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

2024/1/17 11:35-12:20

演習10
回帰3

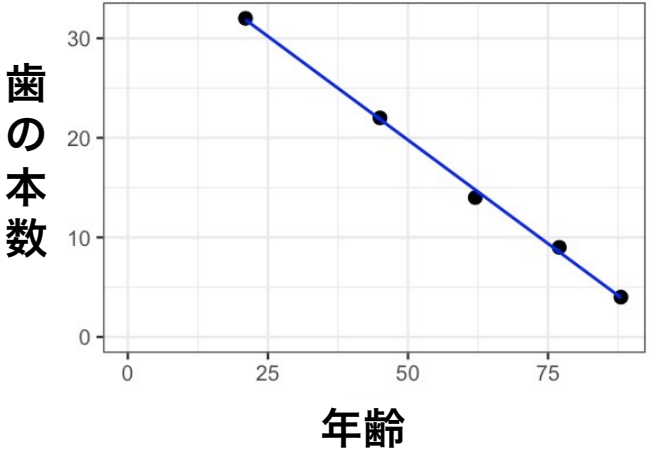
統合教育機構
石丸美穂

学習データと検証データ

予測性能の向上

今までは得られたデータを全て使って機械学習の「学習」を行い、
そのデータで予測性能を推定していた

→実はよくない方法！
実力よりも良すぎる正解率が出る可能性がある(過学習 overfitting)



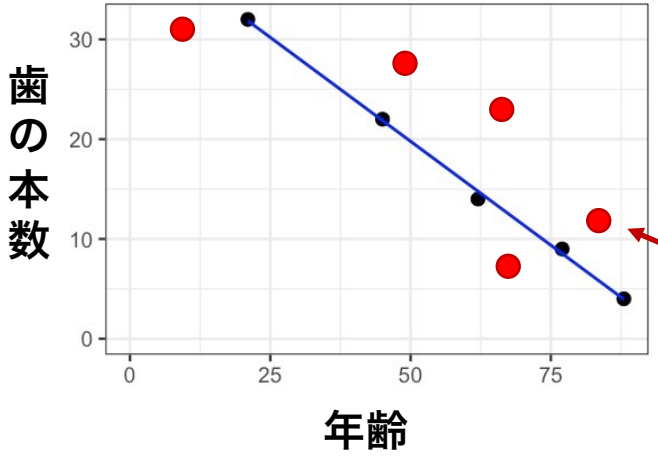
得られた5個のデータにはfitしている

決定係数R^2
0.998

予測性能がものすごく
高い良いモデルに見える

本当なのか？

過学習 overfitting

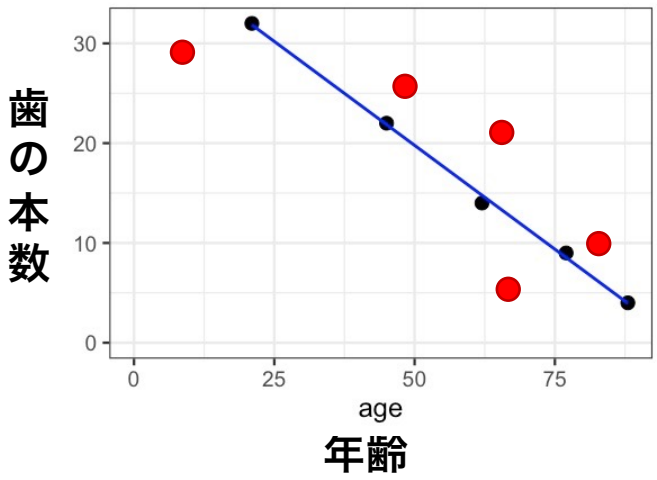


一般的に考えて、すべての人が
この直線上にいるとは思えない

別の5人のデータを追加

追加分の人に対しての予測性能(R^2)は0.998よりは低い
学習したデータ（最初の5人）に過剰に適合したモデルになっている

過学習 overfitting



過学習が起こりやすい条件

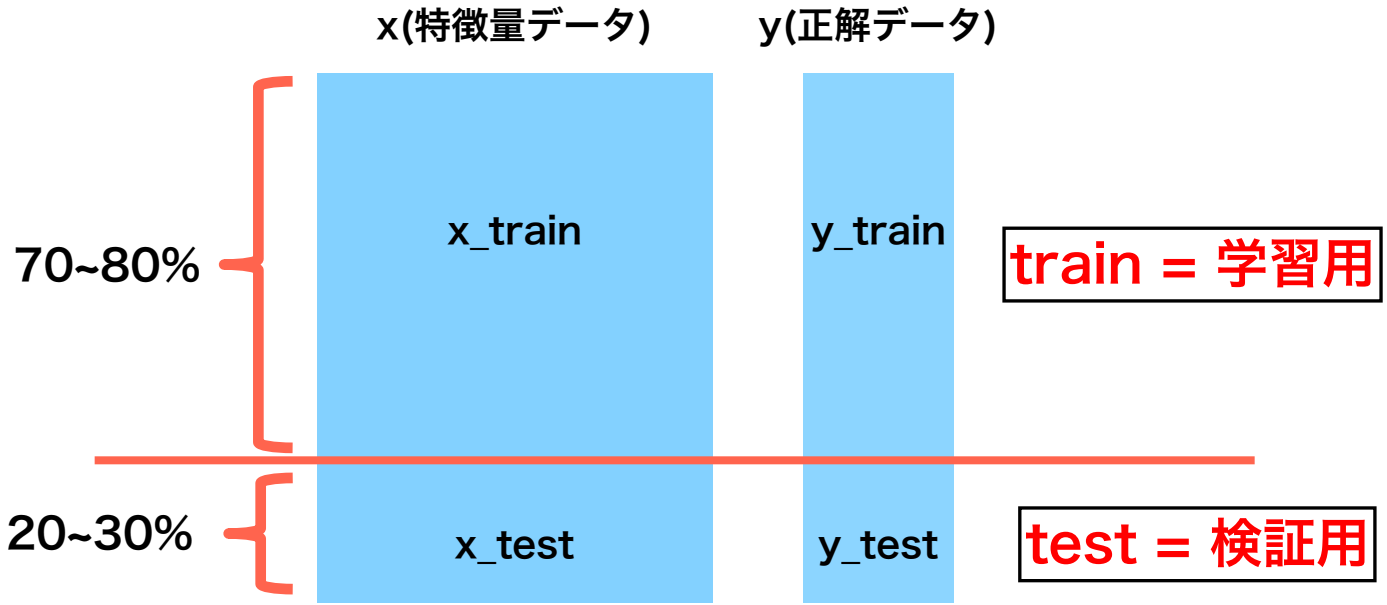
- データ数が少ない
- データに偏りがある
- モデルが複雑

まだ学習していない未知のデータで良い予測が出るか検証用データが必要

機械学習ではそのままデータを丸ごと学習させない！

ホールドアウト法

新たにデータを用意するのではなく既にあるデータを学習用と検証用に分割する方法　(検証用が20~30%で分割するのが一般的)



要注意！ 全部のデータを使って機械学習の訓練をしない

手持ちのデータはあくまで膨大な母集団の中のごく一部の標本にすぎないが、その標本データにフィットしすぎていて、まだ見たことのない新しいデータには適応できない状態を過学習 (overfitting) という。

機械学習のモデルを構築する真の目的は，未知のデータや将来のデータに対して高い精度で予測できることだ。例えば今までに来た患者さんのデータを使っていくらすごいモデルをつくっても、次に来る患者さんに対して高い精度で予測できなかったら意味がない。

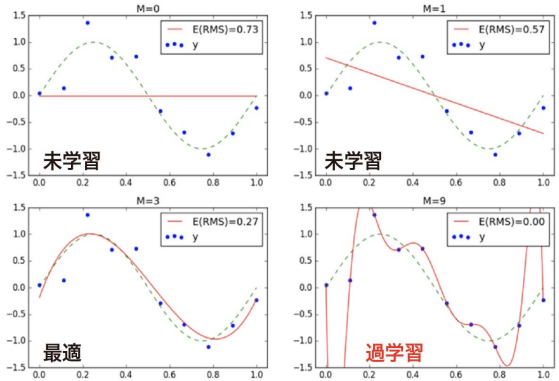
このように，学習データだけでなく未知のデータに対しても正しく予測できる性能のことを汎化性能という。

汎化性能の計り方は色々あるが、最も確実なのは学習に使うデータセットとは別に全く新しいテスト用のデータセット (external dataset) を用意することだ。

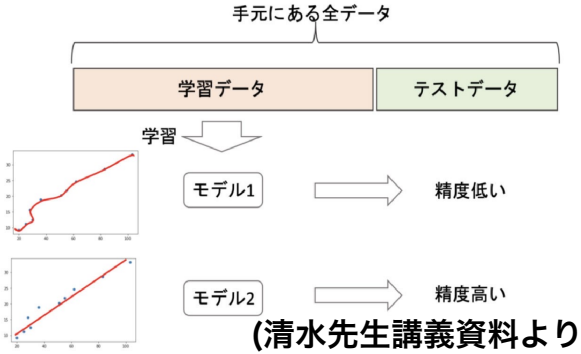
だが実際にはそのような外部データセットを用意することは難しい場合が多い。その時に使われるのが hold-out 法で、訓練データとテストデータに分けて、訓練データのみを使ってモデルを訓練し、最後に訓練では全く使っていないテストデータで汎化性能を調べるという流れになる。

学習データとテストデータは 7:3 でも 9:1 でも好きなように分ければよい。学習データを多くすることが大半で、一方でテストデータもある程度の数が残るように比率を決める。

データの分割方法は厳密にはいろいろあるが、この入門編では完全にランダムに分割するものと考えて良い。



青い 10 個のデータを M 次関数でフィッティング (赤)。
3 次関数が最適で、9 次関数は全ての点を通るが
新しいデータには対応できなさそう (= 過学習)。



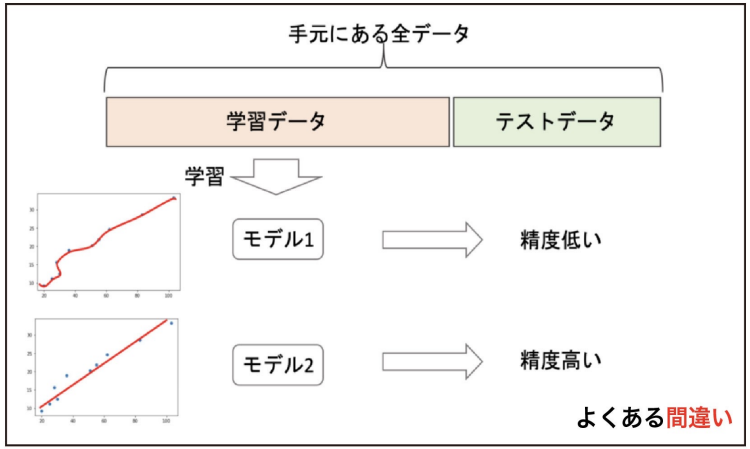
(清水先生講義資料より引用)

要注意！ 複数のモデルを比べるなら、３分割しないといけない

学習データで学習をしてテストデータで評価した際に，思ったより精度が低かったのでさらにモデルのパラメータを変更して学習してテストデータで評価して... というのを繰り返すと，本来テストデータは未知のデータという前提なのに，テストデータに対して過学習したモデルができてしまう。

それを避けるために，検証データ (validation data) というデータを作っておき，学習データで学習したあとにテストデータを使うのではなく検証データを使ってモデルを比較し１つに決める。

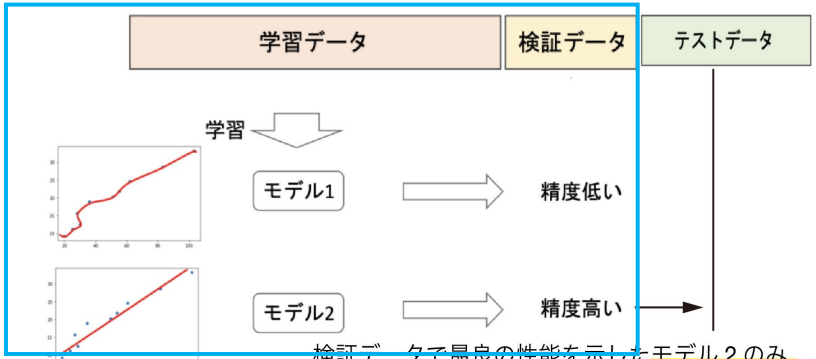
そしてチャンピオンモデルに対してテストデータを１回のみ使う。



テストデータを見ることができるのは

たった１回

であることを肝に命じること



検証データで最良の性能を示したモデル２のみ、テストデータで汎化評価する

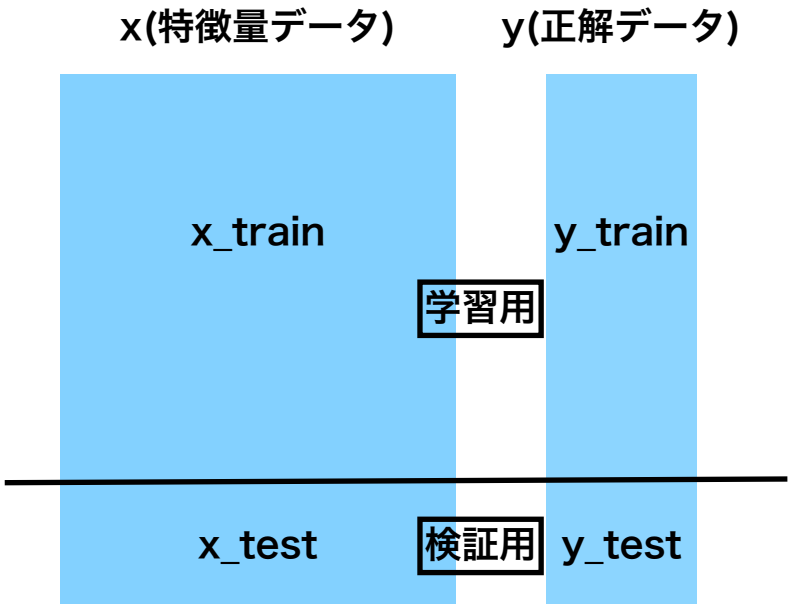
同じ「線形回帰」であっても、
パラメーターが異なればそれらは異なるモデル
であることに注意。

今回はテストデータで汎化評価はしない

(清水先生講義資料より引用)

機械学習ではそのままデータを丸ごと学習させない！

`train_test_split()` ： 学習用データと検証用データに分ける関数



x_train：特徴量の学習用

x_test：特徴量の検証用

y_train：正解データの学習用

y_test：正解データの検証用

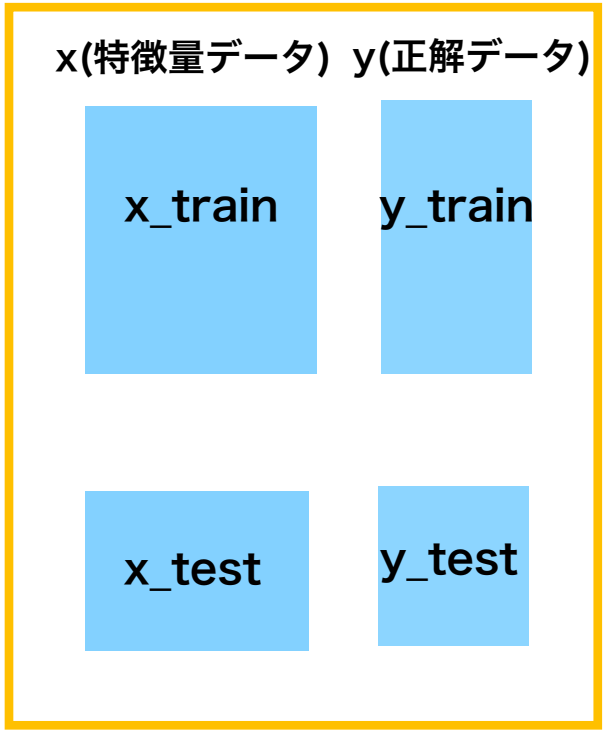
*今回作成するのは検証用のデータだけど、train_test_split()関数を使うため、データ名はx_testとy_testを作ることに注意

機械学習の流れのまとめ

STEP1:データ分割

機械学習の流れのまとめ

STEP1:データ分割



機械学習の流れのまとめ

STEP1:データ分割

x(特徴量データ)　y(正解データ)

