

基礎科学チュートリアル

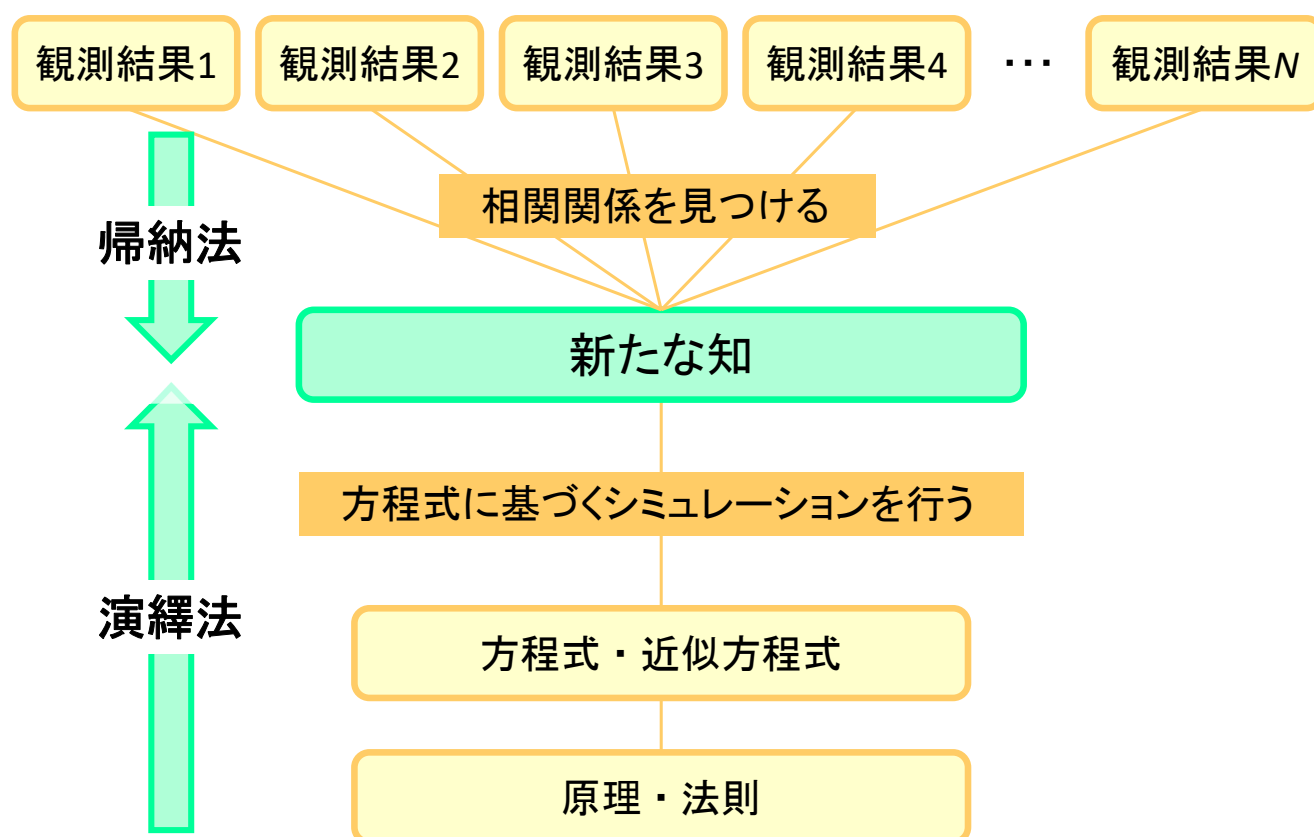
すぐできるマテリアルズ・インフォマティクス

～材料×機械学習の融合～

慶應義塾大学理工学部化学科
畑中美穂

研究スタイル

2



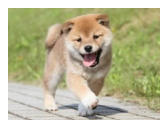
はじめの一步

犬・猫写真を判定するAI(機械学習モデル)を作る

x_1, x_2, x_3, \dots

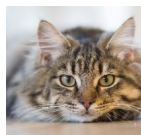
$$y = f(x_1, x_2, x_3, \dots)$$

y



特徴量

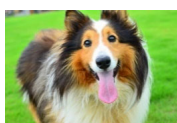
(1,0,2,0,..)



(1,3,-1,1,..)



(0,-1,2,1,..)



(-1,1,-1,1,..)

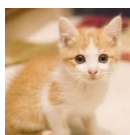
モデル f を
学習

犬 0

猫 1

猫 1

犬 0



(2,1,-1,0,..)

