機械学習応用 実践演習 テキスト

荒木雅弘

2018 年 8 月 10 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[*1]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も残せる 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[*2]をインストールすることをお勧めします。

[*1] 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。

[*2] `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[*3]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。

[*3] 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[*1]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.argmin(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)
```

[*1] 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作成や、過学習への対処法について学びます。

3.2　Weka J48

3.2.1　カテゴリ特徴

実践演習 3-1

Weka の J48 アルゴリズムを用いて，カテゴリ特徴からなる breast-cancer.arff から決定木を作成せよ．その際、パラメータ unpruned の値（False/True）と、Test options の設定（use training set/Cross-validation: Folds=10）のそれぞれを組み合わせて、枝刈りの効果を確認せよ。

3.2.2　数値特徴

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。

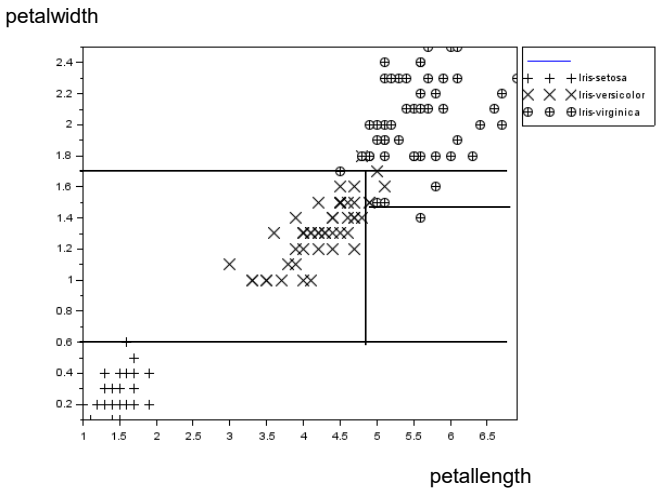

図 3.1　iris.2D データの決定木による識別

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 3-2

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 3-3

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 3-4

credit-g データ[*1] 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 3-5

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。

[*1] 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。

第 4 章

ベイズ識別

4.1 目的

カテゴリカルデータからなる特徴ベクトルや、数値データとカテゴリカルデータが混在した特徴ベクトルに対して、ナイーブベイズ識別器を作成し、何が学習されているのかを観察します。また、カテゴリカルデータに対するベイジアンネットワークの作成・学習・評価について学びます。

4.2 Weka NaiveBayes

実践演習 4-1

weather.nomimal データを用いて Use training set でナイーブベイズ識別器を学習し、学習結果を観察せよ。また学習結果を利用して、(rainy,hot,high,TRUE) で yes となる確率を求めよ。

実践演習 4-2

weather.numeric データを用いて Use training set でナイーブベイズ識別器を学習し、学習結果を観察せよ。

4.3 Weka BayesNet

Bayes net editor

Weka の Bayes net editor はノードとリンクをグラフィカルに配置して、その上で条件付き確率表を設定することでベイジアンネットワークを作成することができます。また、特定の変数に値を与えて、他の変数の確率の変化を見ることができます。ときどきレイアウトが乱れますが、そのときは教科書 p.71 に説明してあるように Tools→Layout で再配置します。また、ノードを動かすときに一度クリックして選択してから動かすようにすると、レイアウトが乱れることはないようです。

実践演習 4-3

Weka の Bayes net editor を使って教科書 p.76 図 4.9 のベイジアンネットワークを作成し、教科書 p.72〜p.76 の計算を確認せよ。

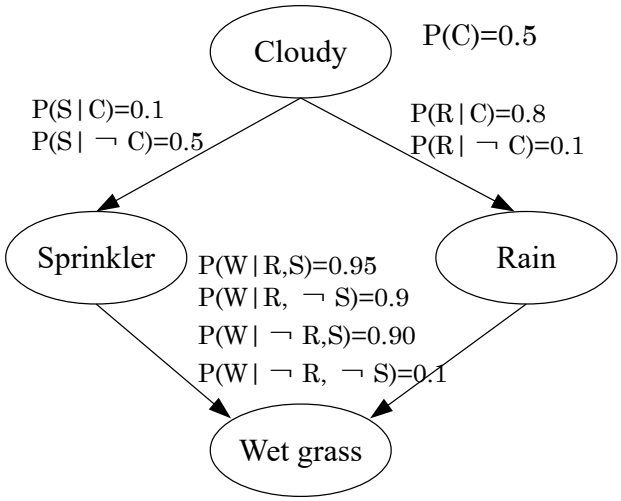

図 4.1　教科書 p.76 図 4.9

実践演習 4-4

weather.nomimal データを用いて Use training set でベイジアンネットワークを学習し、学習結果を観察せよ。学習結果のネットワークは、Result list の該当回を右クリックし、Visualize graph を選ぶと表示される。また、表示されたネットワークのノードを右クリックすると、条件付き確率表を見ることができる。

実践演習 4-5

breast-cancer.arff データ[*1]を用いて 10-fold CV でベイジアンネットワークを学習し、ネットワーク構造の複雑さと性能の関係を観察せよ。ネットワークの複雑さは、ベイジアンネットワークの構造を学習する searchAlgorithm の K2 法において、親ノードの最大数 (MaxNrOfParents) によって調整する。

[*1] breast-cancer データは、患者の年齢・腫瘍の大きさ・悪性度などの情報と、癌再発の有無が記録されたものです。