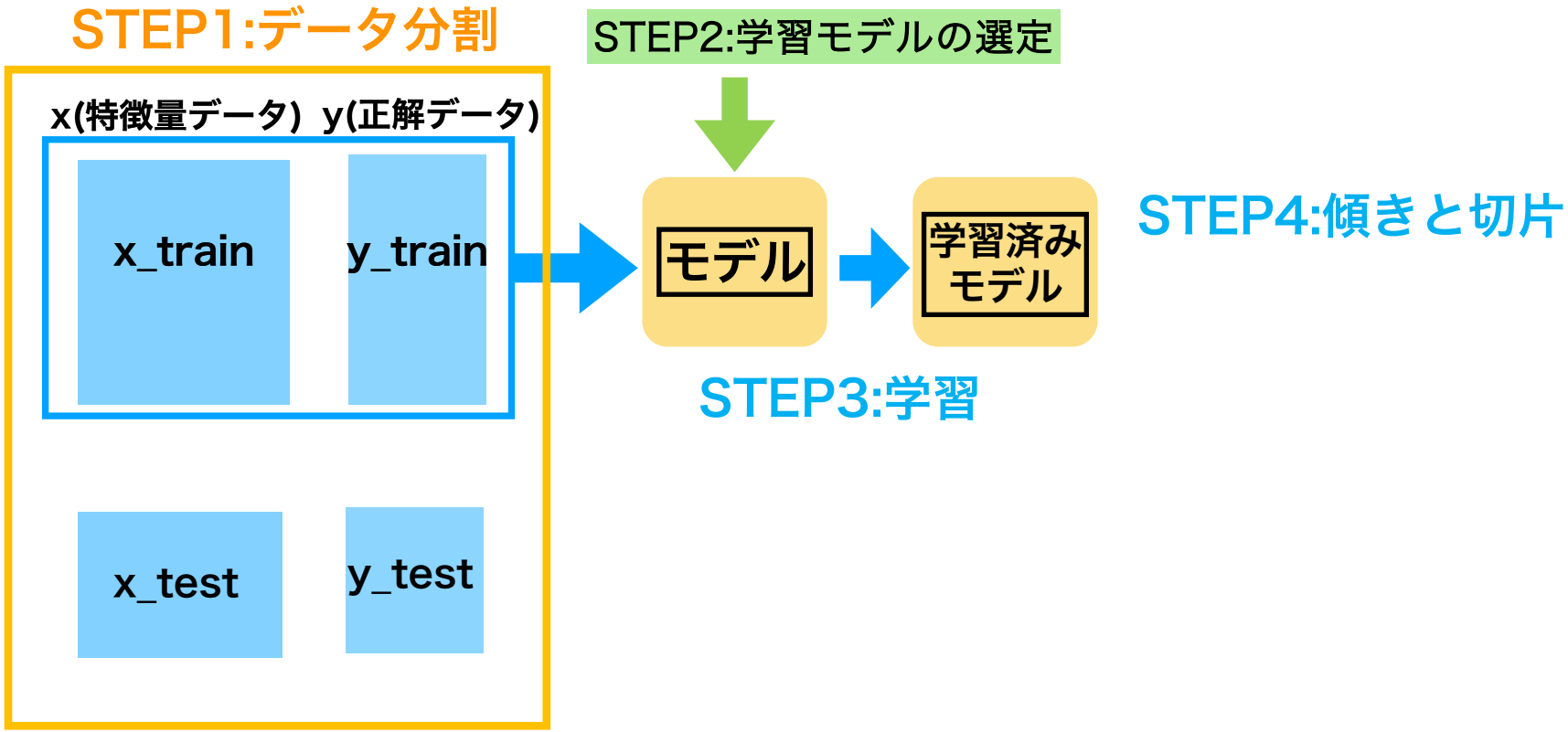
x_train　y_train

x_test　y_test

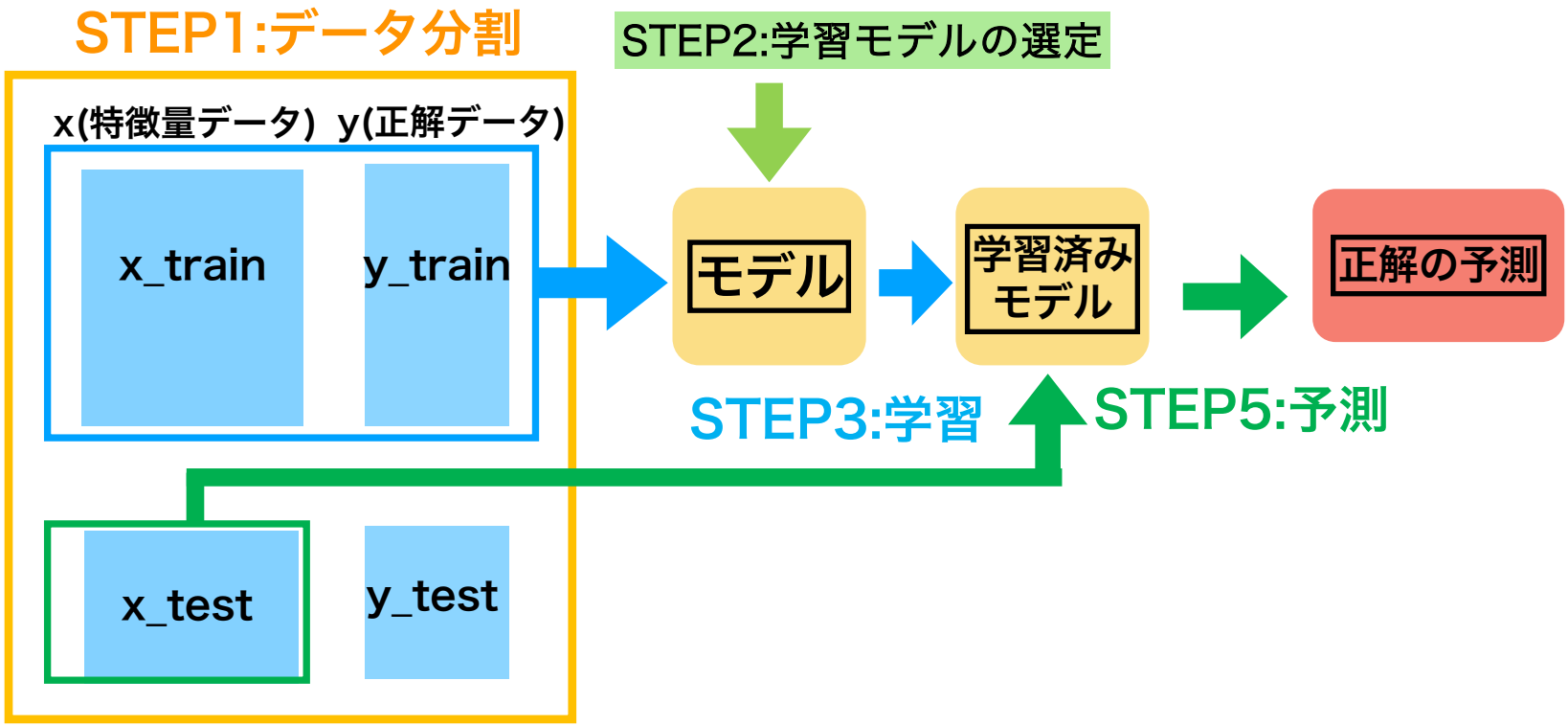
STEP2:学習モデルの選定

モデル

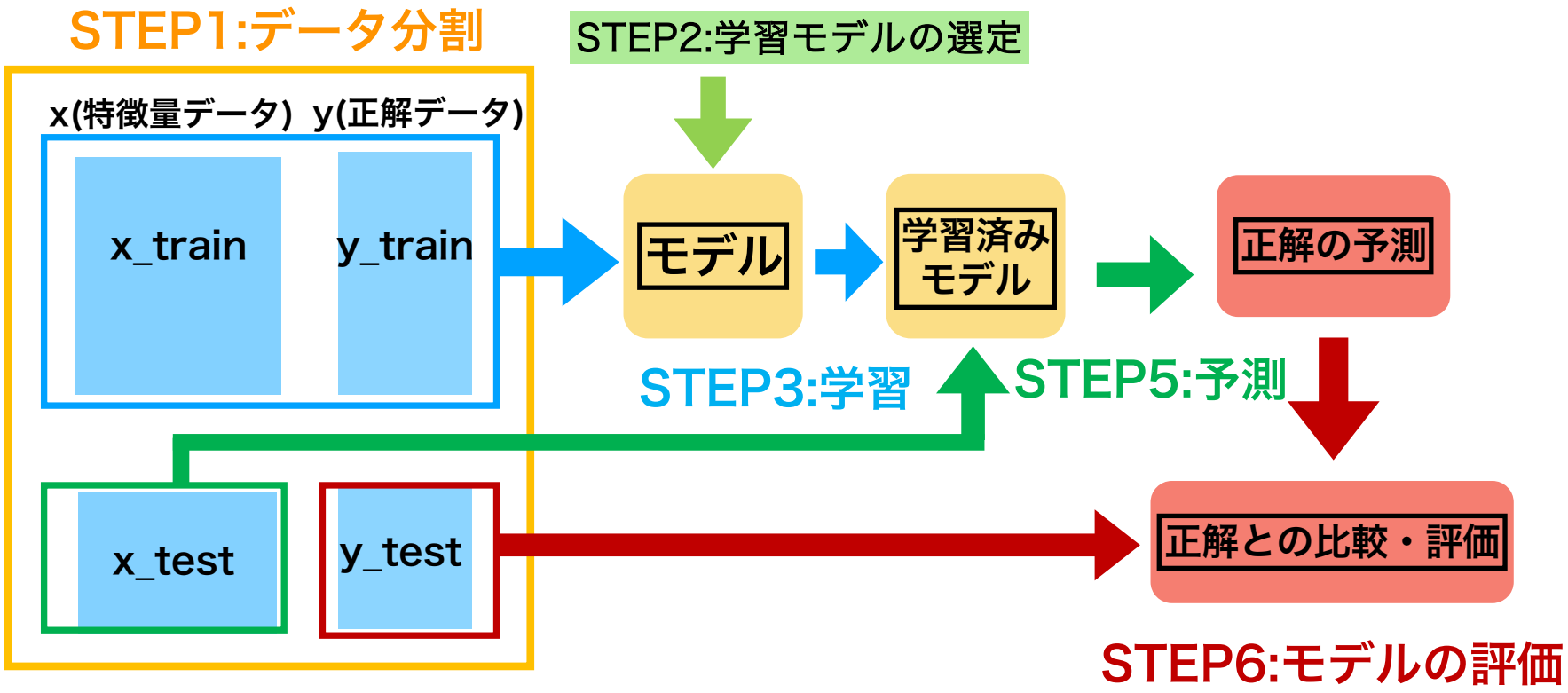
機械学習の流れのまとめ



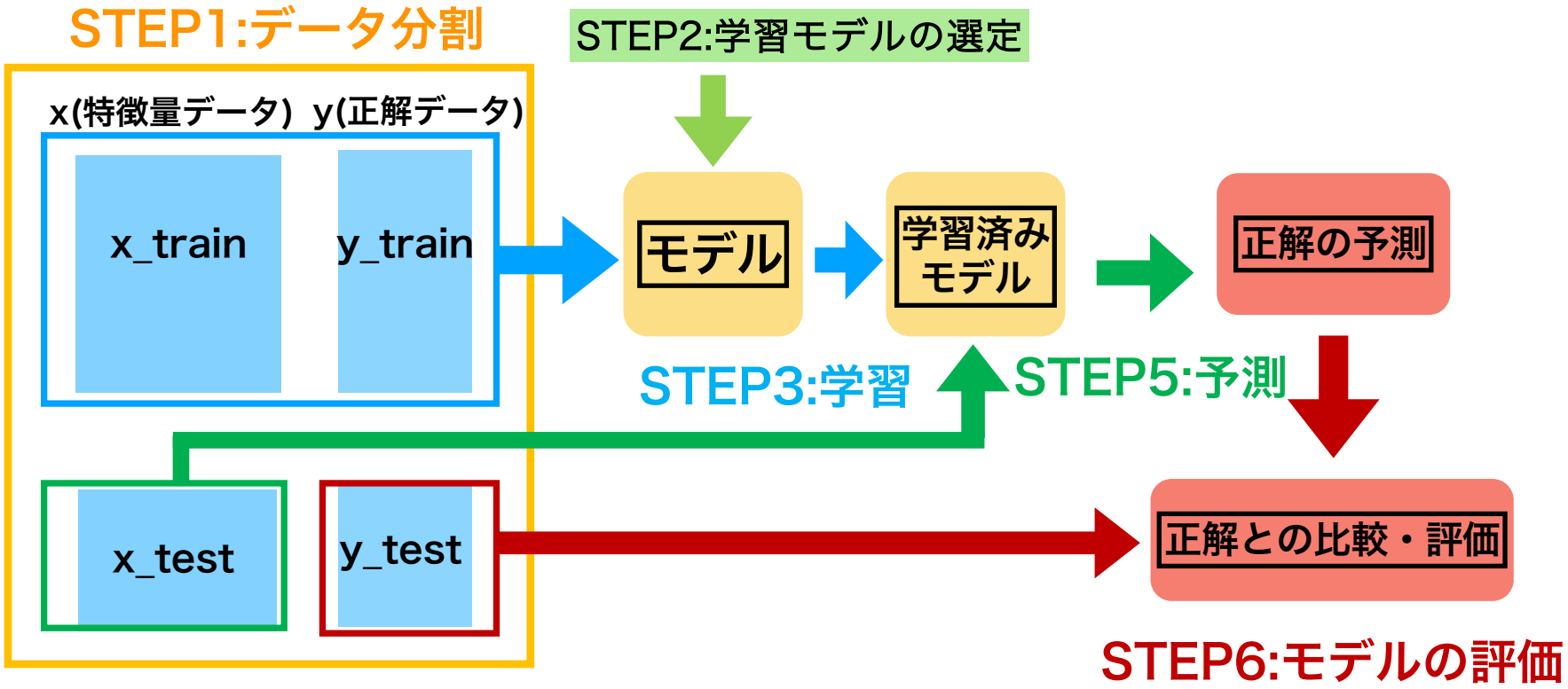
機械学習の流れのまとめ



機械学習の流れのまとめ



機械学習の流れのまとめ



STEP1でデータの分割を行い、STEPごとで使用するデータを変更する

線形回帰分析

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

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

演習10コード.ipynb

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

演習10コード.ipynb

線形回帰分析

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

前回の続きのため、配布した演習10.ipynbの
コード10-1~コード10-2を実行してください

糖尿病のデータを引き続き使う

colab

全データを学習用と検証用に分割して学習

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10-3 学習用と検証用データに分割

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_dm,
y_dm, test_size = 0.3, random_state = 0)
```

●関数をインポート

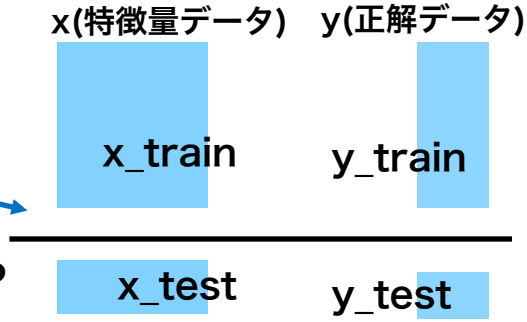
ライブラリ名　モジュール名　関数名

