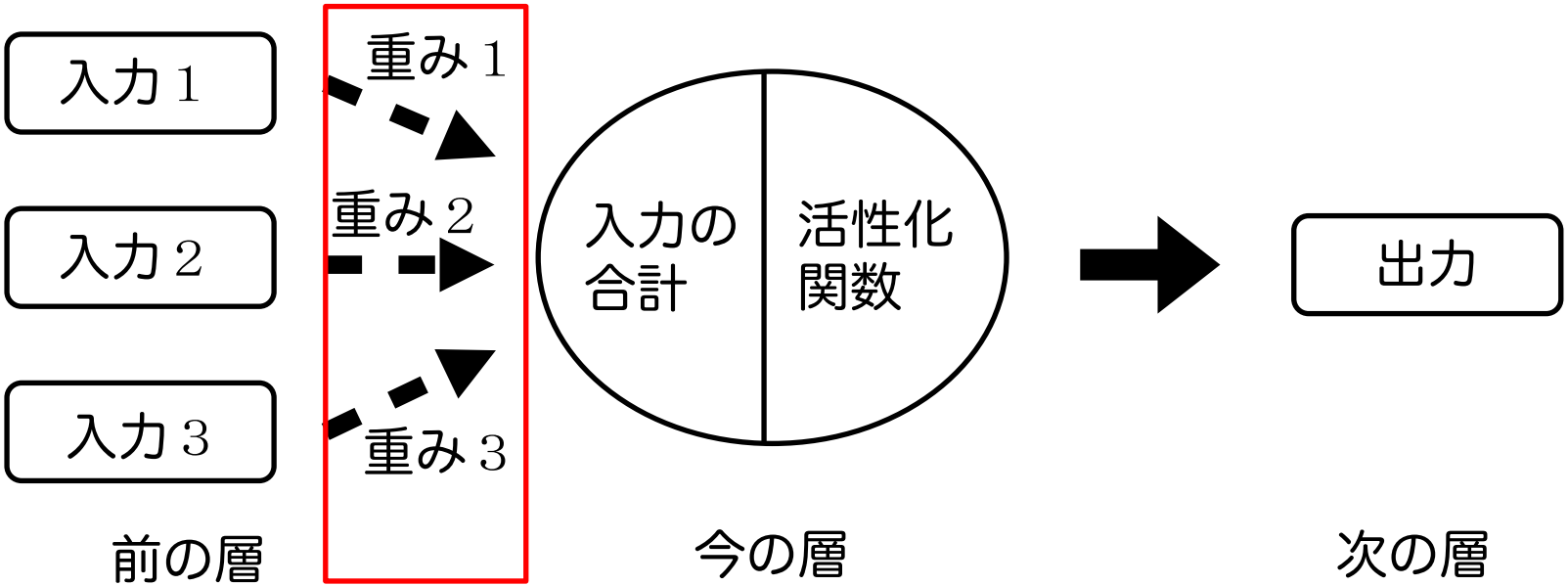
単一の人工ニューロンはこのようなモデルで表すことができる。



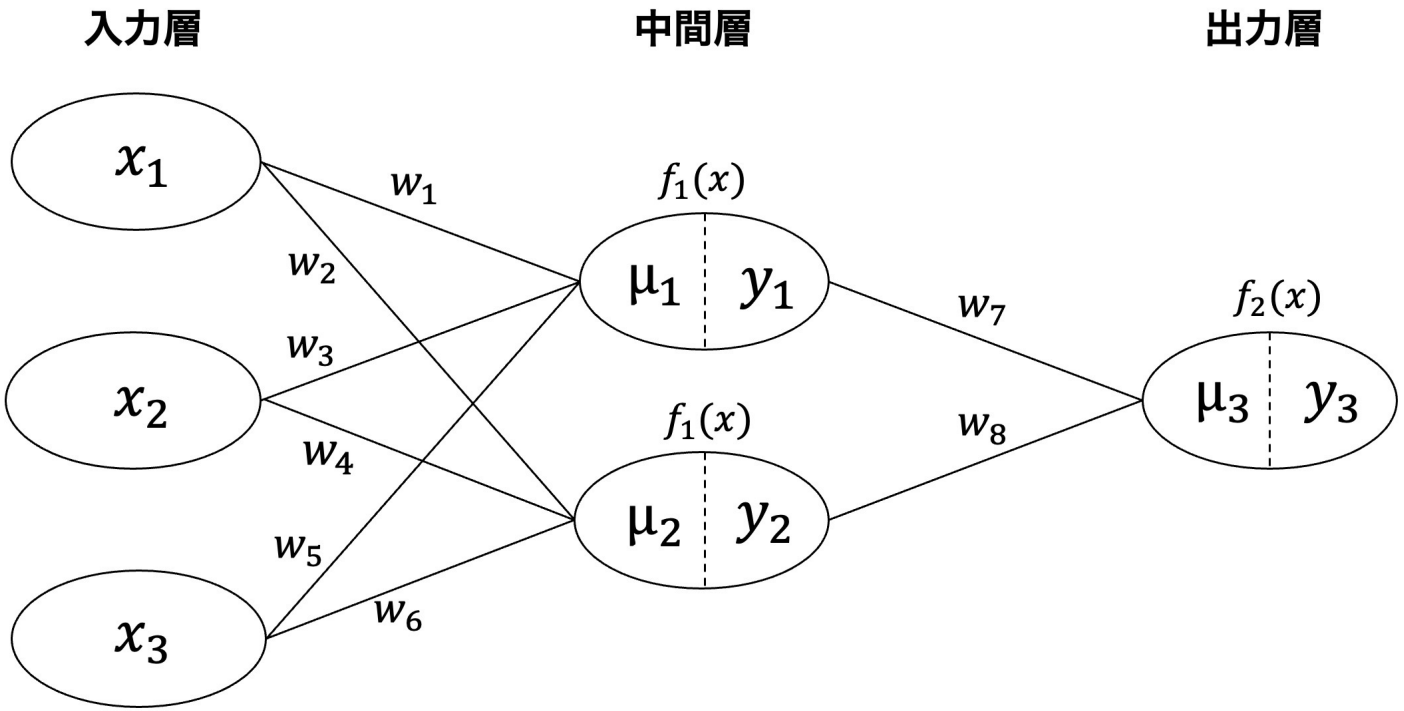
ニューラルネットワークとは

単一の人工ニューロンはこのようなモデルで表すことができる。

深層学習は、この重み(パラメーター)を最適化して出力(予測結果)を正解に近づけるように学習する

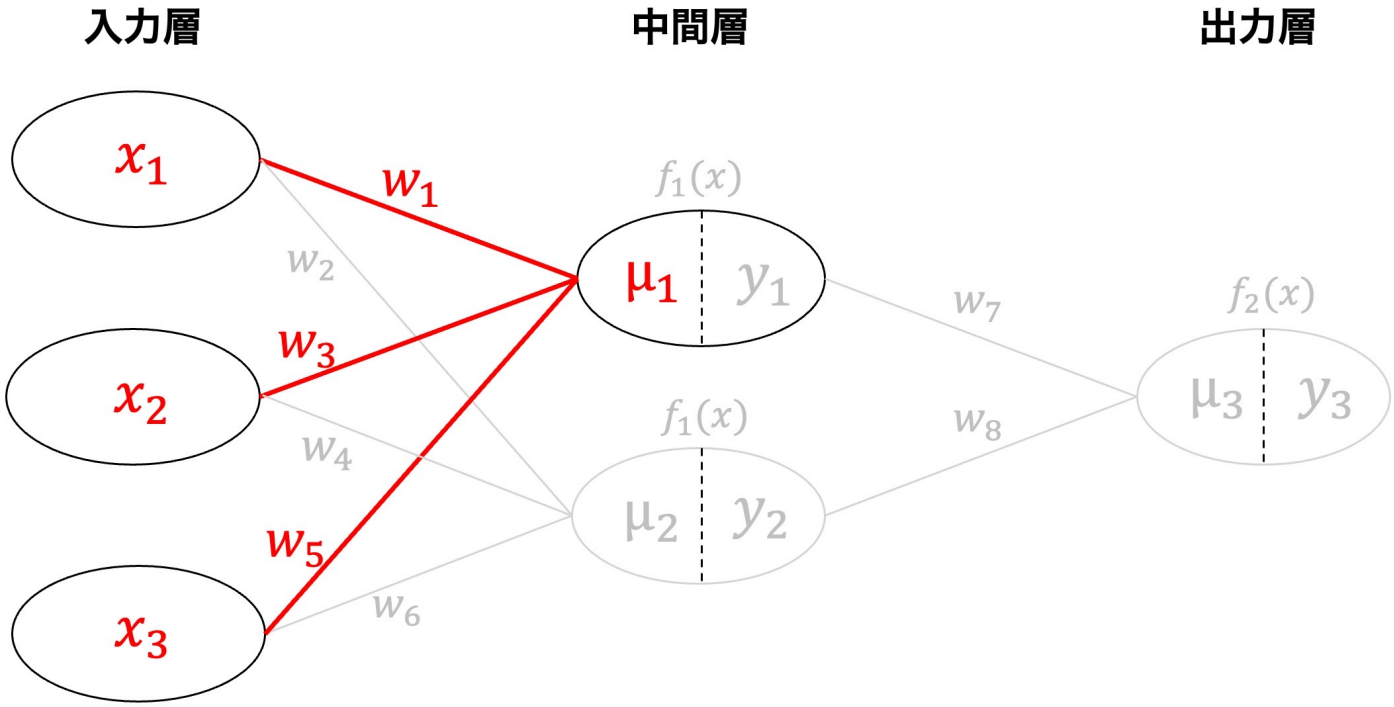


ニューラルネットワークとは



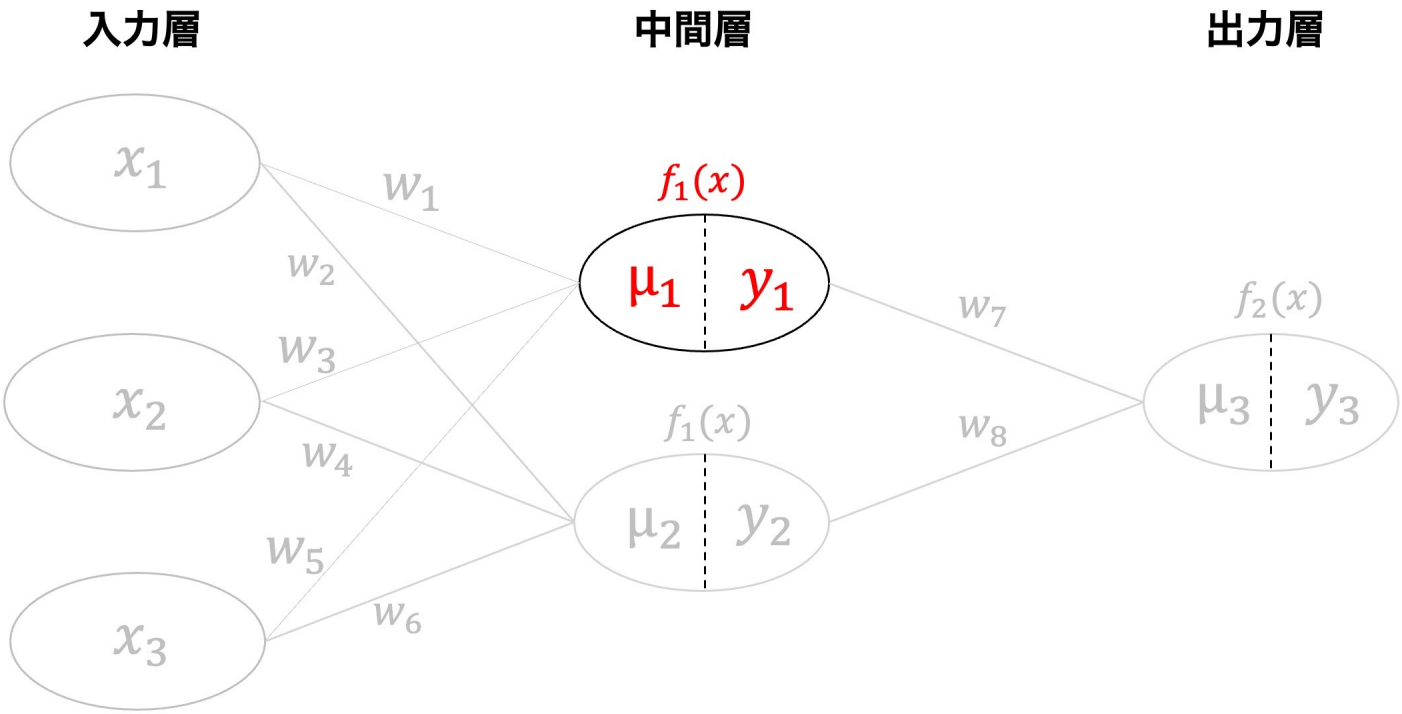
x : 入力、w : 重み、μ : 入力合計、
$f(x)$: 活性化関数、y : (各ニューロンの)出力

