

# 機械学習応用 実践演習 テキスト

荒木雅弘

2018 年 7 月 24 日

## 第 1 章

# Python 入門 (1)

### 1.1 プログラミング言語 Python

プログラミング言語 Python<sup>\*1</sup>はオブジェクト指向スクリプト言語です。コンパイルが不要なので、短いコードを書いてその実行結果を確認しながらプログラムを組んでゆくことができます。

Python ではひとつのファイルからなるスクリプトをモジュールとよび、そのファイル名から拡張子 (.py) を取り除いたものがモジュール名となります。モジュールは、他のスクリプトで `import` 文を使って読み込むことができ、読み込んだモジュールのクラス・関数・変数は「モジュール名. 識別子」で参照することができます。

また、複数のモジュールをまとめたものをパッケージとよびます。Python は数値計算・グラフ表示などのパッケージが充実しており、科学技術計算のプログラム作成に適しています。特に機械学習に関しては、最新の手法のほとんどが Python で実装されており、深層学習も含め、パッケージ化が進んでいます。

### 1.2 環境構築

機械学習に関連する Python のパッケージには以下のものがあります。

- numpy : 多次元配列を効率よく扱うライブラリ
- scipy : 高度な数値計算ライブラリ
- matplotlib : グラフ描画のためのライブラリ
- pandas : データ解析を支援するライブラリ
- scikit-learn : 機械学習のためのライブラリ

また、Python をブラウザ上で対話的に実行でき、Markdown 記法と LaTeX の数式記法で同時にメモを残せる Jupyter notebook を利用することで、小規模なプログラムを対話的に作成できます。これらの環境をすべて事前にインストールした、Anaconda<sup>\*2</sup>をインストールすることをお勧めします。

---

<sup>\*1</sup> 本章ではバージョン 3.6.5 に基づいて解説してゆきます。

<sup>\*2</sup> <https://www.anaconda.com/download/>

## 1.3 Jupyter Notebook の使い方

Anaconda をインストール後、メニューから Jupyter notebook を選択すると、ブラウザが起動して、notebook が作成可能になります。適当なフォルダに移動して、New→Python 3 を選択すると、新しい notebook が作成されます。メニュー・アイコンで提供されている機能や、notebook の名前の変更法を確認しておきましょう。

### 1.3.1 Markdown によるメモ

入力可能なボックスにカーソルを置き、メニューから Markdown を選ぶと、Markdown 記法でメモを挟むことができます。

よく使う Markdown 記法

- 見出し：“#”（レベルは“##”，“###”などで調整）
- 改行：行末で 2 つ以上の空白
- 箇条書き：“\*” と空白を先頭に。入れ子は先頭に空白を入れて、記号を“-”に変更。
- 数式：“\$” で囲み、LaTeX の数式記法で書く

### 1.3.2 Jupyter notebook のキーボードショートカット

以下のキーボードショートカットは便利なのでぜひ憶えておきましょう。

- esc を押してコマンドモード
  - Enter: セルの編集
  - m: マークダウンモード
  - y: コードモード
  - c: セルのコピー
  - v: コピーしたセルのペースト
  - dd: セルの削除
  - Space: スクロールダウン
  - Shift + Space: スクロールアップ
  - h: ショートカット一覧の表示
- セルの編集モードの時
  - Shift+Enter: セルの実行、次のセルへ移動
  - ctrl +Enter: セルの実行のみ

## 1.4 Python の文法

公式チュートリアル<sup>\*3</sup>を使って、変数・データ構造・制御文・関数の定義・モジュールの扱いについて学んでおきましょう。チュートリアルの要点は Tutorial1.ipynb を参照してください。

## 1.5 データの準備とグラフ表示

機械学習用データの扱いに慣れておくために、行列の扱い・グラフ表示・CSV ファイルの読み込みについて学んでおきましょう。手順は Tutorial2.ipynb を参照してください。

また、matplotlib パッケージを用いたグラフ表示については、以下のブログが参考になります。

<http://bicycle1885.hatenablog.com/entry/2014/02/14/023734>

## 実践演習 1-1

パターン認識実践で作成した数字画像の特徴ベクトルを保存した CSV ファイルを読み込み、分散  $x$  に関するヒストグラムおよび分散  $x, y$  の散布図を作成せよ。

---

<sup>\*3</sup> <http://docs.python.jp/3/tutorial/>

## 第 2 章

# Python 入門 (2)

### 2.1 機械学習を行うクラスの使い方

Scikit-learn<sup>\*1</sup>は、識別・回帰・クラスタリング・次元削減などのツールが実装されたパッケージです。各アルゴリズムはクラスとして設計されていて、以下の共通した基本仕様からなります。

- コンストラクタ：クラスの初期化  
引数はアルゴリズムのパラメータ。
- `fit()` メソッド：学習  
引数は学習データと正解ラベル。必要に応じてデータに依存したパラメータ。
- `predict()` メソッド：予測  
学習済みのインスタンスに対して、予測対象のデータを引数として与えると、結果を返す。