```
from sklearn.model_selection import train_test_split
```

● `train_test_split(x,y,test_size=, random_state=)`
で引数を指定

`test_size = 0.3`

testのデータを全体の30%にする

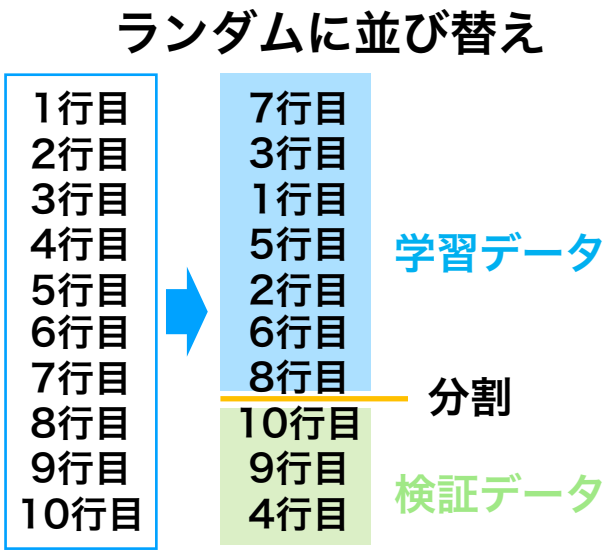


全データを学習用と検証用に分割して学習

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

random_state = 数値 は乱数シード値を固定する

- 同じ数値を指定すると、どんな環境でも同じ乱数が生成される
- train_test_split()はデータを分割する前にデータの行の順番をランダムに変更する
- random_stateを指定しないと、train_test_splitをするたびに異なるtrainデータとtestデータができる
- 教員と学生の全員の分割データを一致させるため、今回はrandom_state=0を指定する



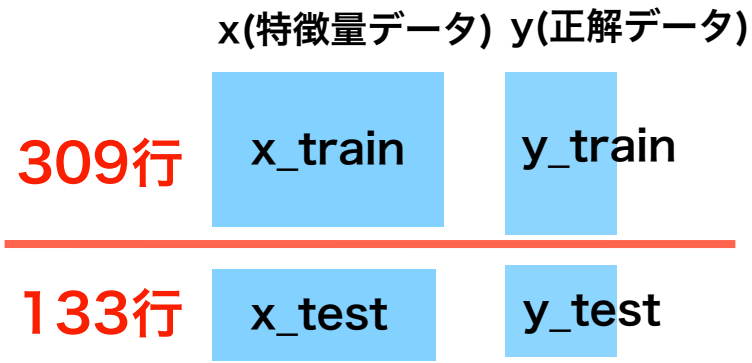
全データを学習用と検証用に分割して学習

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10-4 分割されたデータを調べる

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(309, 10)
(133, 10)
(309,)
(133,)
```



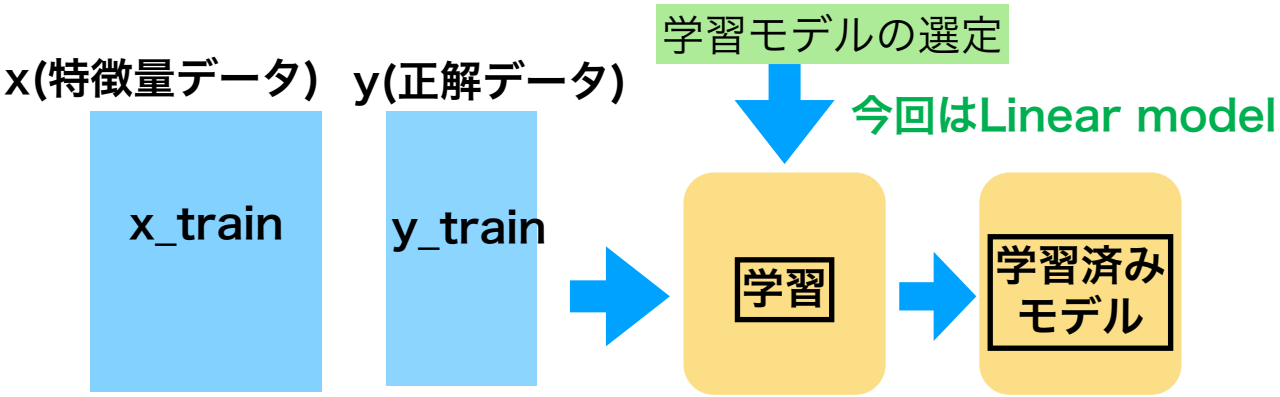
colab

全データを学習用と検証用に分割して学習

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10-5 学習用(train)データで学習させる

```
from sklearn.linear_model import LinearRegression
model_dm = LinearRegression()
model_dm.fit(x_train, y_train)
```



colab

全データを学習用と検証用に分割して学習

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

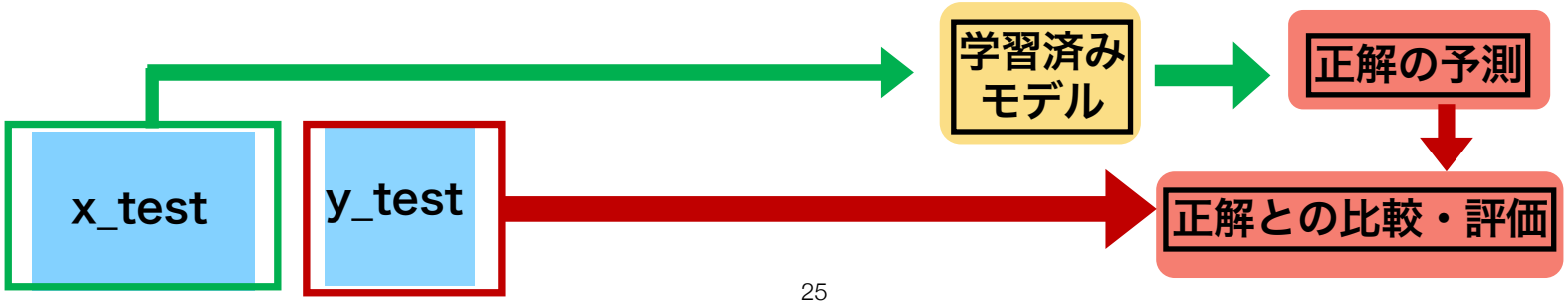
コード 10-6 検証用データで予測結果を評価

```
print(model_dm.predict(x_test))

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

x_testを用いた予測値と、y_test(実際の正解)との違いを比較・評価する

*#np.array()*はNumPy配列の生成をするが、今回はprint(model~)と同じ形式でデータを表示するために使用しているだけ



全データを学習用と検証用に分割して学習

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

予測の値と実際の値でどれぐらい合っているかを目視する

1人目の人は予測値と正解値が近くない

予測の値

[239.67646226 250.52785246 164.85108003 120.27660503 181.73443919
262.21130761 112.23506474 191.94794432 151.49899378 236.97230238
172.17064249 181.76621901 112.11999874 93.10380711 242.61242687
91.18595934 153.65788928 64.67647092 99.36484095 212.26056138
197.04060446 162.46365074 164.1435797 157.73720559 207.25521787
170.20643739 111.97244286 82.77403445 186.91422667 164.71981545
175.37450608 82.34705002 144.29011545 149.22311094 144.09055592

5人目の人は予測値と正解値が近い

正解（実際)の値

[321. 215. 127. 64. 175. 275. 179. 232. 142. 99. 252. 174. 129. 74.
264. 49. 86. 75. 101. 155. 170. 276. 110. 136. 68. 128. 103. 93.
191. 196. 217. 181. 168. 200. 219. 281. 151. 257. 49. 198. 96. 179.
95. 198. 244. 89. 214. 182. 84. 270. 156. 138. 113. 131. 195. 171.
122. 61. 230. 235. 52. 121. 144. 107. 132. 302. 53. 317. 137. 57.
98. 170. 88. 90. 67. 163. 104. 186. 180. 283. 141. 150. 47. 297.
104. 49. 103. 142. 59. 85. 137. 53. 51. 197. 135. 72. 208. 237.
145. 110. 292. 97. 197. 158. 163. 63. 192. 233. 68. 160. 178. 68.
109. 94. 118. 275. 275. 127. 281. 71. 42. 71. 128. 272. 135. 51.
220. 167. 78. 131. 212. 182. 174.]

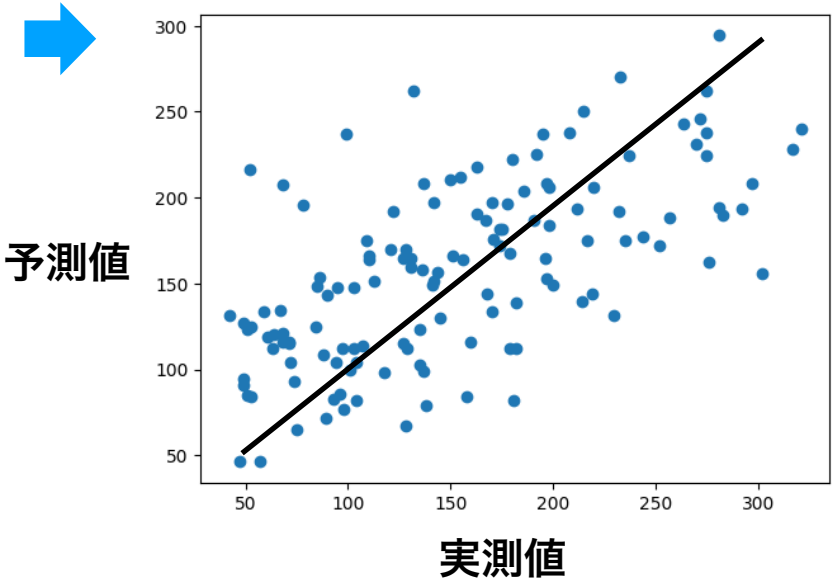
colab

学習モデルの評価

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10-7 検証データで実測値と予測値を比較

```
plt.scatter(y_test,model_dm.predict(x_test))
```



実測値と予測値のズレを散布図で描出

ズレがなければ、直線上に点があるはず

colab

学習モデルの評価

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10-8 検証データでモデルの性能評価

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

```
0.39289927216962883
```

（再掲）全てのデータを使ったモデルの性能

```
print(model_dm10.score(x_dm,y_dm))
 0.5122499966976581
```

予測性能が全てのデータを使った方(model_dm10)がかなり大きいため、過学習していた可能性あり

（再掲）特徴量をbmiだけにしたモデルの性能

```
print(model_bmi.score(x_bmi, y_dm))
 0.3439237602253802
```

colab

ロジスティック回帰

ロジスティック回帰の基礎

重回帰はyが連続変数 (0,1,2,3,4… 好きな数字をとれる)

yが0か1かの二値変数の時はロジスティック回帰を用いる

ロジスティック回帰式

$$P(x) = \frac{1}{1+e^{-(b_0+b_1x_1+b_2x_2+\cdots+bnxn)}}$$

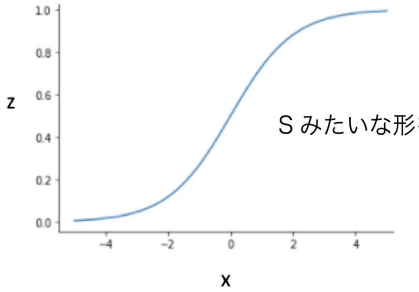
ロジスティック回帰は特徴量xの値でy=1になる確率を予測できる

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

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

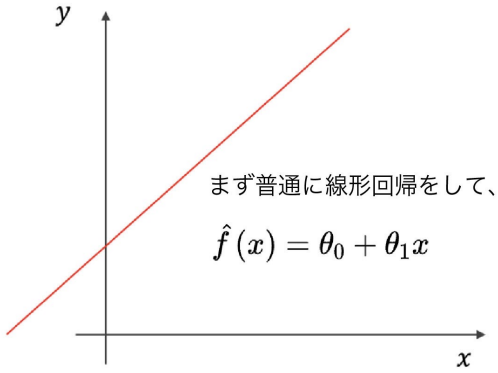
定義域によらず、値域は 0 ~ 1



S みたいな形をとる

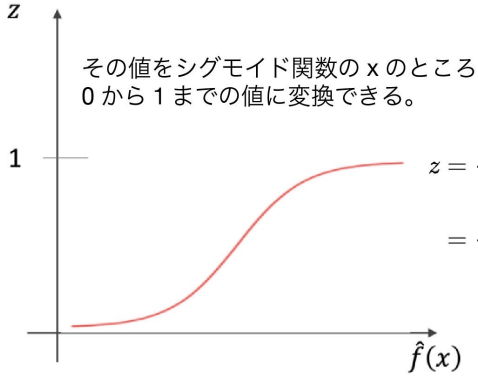
グラフを書いて見ると S のようで、S はギリシャ文字の σ (シグマ) に由来することから、この関数は「シグモイド関数」と呼ばれる。
この関数は x の値によらず 0 から 1 までの値をとる単調増加関数である。

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。



まず普通に線形回帰をして、

$$\hat{f}(x) = \theta_0 + \theta_1 x$$



その値をシグモイド関数の x のところに代入すると、
0 から 1 までの値に変換できる。

$$z = \frac{1}{1 + e^{-\hat{f}(x)}} = \frac{1}{1 + e^{-(\theta_0 + \theta_1 x)}}$$

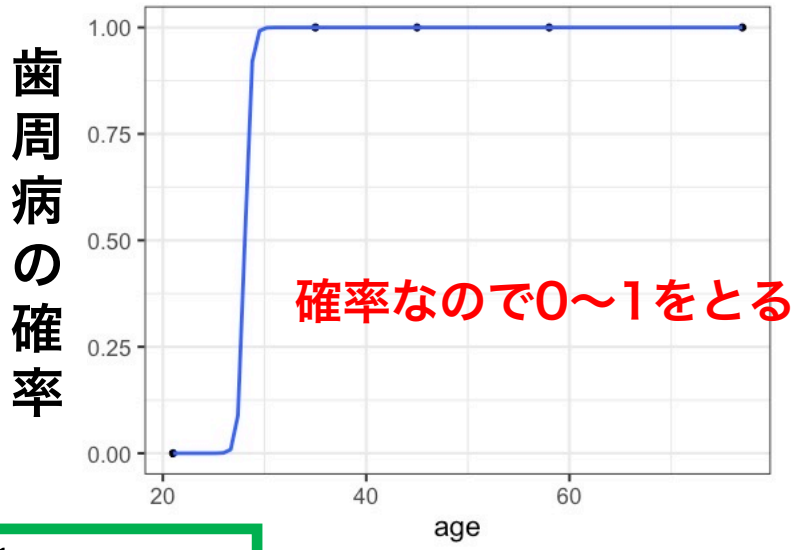
(清水先生講義資料より)

回帰の基礎

yが0か1かの二値変数の時はロジスティック回帰を主に用いる

5人の年齢と歯周病の有無について関係を調べたいとき

被験者	年齢	歯周病の有無 (1：あり、0：なし)
1	35	1
2	21	0
3	45	1
4	58	1
5	77	1



$$P(x) = \frac{1}{1+e^{-(b_0+b_1 age)}}$$

ロジスティック回帰の基礎

y(正解データ）が連続変数のとき「回帰(regression)」と呼び、yがカテゴリ変数(生か死か、りんごかみかんかバナナかなど)を予測することを「分類(classification)」と呼ぶ

*ロジスティック回帰を使って分類するので、紛らわしいので注意

ロジスティック回帰のコードのまとめ

STEP1　データの用意：説明変数のデータフレームと目的変数のデータフレーム

STEP2　学習モデルの選択（今回はロジスティック回帰）

変更点は2つだけ

（モデル名）＝ `LogisticRegression()`

重回帰の時は LinearRegression()だった

STEP3　データを入れて学習させる

（モデル名）.`fit`（説明変数，目的変数）

STEP4　傾き(偏回帰係数)と切片(定数項)を求める

（モデル名）.`coef_`

（モデル名）.`intercept_`

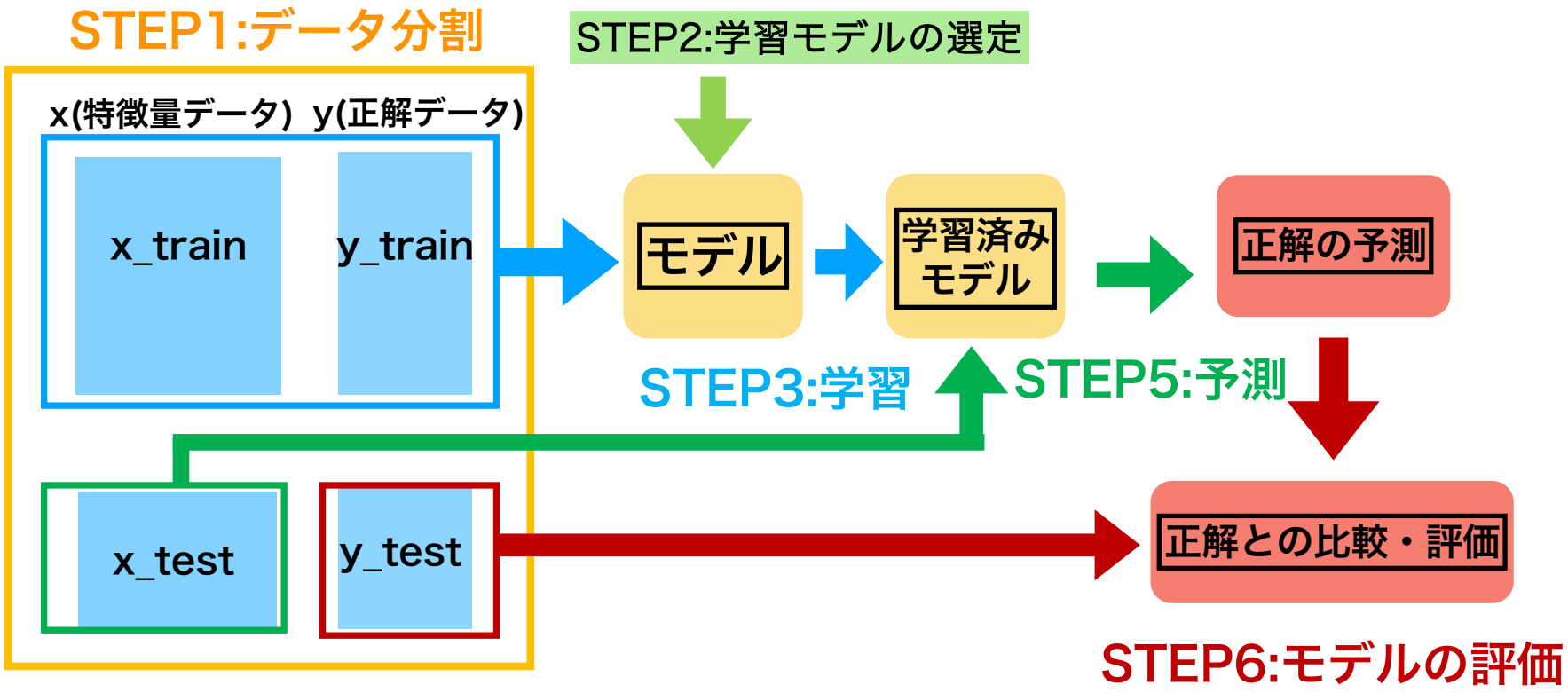
確率の予測が追加

STEP5　予測を行う：（モデル名）.`predict`（新たな説明変数）

（モデル名）.`predict_proba`（新たな説明変数）

STEP6　モデルの評価：（モデル名）.`score`（説明変数，目的変数）

機械学習の流れのまとめ



ロジスティック回帰分析のコードまとめ