ニューラルネットワークとは



$$\mu_1 = w_1 \times x_1 + w_3 \times x_2 + w_5 \times x_3$$

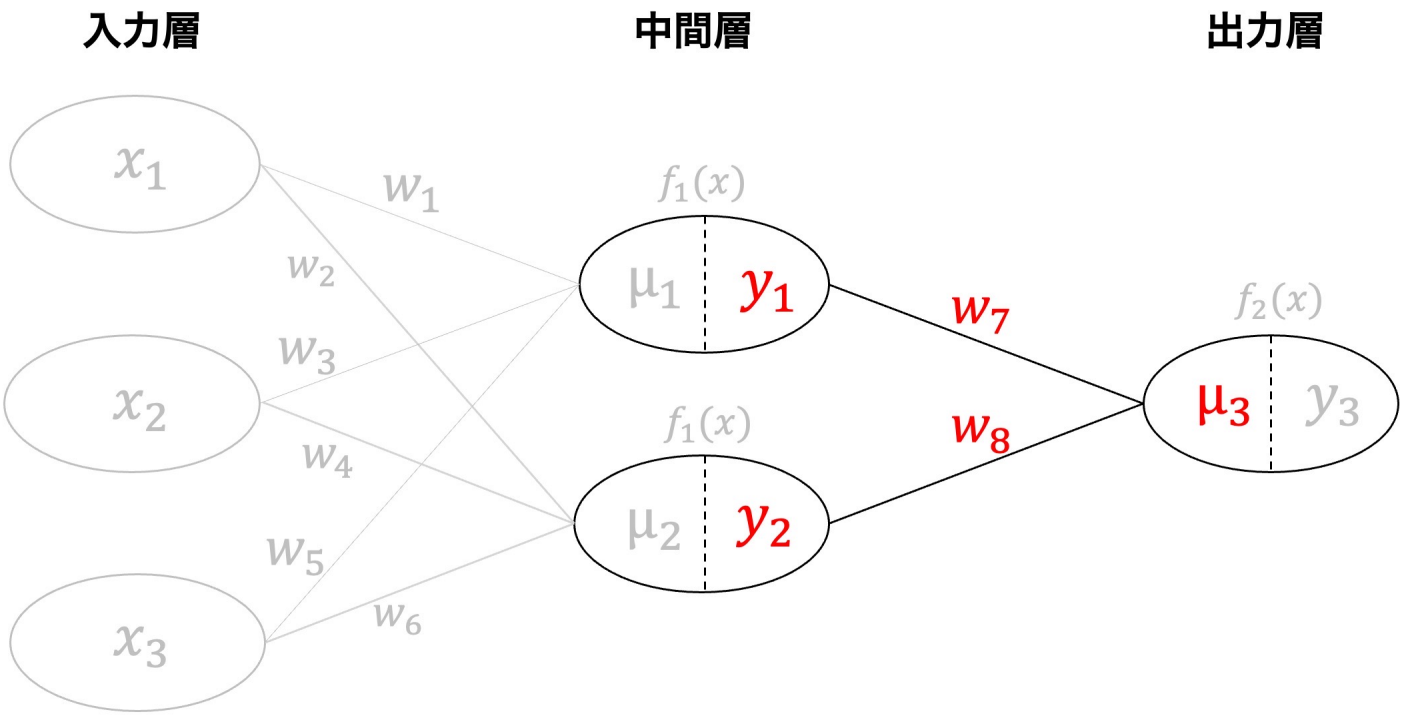
ニューラルネットワークとは



$$y_1 = f_1(\mu_1)$$

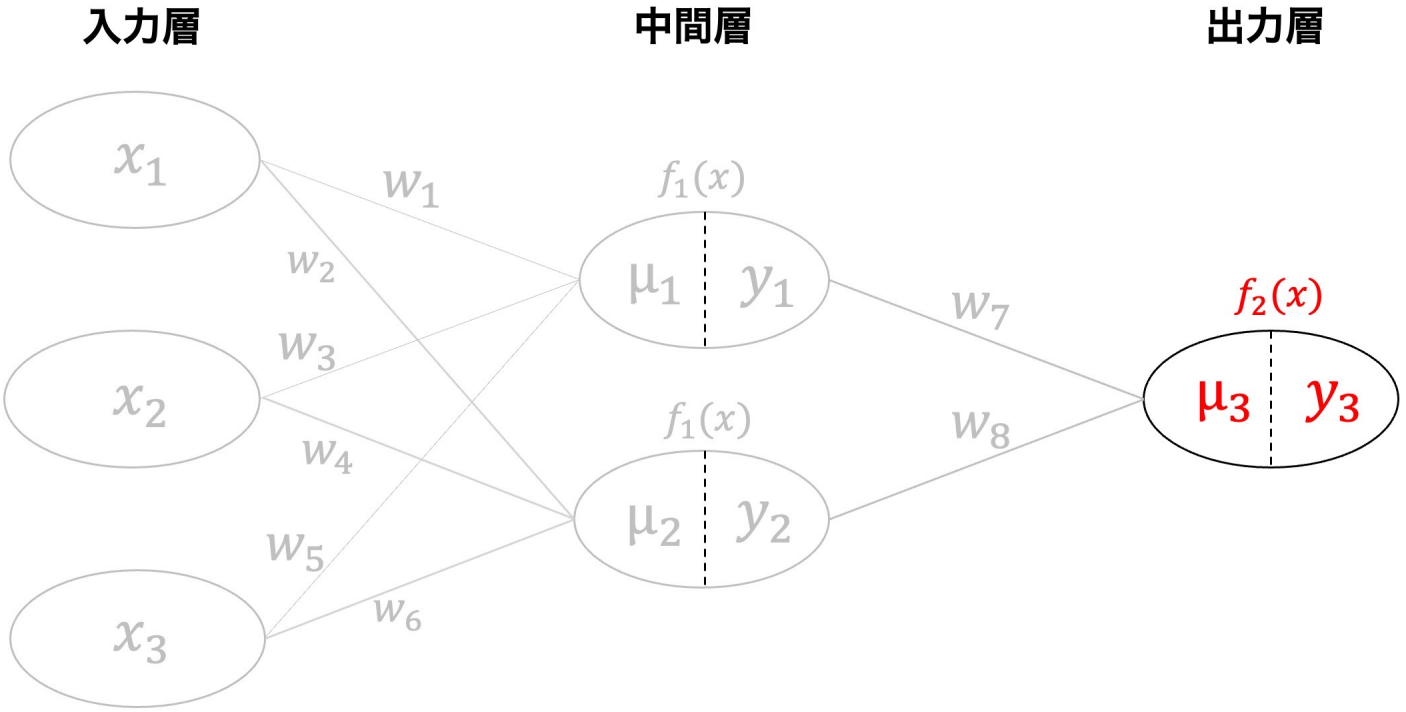
$y_2 = f_1(\mu_2)$ も同様

ニューラルネットワークとは



$$\mu_3 = w_7 \times y_1 + w_8 \times y_2$$

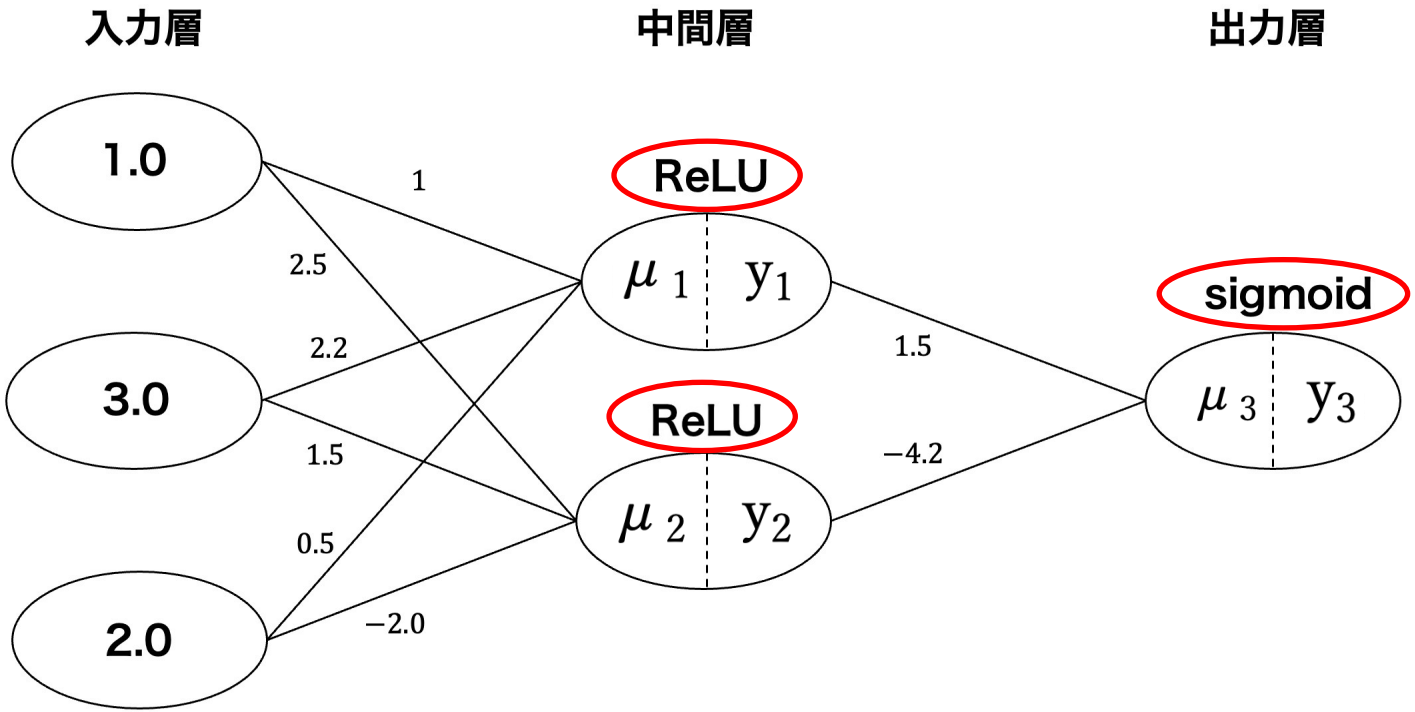
ニューラルネットワークとは



$$y_3 = f_2(\mu_3)$$