未知の写真を
作ったモデルに
代入

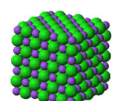
犬か猫かを
予測できる

はじめの一步

x_1, x_2, x_3, \dots

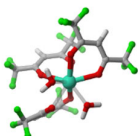
$$y = f(x_1, x_2, x_3, \dots)$$

実験値 y

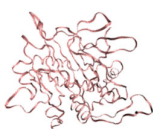


特徴量

(1,0,2,0,..)



(1,3,-1,1,..)



(0,-1,2,1,..)

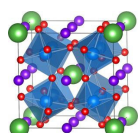
モデル f を
学習

高 90

低 10

中 45

未知の材料



(2,1,-1,0,..)

代入

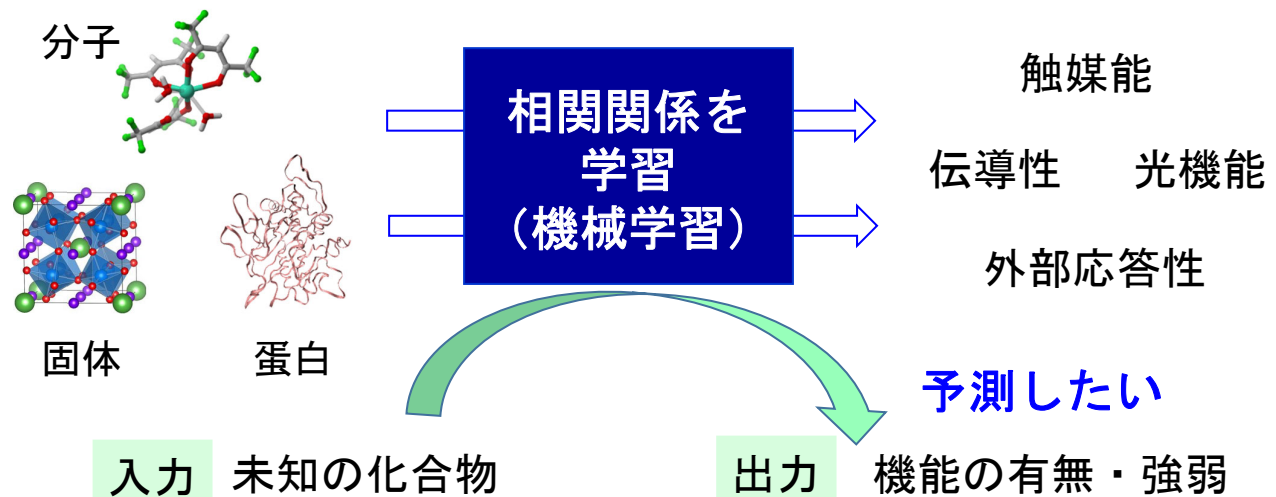
実験結果を
予測

物質の性質(実験結果)をコンピュータ内で予測できれば
望む性質を持つ物質をコンピュータ内で探索できる！

Materials Informaticsの基本概念

化合物(説明変数)

機能・物性(目的変数)



考えるべき
3項目

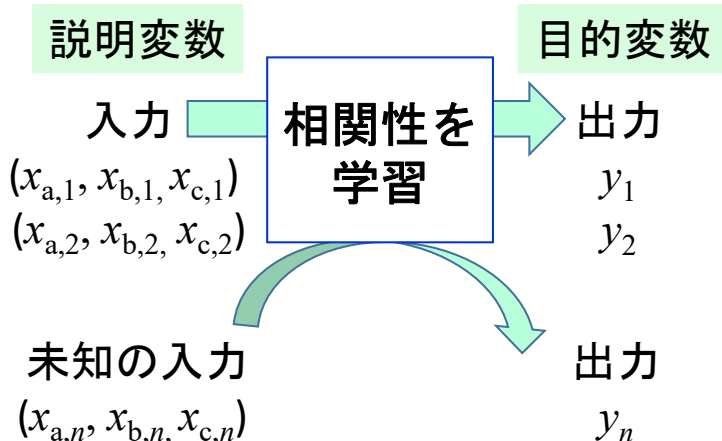
- Q1. 相関関係をどう学習するか？(機械学習)
- Q2. 化合物をどう表現するか？(記述子・特徴量)
- Q3. データをどう集めるか？(データベースの扱い)