```
import numpy as np
import pandas as pd
```

STEP0： ライブラリの読み込み

```
from sklearn.datasets import load_brest_cancer
bc = load_breast_cancer(as_frame = True)
y_bc = bc.target
x_bc = bc.data
x_bc10 = x_bx.iloc[:, 0:10]
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test =
train_test_split(x_bc10,y_bc,test_size=0.3,random_state=0)
```

STEP1：データの準備

```
from sklearn.linear_model import LogisticRegression
model_bc = LogisticRegression()
```

STEP2：学習モデルの選択

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

STEP3：データを入れて学習

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

STEP4：傾きと切片を求める

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

STEP5： 予測を行う

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

STEP6：モデルの評価

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

STEP0：利用ライブラリのインポート

コード 10-9 利用するライブラリを2つインポートする

```
import numpy as np
import pandas as pd
```

- numpyとpandasは前回までの演習で使用
- (今回は作図しないので matplotlibは使わない)

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

STEP1：データの用意

コード 10-10 breast_cancer(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)
```

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

ライブラリ　モジュール　関数

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

*load_breast_cancer()にはscaledの引数はないので、指定しない

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

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

STEP1：データの内容を調べる

コード10-11 bcの内容を調べる①

```
bc                    #データ自体を出力
```

```
type(bc)              #データの型を出力
```

→ `sklearn.utils._bunch.Bunch`

(復習)Bunch型はdictionary(辞書)型の一種

dataset = {“key1”:”a”, “key2”:”b”, “key3”:”c”}

辞書名　キー1　値1　キー2　値2　キー3　値3

要素1　要素2　要素3

辞書型は順序づけたキーと値の集まりで、(データ名).(キー) で格納したデータ(値)を呼び出す

キー

```
{'data':      mean radius  mean texture  mean perimeter  mean area
0        17.99        10.38        122.80        1001.0    0.11840
1        20.57        17.77        132.90        1326.0    0.08474
2        19.69        21.25        130.00        1203.0    0.10960
3        11.42        20.38         77.58         386.1    0.14250
4        20.29        14.34        135.10        1297.0    0.10030
..         ...          ...           ...           ...        ...
564      21.56        22.39        142.00        1479.0    0.11100
565      20.13        28.25        131.20        1261.0    0.09780
566      16.60        28.08        108.30         858.1    0.08455
567      20.60        29.33        140.10        1265.0    0.11780
568       7.76        24.54         47.92         181.0    0.05263

'target': 0    0
1    0
2    0
3    0
4    0
..
564  0
565  0
566  0
567  0
568  1
Name: target, Length: 569, dtype: int64,

'target_names': array(['malignant', 'benign'],
'DESCR': '.. _breast_cancer_dataset:\n\nBrea
:Number of Instances: 569\n\n    :Number of A
distances from center to points on the perime
variation in radius lengths)\n        - compactne
(number of concave portions of the contour)\
"worst" or largest (mean of the three\n        w
field 0 is Mean Radius, field\n        10 is Radius
```

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

STEP1： データの内容を調べる

コード10-12 bcの内容を調べる②

```
print(bc.DESCR)
```

辞書型は順序づけられたキーと値の集まりで、(データ名).(キー) で格納したデータ(値)を呼び出す

bcのキーの一つであるDESCRはデータセットの中身の定義について記載されている部分 (*load_diabetes()と同じ構造)

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

STEP1： データの内容を調べる

bc.DESCR①

Breast cancer wisconsin (diagnostic) dataset

Data Set Characteristics:

:Number of Instances: 569

:Number of Attributes: 30 numeric, predictive attributes and the class

:Attribute Information:
- radius (mean of distances from center to points on the perimeter)
- texture (standard deviation of gray-scale values)
- perimeter
- area
- smoothness (local variation in radius lengths)
- compactness (perimeter^2 / area - 1.0)
- concavity (severity of concave portions of the contour)
- concave points (number of concave portions of the contour)
- symmetry
- fractal dimension ("coastline approximation" - 1)

The mean, standard error, and "worst" or largest (mean of the three worst/largest values) of these features were computed for each image, resulting in 30 features. For instance, field 0 is Mean Radius, field 10 is Radius SE, field 20 is Worst Radius.

- class:
  - WDBC-Malignant
  - WDBC-Benign

bc.DESCR②

:Summary Statistics:

	Min	Max
radius (mean):	6.981	28.11
texture (mean):	9.71	39.28
perimeter (mean):	43.79	188.5
area (mean):	143.5	2501.0
smoothness (mean):	0.053	0.163
compactness (mean):	0.019	0.345
concavity (mean):	0.0	0.427
concave points (mean):	0.0	0.201
symmetry (mean):	0.106	0.304
fractal dimension (mean):	0.05	0.097
radius (standard error):	0.112	2.873
texture (standard error):	0.36	4.885
perimeter (standard error):	0.757	21.98
area (standard error):	6.802	542.2
smoothness (standard error):	0.002	0.031
compactness (standard error):	0.002	0.135
concavity (standard error):	0.0	0.396
concave points (standard error):	0.0	0.053
symmetry (standard error):	0.008	0.079
fractal dimension (standard error):	0.001	0.03
radius (worst):	7.93	36.04
texture (worst):	12.02	49.54
perimeter (worst):	50.41	251.2
area (worst):	185.2	4254.0
smoothness (worst):	0.071	0.223
compactness (worst):	0.027	1.058
concavity (worst):	0.0	1.252
concave points (worst):	0.0	0.291
symmetry (worst):	0.156	0.664
fractal dimension (worst):	0.055	0.208

bc.DESCR③

:Missing Attribute Values: None

:Class Distribution: 212 - Malignant, 357 - Benign

:Creator: Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

:Donor: Nick Street

:Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
https://goo.gl/U2Uwz2

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

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

STEP1： データの内容を調べる

bc.DESCR①

Breast cancer wisconsin (diagnostic) dataset

Data Set Characteristics:

:Number of Instances

:Number of Attribute

:Attribute Informatio
- radius (mean of d
- texture (standard
- perimeter
- area
- smoothness (loca
- compactness (pe
- concavity (severit
- concave points (n
- symmetry
- fractal dimension

The mean, standar
worst/largest value
resulting in 30 feat
10 is Radius SE, fie

- class:
- WDBC-Malign
- WDBC-Benign

bc.DESCR②

:Summary Statistics:

========== ====== ======
Min Max
========== ====== ======

========== ====== ======

- このデータは UCI (カリフォルニア大学アーバイン校) ML(機械学習)用に提供されているウィスコンシン大学のBreast Cancer の診断のデータセットである
- 1995年11月のデータ
- 特徴量は乳房腫瘤の穿刺吸引（FNA）のデジタル化画像から計算

*穿刺吸引は腫瘍に細い針を刺して細胞を回収し、組織診を行うこと

bc.DESCR③

:Missing Attribute Values: None

:Class Distribution: 212 - Malignant, 357 - Benign

:Creator: Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

:Donor: Nick Street

:Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
https://goo.gl/U2Uwz2

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

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

STEP1： データの内容を調べる

bc.DESCR①

Breast cancer wisconsin (diagnostic) dataset

Data Set Characteristics:

:Number of Instances: 569

行は569人

:Number of Attributes: 30 numeric, predictive attributes and the class

:Attribute Information:
- radius (mean of distances from center to points on the perimeter)
- texture (standard deviation of gray-scale values)
- perimeter
- area
- smoothness (local variation in radius lengths)
- compactness (perimeter^2 / area - 1.0)
- concavity (severity of concave portions of the contour)
- concave points (number of concave portions of the contour)
- symmetry
- fractal dimension ("coastline approximation" - 1)

The mean, standard error, and "worst" or largest (mean of the three worst/largest values) of these features were computed for each image, resulting in 30 features. For instance, field 0 is Mean Radius, field 10 is Radius SE, field 20 is Worst Radius.

- class:
  - WDBC-Malignant
  - WDBC-Benign

30個の数値型の特徴量データがある

腫瘍細胞核のradius(半径)、texture(グレースケール値の標準偏差)、perimeter(周囲径)、area(面積)、smoothness(滑らかさ)、compactness(コンパクト度)、concavity(輪郭の凹部)、concave points(凹部の数)、symmetry(対称性)、fractal dimension(フラクタル次元の数値)

10個の変数について、それぞれ「平均値」「標準誤差」「最大値」の3つの変数を作成して、計30個

*標準誤差は 不偏分散/√n

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

STEP1： データの内容を調べる

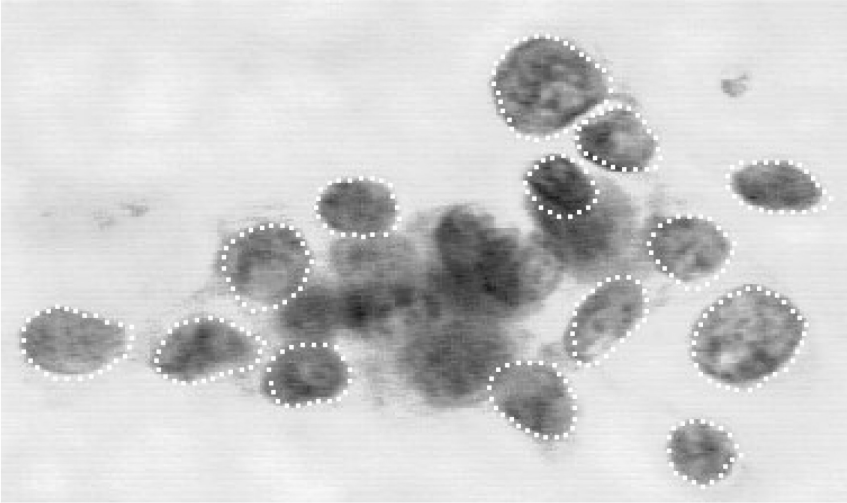


Figure 1: Initial Approximate Boundaries of Cell Nuclei
The user first draws a rough initial outline of some cell nucleus boundaries. Each outline serves as the initial position for a deformable spline which converges to an accurate boundary of the nucleus.

1993 W.N. Street, W.H. Wolberg and O.L. Mangasarian
Nuclear feature extraction for breast tumor diagnosis
IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology, volume 1905, pages 861-870, San Jose, CA, 1993.
Figure1 引用

30個の数値型の特徴量データがある

腫瘍細胞核のradius(半径)、texture(グレースケール値の標準偏差)、perimeter(周囲径)、area(面積)、smoothness(滑らかさ)、compactness(コンパクト度)、concavity(輪郭の凹部)、concave points(凹部の数)、symmetry(対称性)、fractal dimension(フラクタル次元の数値)

ロジスティック回帰分析

STEP1： データの内容を調べる

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：傾きと切片を求める
STEP5：予測を行う
STEP6：モデルの評価

それぞれの変数の最小値と最大値の記述

bc.DESCR②

```
:Summary Statistics:

===================================== ====== ======
                          Min    Max
===================================== ====== ======
radius (mean):                6.981  28.11
texture (mean):               9.71   39.28
perimeter (mean):             43.79  188.5
area (mean):                143.5  2501.0
smoothness (mean):              0.053  0.163
compactness (mean):             0.019  0.345
concavity (mean):             0.0    0.427
concave points (mean):          0.0    0.201
symmetry (mean):              0.106  0.304
fractal dimension (mean):       0.05   0.097
radius (standard error):      0.112  2.873
texture (standard error):       0.36   4.885
perimeter (standard error):     0.757  21.98
area (standard error):        6.802  542.2
smoothness (standard error):      0.002  0.031
compactness (standard error):     0.002  0.135
concavity (standard error):     0.0    0.396
concave points (standard error):  0.0    0.053
symmetry (standard error):      0.008  0.079
fractal dimension (standard error):  0.001  0.03
radius (worst):               7.93   36.04
texture (worst):              12.02  49.54
perimeter (worst):            50.41  251.2
area (worst):               185.2  4254.0
smoothness (worst):             0.071  0.223
compactness (worst):            0.027  1.058
concavity (worst):            0.0    1.252
concave points (worst):         0.0    0.291
symmetry (worst):             0.156  0.664
fractal dimension (worst):      0.055  0.208
===================================== ====== ======
```

bc.DESCR③

:Missing Attribute Values: None

:Class Distribution: 212 - Malignant, 357 - Benign

:Creator: Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

:Donor: Nick Street

:Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
https://goo.gl/U2Uwz2

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

正解値は
212人がMalignant(悪性, がん)、
357人がBenign(良性)

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

STEP1： データの内容を調べる

569個の腫瘍のデータ

細胞核の特徴
半径：8
面積：200
滑らかさ：0.100
・・・

target：0

30個の特徴量(data)
乳房腫瘤の穿刺吸引（FNA）のデジタル化画像

正解値（target）
良性腫瘍か悪性腫瘍(がん)かの診断
良性腫瘍が1,悪性腫瘍が0

腫瘍が良性か悪性かを、細胞診のデータで予測するモデルを作成する

演習10：課題

Webclassで課題を提出してください。締め切りは2024/01/31 23:59まで

(1) dmデータをtrain_test_split()を使って、データを分割してください。その際、test_sizeを15%に設定し、乱数シード値を自分の学籍番号の下四桁(“1234”,”0058”など)にしてデータを作成するコードを記載してください。

(2) (1)のデータセットで全ての特徴量を使用して重回帰を行ってください。(コードを記載してください)

(3) 決定係数 R^2 を回答してください。（出力結果)

(4) bcデータの正解値=1の時、腫瘍は良性ですか悪性ですか？

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

2024/1/18 11:35-12:20

演習11
回帰4

統合教育機構
石丸美穂

ロジスティック回帰分析のコードまとめ

```
import numpy as np
import pandas as pd
```

STEP0： ライブラリの読み込み

```
from sklearn.datasets import load_brest_cancer
bc = load_breast_cancer(as_frame = True)
y_bc = bc.target
x_bc = bc.data
x_bc10 = x_bx.iloc[:, 0:10]
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test =
train_test_split(x_bc10,y_bc,test_size=0.3,random_state=0)
```

STEP1：データの準備

```
from sklearn.linear_model import LogisticRegression
model_bc = LogisticRegression()
```