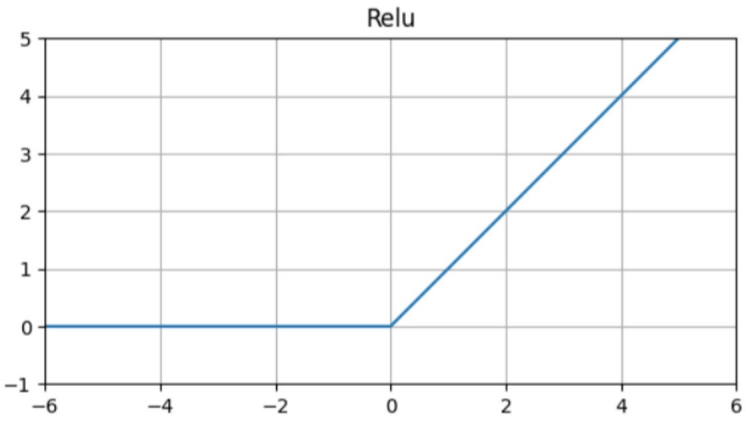
ニューラルネットワークとは

実際に計算してみる



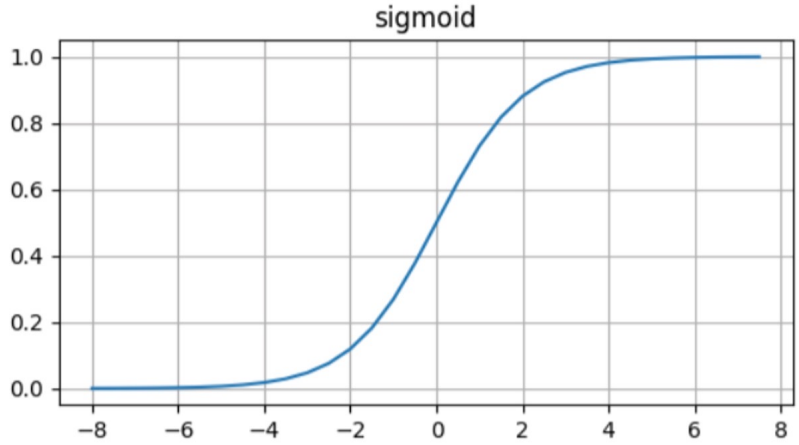
ニューラルネットワークとは

(a) ReLU関数



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

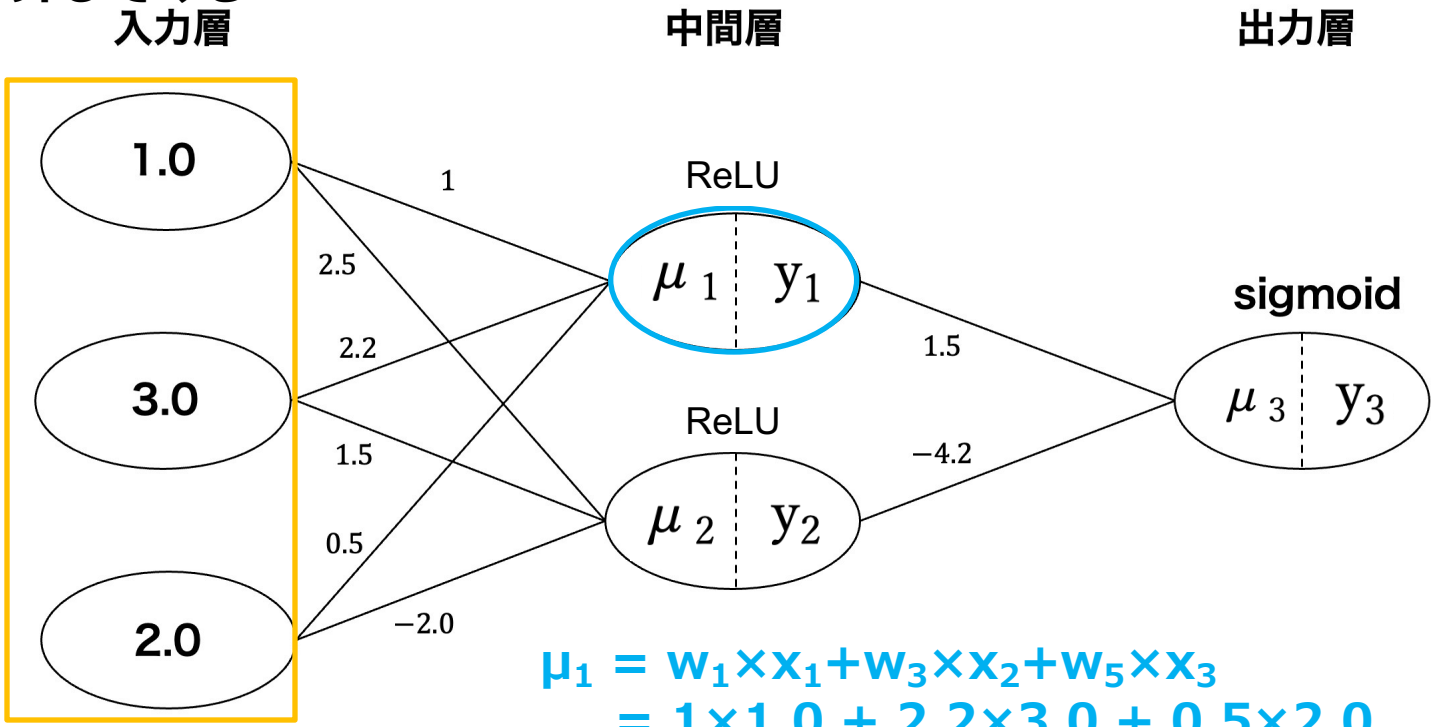
(b) sigmoid関数



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

ニューラルネットワークとは

実際に計算してみる



$\mu_1 = w_1 \times x_1 + w_3 \times x_2 + w_5 \times x_3$