機械学習

③ 強化学習

6

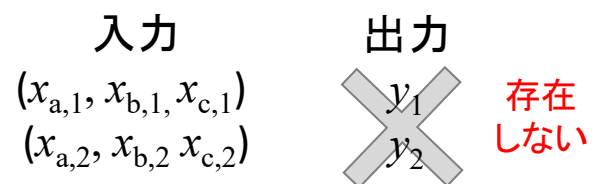
① 教師あり学習



- ・ 出力が連続な数値 : **回帰**
- ・ 出力が不連続 or カテゴリ : **分類**

(例) スпамメールの識別
犬・猫写真の識別

② 教師なし学習



- ・ 入力 x をたくさん与えると入力 x の性質に関して何かしらの結果を返す
- ・ **データの特徴を学習**
- ・ 制御は困難
- ・ 結果の意味の解釈も困難

(例) クラスタリング
異常検知

代表的な教師あり学習①

線形回帰

既知のデータセット

(x_i, y_i) ($i = 1, \dots, n$) に対して

y が x の一次関数で表現できる場合

($y = ax + b$ が成り立つ場合)

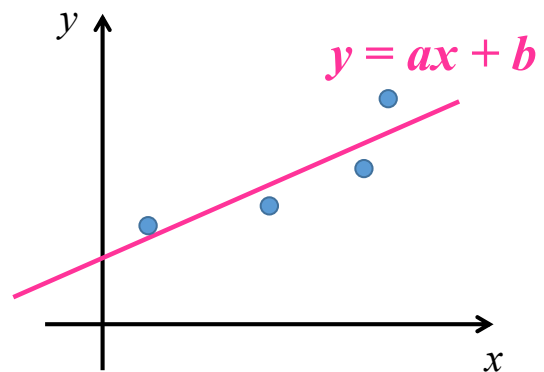
「線形関係にある」と言う

$y_i = ax_i + b + \delta_i$ (δ_i : ノイズ)

と表せる場合、

δ_i を最小化するような (a, b) を求めれば

未知の x に対する y の値を予測することができる



$y = ax + b$ の係数 (a, b) を求める・・・(線形)回帰分析

y が1種の説明変数 x で表現できる場合: 単回帰分析

2種以上

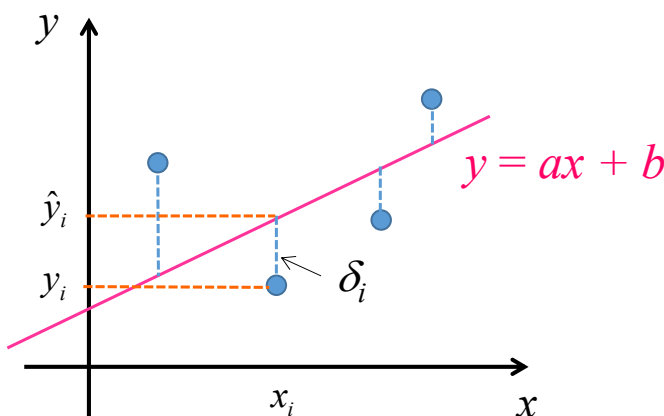
〃

重回帰分析

$$y = w_1x_1 + w_2x_2 + \dots + w_mx_m + b = \sum_{j=1}^m w_jx_j + b$$

回帰係数 (a, b) をどう決めるか?

観測値 (x_i, y_i) と予測値 (x, y) ができるだけ近くなるような (a, b) を決めたい



各データ (x_i, y_i) の
近似直線からのずれを
最小化したい

(x_i, y_i) における
近似直線からの差 (残差)

$$\delta_i = y_i - (ax_i + b)$$

$\nearrow \hat{y}_i$

$$s = \sum_{i=1}^n \delta_i^2 = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

残差の二乗の和 s が最小値を取るような (a, b) を決定する

最小二乗法

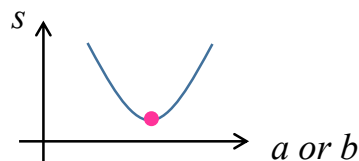
最小二乗法

s が最小値を取る時の (a, b) を決定するには...

☺ 数学の知識を思い出そう

(a, b) で s が最小である
 $\rightarrow (a, b)$ で s の微分が 0 になる

$$s = \sum_{i=1}^n \delta_i^2 = \sum_{i=1}^n (y_i - \hat{y}_i)^2$$



$$\frac{\partial s}{\partial a} = 0, \quad \frac{\partial s}{\partial b} = 0 \quad \text{となる } (a, b) \text{ を探せば良い！}$$