STEP2：学習モデルの選択

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

STEP3：データを入れて学習

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

STEP4：傾きと切片を求める

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

STEP5： 予測を行う

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

STEP6：モデルの評価

ロジスティック回帰分析

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

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

演習11コード.ipynb

ロジスティック回帰分析

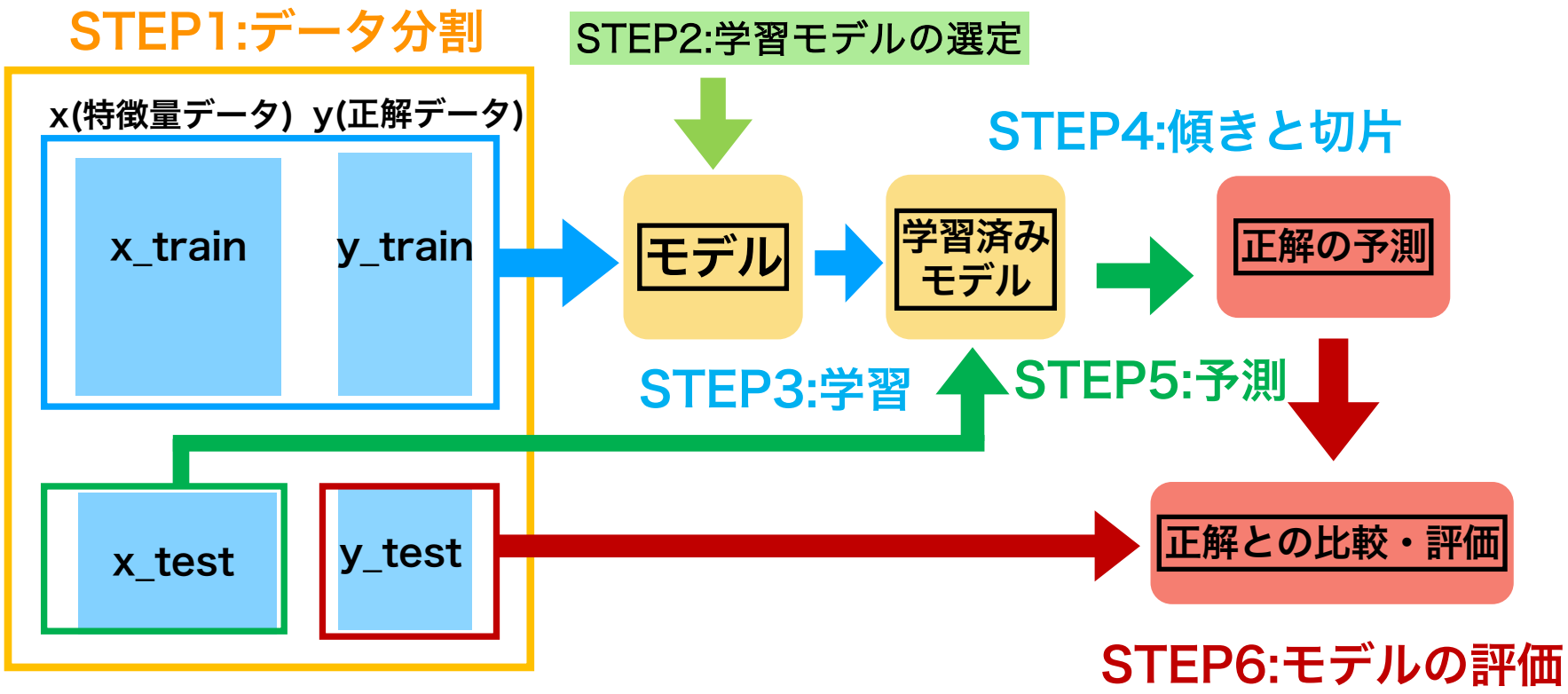
STEP1：データの用意

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

前回の続きのため、配布した演習11コード.ipynbの
コード11-1~コード11-2を実行してください

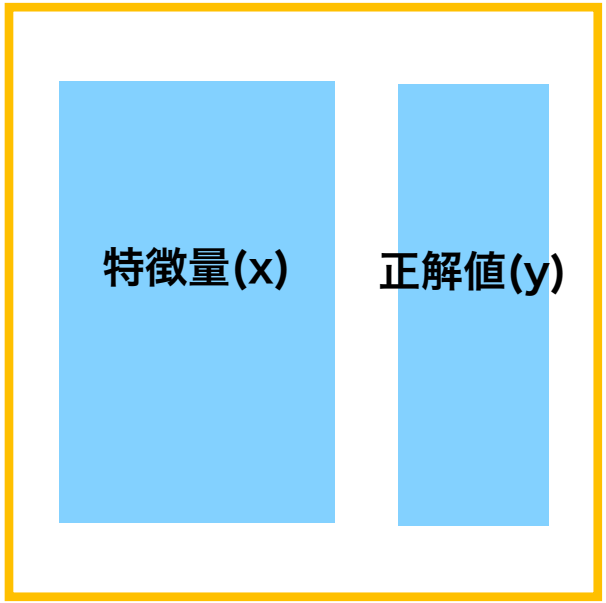
colab

機械学習の流れのまとめ



機械学習の流れのまとめ

STEP1:データの用意



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

STEP1： データの内容を調べる

今回使うのは乳がんの細胞診のデータ

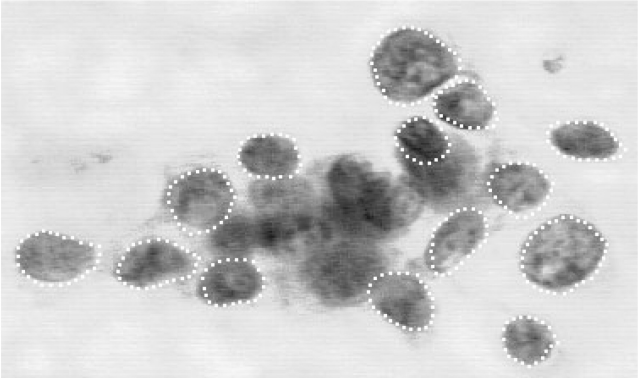


Figure 1: Initial Approximate Boundaries of Cell Nuclei
The user first draws a rough initial outline of some cell nucleus boundaries. Each outline serves as the initial position for a deformable spline which converges to an accurate boundary of the nucleus.

1993 W.N. Street, W.H. Wolberg and O.L. Mangasarian
Nuclear feature extraction for breast tumor diagnosis
IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology, volume 1905, pages 861-870, San Jose, CA, 1993.
Figure1 引用

569人分のデータ
target(正解値）は良性腫瘍(1)か悪性腫瘍(0)かの分類

特徴量は30個のデータがある

腫瘍細胞核のradius(半径)、texture(グレースケール値の標準偏差)、perimeter(周囲径)、area(面積)、smoothness(滑らかさ)、compactness(コンパクト度)、concavity(輪郭の凹部)、concave points(凹部の数)、symmetry(対称性)、fractal dimension(フラクタル次元の数値)

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

STEP1：正解値データの用意

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

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

STEP1：正解値データの記述統計

コード11-4 y(腫瘍の分類)の頻度を調べる

```
y_bc.value_counts()
```

```
1    357
0    212
Name: target, dtype: int64
```

1(良性腫瘍)が357,0(悪性腫瘍)が212

y_bcはpandasのSeries型なので、(データ名).value_counts()でユニークな要素の値とその出現頻度を返す

ユニークな要素の値 → 1/0

その出現頻度 → 全ての行の中で何回その値が出現したかの集計値

*(データ名).describe()は平均値などを返しカテゴリ変数の時は意味がないため、今回はvalue_counts()を使う

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

STEP1：特徴量データの用意

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

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

STEP1：特徴量データの用意

コード11-6 列名を出力

```
x_bc.columns
```

```
Index(['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='object')
```

mean~から始まる10個の特徴量

~error の10個の特徴量

worst~の10個の特徴量

colab

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

30個の変数は同じ指標(例えば細胞核の半径）の平均値・標準誤差・最大値が含まれる。変数同士の関連が強すぎて、回帰に同時に入れると収束しない

↓

今回は10個の平均値の変数（1～10個目）のみを選択し、x_bc10を作成

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

STEP1：特徴量データの用意

コード11-7 特徴量データから1~10列抽出

```
x_bc10 = x_bc.iloc[:, 0:10]
x_bc10.head()  #最初の5行を出力
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness
0	17.99	10.38	122.80	1001.0	0.11840	0.27760
1	20.57	17.77	132.90	1326.0	0.08474	0.07864
2	19.69	21.25	130.00	1203.0	0.10960	0.15990
3	11.42	20.38	77.58	386.1	0.14250	0.28390
4	20.29	14.34	135.10	1297.0	0.10030	0.13280

mean~から始まる
10個の特徴量

(データフレーム名).iloc[行番号,列番号]で特定の行と列をインデックス番号で指定して抽出　(演習5の復習)

<u>行番号は全てなので ':' 、列番号は0:10(0~10)で指定する</u>

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

STEP1：正解値データの整形

コード11-8 x_bc10の配列構造と型を出力

```
x_bc10.shape  #データの配列構造を出力
```

➡ (569,10)　569行×10列の2次元配列(行列)

```
type(x_bc10)  #データの型を出力
```

➡ pandas.core.frame.DataFrame

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

STEP1：特徴量データの記述

コード11-9 x_bc10の記述統計値を出力

```
x_bc10.describe()
```

(データフレーム名).describe()で記述統計値を出力

radius(半径)、texture(グレースケール値の標準偏差)、perimeter(周囲径)、area(面積)、smoothness(滑らかさ)、compactness(コンパクト度)、concavity(輪郭の凹部)、concave points(凹部の数)、symmetry(対称性)、fractal dimension(フラクタル次元の数値)

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000

配列

回帰で学習を行う時、xとyの配列が指定されている

`x_bc10`, `y_bc`

xの配列構造は2次元配列（行列の形）

(569,10)

yの配列構造は1次元配列（ベクトルの形）

(569,)

配列を間違うとエラーになるので、作成したデータの配列を確認するように気を付ける

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

STEP1：データの用意

コード11-10 学習データと検証データに分割

*今回は最初から学習データと検証データ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(
    x_bc10, y_bc, 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

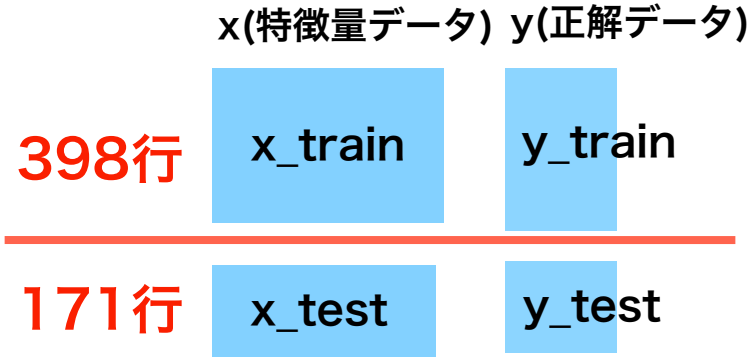
全データを学習用と検証用に分割して学習

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：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11-11 分割されたデータを調べる

```
print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

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



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

STEP2：学習モデルの選択

コード11-12 ロジスティック回帰モデルを選択

```
from sklearn.linear_model import LogisticRegression
model_bc = LogisticRegression()
```

クラスだけをインポート `from sklearn.linear_model import LogisticRegression`

ライブラリ名　モジュール名　クラス名

`LogisticRegression()`クラスから`model_bc`をインスタンスを作成する
（この時点ではモデル構造のみが指定されている）

colab

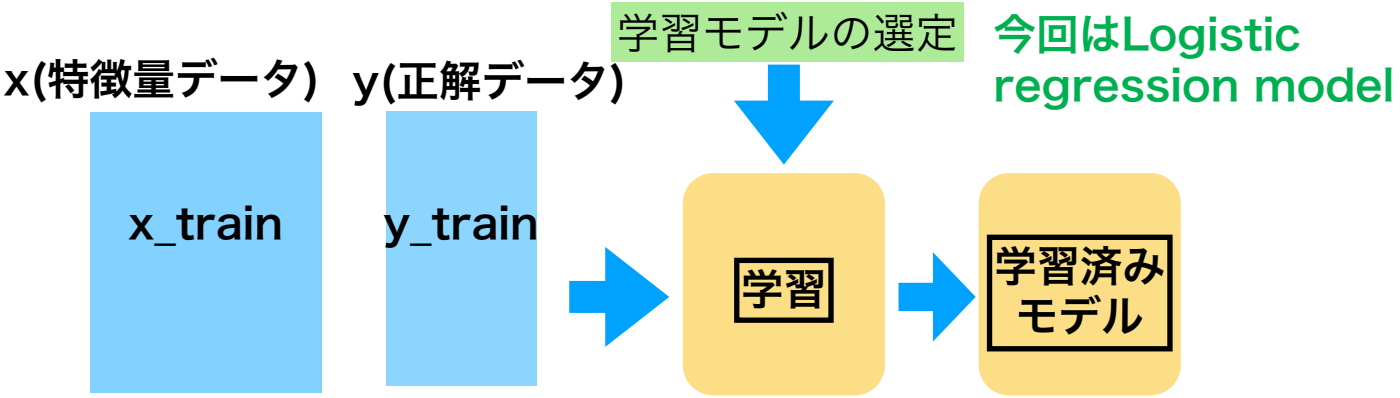
ロジスティック回帰分析

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

STEP3：データを入れて学習させる

コード11-13 学習用データで学習させる

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



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

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11-15 検証用データで分類を行う

```
print(model_bc.predict(x_test))

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

(モデル名).predict()で分類

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

x_testのデータからyが0/1かを分類

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

y_testの実際の値

1人目は不正解で、3人目は正解の値が入っている

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

ロジスティック回帰分析

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11-17 検証データでモデルの評価を行う

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

➡ 0.9181286549707602

(モデル名).score()でロジスティック回帰の時は正解率が出力される

colab

(清水先生の講義引用)

どんな分類器も検査も完璧ではない

2 値分類において，0 を陰性 (**negative**), 1 を陽性 (**positive**) と呼ぶことがある。例えば病気の検査であれば，
病気であるケースを 1 とし陽性 positive, 病気でないケースを 0 とし陰性 negative とするのが一般的だ。
そしてどんな分類器も検査も完璧ではない。

これらの positive/negative のデータに対して予測した結果，正解していることを真 (**True**)，間違っていることを偽 (**False**) という。

本当は陽性であるデータに対して正しく陽性と予測したケースを真陽性 (True Positive: TP)
間違えて陰性と予測したケースを偽陰性 (False Negative: FN)

本当は陰性であるデータに対して正しく陰性と予測したケースを真陰性 (True Negative: TN)
間違えて陽性と予測したケースを偽陽性 (False Positive: FP)

これらの 4 通りが考えられ、表形式で表したものを
混同行列 confusion matrix という。

		予測	
		Positive	Negative
正解	Positive	TPの数	FNの数
	Negative	FPの数	TNの数

そして全体のうち、正しく予測できた TP, TN (右図赤字) の割合のことを、
その分類器なり検査法なりの正解率 accuracy という。

今回の正解率0.918128654970760 2 ＝
予測した分類の92%が正解していた

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

STEP6：モデルの評価

コード11-18 混同行列の出力

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

➡ array([[102, 6],
       [8, 55]])

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

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

STEP6：モデルの評価

コード11-18 混同行列の出力

