解　説

Python で始めるケモメトリックス・機械学習

データサイエンスの分野でよく用いられているプログラミング言語の Python を，汎用のコンピュータで扱えるように，無料で簡便に環境構築する方法を紹介し，それを使って機器分析データをケモメトリックス・機械学習によって解析する手順を解説する。Python を用いた機械学習の教科書はたくさんあるので，それをどのように読めば分析化学に応用できるのか，そのコツを示したい。

森　田　成　昭

1　はじめに

分析化学はデータサイエンス[1]との親和性が高く，それによって正確さや精度[2]の向上が期待できる分野である。むしろ，時代に先駆けて計算機によるデータ解析の可能性を模索してきた分野ともいえる[3]。今後，逆解析やケモインフォマティクス/マテリアルズインフォマティクス[4]といったデータ駆動型の計算を駆使して，新たな機能性化合物を予測できるのは，膨大な実験データを系統的に蓄積している分析化学者なのかもしれない。しかし国内では，産・官・学のいずれにおいても，分析化学の専門家がデータサイエンスのエキスパートであることは稀であるし，そのスキルを身につけようとしている人材が少ないのは残念である。筆者が専門とする分光分析の分野では，国際会議において，多くのデータ解析に関する発表が活発に行われており，日本はこの分野で後れを取っているように感じる[5][6]。

そこで本稿では，分析化学に携わる読者に，データサイエンスの分野でよく使われているプログラミング言語の Python[7]~[9]を用いて，ケモメトリックス[10][11]・機械学習[12]~[14]を始めるきっかけを与えたい。この二つを併記する理由は後述する。もちろん Python にこだわる必要はなく，R や Matlab を使いこなしているのであればそれでよい。人生で初めて機器分析をしたときのように，最初は心細く，多くの失敗をするかもしれないが，経験を積んで，誰よりもエレガントに結果を導き出せるようになってほしい。そういった鍛錬や試行錯誤を積み重ねて目標を達成するプロセスは分析化学者が得意とするところのように思える。

Python は汎用のプログラミング言語であり，Raspberry Pi[15]のようなシングルボードコンピュータによる制御にも，GPU を搭載したワークステーションによるビッグデータ解析にも使われている。Microsoft Windows や Mac OS といった手頃な環境で Python によるプログラミングをするには Anaconda[16]のような無料の統合環境をダウンロードしてインストールすればよい。Anaconda には様々なツールが含まれており，Jupyter Notebook[17][18]を選んで起動すると，すぐにブラウザ上で Python によるプログラミングを始めることができる。Anaconda には後述する多くのライブラリが梱包されており，それらは自動的にインストールされるため，煩雑な環境構築をする必要がない。筆者が勤める大学では演習室のコンピュータに Anaconda がインストールされており，Jupyter Notebook を用いて Python のプログラミング教育を行ったり，ハンズオン講習会を開催したりしている。

2　データファイルの準備

機器分析を行うと，スペクトルやクロマトグラムといった波形データが得られ，現在ではデジタルデータとして保存される。本稿ではこのようなデータをひとまずスペクトルと呼ぶが，横軸はエネルギーである必要はなく，保持時間でも温度でも電位でもかまわない。以降，得意な分析手法を想定して，スペクトルをクロマトグラムやサイクリックボルタモグラムに置き換えて読んでほしい。ここで，スペクトルにおける横軸をスペクトル変数 x，縦軸を信号強度 y と呼ぼう。可視吸収スペクトルの場合，スペクトル変数 x は波長，信号強度 y は吸光度である。濃度や時間を変えて複数のスペクトルを測定することを考える。このとき，それぞれのスペクトルをサンプル変数 t で区別しよう。例えば，測定波長範囲を 300～700 nm，試料濃度を 10～100 % としたときの可視吸収スペクトルをイメージして，図 1 に示すモデルデータを準備した。このデータはガウス関数

$$y(t,x) = y_0 t \exp\left(-4\ln 2\,\frac{(x-x_0)^2}{w^2}\right) \quad \cdots\cdots(1)$$

において，中心波長 x_0＝500，吸光係数 y_0＝0.01，半値

Introduction to Chemometrics and Machine Learning with Python.