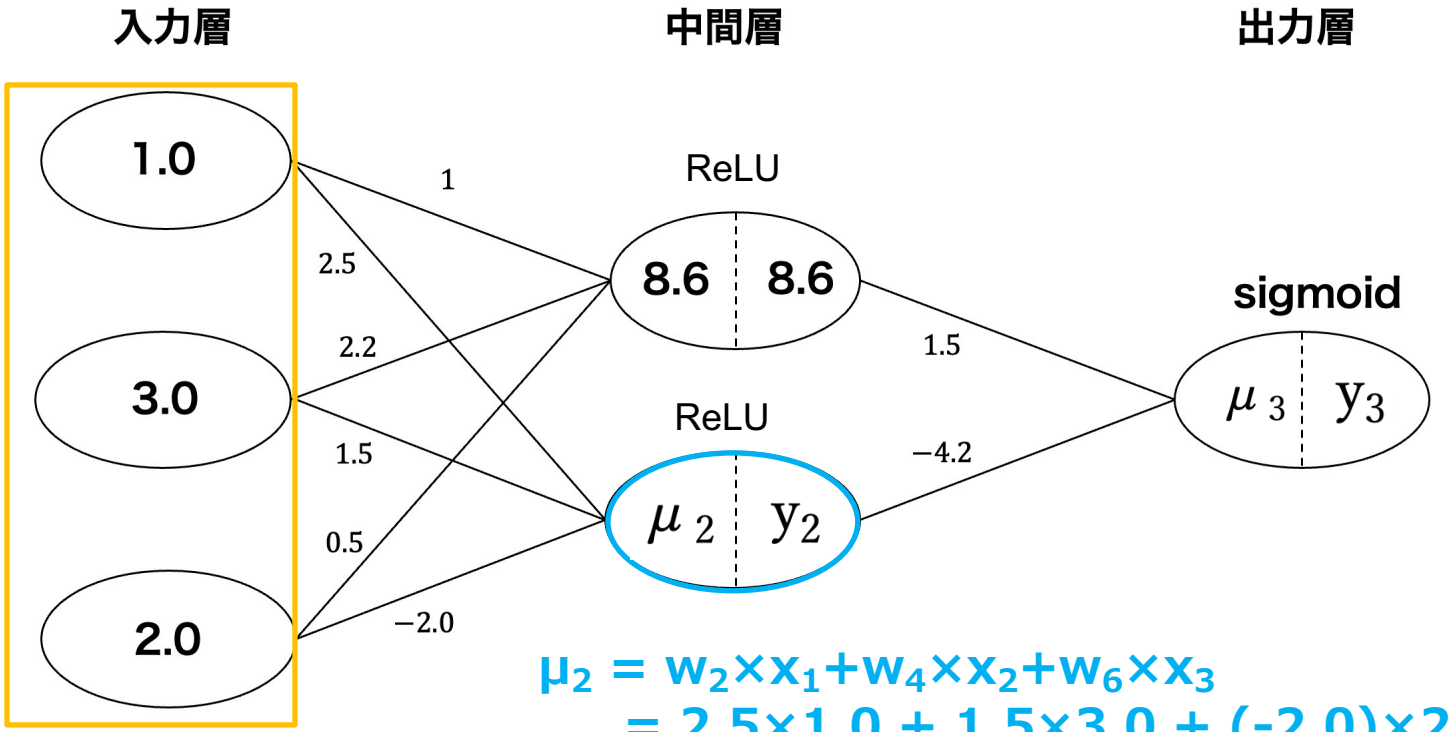
$= 1 \times 1.0 + 2.2 \times 3.0 + 0.5 \times 2.0$

$= \underline{8.6}$

$y_1 = \mathrm{ReLU}(\mu_1) = 8.6$

ニューラルネットワークとは

実際に計算してみる



$\mu_2 = w_2 \times x_1 + w_4 \times x_2 + w_6 \times x_3$

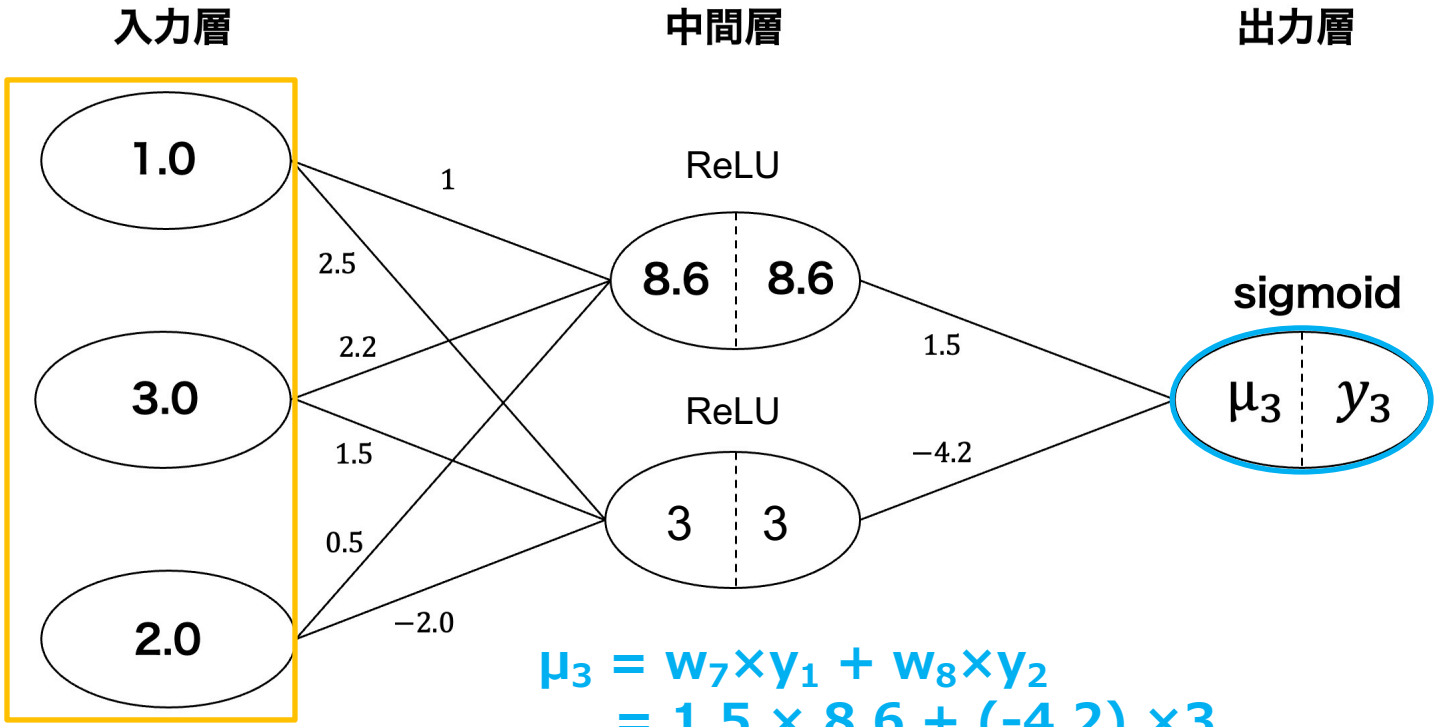
$= 2.5 \times 1.0 + 1.5 \times 3.0 + (-2.0) \times 2.0$