```
array([[102,  6],
       [ 8, 55]])
```

		予測結果	
		positive(正)	Negative(負)
実際の分類結果	positive(正)	真陽性True Positive **102**	偽陰性False Negative **6**
	Negative(負)	偽陽性False Positive **8**	真陰性True Negative **55**

正解率

= 102+55/102+6+8+55

= 157/171

= 0.918

colab

正解率だけで分類器を評価していいのか？

(清水先生の講義引用)

あなたは病院勤務の新人医療スタッフです。患者さんから相談を受けました。

「先生には怖くて聞けないのですが、区民健診で便潜血を受けたら陽性で、大腸がんの疑いがあるから
精密検査を受けるように言われました。私、大腸がんなんでしょうか？」

このケースにおいては、「いいえ、あなたは大腸がんではありません」と (全ての患者に対して)
答えておけば、正解率は非常に高くなるでしょう。

というのも、便潜血陽性のうちで大腸がんがある方は数 % しかいませんので、100 人の患者さんのうち
TN が 97 くらいになって、患者もデータも全く見なくても正解率 97% で的中させられます。
もちろん、残り 3 人の大腸がん患者さんは完全に見逃してしまいますが .....

このように、現実のデータ分析においては特定のクラス (今回のケースだと「大腸がんではない」) に大きく偏っている
場合がほとんどであり、このようなデータのことを不均衡データ (unbalanced data) といいます。
その場合、分類タスクにおける正解率を高めるだけなら一番可能性の高い選択肢を迷わず選べばよいだけです。
なので、正解率だけで分類システムを評価してはいけません。

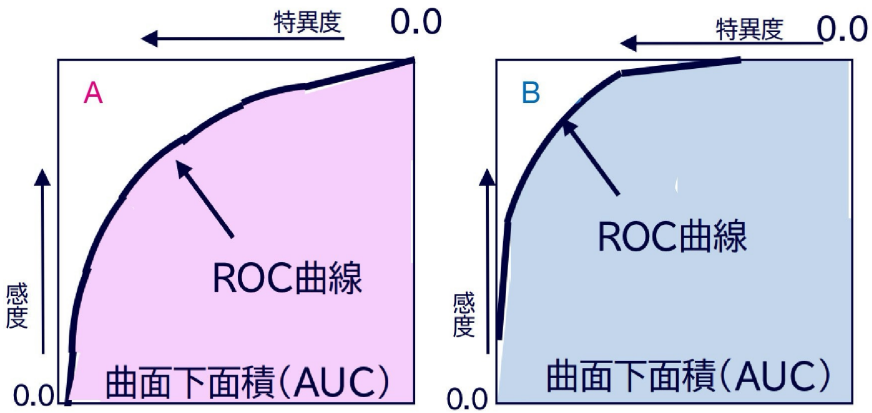
(清水先生の講義引用)

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。

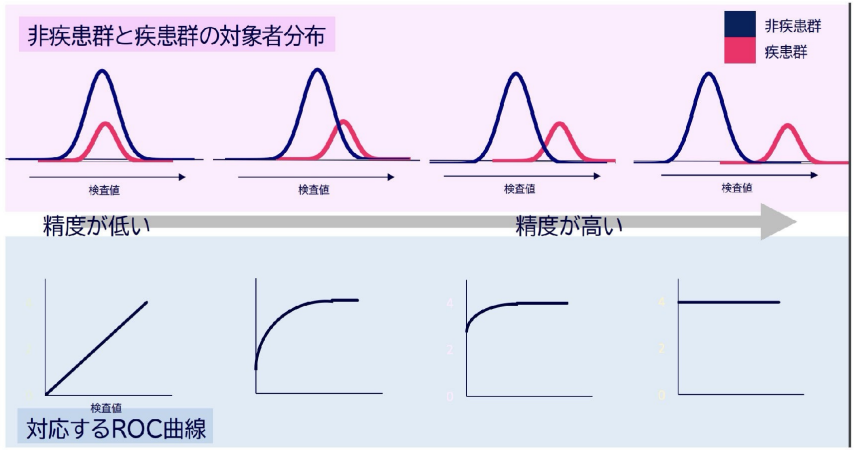
さまざまなカットオフで感度・特異度を計算し、横軸に 1 - 特異度、縦軸に感度をプロットしたものだ。
左上が感度 1、特異度 1 という理想的な分類器を示す。

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

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

STEP6：モデルの評価

コード11-19 AUCを算出

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

➡ 0.982363315696649

- 関数をインポート　`from sklearn.metrics import roc_auc_score`

  ライブラリ名　モジュール名　関数名

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

  (モデル名).predict_proba()[:,1]

colab

まとめ

機械学習の一歩目として、
線形回帰とロジスティック回帰を実践しました

STEP1： データの用意
特徴量と正解値のデータフレーム

STEP2：学習モデルの選択
(モデル名) = LinearRegression()

STEP3：データを入れて学習させる
(モデル名).fit(特徴量,正解値)

STEP4：傾きと切片を求める
(モデル名).coef_
(モデル名).intercept_

STEP5：予測を行う
(モデル名).predict(新たなx)

STEP6：モデルの評価：
(モデル名).score(特徴量,正解値)

STEP1：データの用意
特徴量と正解値のデータフレーム

STEP2：学習モデルの選択
(モデル名) = LogisticRegression()

STEP3：データを入れて学習させる
(モデル名).fit(特徴量,正解値)

STEP4：傾きと切片を求める
(モデル名).coef_
(モデル名).intercept_

STEP5：予測を行う
(モデル名).predict(新たなx)
(モデル名).predict_proba(新たなx)

STEP6：モデルの評価
(モデル名).score(特徴量,正解値)

まとめ

機械学習の一歩目として、
線形回帰とロジスティック回帰を実践しました

回帰モデルは回帰式ではっきりと学習済みモデルの内容がわかるため
「ホワイトボックスモデル」と呼ばれる

個人がどの特徴量によって、1/0に分類されるかなど理屈がわかりやすい

しかし、線形関数やシグモイド関数にデータが当てはまっているとは限らないため、予測性能がそれほど高くならないこともある

今後は予測性能を高める様々なアルゴリズムの機械学習を行う

演習11：課題

Webclassで課題を提出してください。締め切りは2024/02/01 23:59まで

ばらつきに着目した変数(標準誤差)を特徴量として使って解析してください

(1) breast_cancerデータのデータセットで特徴量を11~20個目(インデックス番号10~19)の特徴量データ(x_bc20)を作成してください。（コード)

(2) (1)のデータを用いてradius errorの平均値、最小値、最大値を記載してください（出力結果)

(3) (1)のデータを用いてx_bc20データとy_bcデータを分割し、testデータを作成するときに全体の20%で、乱数シード値を”0”にしてデータを作成するコードを記載してください。（コード)

(3)のデータでロジスティック回帰を行なってください。

(4)正解率、(5)AUCを回答してください。（出力結果)

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　10:40-11:25

医療とAI・ビッグデータ入門
演習12

教師あり機械学習-SVM-

統合教育機構
石丸美穂

前回(演習11)
ロジスティック回帰の修正

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

STEP6：モデルの評価

コード11-19 AUCを算出

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

➡ 0.982363315696649

- 関数をインポート　`from sklearn.metrics import roc_auc_score`

  ライブラリ名　モジュール名　関数名

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

  (モデル名).predict_proba()[:,1]

ロジスティック回帰分析

Y=1/0になる確率

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

*(モデル名).predict_proba()で予測確率を求められる