	10.0	20.0	30.0	40.0	50.0	60.0	70.0	80.0	90.0	100.0
300.0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
310.0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
320.0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
330.0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
340.0	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.001	0.001
350.0	0.000	0.000	0.001	0.001	0.001	0.001	0.001	0.002	0.002	0.002
360.0	0.000	0.001	0.001	0.002	0.002	0.003	0.003	0.003	0.004	0.004
370.0	0.001	0.002	0.003	0.004	0.005	0.006	0.006	0.007	0.008	0.009
380.0	0.002	0.004	0.006	0.007	0.009	0.011	0.013	0.015	0.017	0.018
390.0	0.003	0.007	0.010	0.014	0.017	0.021	0.024	0.028	0.031	0.035
400.0	0.006	0.013	0.019	0.025	0.031	0.038	0.044	0.050	0.056	0.063
410.0	0.011	0.021	0.032	0.042	0.053	0.064	0.074	0.085	0.095	0.106
420.0	0.017	0.034	0.051	0.068	0.085	0.102	0.119	0.136	0.153	0.170
430.0	0.026	0.051	0.077	0.103	0.129	0.154	0.180	0.206	0.231	0.257
440.0	0.037	0.074	0.111	0.147	0.184	0.221	0.258	0.295	0.332	0.369
450.0	0.050	0.100	0.150	0.200	0.250	0.300	0.350	0.400	0.450	0.500
460.0	0.064	0.128	0.193	0.257	0.321	0.385	0.449	0.513	0.578	0.642
470.0	0.078	0.156	0.234	0.312	0.390	0.467	0.545	0.623	0.701	0.779
480.0	0.090	0.179	0.269	0.358	0.448	0.537	0.627	0.716	0.806	0.895
490.0	0.097	0.195	0.292	0.389	0.486	0.584	0.681	0.778	0.875	0.973
500.0	0.100	0.200	0.300	0.400	0.500	0.600	0.700	0.800	0.900	1.000
510.0	0.097	0.195	0.292	0.389	0.486	0.584	0.681	0.778	0.875	0.973
520.0	0.090	0.179	0.269	0.358	0.448	0.537	0.627	0.716	0.806	0.895
530.0	0.078	0.156	0.234	0.312	0.390	0.467	0.545	0.623	0.701	0.779
540.0	0.064	0.128	0.193	0.257	0.321	0.385	0.449	0.513	0.578	0.642
550.0	0.050	0.100	0.150	0.200	0.250	0.300	0.350	0.400	0.450	0.500
560.0	0.037	0.074	0.111	0.147	0.184	0.221	0.258	0.295	0.332	0.369
570.0	0.026	0.051	0.077	0.103	0.129	0.154	0.180	0.206	0.231	0.257
580.0	0.017	0.034	0.051	0.068	0.085	0.102	0.119	0.136	0.153	0.170
590.0	0.011	0.021	0.032	0.042	0.053	0.064	0.074	0.085	0.095	0.106
600.0	0.006	0.013	0.019	0.025	0.031	0.038	0.044	0.050	0.056	0.063
610.0	0.003	0.007	0.010	0.014	0.017	0.021	0.024	0.028	0.031	0.035
620.0	0.002	0.004	0.006	0.007	0.009	0.011	0.013	0.015	0.017	0.018
630.0	0.001	0.002	0.003	0.004	0.005	0.006	0.006	0.007	0.008	0.009
640.0	0.000	0.001	0.001	0.002	0.002	0.003	0.003	0.003	0.004	0.004
650.0	0.000	0.000	0.001	0.001	0.001	0.001	0.001	0.002	0.002	0.002
660.0	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.001	0.001
670.0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
680.0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
690.0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
700.0	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