$= \underline{3}$

$y_2 = \mathrm{ReLU}(\mu_2) = 3$

ニューラルネットワークとは

実際に計算してみる



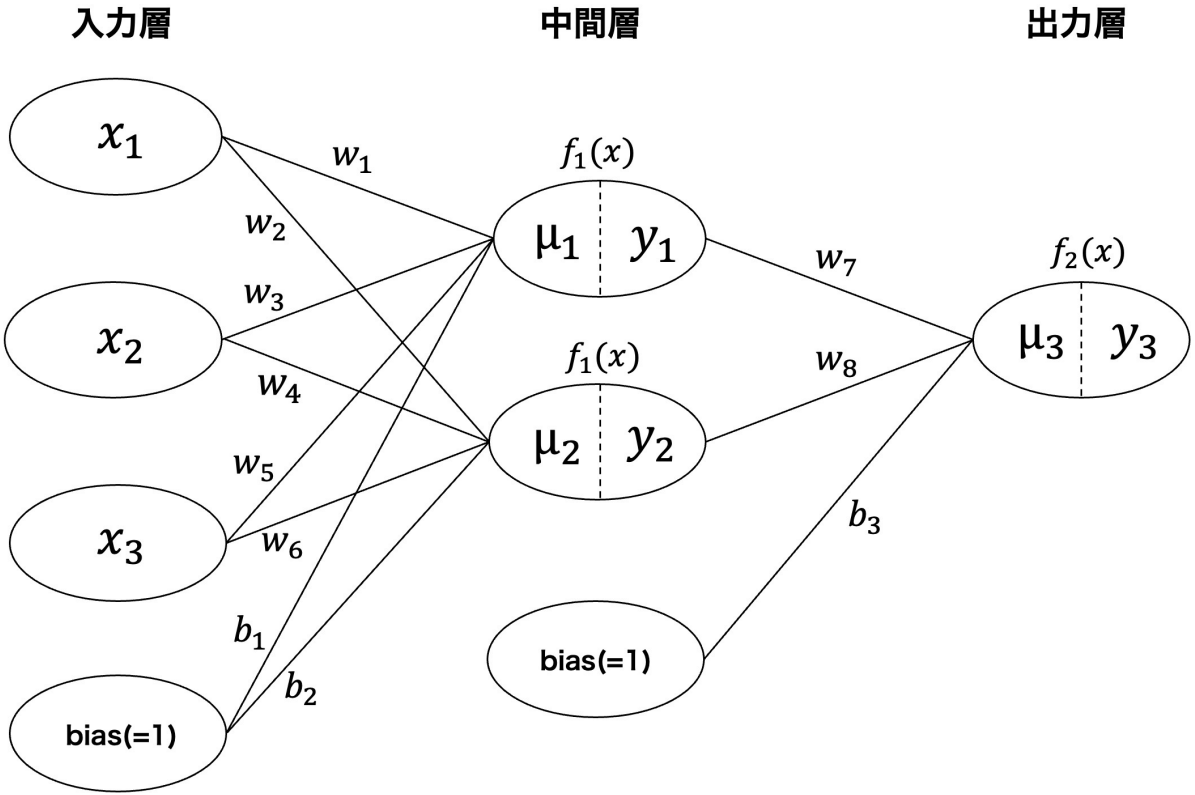
$\mu_3 = w_7 \times y_1 + w_8 \times y_2$

$= 1.5 \times 8.6 + (-4.2) \times 3$

$= \underline{0.3}$

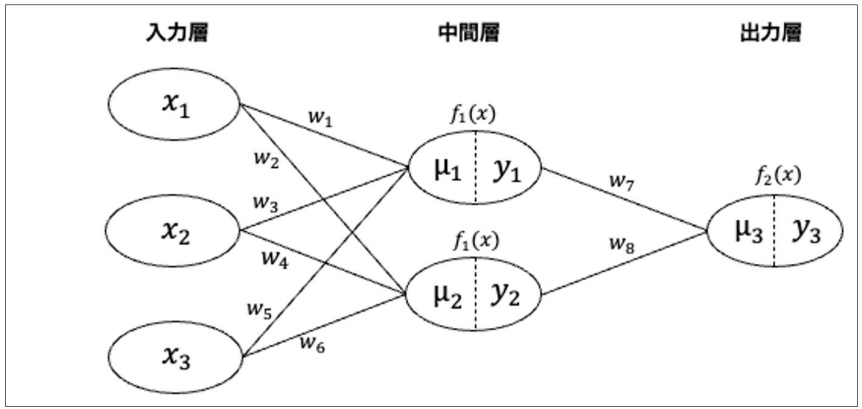
$y_3 = \mathrm{Sigmoid}(\mu_3) = \underline{0.5744\cdots}$

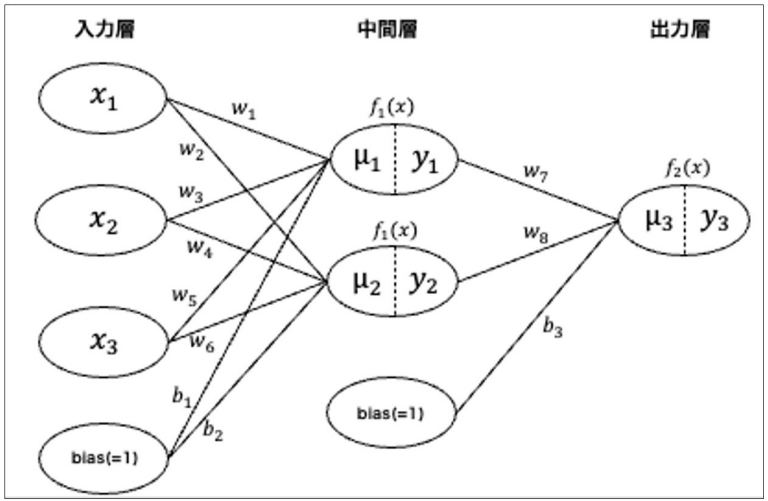
ニューラルネットワークとは



実は、各ニューロンには、重み×入力の他に、バイアス（前のニューロンと繋がっていない定数)も足される

ニューラルネットワークとは





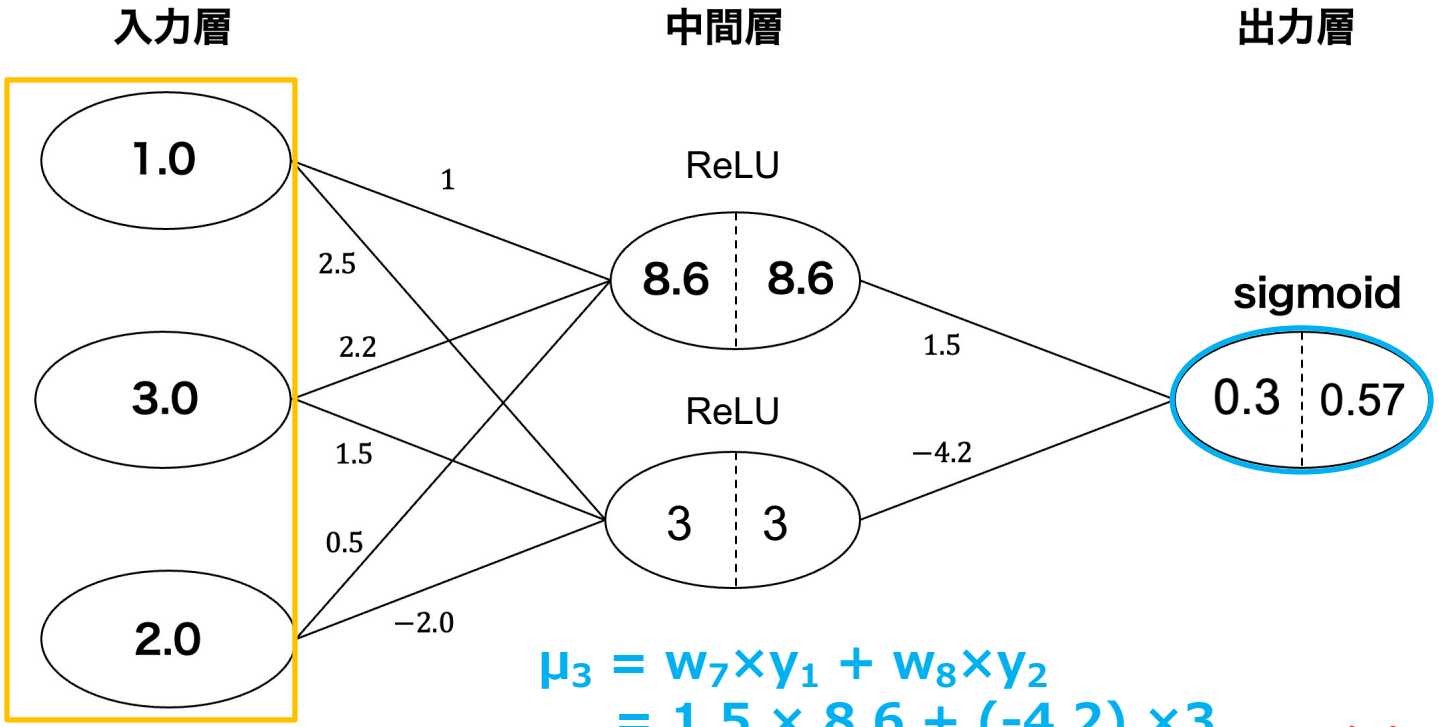
$$\mu_1 = w_1 \times x_1 + w_3 \times x_2 + w_5 \times x_3$$
$$\mu_2 = w_2 \times x_2 + w_4 \times x_2 + w_6 \times x_3$$
$$\mu_3 = w_7 \times y_1 + w_8 \times y_2$$

$$\mu_1 = w_1 \times x_1 + w_3 \times x_2 + w_5 \times x_3 + 1 \times b_1$$
$$\mu_2 = w_2 \times x_2 + w_4 \times x_2 + w_6 \times x_3 + 1 \times b_2$$
$$\mu_3 = w_7 \times y_1 + w_8 \times y_2 + 1 \times b_3$$