```
model_bc.predict_proba(x_test)[:,1])
```

	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]

演習12-14の概要

- 演習8-11では教師あり機械学習の中で、モデルがわかりやすい回帰分析を実行した
- 他にも教師あり機械学習には多くのアルゴリズムが存在する
- それぞれ利点・欠点があるため、適切なモデルの設定できるように、代表的なモデルを実行できるようにする

機械学習が目指すこと

広さ [平方メートル] を X_1 とし，家賃 [万] を Y とすると，これら 2 つの変数の間には何かしらの関係があるとし以下のような式で表せると考えることができる。

$$Y = f(X_1) + \epsilon$$

$f(X_1)$ は X_1 を引数とした何かしらの関数という意味。ε は誤差項 (error term) と呼ばれるもので Y のうち X_1 では説明できない部分。
Y は目的変数、X_1 は説明変数だが、機械学習では説明変数のことを特徴量 (feature) と呼ぶ。目的変数を特徴付ける変数という意味合いだ。

一般に「家賃」を特徴付ける特徴量は「広さ」以外にも「築年数」や「駅からの距離」など，他にも多くの特徴量があると考えられる。

$$Y = f(X_1, X_2, X_3, \cdots) + \epsilon$$

これら n 個の特徴量 X_1 X_2 X_3 ... X_n をまとめて X (というベクトル) と書くことにすると、次のように表すことができる。

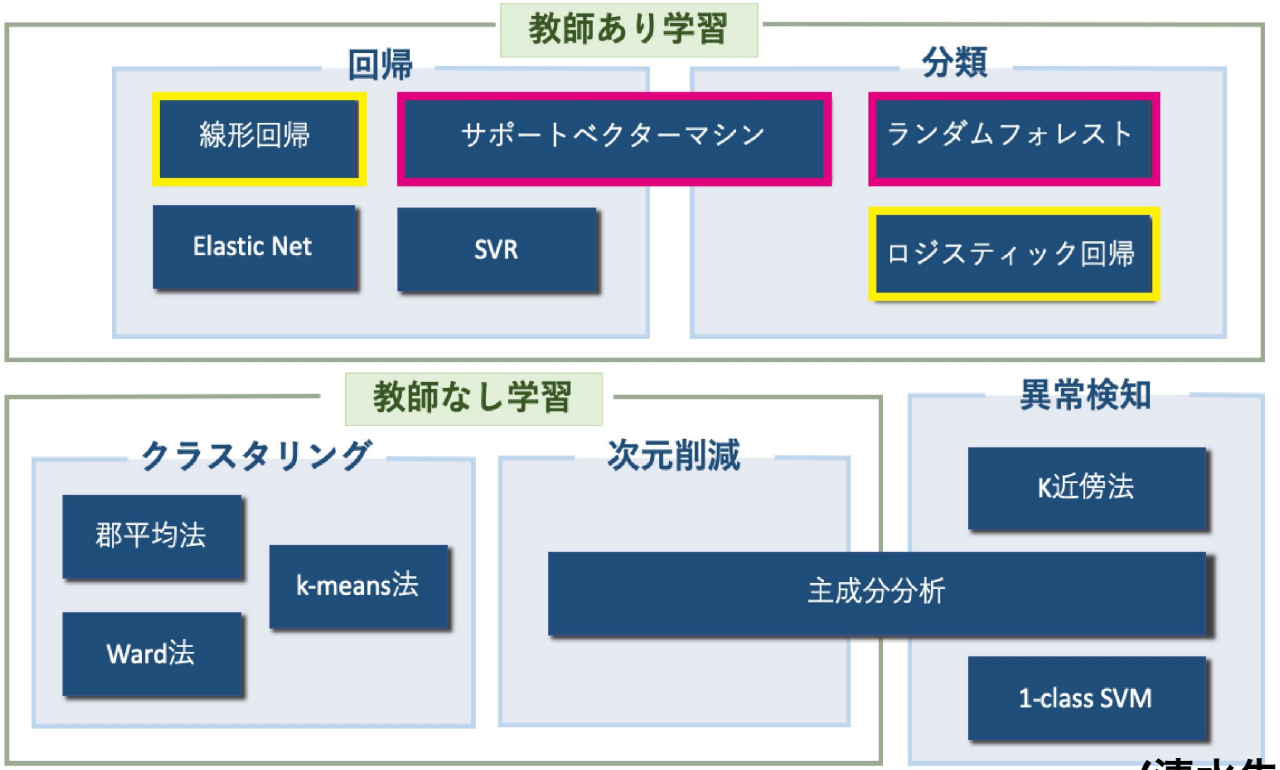
$$Y = f(X) + \epsilon$$

機械学習では，手元にある学習データを使って**真の** f(X) を推測することを目指す。

最もよくxでyを予測できるモデルを探し出す

(清水先生の講義資料より引用)

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、

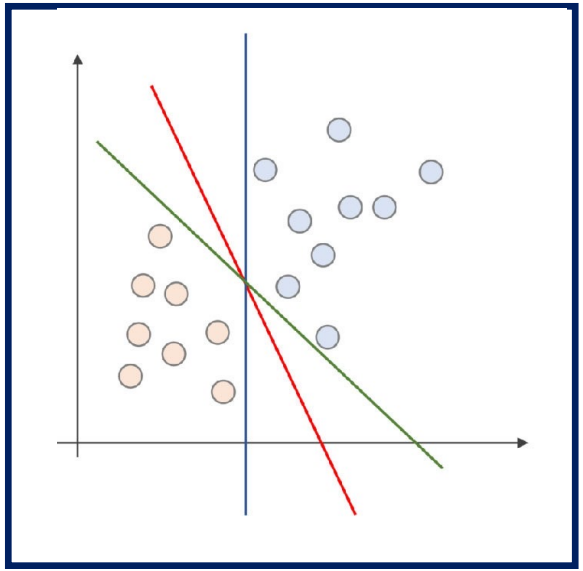
2010 年くらいまでの最強の手法、
サポートベクターマシンを紹介する。

(清水先生の講義資料より引用)

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

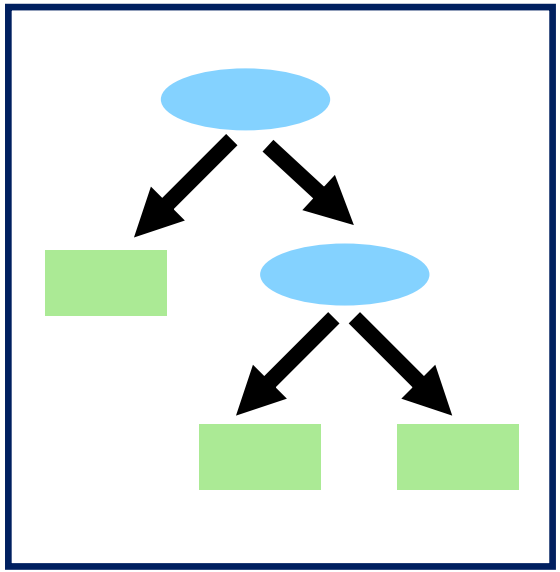
教師あり学習の種類

1.サポートベクターマシン



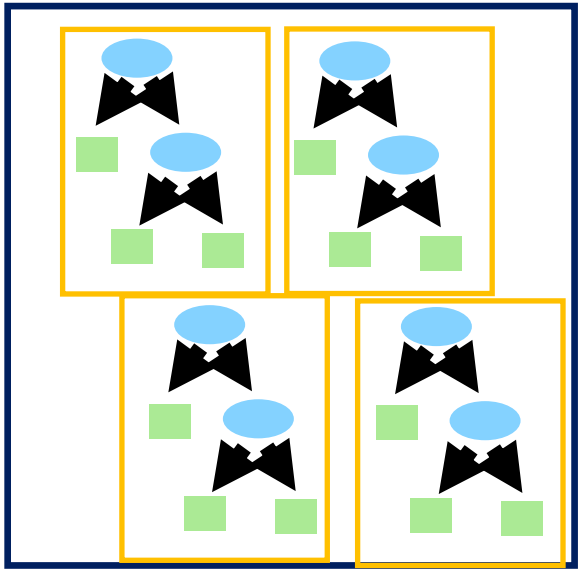
演習12

2.決定木



演習13

3. ランダムフォレスト



演習14

サポートベクターマシン

サポートベクターマシン (SVM, support vector machine)

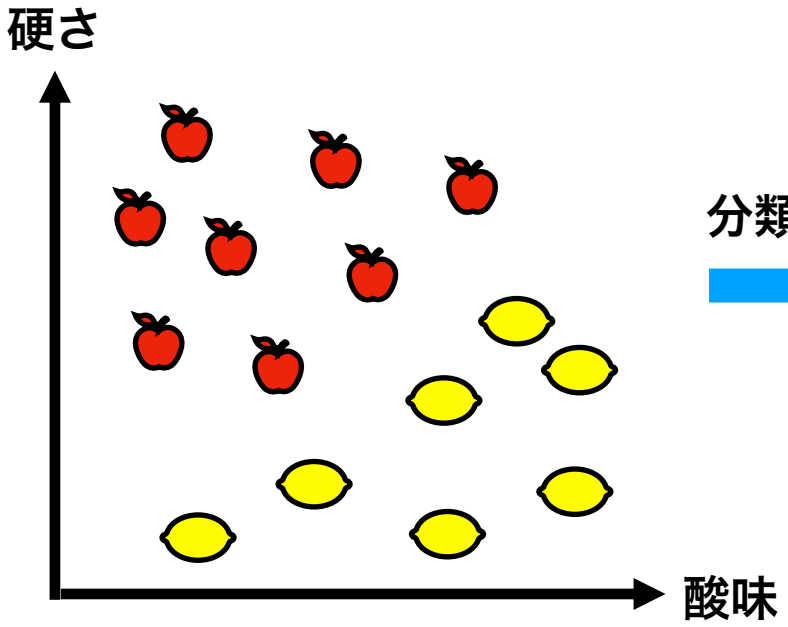
- 1990年代に開発された教師あり学習のアルゴリズム
- 分類も回帰も行えるが、主に分類(classification)に利用される
- 未学習データに対しても高い識別性能を得られる

サポートベクターマシン (SVM, support vector machine)

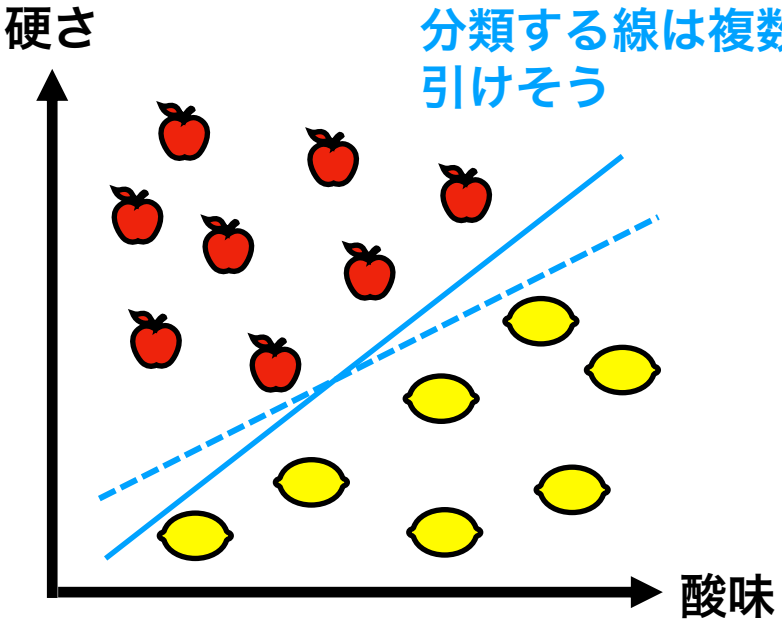
例：りんごとレモンを区別する

正解値：りんご/レモン

二つの特徴量 ：硬さと酸味



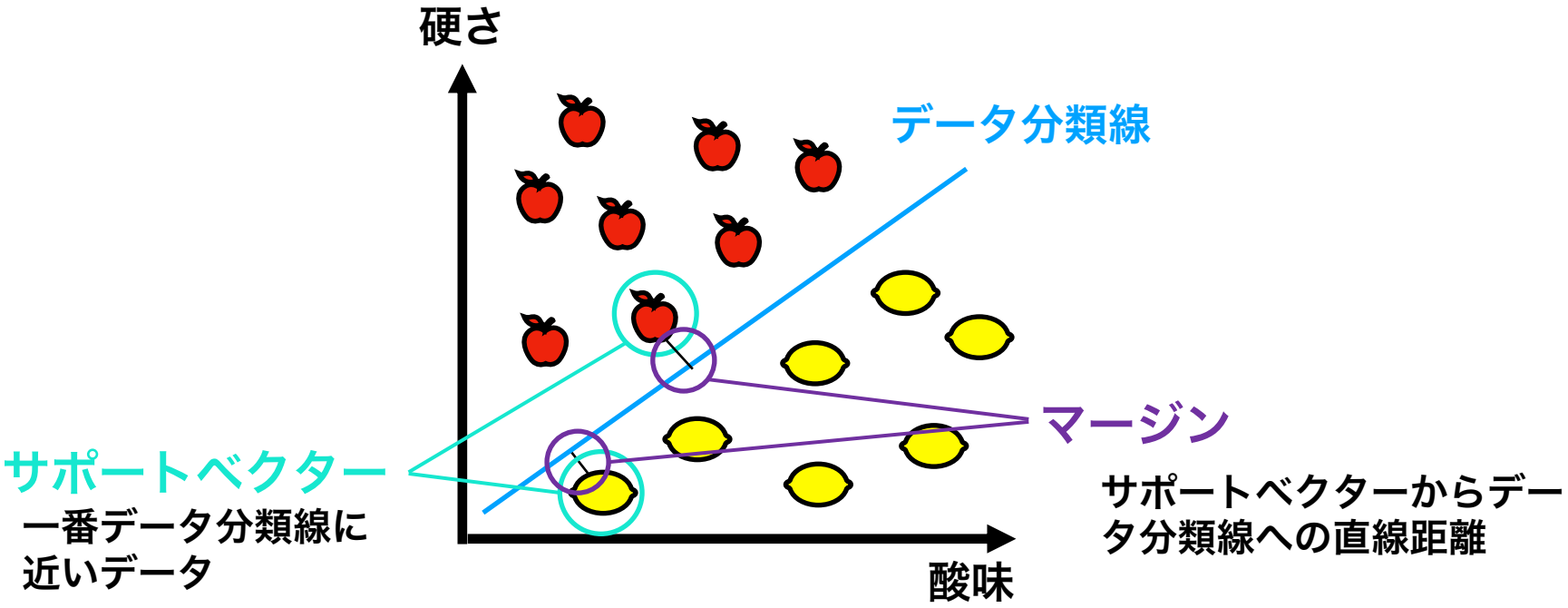
分類線



りんごとレモンを
分類する線は複数
引けそう

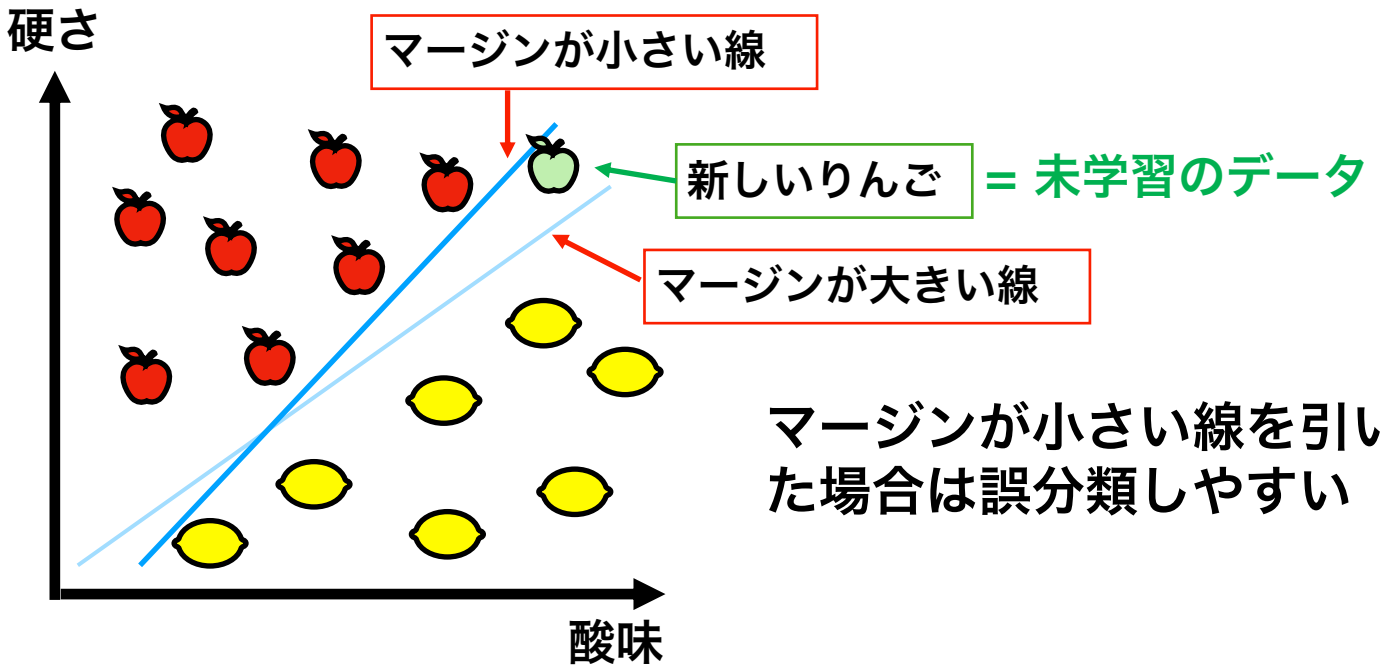
サポートベクターマシン (SVM, support vector machine)

- 分類線を引くアルゴリズムとしてサポートベクターを使う



<u>マージンが最大化したデータ分類線を引くのがSVMのアルゴリズム</u>

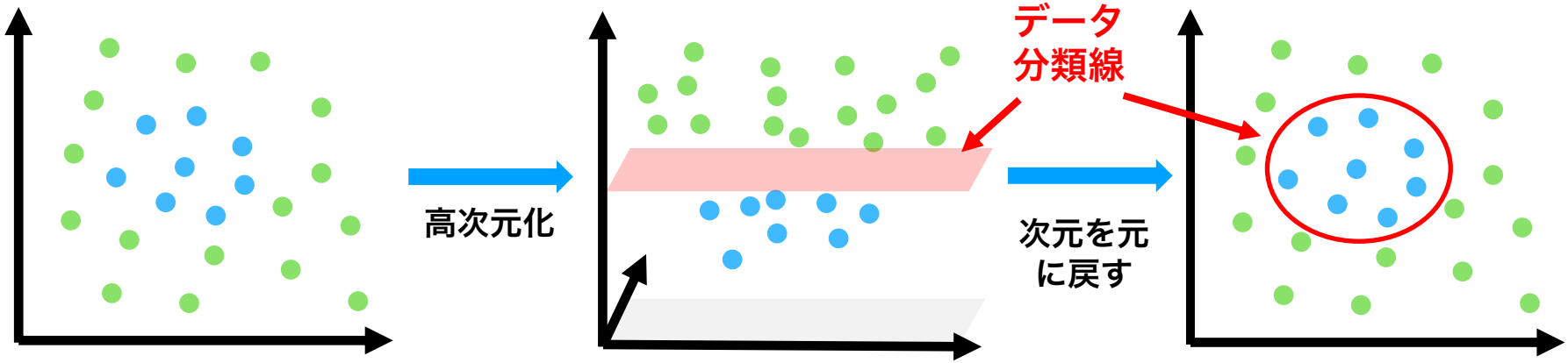
サポートベクターマシン (SVM, support vector machine)



マージンが小さい線を引いた場合は誤分類しやすい

線形で分けている場合、linear SVMと呼ぶ

線形では分けられない場合にもSVMは拡張できる



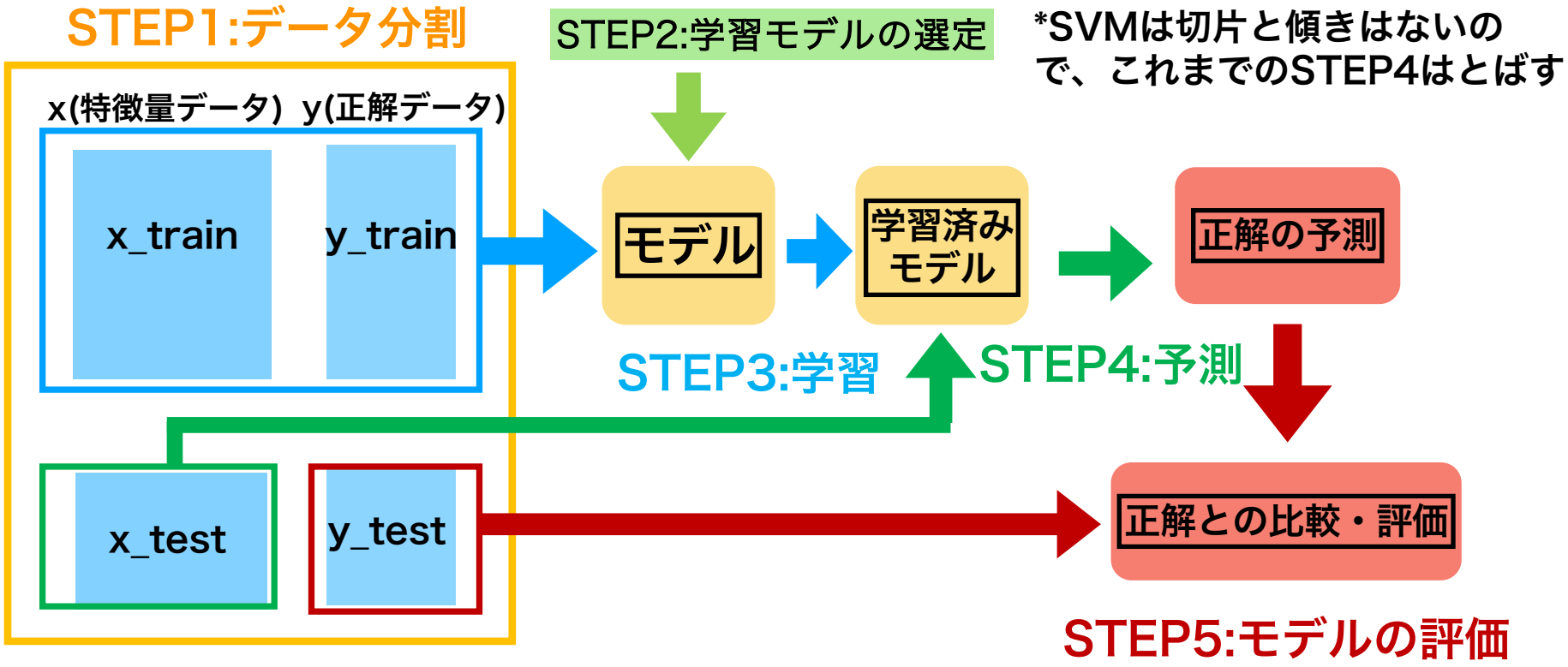
データを高次元の特徴空間に写像する方法：カーネル法
→非線形の場合に高い識別性能

演習

サポートベクターマシン

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



*SVMは切片と傾きはないので、これまでのSTEP4はとばす

SVM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SVM）

（モデル名）＝ SVC（）

回帰の時は
LinearRegression()だった

STEP3　データを入れて学習させる

（モデル名）.fit（特徴量，正解値）

回帰のSTEP4（切片と傾き)は
SVMでは存在しないので、すぐに予測を行う

STEP4　予測を行う：（モデル名）.predict（新たなxの値）

STEP5　モデルの評価：（モデル名）.score（特徴量，予測値）

1.サポートベクターマシン (SVM, support vector machine)

STEP0： ライブラリの読み込み

```
import numpy as np
import pandas as pd
```

STEP1：データの準備

```
from sklearn.datasets import load_breast_cancer
df_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(df_bc.data,
df_bc.target,test_size = 0.3, random_state = 0)
```

STEP2：学習モデルの選択

```
from sklearn.svm import SVC
model_svm = SVC(kernel = 'rbf', C = 1, random_state = 0)
```

STEP3：データを入れて学習させる

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

STEP4： 予測を行う

```
print(model_svm.predict(x_test))
print(np.array(y_test))
```

STEP5：モデルの評価

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

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

SVM

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させる
STEP4：予測を行う
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

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

演習12コード.ipynb

SVM

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：予測を行う
STEP5：モデルの評価

STEP0：ライブラリの読み込み

コード12-1 ライブラリを2つインポート

```
import numpy as np
import pandas as pd
```

- いつも通りnumpy、pandas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

SVM

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させる
STEP4：予測を行う
STEP5：モデルの評価

コード12-2 乳がんデータセットを読み込む

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

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

ライブラリ　モジュール　関数

● 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

colab

今回使うのは乳房腫瘤の細胞の画像のデータ

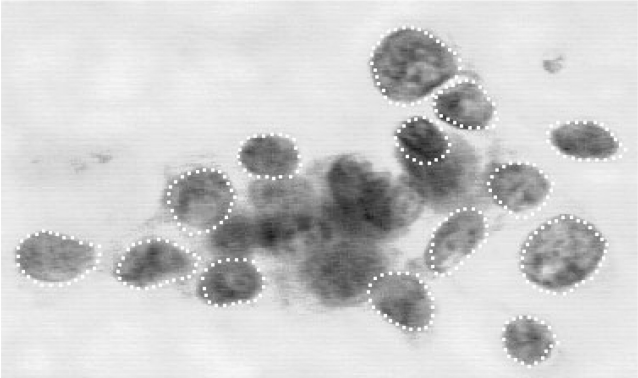


Figure 1: Initial Approximate Boundaries of Cell Nuclei
The user first draws a rough initial outline of some cell nucleus boundaries. Each outline serves as the initial position for a deformable spline which converges to an accurate boundary of the nucleus.

1993 W.N. Street, W.H. Wolberg and O.L. Mangasarian
Nuclear feature extraction for breast tumor diagnosis
IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology, volume 1905, pages 861-870, San Jose, CA, 1993.
Figure1 引用

569人分のデータ
target(正解値）は良性腫瘍(1)か悪性腫瘍(0)かの分類

特徴量は30個のデータがある

腫瘍細胞核のradius(半径)、texture(グレースケール値の標準偏差)、perimeter(周囲径)、area(面積)、smoothness(滑らかさ)、compactness(コンパクト度)、concavity(輪郭の凹部)、concave points(凹部の数)、symmetry(対称性)、fractal dimension(フラクタル次元の数値)

SVM

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：予測を行う
STEP5：モデルの評価

STEP1：データの用意

コード12-3 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(全員同じ分割セットにするため)
- 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
- ロジスティック回帰と異なり、特徴量は30個全て使うことができる

colab

SVM

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させる
STEP4：予測を行う
STEP5：モデルの評価

コード12-4 学習モデルにSVCを選択