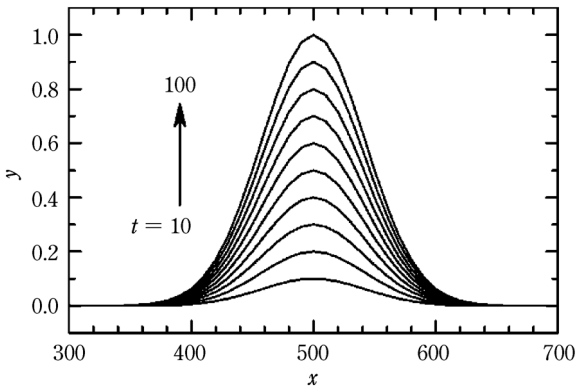

図 1　解析のモデルデータ（data.csv）。（上）csv ファイルにおけるデータ構造と（下）そのプロット。第 1 列をスペクトル変数 x，第 1 行をサンプル変数 t とし，それらの間に挟まれた行列を信号強度 $y(t, x)$ とする。左上の 1 行 1 列セルは空白としておく。

幅 w＝100 とし，サンプル変数 t を 10～100 まで 10 ごとに増加させて 10 本のスペクトルを生成させたものである。

このようなデータセットを図 1 に示す csv ファイルとして data.csv のファイル名で保存したとしよう。このとき，第 1 列はスペクトル変数（波長），第 1 行はサンプル変数（濃度）であり，左上の 1 行 1 列セルは空白にしておく。スペクトル変数とサンプル変数に挟まれた 41 行 10 列の巨大な信号強度（吸光度）のかたまりを「縦長」の行列として考えよう。

筆者の研究室では，例えば赤外スペクトルを 4000～400 cm^{-1} の範囲で 1 cm^{-1} ごとに，温度を 20～120 ℃の範囲で 1 ℃ ごとに，自動的に測定するようなことがある。これによって 101 本のスペクトルが得られるが，図 1 のフォーマットでファイルに保存すると，吸光度データは 3601 行 101 列の「縦長」の行列となる。このように，サンプル変数，即ち測定試料や測定条件，を変えて測定した機器分析データは，「縦長」のフォーマットでファイルに保存されることが多い。しかし残念なことに，後述するケモメトリックスや機械学習では，サンプル変数（目的変数）を縦方向に，スペクトル変数（説明変数）を横方向にとって，このようなデータを「横長」の行列として扱うことになる。そこで本稿でも，データファイルは図 1 のように「縦長」のフォーマットで読み書きするが，実際に計算をするときは，データを転置して「横長」の行列に変換してから行うこととする。

3　Python を用いたデータ解析の例

ここでは，ケモメトリックスでも機械学習でもよく扱われる主成分分析（principal component analysis, PCA）を例に，Python によるプログラミングを紹介する。図 2 に，図 1 で示した data.csv を用いて主成分分析を実行するための Python プログラム pca.ipynb を示す。Jupyter Notebook でプログラムを作成して保存すると拡張子が ipynb となるが，これは Jupyter Notebook 固有の情報も含まれるためであり，他のシステムで実行する場合には一般的な Python 形式（拡張子 py）で保存する。もちろん data.csv はモデルとして示したガウス関数である必要はなく，機器分析データをこのフォーマットで保存したものを用いてもよい。説明の便宜上，図 2 の左側に 001～017 の行番号を記したが，実際のプログラミングではこの行番号を付ける必要はない。

3･1　ライブラリの読み込み

001～002 行ではライブラリの読み込みを行っている。ライブラリは import 文で読み込む。このプログラムでは，データ構造を扱うためのライブラリである pandas と，機械学習ライブラリの scikit-learn（sklearn）を使う。pandas は，一次元配列（Series），二次元配列（DataFrame），三次元配列（Panel）を扱うことができるが，このプログラムでは図 1 に示したような二次元配列のみを扱う。sklearn は多くの機械学習アルゴリズムを備えているが，ここでは行列分解を行う decomposition モジュールにある PCA クラスのみを使うので，002 行のように書く。

```
001 import pandas
002 from sklearn.decomposition import PCA
003
004 filename='data.csv'
005 spec=pandas.read_csv(filename,header=0,index_col=0).T
006 spec.T.plot(legend=None)
007
008 pca=PCA(n_components=3).fit(spec)
009 print(pca.explained_variance_ratio_)
010
011 loading=pandas.DataFrame(pca.components_,columns=spec.columns)
012 loading.T.plot()
013 loading.T.to_csv(filename[:-4]+'_loading.csv')
014
015 score=pandas.DataFrame(pca.transform(spec),index=spec.index)
016 score.plot()
017 score.to_csv(filename[:-4]+'_score.csv')
```