連立方程式を解いた結果

$$a = \frac{n \sum_{i=1}^n x_i y_i - \sum_{i=1}^n x_i \sum_{i=1}^n y_i}{n \sum_{i=1}^n x_i^2 - \left(\sum_{i=1}^n x_i \right)^2} \quad b = \frac{\sum_{i=1}^n x_i^2 \sum_{i=1}^n y_i - \sum_{i=1}^n x_i y_i \sum_{i=1}^n x_i}{n \sum_{i=1}^n x_i^2 - \left(\sum_{i=1}^n x_i \right)^2} \quad \dots \text{式(1)}$$

最小二乗法による線形回帰では、 (a, b) の値を式(1)を用いて計算

代表的な教師あり学習②

制約付き線形回帰

✓ Ridge回帰における制約条件

回帰係数の二乗の合計をなるべく小さくする

$$\sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^m w_j^2$$

回帰係数

利点：不自然に大きな係数がでないため
 データに『過剰適合』するのを防ぐ

『ハイパーパラメタ』
 ユーザーが決める
 決め方：p.16-17参

✓ Lasso回帰における制約条件

回帰係数の絶対値の合計をなるべく小さくする
 (L1正則化と呼ぶ)

$$\sum_{i=1}^n (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^m |w_j|$$

利点：0になる係数が出てくるので
 どの説明変数が重要か解釈しやすくなる

代表的な教師あり学習③

Support Vector Machine (SVM)

データを**非線形写像**で変数変換

単純に非線形性を取り入れようと思ったら... 計算量増！

$$y = p x_a + q x_b + r x_a^2 + s x_a^3 + t x_b^2 + u x_b^3 + v x_a x_b + w x_a^2 x_b + \dots$$

「カーネルトリック」を用いることで

変数変換をせずに回帰の結果の係数だけ求めることが可能

代表的カーネル（非線形写像の種類）

- ・ 多項式カーネル

説明変数の特定の次数までの全多項式 (x^2, x^3, x^4, x^5)

- ・ ガウシアンカーネル

(別名：radial basis function (RBF)カーネル) $\exp(-a\|x - x'\|^2)$

代表的な教師あり学習④

Neural Network (Deep Learning)

- ・ 非線形写像を多重化

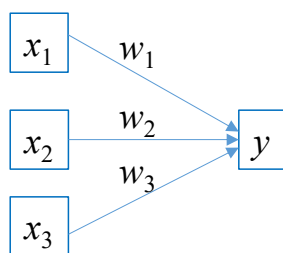
隠れ層を導入し、重み付き和の計算を繰り返し行う

線形回帰

$$y = w_1 x_1 + w_2 x_2 + w_3 x_3$$

入力

出力

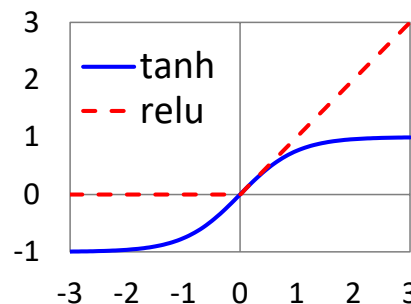
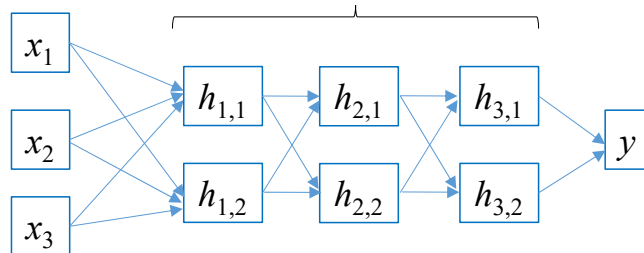


Neural Network

入力

隠れ層

出力



- ・ 出力=入力の重み付き和+**活性化関数**
 - ・ 活性化関数: tanh, reluなどの非線形関数

$$h_{1,1} = \tanh(w_1 x_1 + w_2 x_2 + w_3 x_3)$$

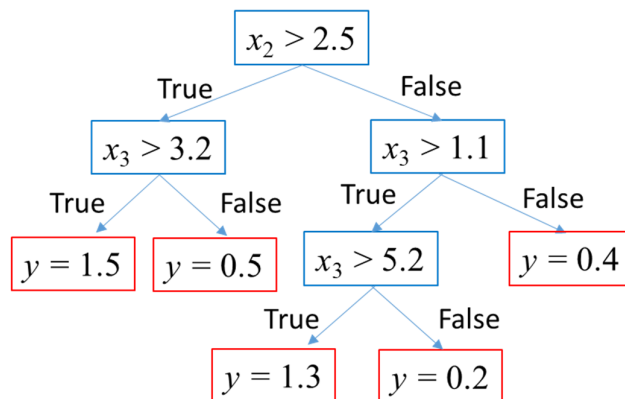
- ・ 層が多いほど複雑な学習が可能に>**深層学習**

代表的な教師あり学習⑤

決定木・Random Forest

✓ 決定木

- ・ Yes/Noで答えられる分岐で構成された階層的な木構造を構築
- ・ 分岐の数と閾値をさまざまに変えて入力データに最適化
- ・ 分類木、回帰木と呼び分ける場合も
- ・ 過学習に注意