```
from sklearn.svm import SVC
```

今回のモデルはSVMで分類を行うので、SVC(support vector classification)を選択する

クラスだけをインポート

```
from sklearn.svm import SVC
```

ライブラリ名　モジュール名　クラス名

SVM

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：予測を行う
STEP5：モデルの評価

STEP2：学習モデルの選択

コード12-4 学習モデルにSVCを選択

```
model_svm = SVC(kernel = 'rbf', C = 1)
```

SVCクラスからmodel_svmインスタンスを作成する
SVC()でモデルのハイパーパラメータを引数として指定する

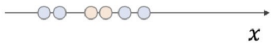
- 引数 `kernel =` では、アルゴリズムで使用されるカーネル関数を指定
  - `'rbf'`はガウスカーネル(Radial basis function)を指定し、複雑な決定境界線を引くことができるもので、今回指定する
  - 線形の場合は `kernel = 'linear'`

非線形分離を可能にするカーネルトリックの概要

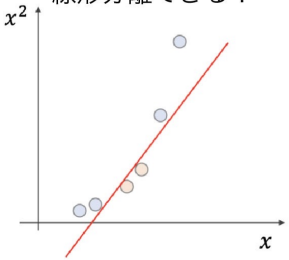
一見すると線形分離できなくても、カーネルトリック (kernel trick) と呼ばれる手法によって実現することができる。
特徴量を非線形変換し特徴量を高次元空間に変換することで線形モデルを使って予測するというアプローチをとる。

例えば、仮に 1 次元の特徴量 x に対しては線形分離できなくても，x^2 という新たな特徴量空間を作り，x と x^2 の二次元空間に変換することで線形分離することができる場合がある。
例えば、仮に 2 次元の特徴量 (x, y 座標の点) について線形分離できなくても、$z= - (x^2 + y^2)$ という新たな特徴量 z を計算すればこれをもとに線形分離することができる場合がある。

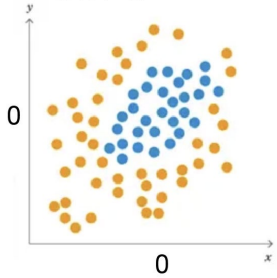
線形分離できない



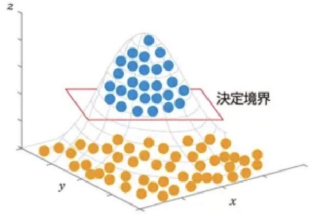
線形分離できる！



線形分離できない



線形分離できる！



このような変数変換 (写像という) をしてから SVM に必要な計算をするという 2 段構成になるが、最初の写像が複雑なものであればあるほど計算が大変になってしまう。そこで、この 2 ステップと同等のことができる 1 つの関数 K を考え、これをカーネルという。

カーネルをうまく使うことで、計算コストを抑えて (= ずっと早く) SVM ができ非線形な決定境界が見つかるというわけだ。

よく使われるカーネル関数には、多項式カーネル、ガウスカーネル (RBF カーネル)、シグモイドカーネルなどがある。
どのカーネルを使えばよいかは、機械学習モデルを作る人が自分で選ぶ必要がある。

SVM

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させる
STEP4：予測を行う
STEP5：モデルの評価

```
model_svm = SVC(kernel = 'rbf', C = 1)
```

- C = 1と指定して、大きくも小さくもない値に設定する

$$\min\left\{\frac{1}{M}+C\sum_{i=1}^{m}\xi_i\right\}$$

ただし M はマージン，ξ_i は i 番目のデータがマージンを侵した具合 (つまり残差) を表す変数

学習データの総数を m としている。ハイパーパラメーター C を (人が) 調整することで，どれだけ誤分類を許すかが決まる。
C が大きいとたくさん誤分類することができなくなり、逆に C が小さいと比較的誤分類を許容するようになる。

colab

ハイパーパラメーターの最適化

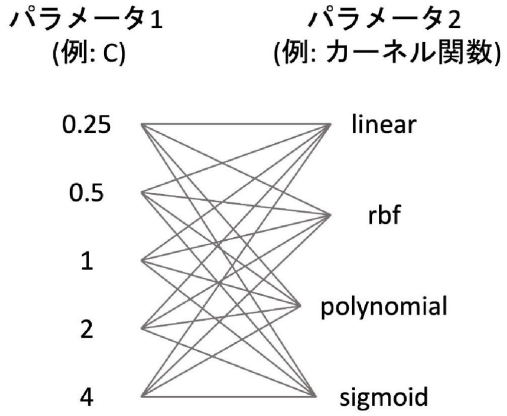
決定木の「深さ」や SVM の「C」「カーネル」のように，それぞれの機械学習アルゴリズムにはハイパーパラメータ (hyperparameter) がある。
モデルを作る人が値を指定するものをハイパーパラメータといいモデルが学習の過程で自動で最適化するパラメータとは区別されている。

高精度のモデルを構築するためには最適なハイパーパラメータを見つける必要があり、この作業をハイパーパラメータチューニングという。

シンプルなやり方としては 2 通りあり、

- Random Search:
  あらかじめ決められた範囲でランダムなハイパーパラメータを組み合わせて最も精度が高い組み合わせを探す

- Grid Search:
  あらかじめ決めた値を総当たりで組み合わせて最も精度が高い組み合わせを探す



例えば SVM の C パラメーターを 0.25, 0.5, 1, 2, 4 の 5 通りの中から、カーネル関数を 4 種類の設定の中から選びたいとする。
これを Grid search で探索する場合には、まずデータ全体を訓練 (training), 検証 (validation), テスト (test) データセットに分けておく。

そして C パラメーターで 5 通り x カーネル関数で 4 通り、合わせて 20 通りのハイパーパラメーターの組み合わせ全てについて SVM モデルを構築し訓練データで学習させ、20 個のモデルの性能を検証データを使って確認する。こうすることで 20 通りのどの組み合わせが最も良いかを知ることができる。

最後に、その最良のハイパーパラメーターの組み合わせで学習した SVM モデルを使ってテストデータでの性能を調べ、汎化性能がどの程度かを見る。前回勉強したように、テストデータセットは 1 回だけしか見ることができないことに注意。

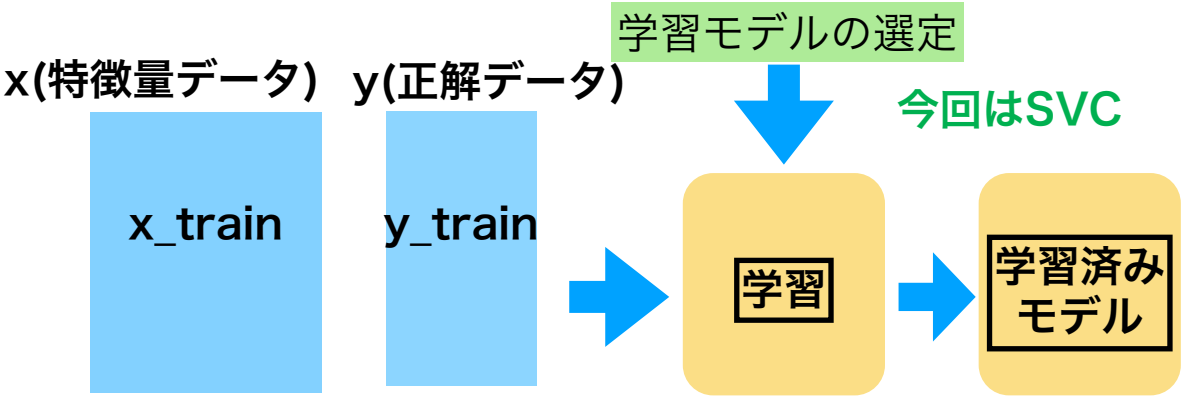
SVM

STEP3：データを入れて学習させる

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させる
STEP4：予測を行う
STEP5：モデルの評価

コード12-5 学習用データで学習させる

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



SVM

STEP4：予測を行う

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：予測を行う
STEP5：モデルの評価

コード12-6 検証用データで予測を行う

```
print(model_svm.predict(x_test))

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

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

x_testデータで
予測した分類の値

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

y_testの実際の値

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させる
STEP4：予測を行う
STEP5：モデルの評価

STEP4：予測を行う

コード12-7 混同行列の出力

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

```
array([[107,  1],
       [ 12, 51]])
```

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

どんな分類器も検査も完璧ではない

2 値分類において，0 を陰性 (**negative**), 1 を陽性 (**positive**) と呼ぶことがある。例えば病気の検査であれば，
病気であるケースを 1 とし陽性 positive, 病気でないケースを 0 とし陰性 negative とするのが一般的だ。
そしてどんな分類器も検査も完璧ではない。

これらの positive/negative のデータに対して予測した結果，正解していることを真 (**True**)，間違っていることを偽 (**False**) という。

本当は陽性であるデータに対して正しく陽性と予測したケースを真陽性 (True Positive: TP)
間違えて陰性と予測したケースを偽陰性 (False Negative: FN)

本当は陰性であるデータに対して正しく陰性と予測したケースを真陰性 (True Negative: TN)
間違えて陽性と予測したケースを偽陽性 (False Positive: FP)

これらの 4 通りが考えられ、表形式で表したものを
混同行列 confusion matrix という。

そして全体のうち、正しく予測できた TP, TN (右図赤字) の割合のことを、
その分類器なり検査法なりの正解率 accuracy という。

		予測	
		Positive	Negative
正解	Positive	TPの数	FNの数
	Negative	FPの数	TNの数

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させる
STEP4：予測を行う
STEP5：モデルの評価

STEP4：予測を行う

```
array([[107,  1],
       [ 12, 51]])
```

		予測結果	
		Positive(正)	Negative(負)
実際の分類結果	Positive (正)	真陽性True Positive **107**	偽陰性False Negative **1**
	Negative (負)	偽陽性False Positive **12**	真陰性True Negative **51**

colab

SVM

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させる
STEP4：予測を行う
STEP5：モデルの評価

コード12-8 検証用データで正解率を出力

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

0.9239766081871345

SVMの方が少し予測性能が高く出力された
（たまたまの可能性もあり）

ロジスティック回帰の正解率：0.9181286549707602

（モデル名）.scoreで分類（SVC）の時は正解率が出力される

colab

分類問題の性能指標（実臨床でも重要）

		予測	
		Positive	Negative
正解	Positive	TP	FN
	Negative	FP	TN

Precision（適合率）

$$\text{Precision} = \frac{TP}{TP + FP}$$

“陽性”と分類（あるいは予測）した中で実際に”陽性”だった割合。
「あなたは病気です！」って言った人のうち，どれだけの人が本当に病気だったかってこと。
陽性適中率ともいう。

Recall（再現率）

$$\text{Recall} = \frac{TP}{TP + FN}$$

実際には Positive のデータをどれだけ正しく Positive と判定できたか。つまり，「病気の人をどれだけ正しく病気と言えたか」。
特に医療系の分野では Sensitivity（感度）と呼ばれるが、同じ意味。医療では陽性の方を見逃したくないので、高い Recall を目指す。

Specificity（特異度）

$$\text{Specificity} = \frac{TN}{TN + FP}$$

実際には Negative のデータをどれだけ正しく Negative と判定できたか。つまり「病気でない人をどれだけ正しく病気でないと言えたか」。
Recall の Negative 版といえる。

Recall と Precision の調和平均 harmonic mean をとったものが、F1 値 (F 値) である。いろいろな指標の総合評価。

$$\text{F-score} = \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}$$

SVM

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させる
STEP4：予測を行う
STEP5：モデルの評価

コード12-9 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_svm.predict(x_test), output_dict = True))
```

	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

●関数をインポート

```
from sklearn.metrics import classification_report
```

ライブラリ名　モジュール名　関数名

SVM

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させる
STEP4：予測を行う
STEP5：モデルの評価

```
pd.DataFrame(classification_report(y_test,
  model_svm.predict(x_test), output_dict = True))
```

- classification_report(実際の分類,予測分類)でPrecision(適合率)、Recall（再現率)、F1値、accuracyを算出
- 引数output_dict = True　で辞書型で出力
- pd.DataFrame()で綺麗に出力される

SVM

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させる
STEP4：予測を行う
STEP5：モデルの評価

コード12-9 検証用データでPrecision(適合率)、Recall（再現率)、F1値を算出

悪性腫瘍　良性腫瘍

	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

SVM

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：予測を行う
STEP5：モデルの評価

STEP5：モデルの評価

良性腫瘍 precision：0.899160

recall：0.990741

f1_score：0.942731

		予測	
		Positive	Negative
正解	Positive	TP 107	FN 1
	Negative	FP 12	TN 51

Precision(適合率) = TP/(TP + FP)
= 107/119 = 0.899

Recall(再現率) = TP/(TP + FN)
= 107/108 = 0.990

F1値 =(2×Precision×Recall) /(Precision+Recall)
= 2× 0.899×0.990/ 0.899+ 0.990
= 0.942

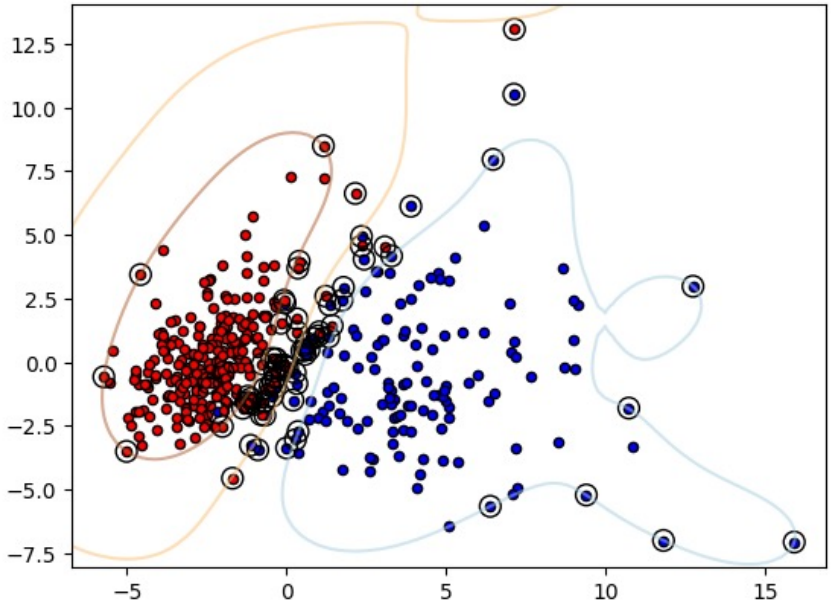
colab

SVM

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：予測を行う
STEP5：モデルの評価

STEP5：モデルの評価

特徴量データを二次元に次元削減することで図示することができる（xもyもどれかの特徴量ではない）



次元削減は教師なし機械学習で学ぶため、今回はPythonコードは割愛

青が悪性腫瘍、赤が良性腫瘍で非線形で分類している

正解率92.3%のため、かなりうまく分類できていることがわかる
一方、過剰に丸くなっている部分は、過学習の可能性がある

演習12：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

- SVC()モデルのハイパーパラメータの指定を(kernelをlinear、Cを100) に変更して、STEP2~STEP5まで実行してください。

以下の4つの性能指標について「良性腫瘍」のものを回答してください。
(1)Accuracy(正解率)
(2)Precision(適合率)
(3)Recall(再現率)
(4)F1値

演習12：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

- サポートベクターマシン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：サポートベクターとは何ですか？

1. 最もエラーが小さいデータ
2. 各分類クラスの中心に最も近いデータ
3. 分類境界に最も近いデータ
4. 最も頻繁に出現するデータ

(6) 問題2：SVMにおいてカーネルトリックは何を行いますか？

1. 次元削減をする
2. 過学習を防ぐ
3. 線形分離を可能にする
4. サポートベクターの数を増やす