バイアス（前のニューロンと繋がっていない定数)もμに足す

ニューラルネットワークとは

実際に計算してみる



$\mu_3 = w_7 \times y_1 + w_8 \times y_2$

$= 1.5 \times 8.6 + (-4.2) \times 3$

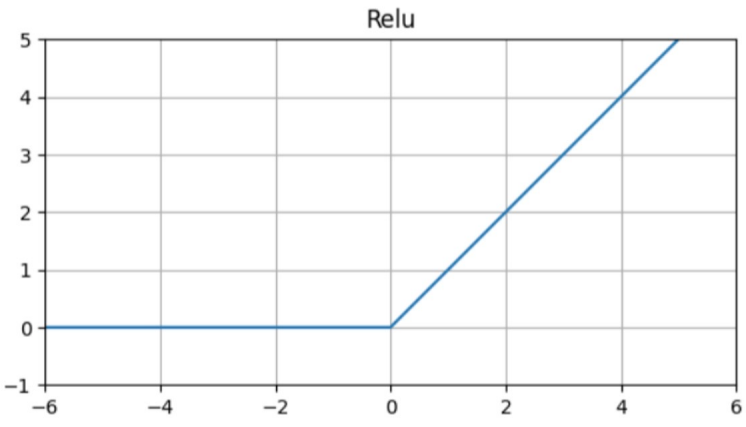
$= \underline{0.3}$

$y_3 = \mathrm{Sigmoid}(\mu_3) = \underline{0.5744\cdots}$

確率で考える

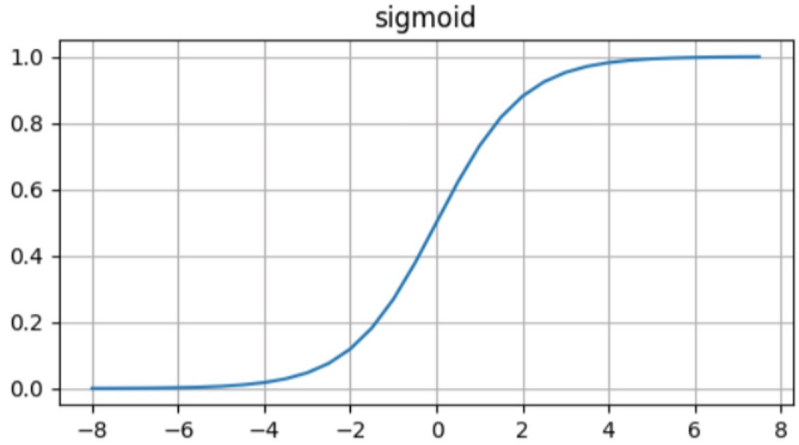
ニューラルネットワークとは

(a) ReLU関数



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

(b) sigmoid関数



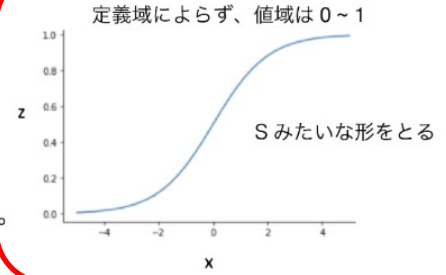
$$y = \frac{1}{1 + e^{-x}}$$

ニューラルネットワークとは

シグモイド関数とロジスティック回帰

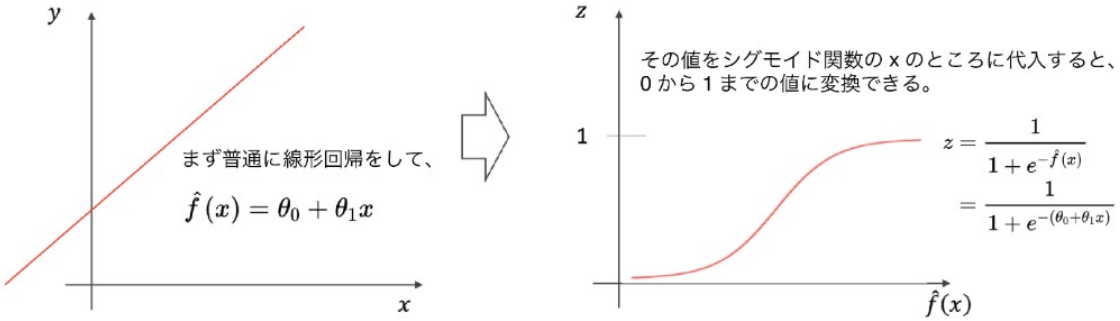
突然だが、次の関数を考えたい

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



グラフを書いて見ると S のようで、S はギリシャ文字の σ (シグマ) に由来することから、この関数は「シグモイド関数」と呼ばれる。
この関数は x の値によらず 0 から 1 までの値をとる単調増加関数である。

これを活用して線形回帰の結果を 0~1 の間に収めることを考える。



ニューラルネットワークとは

ロジスティック回帰分析

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させる
STEP4：傾きと切片を求める
STEP5：予測を行う
STEP6：モデルの評価

STEP4：傾きと切片を求める

コード11-14 学習済みモデルの傾きと切片を求める

```
print(model_bc.coef_)
print(model_bc.intercept_)
```

```
[[ 4.43803517 -0.15747975 -0.49120368 -0.02299483 -0.1977349  -0.88107831
  -1.17436636 -0.51610975 -0.32416124 -0.06441917]]
[0.71206521]
```

係数(傾き）は10個の特徴量について、それぞれ求められている

$$P(x) = \frac{1}{1+e^{-((0.71+(4.44\times radius)+(-0.16\times texture)+(-0.49\times perimeter)+(-0.02\times area)+(-0.20\times smoothness)+(-0.88\times compactness)+(-1.17\times concavity)+(-0.52\times concave\ points)+(-0.32\times symmetry)+(-0.06\times fractal\ dimension))}}$$

学習済みのロジスティック回帰式が完成

colab

ロジスティック回帰分析

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させる
STEP4：傾きと切片を求める
STEP5：予測を行う
STEP6：モデルの評価

STEP5：予測を行う

コード11-16 Y=1/0になる確率を求める

```
print(model_bc.predict_proba(x_test))
```

*(モデル名).predict_proba()で予測確率を求められる

	Y=0になる確率	Y=1になる確率
1人目	4.57042606e-01	5.42957394e-01
	1.79395138e-01	8.20604862e-01
3人目	8.66171843e-02	9.13382816e-01
	1.65251193e-01	8.34748807e-01
	5.37373549e-02	9.46262645e-01
	4.12045708e-02	9.58795429e-01
	2.02153766e-01	7.97846234e-01
	3.44362161e-02	9.65563784e-01
	3.93259776e-02	9.60674022e-01
	2.17382152e-02	9.78261785e-01
	5.21246619e-01	4.78753381e-01

Y=1になる確率は 0.54=54%
→なので、分類は「1」になる

Y=1になる確率は 0.91=91%
→なので、分類は「1」になる

一人目は不正解(本当は0なのに1と予測していた）ので、確率が低いのは整合性がある

colab

出力層の最後の活性化関数をsigmoid関数にすると、
(ロジスティック回帰分析と同様に)
y(正解ラベル)が1になる確率が得られる

ニューラルネットワークとは

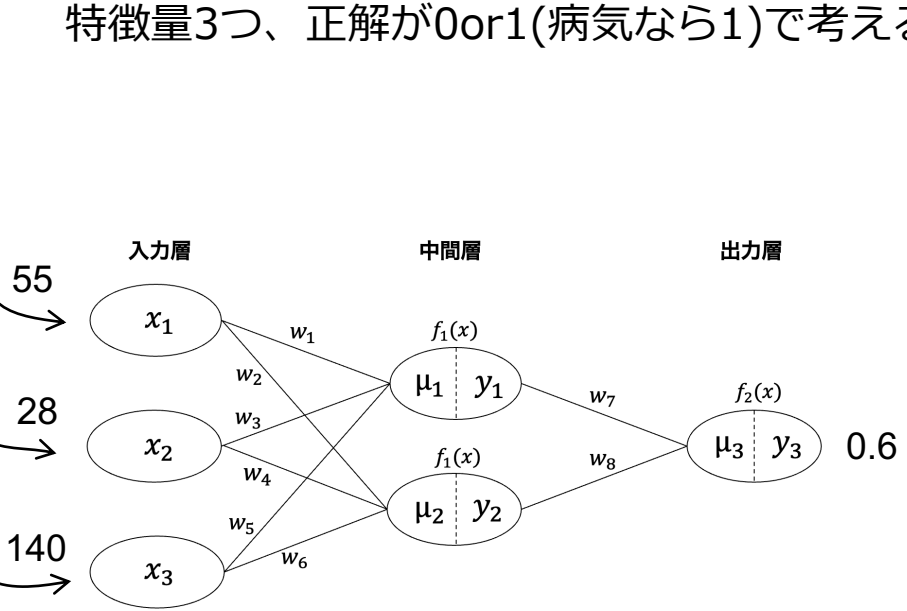
特徴量3つ、正解が0or1(病気なら1)で考える

	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
6	74	38	109	0
7	59	46	137	1
8	52	41	140	1
9	38	56	110	0
10	47	53	121	1

ニューラルネットワークとは

特徴量3つ、正解が0or1(病気なら1)で考える

	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
6	74	38	109	0
7	59	46	137	1
8	52	41	140	1
9	38	56	110	0
10	47	53	121	1