図 2　主成分分析を実行するための **Python** プログラム（**pca.ipynb**）。図 1 に示したデータファイル data.csv がこれと同じフォルダにある必要がある。

3･2　データファイルの読み込み

004～006 行ではデータファイルの読み込みを行っている。004 行はファイル名 data.csv を文字列型の変数 filename に代入する命令である。ここでは作業フォルダを指定していないので，data.csv は pca.ipynb と同じフォルダにある必要がある。005 行では pandas の read_csv 関数を用いて data.csv を読み込んでいる。read_csv 関数で読み込んだ情報は DataFrame オブジェクトとなる。read_csv 関数の引数には，ファイル名の他に，ヘッダーを読み込む行とインデックスを読み込む列を指定している。Python では要素のインデックスを 0 から始めることになっているので，1 列目を読み込むときのインデックスは 0 である。先述のように，データファイルは「縦長」のフォーマットであるが，実際に計算に用いるデータは「横長」の構造に転置しなければならない。データを転置するためのメソッドは T であり，read_csv で読み込んだ DataFrame オブジェクトに `.T` とするだけで，転置してデータを読み込むことができる。コンソールへの出力を行う print 関数を用いて 005 行の後に `print(spec)` を加えると，DataFrame オブジェクトである spec が横長の構造となっていることを確認できる。サンプル変数 t は `spec.index` で，スペクトル変数 x は `spec.columns` で，信号強度 y は `spec.values` で，それぞれ取り出すことができ，`print(spec.index)` のようにprint関数を用いることで中身を確認することができる。$y(t,x)$ において，i 番目のサンプル変数 t_i を取り出すには `spec.index[i]`，j 番目のスペクトル変数 x_j を取り出すには，`spec.columns[j]`，サンプル変数 t_i，スペクトル変数 x_j における 1 点の信号強度 $y(t_i, x_j)$ を取り出すには DataFrame オブジェクトの iloc プロパティを使って `spec.iloc[i,j]` とする。サンプル変数 t_i における信号強度変化（スペクトル）を取り出すには `spec.iloc[i,:]`，スペクトル変数 x_j における信号強度変化を取り出すには `spec.iloc[:,j]` とすればよい。1 本のスペクトルを取り出すときに `spec.iloc[i,:]` を `spec.iloc[i]` と省略してもよい。サンプル変数 t_i におけるスペクトルを $x_j \leq x < x_k$ の範囲で取り出すときは `spec.iloc[i,j:k]` とする。

pandas には DataFrame オブジェクトを簡易にプロットするための plot メソッドがあり，006 行のように書くことでデータを可視化できる。ここで注意が必要なのは，DataFrame.plot によってプロットをするときの横軸は DataFrame.index の値が用いられるので，DataFrame オブジェクトである spec をプロットするときには，転置によりスペクトル変数が DataFrame.index になるようにしなければならない。006 行で `legend＝None` はプロットの凡例を省略する命令で，大量のスペクトルをプロットするときには入れておくとよい。サンプル変数 t_i における 1 本のスペクトルだけをプロットしたいときは `spec.iloc[i].plot()` とする。