第 5 章

生成モデルと識別モデル

5.1　目的

数値データからなる特徴ベクトルに対して、生成モデルに基づいた識別器（ナイーブベイズ）と識別モデルに基づいた識別器（ロジスティック識別）で学習を行い、その違いを確認します。

5.2　Weka

Weka のナイーブベイズの実装は NaiveBayes、ロジスティック識別の実装は SimpleLogistic です。

実践演習 5-1

Weka 付属の iris データに対して、NaiveBayes と SimpleLogistic で識別器を構成し、結果を比較せよ。

実践演習 5-2

Weka 付属の glass データ[*1]に対して、NaiveBayes と SimpleLogistic で識別器を構成し、結果を比較せよ。

5.3　sklearn

Scikit-learn のナイーブベイズの実装は GaussianNB、ロジスティック識別の実装は LogisticRegression です。

実践演習 5-3

Scikit-learn の GaussianNB と LogisticRegression で実践演習 5-1 と同様の実装を行え。iris データは scikit learn 組み込みのデータを用いること。

[*1] glass データは、屈折率および組成（Na, Mg などの含有量）からガラスの用途（建物、車、食器など）を判定するものです。

実践演習 5-4

Scikit-learn の GaussianNB と LogisticRegression で実践演習 5-2 と同様の実装を行え。arff 形式ファイルの読み込みは、教科書 p.87 例題 5.2 を参照のこと。

第 6 章

回帰

6.1 目的

数値特徴および数値・カテゴリの混合特徴を入力して数値を出力する関数を学習する回帰について学びます。線形回帰では、正則化の効果について学びます。回帰木では性能を高める方法について学びます。

6.2 Weka

実践演習 6-1

アッシェンフェルターのワイン方程式を導く元となったデータ `http://www.liquidasset.com/winedata.html` から作成した wine.csv から、Weka の LinerRegression を用いて回帰式を導け。ただし回帰式の学習には 1980 年までのデータを用いること。

実践演習 6-2

例題 6.1 に従い、Weka の LinerRegression を用いて cpu データで回帰を行え。その際、正則化係数 (ridge) を変化させ、学習された線形関数の重みを観察せよ。

実践演習 6-3

Weka の LinerRegression を用いて cpu.with.vendor データで回帰を行い、学習結果を解釈せよ。

実践演習 6-4

例題 6.4 に従い、Weka の REPTree を用いて cpu データで回帰を行え。その際、木の大きさを変えて相関係数の変化を観察せよ。

実践演習 6-5

例題 6.5 に従い、Weka の M5P を用いて cpu データで回帰を行え。その際、木の大きさを変えて相関係数の変化を観察せよ。

6.3　sklearn

実践演習 6-6

sklearn の LinerRegression を用いて boston データで回帰を行え。その際、正則化係数を変化させ、学習された線形回帰式の重みを観察せよ。

1. boston データの読み込み（パターン行列 X、ターゲット y）
2. LinearRegression の学習
3. 学習結果オブジェクトの coef_の値を表示して係数を確認
4. sklearn.model_selection.cross_val_score を使って交差確認を行い、決定係数を表示
5. L2 正則化の Ridge で同様の手順
6. L1 正則化の Lasso で同様の手順

実践演習 6-7

sklearn の DecisionTreeRegressor を用いて boston データで回帰木を作成せよ。その際、木の大きさを変えて、誤差の変化を観察せよ。

第 7 章

SVM

7.1 目的

SVM を用いた識別と回帰を行います。また、学習の際のパラメータの最適値を、sklearn の Grid search および Randomized search を使って求めます。

実践演習 7-1

Weka の SMO で例題 7.3 に基づいて ReutersGrain の学習と評価を行え。その際、カーネルの種類やハイパーパラメータをいくつか変化させて、識別率への影響を観察せよ。

余裕があれば、StringToWordVector フィルタ適用時に、(1) stopwordsHandler の値を Rainbow[*1]にする、(2) IDFTrandform, TFTransform を True にする、などの変更を行って、変化を観察せよ。

実践演習 7-2

実践演習 7-1 の内容を sklearn の SVC を用いて行え。その際、(1) CountVectorizer で stop_words 引数 (値は'english') を用いる、(2) TfidfVectorizer を用いるなどして、単語ベクトルを変更して変化を観察せよ。

7.2 Grid search

Grid search は、パラメータの可能な値をリストアップし、そのすべての組み合わせについて性能を評価して最も性能の高い組み合わせを求めます。

パラメータを組み合わせる空間をグリッドとよび、グリッドは Python のディクショナリ（辞書型オブジェクト）で表現します。試みたいパラメータの組み合わせは、ディクショナリのリストで表します。各グリッドでは、軸をディクショナリのキーに、可能な値をリスト形式の要素で表現します。

例えば、サポートベクトルマシンの「スラック変数の重み C」と「多項式カーネルの次数 degree」でグリッドを構成する場合、C の値 1, 10, 100, 1000、degree の値 1,2,3 と変化させたいときは以下のようなディクショナリのリストを作ります。