✓ Random Forest

- ・ 複数の機械学習を組み合わせより強力なモデルを構築する（アンサンブル法）
- ・ N 個の決定木の多数決、または平均値を予測結果に用いる
- ・ 『外挿』は不得意

過剰適合（過学習）

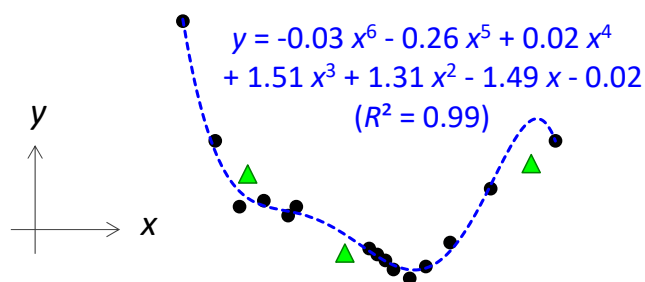
✓ 過剰適合 (overfitting, 過学習とも言う) :

持っている情報の量に比べ、過度に複雑なモデルを作ってしまうこと
未知のデータを用いた場合、かえって誤差が大きくなる

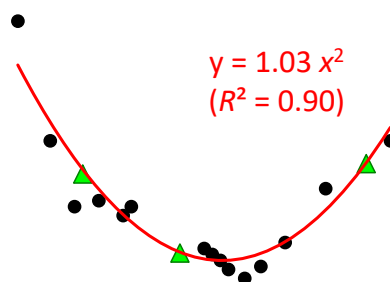
✓ 汎化 (generalization):

新たな入力データに対しても少ない誤差で予測できること

多項式関数の学習に用いたデータ(●)
用いなかったデータ(▲)



黒●との誤差小／緑▲とは誤差大
(汎化できていない)



黒●との誤差小／緑▲との誤差小
(汎化できている)

内挿・外挿

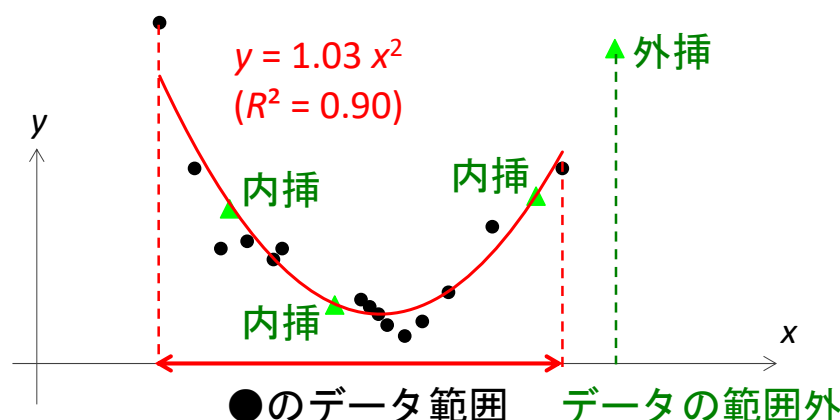
✓ 内挿

ある既知の数値データを基にして、
そのデータ列の各区間の範囲内を埋める数値を求めること

✓ 外挿

ある既知の数値データを基にして、
そのデータの範囲の外側で予想される数値を求めること

多項式関数の学習に用いたデータ(●) 用いなかったデータ(▲)



モデルの妥当性を検証する①

✓ ホールドアウト検証:

- ・ データを学習用 (Train data) とテスト用 (Test data) に分割
- ・ Train data で機械学習を実施 (モデルを作成)
- ・ Test data でモデルの妥当性を検証

Train	Test
-------	------

✓ K-分割交差検証 (K-fold cross-validation):

- ・ データをK個のグループに分割
- ・ K-1グループでモデルを作成、
残り1グループで検証
- ・ K回繰り返した平均によって妥当性を検証

K = 3の場合

1回目	Train	Train	Test
2回目	Train	Test	Train
3回目	Test	Train	Train

✓ リーブワンアウト法:

- ・ N個のデータがある場合、N-1個をTrain data、残り1個をTest dataに
- ・ Test dataを変えてN回繰り返した平均によって妥当性を検証
- ・ データ数が少ない場合に利用