3･3　主成分分析の計算

008～009 行では主成分分析を行っている。008 行で PCA インスタンスを生成するときに `n_components＝3` としている。これは主成分分析における成分数の指定であり，スペクトル変数によって張られた空間（ここでは 300～700 の 401 次元）をこの場合は 3 次元に次元削減することを決めている。続く `.fit(spec)` で DataFrame オブジェクトである spec を用いて主成分分析を行い，結果を PCA インスタンスである pca に格納している。PCA インスタンスである pca には様々な情報が格納されており，対応するメソッドでそれらを呼び出すことができる。例えば各成分の寄与率を呼び出すには 009 行のように explained_variance_ratio_ メソッドを使えばよい。ここでは 008 行で `n_components＝3` としているので，第 1～3 主成分の寄与率がそれぞれ大きい順に表示される。この寄与率の変化は主成分分析を行う際の成分数を決める目安となる。機器分析を行うと，どうしてもデータにノイズが含まれてしまうので，寄与率が小さい主成分はノイズを説明していると考えられる。そこで，適切な寄与率となる範囲で成分数を打ち切って，あらためて主成分分析を行うとよい。

3･4　ローディング

011～013 行では各主成分におけるローディングを扱っている。011 行では 008 行で計算した PCA インスタンスである pca から components_ 属性を用いてローディングの情報を取り出している。011 行全体では，loading という名前の DataFrame オブジェクトを準備し，そこに pca.components_ の値を代入して，loading.columns の値を spec.columns と同じとなるように指定している。012 行はローディングの可視化であり，可視化結果に表示される凡例の 0～2 はそれぞれ第 1～3 主成分に対応している。第 1 主成分のローディングの値は `loading.iloc[0]`，第 i 主成分のローディングの値は `loading.iloc[i−1]` で扱うことができる。例えば，第 2 主成分のローディングだけを可視化したい場合は `loading.iloc[1].T.plot()` とすればよい。013 行は DataFrame オブジェクトである loading を csv ファイル形式で保存するための命令である。DataFrame オブジェクトを csv ファイル形式で保存するには to_csv 関数を用いる。この関数の引数にはファイル名を文字列型で指定する。`filename[:−4]` は `filename[0:−4]` の省略であり，004 行で指定した文字列変数 filename を，0 文字目（最初）から，−4 文字目（最後から 4 文字目）の直前である最後から 5 文字目まで抜き出して新たな文字列としている。これにより文字列 'data.csv' から，拡張子より前の 'data' だけを抜き出している。013 行ではこれと文字列 '_loading.csv' を結合させて，保存するファイル名を 'data_loading.csv' とした。

3･5　スコア

015～017 行では各主成分におけるスコアを扱っている。015 行では transform メソッドを用いて，DataFrame オブジェクトである spec の値（元の空間における座標）を，PCA インスタンスである pca で次元削減した新たな空間における座標（スコア）に変換している。ローディングと同様に，第 i 主成分のスコアの値は `score.iloc[:,i−1]` で扱うことができる。spec によってモデル化した PCA インスタンスである pca に新たなスペクトルデータ spec_new を作用させて次元削減するときには `pca.transform(spec_new)` とすればよい。

ここで，data.csv を図 1 で示したガウス関数とし，図 2 の 008 行を `pca＝PCA(n_components＝1).fit(spec)` と書き換えて成分数を 1 とし，主成分分析を行ったときの結果をみてみよう（図 3）。第 1 主成分のローディングの波形は元データの波形と相似であり，第 1 主成分の寄与率は 1.0，即ち 100 % となっている。これは，元データがこのローディングの波形に対応する新たな 1 次元に次元削減され，この波形の定数倍であらわされることを意味する。スコアの値がサンプル変数に対して直線的に変化しているのは，式(1)で信号強度 y がサンプル変数 t に比例する，即ちランベルト―ベール則に従うことに対応している。しかし，t=10 におけるスコアの値をローディングの値に掛けて元のスペクトルを再現しようとすると，元のスペクトルは正のピークを持つ波形なのに対し，再現スペクトルは負のピークを持つ波形となってしまうことに気づく。その確認は `pandas.DataFrame(score.iloc[0,0]*loading.iloc[0]).plot()` とすればよい。これは sklearn の PCA クラスが，前処理としてデータのセンタリングを自動的に行っているためである。センタリングとは，各スペクトル変数におけるサンプル変数方向の平均値が 0 となるように平均値を引く操作であり，`spec−spec.mean()` によって計算される。実際に，次元削減されたスコアの値から元のスペクトルを再現するには `pca.inverse_transform`