```
param_grid = [
  {'C': [1, 10, 100, 1000], 'degree': [1, 2, 3]}
]
```

[*1] Rainbow は Weka 組み込みの Stopwords list です。

実践演習 7-3

GridSearchCV で iris データに対するサポートベクトルマシン (SVC) 識別の最適なパラメータを求めよ。なお、多項式カーネル以外に RBF カーネルも探索の対象とすること。

7.3　Randomized search

RandomizedsearchCV は、パラメータの可能な値の範囲を指定しておき、そこから乱数で値をリストアップし、そのすべての組み合わせについて性能を評価して最も性能の高い組み合わせを求めます。生成する組み合わせの数は n_iter（デフォルトは 10）で指定します。

パラメータが連続値の場合は、値に確率密度関数を与えて、RandomizedsearchCV にパラメータ値を生成させます。

```
param_grid =   {'C': scipy.stats.expon(scale=100), 'gamma': scipy.stats.expon(scale=1)}
```

scipy.stats.expon は $exp(-x)/scale$ に基づいて乱数を発生させるものです。

パラメータが整数の場合は、range 関数で範囲を与えておきます。

```
param_grid =  {'C': range(1,1000), 'degree': range(1,5)}
```

実践演習 7-4

RandomizedSearchCV で boston データに対する rbf カーネル SVR 回帰の最適なパラメータを求めよ。