モデルの妥当性を検証する②

評価関数

・ 回帰の場合

$$\text{二乗平均平方根誤差 (RMSE)} = \sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2}$$

$$\text{決定係数 (R}^2\text{)} = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y}_i)^2} \quad \left[\hat{y}_i: \text{予測値}, \bar{y}_i: \text{平均値} \right]$$

・ 二値分類の場合

		予測値	
		YES	NO
正解値	YES	True Positive (TP)	False Negative (FN)
	NO	False Positive (FP)	True Negative (TN)

$$\text{正解率 (Accuracy)} = (TP + TN) / (TP + FP + FN + TN)$$

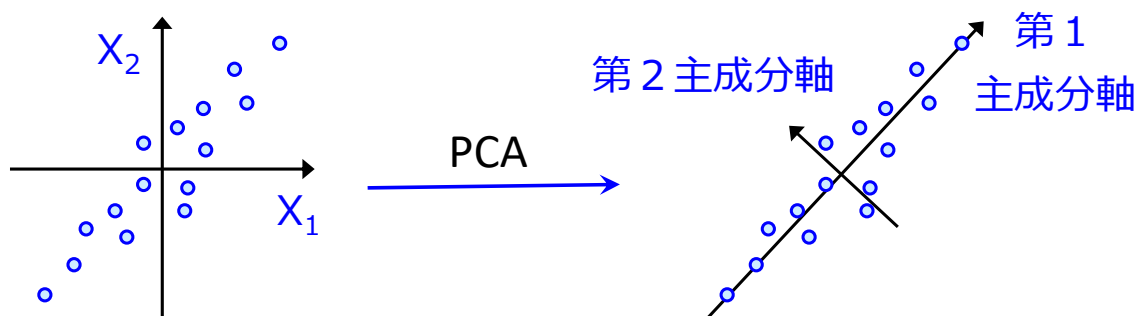
代表的な教師なし学習①

次元削減

✓ 主成分分析 (Principal Component Analysis, PCA)

データセットのもつ情報量をなるべく失わないように
多次元のデータセットを低次元化する方法

データの分散(ばらつき)が最大になる方向を第1主成分軸とする
第1主成分に直交&分散が最大になる方向を第2主成分軸に...



データのばらつきが大きいところに着目する

元のデータの次元より、少ない次元数の主成分で説明する
データの次元を削減できる

材料分野のデータ数は少ない…

スモールデータ解析

スモールデータの種類

- ・ case(1) 全体的にデータ数が少ない
 - 説明変数の次元削減
 - ・ 主成分分析(PCA)
 - ・ (手動)入力変数選択手法
 - 変数の組み合わせは膨大！ → 正則化/クラスタリング
 - モデルはなるべくシンプルにする
 - ・ 線形回帰モデル (部分的最小二乗(PLS)法など)
- ・ case(2) 特定のクラスのサンプルが稀少
 - 不均衡データ問題
- ・ case(3) 特定のクラスのサンプルが極端に少ない (ほぼない)
 - 異常検知問題

スモールデータ解析のコツ

case(1) 全体的にデータ数が少ない場合

- ① データをよく観察する/可視化する
 - 1つの外れ値がモデル全体に大きく影響してしまう
 - 相関係数の大きい説明変数の組がある場合、片方を削除する
- ② モデルはなるべくシンプルにする
 - 線形回帰 + α がおすすめ
 - ・ 次元削減 + 線形回帰： 部分的最小二乗(PLS)回帰
- ③ 特徴量を工夫する
 - ターゲットとする物性のメカニズムに関わる量を用いる