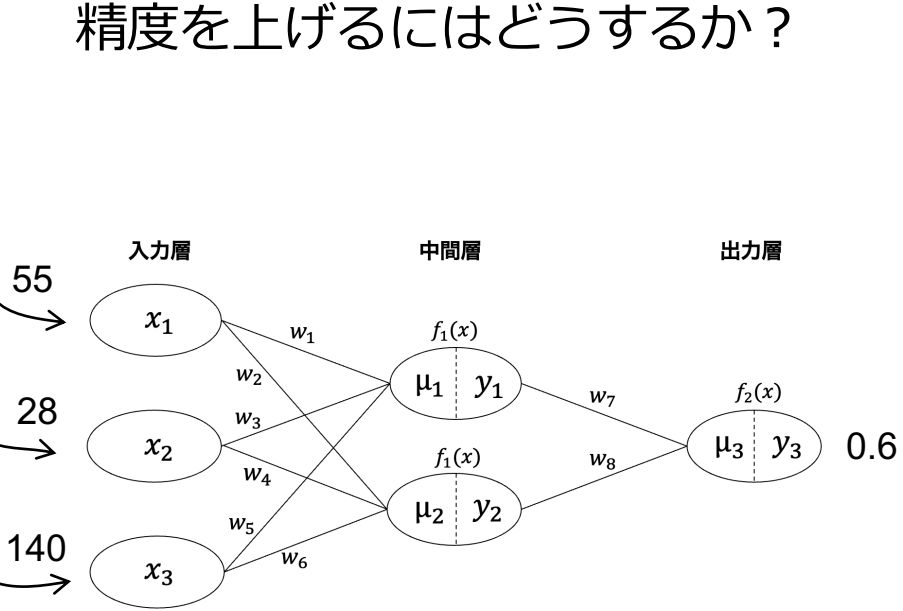
	体重	年齢	血圧	正解	予測
1	55	28	140	1	0.6
2	48	44	130	0	
3	77	64	145	0	
4	49	42	130	1	
5	59	66	128	1	
6	74	38	109	0	
7	59	46	137	1	
8	52	41	140	1	
9	38	56	110	0	
10	47	53	121	1	

ニューラルネットワークにはデータが１つずつ入力される
入力層のニューロンの数は1つのデータが持つ説明変数の数

ニューラルネットワークとは

精度を上げるにはどうするか？

	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
6	74	38	109	0
7	59	46	137	1
8	52	41	140	1
9	38	56	110	0
10	47	53	121	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
6	74	38	109	0	0.2
7	59	46	137	1	0.4
8	52	41	140	1	0.9
9	38	56	110	0	0.6
10	47	53	121	1	0.5

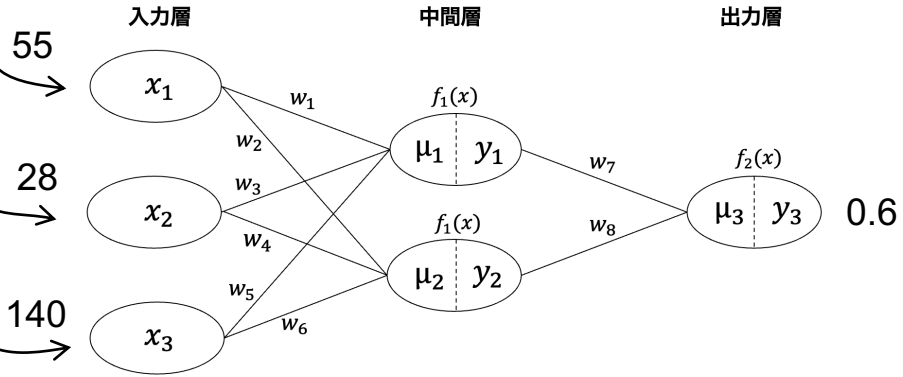
ニューラルネットワークにはデータが１つずつ入力される
入力層のニューロンの数は1つのデータが持つ説明変数の数

ニューラルネットワークとは

精度を上げるにはどうするか？

正解と予測の誤差が小さくなるように
重みとバイアスを更新する

	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
6	74	38	109	0
7	59	46	137	1
8	52	41	140	1
9	38	56	110	0
10	47	53	121	1



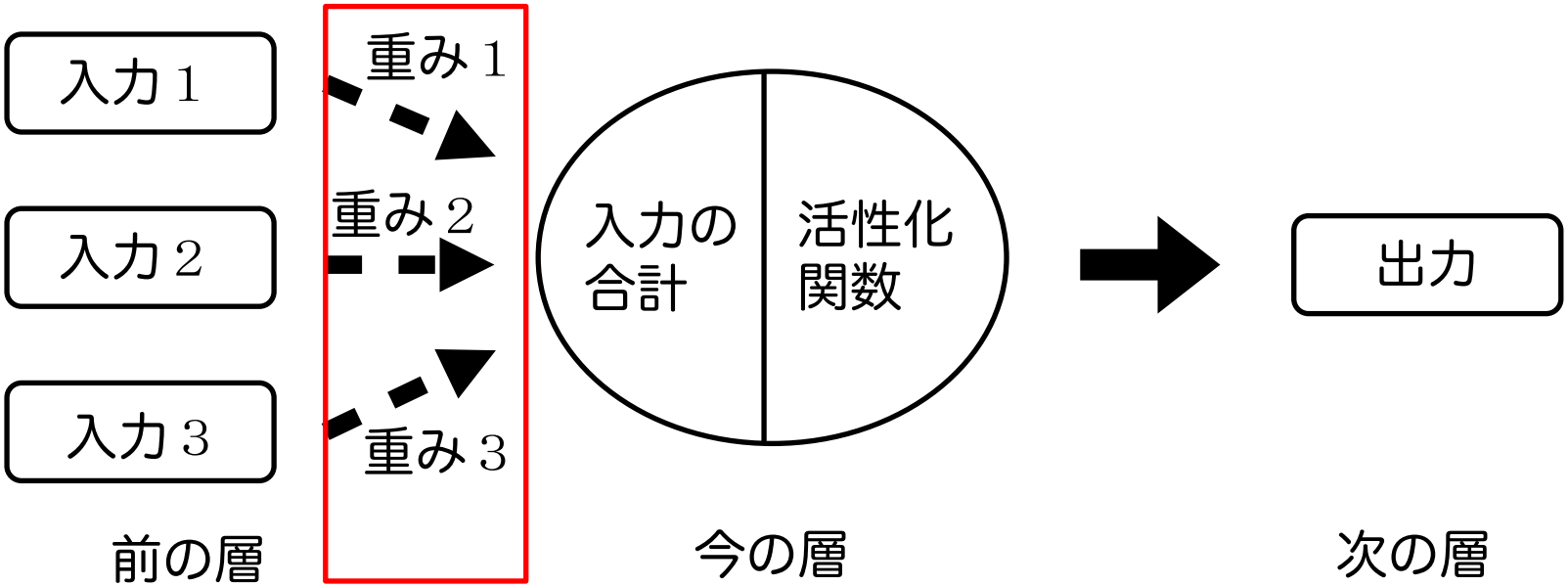
	体重	年齢	血圧	正解	予測
1	55	28	140	1	0.6
2	48	44	130	0	0.3
3	77	64	145	0	0.4
4	49	42	130	1	0.9
5	59	66	128	1	0.6
6	74	38	109	0	0.2
7	59	46	137	1	0.4
8	52	41	140	1	0.9
9	38	56	110	0	0.6
10	47	53	121	1	0.5

ニューラルネットワークにはデータが１つずつ入力される
入力層のニューロンの数は1つのデータが持つ説明変数の数

ニューラルネットワークとは

深層学習は、

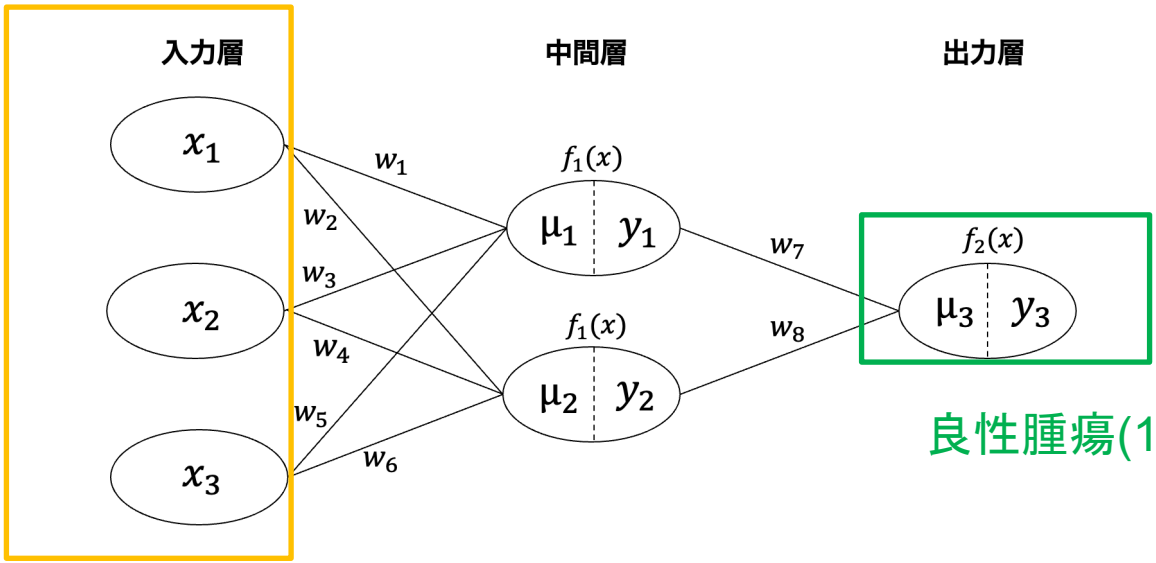
①入力したデータの予測結果を算出する
②正解と予測結果がどれくらい異なっているかという誤差を計算する
③誤差が小さくなるように重みとバイアスを変える
を何度も繰り返すことで、誤差を減らしていき精度を高める