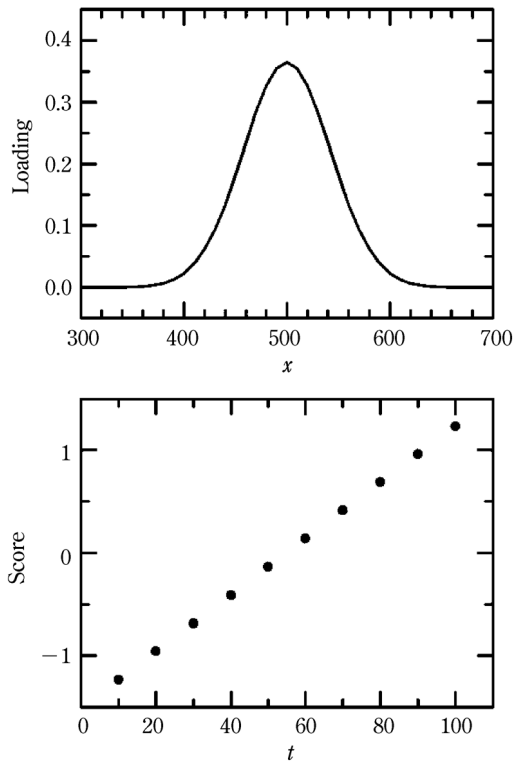

図 3　図 1 のデータ（**data.csv**）を図 2 のプログラム（**pca.ipynb**）で解析した結果。（上）ローディング，（下）スコア。

(score) によって逆変換すればよい。

4　Python を用いたその他のケモメトリックス・機械学習

一般的な機械学習の教科書を読むと，機械学習には教師なし機械学習と教師あり機械学習があり，教師なし機械学習の例として次元削減とクラスタリングが，教師あり機械学習の例としてクラス分類と回帰あたりが紹介されている。sklearn の cheat sheet[19] をみても，この四つに分類されており，次元削減として主成分分析，クラスタリングとして k−means，クラス分類として support vector machine（SVM）による分類，回帰として SVM による回帰，あたりが紹介されている。この他にも sklearn には様々な解析ツールが含まれており，例えば分析化学の分野でよく使われている線形判別分析（linear discriminant analysis，LDA）や部分最小二乗（partial least squares, PLS）回帰も選ぶことができる[9]。階層的クラスタリングによりデンドログラムを作図するには scipy ライブラリにある scipy.cluster.hierarchy を使えばよい。これらの解析手法はケモメトリックスの教科書にも紹介されていることが多く，化学の分野におけるケモメトリックスとデータサイエンスの分野における機械学習を実用面で区別する意味はないのかもしれない。ケモメトリックスにおいては，化学の分野で有用な，実験計画法，スペクトル分解[5]，二次元相関法[20,21] といった解析法もしばしば紹介されていることが多い。