ここまで出てきた方法は 今日から皆さんも利用できます

必要な準備は2つだけ！

- ① データを集める（Excelにまとめる）
- ② Google Colaboratoryを利用する
（or Anacondaをインストールする）



プログラミング言語



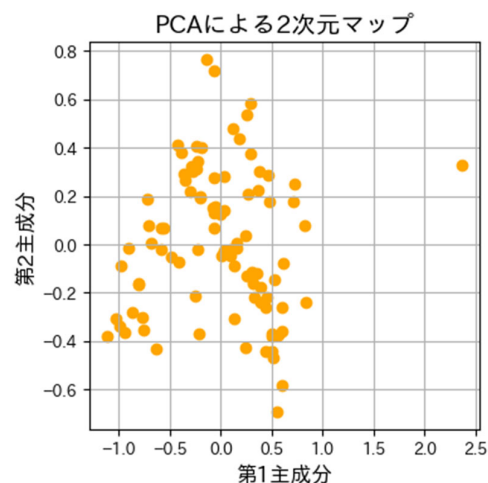
機械学習ライブラリ

演習用セットはこちら

https://github.com/hatanaka-lab/Getting_started_with_MI/tree/main

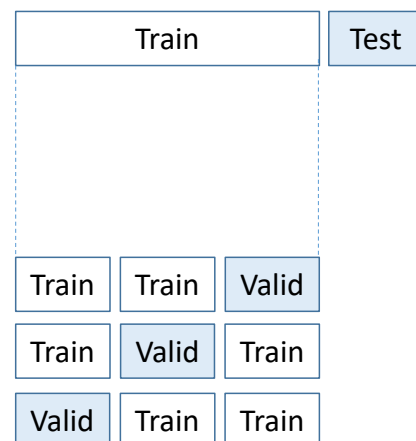
演習[1]の流れ

- (STEP 0 : ライブラリの読み込み)
- STEP 1 : データ読み込み
- STEP 2 : 不要データの削除
- STEP 3 : データの観察
- STEP 4 : 主成分分析(PCA)による次元圧縮
- STEP 5 : データの散布図描画



演習[2]の流れ

- (STEP 0 : ライブラリの読み込み)
- STEP 1 : データ読み込み
- STEP 2 : データを観察
- STEP 3 : ホールドアウト検証用に
データを2つ(訓練/テスト用)に分割
- STEP 4 : ハイパーパラメタを決定
(訓練データを用いてK分割交差検証)
- STEP 5 : 機械学習実行
ホールドアウト検証
- STEP 6 : 目的変数が未知のデータに対する予測



K個のモデルの
検証(Validation)データの
スコアの平均値を計算
↳ 最良のスコアを与える
ハイパーパラメタを
STEP 4で採用

はじめての プログラミング Python編



基礎科学チュートリアル2023用
畑中美穂

プログラミングを始める前に…

• プログラミングとは

手順の明文化

操作を適切な順番で行うよう命令する（＝プログラムを書く）

• 基本操作

① 四則演算（＋／－／×／÷）

② 変数への代入

③ 条件分岐

④ 繰り返し

• その他諸々必要な操作（Python版）

⑤ ライブラリ（関連する機能（関数・クラス）をまとめたもの）

⑥ データの塊の操作（配列・リスト・タプル etc）

⑦ ファイルの読み込み・書き出し

⑧ 関数(def)の定義（⑧' データ設計図(クラス)）

基本操作① 四則演算（+他の数学的な演算たち）

◆四則演算 「+」, 「-」, 「*」, 「/」

(例)	入力	1+1	1+1.0	2-3	2*3	4/2	4/3
	出力	2	2.0	-1	6	2.0	1.33333

※ 整数と小数の区別に注意

◆知っていると便利な演算（Python標準搭載）

累乗 x^y : `x**y` $x \div y$ の商: `x//y` $x \div y$ の余り: `x%y` 括弧()も使用可能

◆複雑な数学的演算（numpyが便利）

(例) `import numpy as np` 実行後

入力	<code>np.sqrt(4)</code>	<code>np.exp(2)</code>	<code>np.sin(3.14/2)</code>
出力	2.0	7.38905609893...	0.99999968293...
説明	$\sqrt{4}$	e^2	$\sin(3.14/2)$ ()内はradian単位

基本操作② 変数への代入 (+引数の表示)

◆代入 「=」 右辺を左辺に代入する (等号という意味ではない！)

(例) `a = 2` ☞この段階では `a = 2`, `b = 未定義`
`b = a * 3` ☞ `//` `a = 2`, `b = 6`
`a = a + 1` #上書き ☞ `//` `a = 3`, `b = 6`

※大文字・小文字は区別されることに注意

※変数には意味のある名前 (変数名) をつけること

(例) `apple = 120`
`orange = 300`
`price = 4 * apple + 2 * orange` ☞ `price`の中身は1080

◆引数 (変数の中身や計算結果) を表示する関数 「print関数」

使い方: `print(表示したい値)`

(例)

入力	<code>print(a)</code>	<code>print("a")</code>	<code>print(3+2)</code>	<code>print(a, b)</code>	<code>print("a =", a)</code>
出力	3	a	5	3 6	a = 3

基本操作③ 条件分岐

◆if文 条件を満たすか否かで異なる操作をする

使い方:

- (1) ifの後に条件を書き、その行をコロン: で終える
- (2) 条件に当てはまる場合に行う操作を、次の行に「タブ」1つ分あけて書く
タブで下がった部分が「その条件に合致する際に処理するところ」
- (3) 条件が複数ある場合は「elif 条件文:」または「else:」で繋げる