上記仕様を満たした 1-NN 法を実装したクラスの例を以下に示します。

---

```
class NN(object):
    """simple 1-NN classifier"""
    def __init__(self):
        self.X = None
        self.y = None

    def fit(self, X, y):
        self.X = X
        self.y = y

    def predict(self, x):
        return y[np.argmax(np.linalg.norm(
            self.X - np.tile(x,(self.y.size,1)), axis=1))]
```

---

### 2.2 交差確認法

1 つ抜き法の実装は、行列から位置を指定して要素を削除する `np.delete` を用いて識別したいデータを除いた学習データを作成し、`fit` メソッド、`predict` メソッドを順に呼んで識別を行います。

---

```
clf = NN()
for i in range(y.size):
    x = X[i]
    X2 = np.delete(X, i, axis=0)
    y2 = np.delete(y, i)
```

---

<sup>\*1</sup> <http://scikit-learn.org/>

```
clf.fit(X2,y2)
print(clf.predict(x), end = ' ')
```

---

10-fold CV のように学習データを分割するときは、クラスバランスを考えて分割する必要があり、コードが複雑になります。sklearn.model\_selection.cross\_val\_score は、識別器・特徴ベクトル・正解ラベル・分割数を与えるだけで、クラスバランスを考慮した分割を行って、評価した値を返します。

## 2.3 多クラス識別の評価

多クラス識別の評価値を計算する場合は、目的に応じて以下の計算法を使い分けます。

- マイクロ平均  
全体の混同行列から計算する
- マクロ平均  
クラスごとに評価値を求めて平均する
- 加重平均  
クラスごとの評価値に対して、クラスごとの事例数で重みを付けて足し合わせる

### 実践演習 2-1

教科書 2.2 節の手順に従い、scikit-learn を用いた学習手順を確認せよ。また、学習結果の評価について、マイクロ・マクロ・加重それぞれの方法で正解率を求め、このデータの場合、どの値で判断するべきか考えよ。

### 実践演習 2-2

使用するデータを breast cancer に変更して、実践演習 2-1 と同様の手順で学習を行え。

## 第 3 章

# 決定木

### 3.1 目的

決定木による識別器の学習アルゴリズムでの数値特徴の扱いや、枝刈りを使った木の大きさの制御について学びます。

### 3.2 Weka J48

#### 3.2.1 数値特徴の扱いと過学習

決定木の学習アルゴリズムで数値特徴を扱う場合、図 3.1 に示すように、特徴空間を縦横に区切って、同じクラスのものが集まる区画を求めていることになります。

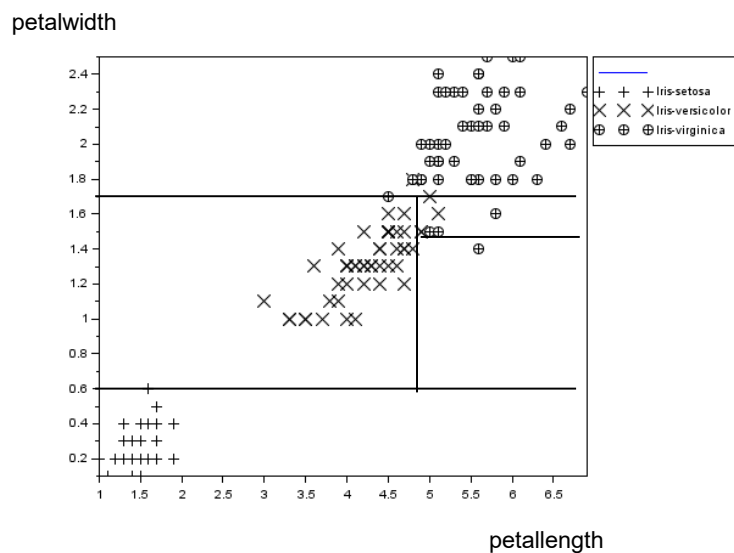


図 3.1 iris.2D データの決定木による識別

この処理を最後まで続けると、学習データに対する識別率はほぼ 100% になる過学習が起きます。そこで決定木学習では、予めデータを学習用と評価用に分けておき、学習用で過学習させたあと、評価用で枝刈りをする処理をします（教科書 p.51）。

## 実践演習 3-1

iris.arff データおよび iris.2D.arff データから決定木を作成し、Use training set でなるべく高い識別率を実現せよ。その際、調節するパラメータは以下のものとせよ。

- minNumObj: リーフとなる事例数の最小数
- unpruned: 枝刈り処理の有無

## 実践演習 3-2

iris.arff データから決定木を作成し、10-fold CV でなるべく高い識別率を実現せよ。

## 実践演習 3-3

credit-g データ<sup>\*1</sup> から決定木を作成せよ。その際、識別性能と識別結果の説明可能性（木の単純さ）とのバランスを考えてパラメータを決定せよ。

## 3.3 sklearn DecisionTreeClassifier

## 実践演習 3-4

sklearn.datasets にある iris データから決定木を作成し、交差確認法で性能を評価せよ。また、決定木をより単純にしたときの性能の変化を調べよ。

---

<sup>\*1</sup> credit-g データは、金融機関が個人顧客に融資をするかどうか (good or bad) を判定したデータです。特徴は 20 種類で、カテゴリ属性として checking\_status (当座預金残高)、saving\_status (普通預金残高)、purpose (目的) など、数値属性として age (年齢)、duration (貸出月数) などがあります。