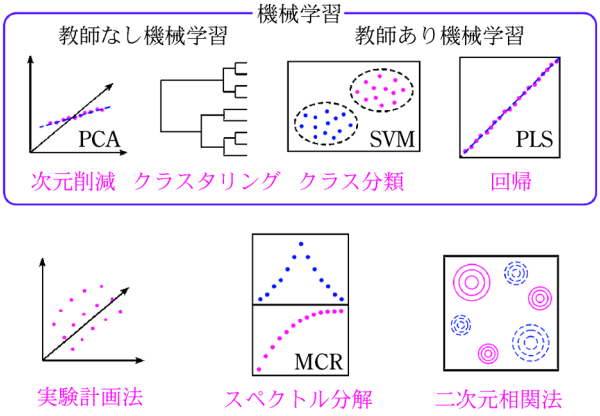

図 4　ケモメトリックス・機械学習の大まかな分類。

これらもライブラリを用いることで Python を使って計算が可能であり，他の総説にまとめてあるので参考にしてほしい[9]。図 4 に，ケモメトリックスの教科書でしばしば取り扱われる解析手法と，その中でも機械学習の教科書でもしばしば取り扱われる解析手法を模式的にあらわした。少なくとも筆者は，データ解析の道具として実践的に使う立場で，ケモメトリックスと機械学習を分けて扱う必要はないと考えている。これまで，残念なことに，ケモメトリックスを学ぶ際の教科書，特に日本語で書かれた教科書はそれほど多くなかった。しかし，現在，機械学習の教科書は，日本語で書かれたものも含めて，選ぶのに困るくらい良書にあふれている[12)~14)]。特に Python による機械学習の教科書や web サイトは，他の解析ツールと比較して，質・量共に豊富であり，初心者も上級者も情報を得やすくなっている。

そこで最後に，機械学習の教科書を使ってケモメトリックスを学ぶコツを紹介しよう。多くの機械学習の教科書が，sklearn に付属しているデータセット（sklearn.datasets）を用いて解説がなされている。例えば次元削減やクラス分類の練習には，アヤメの花の計測データ（iris）が，回帰の練習にはボストンにおける住宅価格のデータ（boston）が，それぞれよく用いられている。iris は 150 サンプル分のデータがまとめられており，それぞれについて，がくの長さ，がくの幅，花びらの長さ，花びらの幅の 4 種類の計測データと，セトナ，バーシクル，バージニカのいずれかの品種の分類が記されている。これを，強引ではあるが，4 種類の計測（説明変数）をスペクトル変数と捉え，3 種類の品種でサンプル変数が記述された 150 本のスペクトルであると想定しよう。そうすると，これは図 5 に示すような「縦長」のデータセットであり，「横長」の実際のスペクトルデータとは構造が異なることに気付く。boston の場合も 13 の説明変数（スペクトル変数）によって住宅価格（サンプル変数）が与えられている 506 サンプルのスペクトルとして捉えると，「縦長」の行列である。これらの「縦長」のデータセットを使って機械学習の教科書は説明がなされるが，我々が読むときに注意しなけ

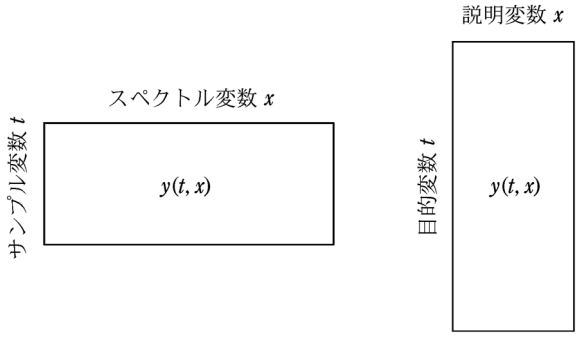

図 5　（左）ケモメトリックスでよく用いられる「横長」のデータセットと（右）機械学習でよく用いられる「縦長」のデータセット。

ればならないのは，実際のスペクトルデータやクロマトデータはしばしば「横長」であるということである。この点に注意しておけば，一般的な機械学習の教科書に従って，機器分析データを様々なデータ解析手法に供することができるようになる。

このとき，場合によっては，まず主成分分析を行って，「横長」のスペクトルデータを次元削減し，「縦長」のデータセットに変換してから，その他の機械学習を試みるのもよい。これによって多重共線性の問題を解決できることがある。回帰分析において，このアイディアを採用したのが主成分回帰（PCR）であり，主成分分析によって次元削減し，得られた「縦長」のデータセットを使って sklearn.linear_model.LinearRegression により重回帰分析を行えばよい。

5　おわりに

本稿では，プログラミング言語の Python と，sklearn 等の有用なライブラリを活用して，機器分析データを解析する手順やコツを紹介した。今回はケモメトリックス・機械学習に限って紹介したが，Python を使うと，ベースライン補正やピーク検出，スムージングや微分処理，関数フィッティング，高速フーリエ変換，*t* 検定や分散分析（ANOVA）のような仮説検定，等々も簡単なプログラミングで行うことができる。先に述べたケモインフォマティクスやマテリアルズインフォマティクスの分野でも Python がよく用いられている。また，新しい解析アルゴリズムを考案したときに，GitHub で公開することによって，多くの人に使ってもらえるし，逆に最新の解析アルゴリズムを試してみることもできる。