(例) `if t <= 0:`
`s = 0`
`else:`
`s = 1`

(例) `if t < 0 or t > 5:`
`s = 0`
`else:`
`s = 1`

(例) `if t < 0:`
`s = 0`
`elif t > 5:`
`s = 5`
`else:`
`s = t`

(例) `if (t > 0) or (t < 5):`
`s = 0`
`elif (t < 10):`
`s = 1`
`else:`
`s = -1`

条件の書き方

比較演算子

`a < b`
`a > b`
`a <= b`
`a >= b`
`a == b` 等しい
`a != b` 等しくない

論理演算子

条件1 `and` 条件2 且つ
 条件1 `or` 条件2 又は

基本操作④ 繰り返し

◆for文 指定した回数繰り返し処理を行う

使い方：(1) for 繰り返し変数 in 変数値の範囲 (or リスト)
 (2) 繰り返し行う操作を、次の行にタブ1つ分あけて書く
 タブで下がった部分が「繰り返す部分」

(例)

```
for i in range(5):  
    print(i)
```

(例)

```
for i in [1, 3, 5]:  
    for j in [2, 4]:  
        print(i, j)
```

変数値の範囲の書き方

range(a) 0からaの手前までの整数
 range(a, b, c) aからbの手前までのc刻みの整数
 [a, b, c, d] a, b, c, d

◆while文 条件を満たす間、繰り返し処理を行う

使い方：(1) while 条件： (2) 同上

(例)

```
a = 0  
while a < 3:  
    print(a)  
    a += 1
```

☞ a = a + 1と同意

基本操作⑤ ライブラリの利用

◆ライブラリ 関連する機能(関数, クラス)をまとめたもの

※ 関数, クラスをまとめたファイル(.py)：モジュール

モジュールのまとめ：パッケージ, パッケージのまとめ：ライブラリ

よく使うライブラリ

- ・ pandas データ解析 (読み込み, 切り出し, 並び替え, etc Excelと相性◎)
- ・ numpy 数学的演算・配列の取り扱い・ベクトル/行列/テンソルの計算
- ・ matplotlib グラフ描画 (seabornと組み合わせるとより美麗)
- ・ scikit-learn 機械学習
- ・ rdkit 原子・分子の取り扱い (ケモインフォマティクス)
- ・ scipy 科学・工学で用いられる数値解析

Google Colabにインストールされている(標準)ライブラリを使う場合

`import matplotlib`

Google Colabにインストールされていない(外部)ライブラリを使う場合

`!pip install japanize-matplotlib`

`import japanize-matplotlib`

基本操作⑥ データの塊の取り扱い pandas

◆pandasのデータ塊 (DataFrame) Excelに似た扱いが可 , `import pandas as pd`

(例) データをExcelファイルから読み込む場合

```
df = pd.read_excel("test.xlsx")
```

```
print(df)
```

☞ 中身確認

```
df.describe()
```

☞ 基礎的な統計量

```
df.mean()
```

☞ 特定の統計量だけ表示も可

```
df.sum(axis=n)
```

☞ n=0 : 各列の和を計算

n=1 : 各行の和を計算

```
df.iloc[0,1]
```

☞ 0行1列目 (Ando, Math) を抽出

```
df.iloc[:,1:]
```

☞ 全ての行, 1列目 (Math) 以降を抽出

```
df.loc[:, "Chem"]
```

☞ Chemの列のみ抽出, `df["Chem"]` でも同意

```
sum_exam = df.iloc[:,1:].sum(axis = 1)
```

☞ 全ての行に対して

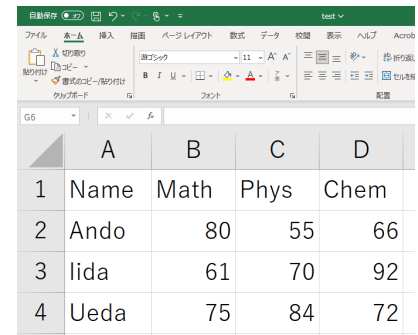
Math以降の和を計算

```
df["Total"] = sum_exam
```

☞ sum_examを列名:Totalとしてdfに追加

```
df=df.drop("Total", axis=1)
```

☞ dfから列Totalを削除



	A	B	C	D
1	Name	Math	Phys	Chem
2	Ando	80	55	66
3	Iida	61	70	92
4	Ueda	75	84	72

	Name	Math	Phys	Chem	Total
0	Ando	80	55	66	201
1	Iida	61	70	92	223
2	Ueda	75	84	72	231