深層学習(乳がんデータ)

深層学習の概念を掴むため、前回まで使ってきた
乳がんデータを使って深層学習を行う

乳がんデータ
の特徴量



良性腫瘍(1)/ 悪性腫瘍(0)

深層学習(乳がんデータの分類)コードまとめ

STEP1　データの用意

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(特徴量,予測値)

STEP4　学習結果の図示

学習結果の図示を行う

STEP5　モデルの評価：　(モデル名).evaluate()

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

STEP0：Google Colaboratoryの立ち上げ

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15-16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15-16コード.ipynb

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

STEP0：ライブラリのインポート

コード15-1 ライブラリとモジュールをインポート

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

ライブラリをインポート：import （ライブラリ名） as 省略形

numpyとpandas

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(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

STEP1：データの用意

コード15-2 乳がんデータの読み込み

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

今回は分類を行うため、**breast_cancer(乳がん)**のデータを読み込む

ライブラリ　モジュール　関数

関数をインポート　`from sklearn.datasets import load_breast_cancer`

● `load_breast_cancer(as_frame = True/False)` の引数で、`True`(真)か`False`(偽)を選択する

読み込むデータの形式を指定
`True` → pdのデータフレーム型
`False`→ numpy配列

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

STEP1：データの用意

コード15-2 乳がんデータ

`bc.target`

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

`bc.data`

```
array([[1.799e+01, 1.038e+01, 1.228e+02, …, 2.654e-01,
4.601e-01, 1.189e-01], [2.057e+01, 1.777e+01, 1.329e+02,
…, 1.860e-01, 2.750e-01, 8.902e-02], [1.969e+01,
```

`as_frame = False`で指定したので、numpy配列になっている

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

STEP1：データの用意

コード15-2 乳がんデータ

`bc.data`

特徴量データ

(569, 30)

569個のデータ

それぞれ30個の特徴量

`bc.target`

正解データ

(569,)

569個のデータ

それぞれ0か1の正解

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

STEP1：データの用意

コード15-2 乳がんデータ

ロジスティック回帰分析

STEP1：正解値データの用意

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させる
STEP4：傾きと切片を求める
STEP5：予測を行う
STEP6：モデルの評価

コード11-3 正解値データの作成

```
y_bc = bc.target
y_bc.head()
```

- bcのtargetキーで格納されている値を呼び出し、y_bcに格納する
- (データフレーム名).head()で最初の5行を出力する(見るだけ)

```
y_bc.shape  #データの配列構造を出力
```

➡ (569,)　569個の1次元配列(ベクトル)

```
type(y_bc)  #データの型を出力
```

➡ pandas.core.series.Series

```
0    0
1    0
2    0
3    0
4    0
Name: target, dtype: int64
```


ロジスティック回帰分析

STEP1：特徴量データの用意

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させる
STEP4：傾きと切片を求める
STEP5：予測を行う
STEP6：モデルの評価

コード11-5 特徴量データを作成

```
x_bc = bc.data
x_bc.head()  #最初の5行を出力
```

bcのdataキーで格納された値を呼び出し、x_bcに格納する

➡

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	25.38
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	24.99
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	23.57
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	14.91
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	22.54

5 rows × 30 columns

5行×30個の特徴量を出力

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

STEP1：データの用意

コード15-3 学習データと検証データに分割

*今まで同じように分割をする

```
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)
```

- trainデータ70%とtestデータ30%にデータを分割
- 分割時の乱数シード値を"0"に指定
- x_bcとy_bcを作成せず、bc.data, bc.targetをtrain_test_splitに直接入れてデータ分割を行う

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

STEP1：データの用意

コード15-3 学習データと検証データに分割

特徴量データ　　正解データ

`bc.data`
(569, 30)

`bc.target`
(569,)

特徴量データ　　正解データ

`x_train`
(398, 30)

`y_train`
(398,)

`x_test`
(171, 30)

`y_test`
(171,)

深層学習(乳がんデータの分類)

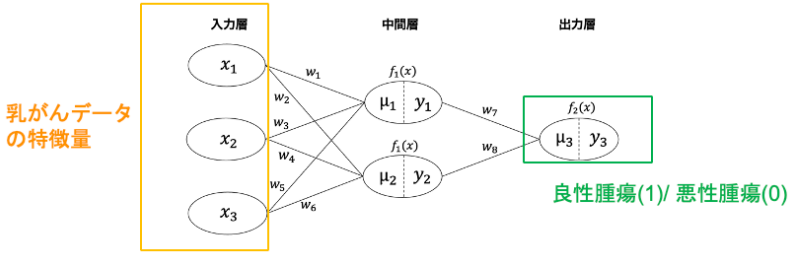
STEP0：事前準備
STEP1：データの用意
STEP2：学習モデルの選択
STEP3：データを入れて学習
STEP4：学習結果の図示
STEP5：モデルの評価

STEP1：データの用意

コード15-4 3つの特徴量だけを抽出する

```
x_train3 = x_train[:, 0:3]

x_test3 = x_test[:, 0:3]
```



- 簡単なニューラルネットワークを行うため、特徴量を1~3番目 (インデックス番号0~2) 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
- 今回は全ての行を意味する (: コロン)、列は0~3未満のインデックス番号を指定する 0:3 で抽出を行う

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

STEP1：データの用意

コード15-5 データの配列構造を確認する

```
print(x_train.shape)

print(x_test.shape)

print(x_train3.shape)

print(x_test3.shape)
```

```
(398, 30)
(171, 30)
(398, 3)
(171, 3)
```

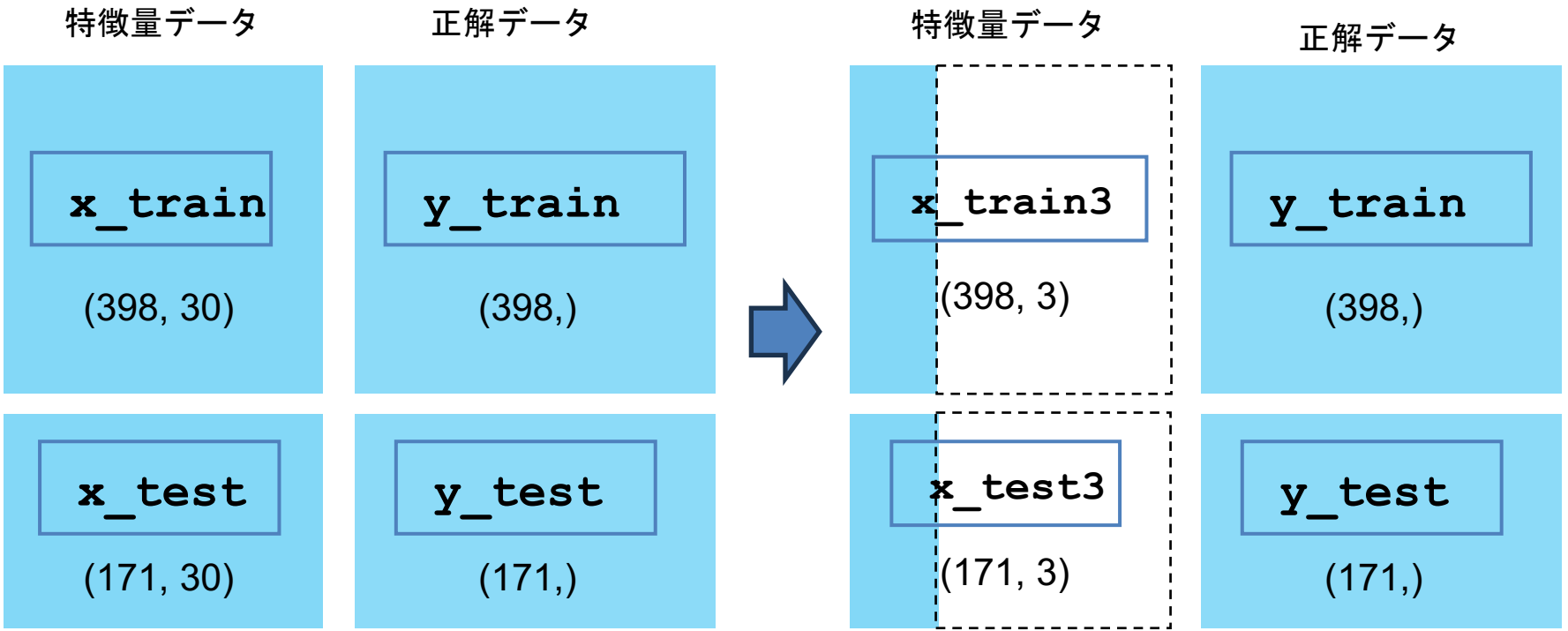
colab

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

STEP1：データの用意

コード15-5 データの配列構造を確認する



演習15 課題

Webclassに課題があります。
締め切りは2024/02/14 23:59まで