このような機器分析データの解析は，情報科学の専門家に任せることなく，化学の専門家が自ら行うべきである。そうしないと，化学としてあり得ない解釈をしてしまいがちであり，それに気付かないまま論文や学会で発表をしてしまうからである。

人生で初めて機器分析をしたときのように，最初は得られた結果の読み取り方すらわからないかもしれない。多くの経験を積みながら，解析手法の選定や計算パラメータの調整について，コツをつかんでほしい。データ解析は，訓練によって，誰にでも習得できるコアな技術である。分析化学者がデータサイエンスのスキルを身につけることは，新たな機器分析装置を導入するより，はるかに低予算であり，汎用性があり，将来が期待できる投資ではないだろうか？

文　献

1）塚本邦尊，山田典一，大澤文孝：“東京大学のデータサイエンティスト育成講座 Python で手を動かして学ぶデータ分析”，(2019)，(マイナビ出版)．
2）化学同人編集部編：“実験データを正しく扱うために”，(2007)，(化学同人)．
3）南　茂夫：“科学計測のための波形データ処理”，(1986)，(CQ 出版)．
4）船津公人（訳），佐藤寛子（訳），増井秀行（訳），J. Gasteiger（編），T. Engel（編）：“ケモインフォマティックス”，(2005)，(丸善)．
5）森田成昭：分析化学，**67**, 179 (2018)．
6）森田成昭，尾崎幸洋：高分子，**67**, 680 (2018)．
7）https://www.python.org, (2020 年 4 月 10 日，最終確認)．
8）G. van Rossum, Python Dev Team: “Python 3.6 Tutorial”, (2016), (Samurai Media Limited)．
9）S. Morita: *Anal. Sci.*, **36**, 107 (2020)．
10）尾崎幸洋，宇田明史，赤井俊雄：“化学者のための多変量解析”，(2002)，(講談社サイエンティフィク)．
11）長谷川健：“スペクトル定量分析”，(2005)，(講談社サイエンティフィク)．
12）金子弘昌：“化学のための Python によるデータ解析・機械学習入門”，(2019)，(オーム社)．
13）S. Raschka（著），株式会社クイープ（訳），福島真太朗(監訳)：“Python 機械学習プログラミング”，(2018)，(インプレス)．
14）A. C. Muller(著)，S. Guido(著)，中田秀基(訳)：“Python ではじめる機械学習”，(2017)，(オライリージャパン)．
15）金丸隆志：“Raspberry Pi で学ぶ電子工作”，(2016)，(講談社)．
16）https://www.anaconda.com，(2020 年 4 月 10 日，最終確認)．
17）池内孝啓，片柳薫子，岩尾エマはるか，@driller: “Python ユーザのための Jupyter［実践］入門”，(2017)，(技術評論社)．
18）掌田津耶乃：“データ分析ツール Jupyter 入門”，(2018)，(秀和システム)．
19）https://scikit-learn.org/stable/tutorial/machine_learning_map/，(2020 年 4 月 10 日，最終確認)．
20）森田成昭，新澤英之，尾崎幸洋：分光研究，**60**, 243 (2011)．
21）S. Morita, Y. Ozaki: *Chemometrics Intellig. Lab. Syst.*, **168**, 114 (2017)．

森田成昭（Shigeaki Morita）
大阪電気通信大学工学部（〒572-8530 大阪府寝屋川市初町 18-8）。東京農工大学大学院生物システム応用科学研究科修了。博士（学術）。≪現在の研究テーマ≫分子分光とデータ解析。≪主な著書と出版社名≫機器分析（共著），(講談社)。≪趣味≫ジャズドラム。
E-mail : smorita@isc.osakac.ac.jp