機械学習 演習資料

手塚太郎

Python

Pyrhon は Ruby とよく似たオブジェクト指向のスクリプト言語であり、他の言語でのプログラミング経験がある場合は容易に習得できる。ただし以下の特徴には注意する。

・if や for などの対象となるブロックは Ruby のように end までの範囲ではなく、また C 言語のように中括弧で括るのでもなく、同一数のスペースやタブからなるインデントを使うことで表す。ゆえに end 文は不要である。たとえばスペース 2 つ以上のインデントを持つすべての行がひとつのブロックとみなされ、その中にあるスペース 3 つ以上のインデントを持つすべての行がまたひとつのブロックとみなされる、などである。

・制御構造を表す各行（たとえば for や if の行）の終わりに「:」を付ける必要がある。

たとえば Python において hello!と 3 回出力させる for ループは以下のように書く。
range(0,3)は 0 以上 3 未満の整数を表す範囲であるため、変数 a は 0,1,2 と動く。

```
for a in range(0,3):
    print("hello!")
```

Google Colaboratory の利用

Google が提供しているクラウドによる計算環境 Google Colaboratory（略称 Colab）では Python を用いて機械学習のプログラミングが行える。

https://colab.research.google.com

上記のページにブラウザでアクセスし、一番下に現れる New Python 3 Notebook という文字列を選択すると、Python スクリプトが実行できるセルが表示される。なお、新しいセルを作成したい場合はメニューバーの下にある「+CODE」というボタンをクリックすればよい。「+TEXT」はコード（スクリプト）への説明を書くためのテキストセルを追加するためのものである。

Colab 上で TPU（Tensor Processing Unit）を使用するため、メニューバーの Runtime から change runtime type を選択し、Hardware accelerator を None から TPU に切り替え、Save を押す。（None、GPU、TPU のそれぞれで同じスクリプトを走らせ、どれだけ実行時間が違うかを比較してもよい。ただし深層学習で使用する Tensorflow など GPU で高速化されるフレームワークを使わなければ違いは見られない）。

Colab のセル内では pip install 等の（Python の一部でない）コマンドは先頭に「!」をつけて実行する。たとえば pillow パッケージをインストールするには以下になる。

```
!pip install pillow
```

同様に git の前に「!」をつけて実行することで git のリポジトリを Colab 上にクローンできる。

```
!git clone https://github.com/tarotez/pyml
```

その他の git のコマンドも!git pull のように実行できる。

クローンされたリポジトリ pyml の中に移動するには以下のコマンドを使う。cd は change directory の略であり、ディレクトリを移動するためのコマンドである。

```
cd pyml
```

続いて以下のようにサンプルコード mnist_tb.py を実行する。

```
run mnist_tb.py
```

なお、これらの UNIX のコマンドをひとつのセルで複数実行させようとするとエラーが出る。UNIX のコマンドはひとつのセルにひとつである。これに対し、Python のプログラムは何行に分けて書いても問題ない。

出力結果のうち、TensorBoard link:の後に表示されているリンク（ngrok.io を含む URL）をクリックすると、別のタブ上でネットワーク構造や訓練誤差、テスト誤差の時間変化を TensorBoard の結果が表示される。グラフは計算がまだ終わっていない段階でも表示できる。

スクリプトの実行を途中で停めるにはセルの左上隅にある赤い再生ボタンをクリックすればよい。その後、+CODE というボタンをクリックすると新しいセルが作られる。

データ解析や機械学習に使いたいデータを Colab 上にアップロードするには、左端の縦長のメニュー領域（隠れている場合は左上隅の「CO」というアイコンの下にある「>」をクリックすると表示される）から「Files」タブを選び、現れる「UPLOAD」というボタンをクリックすればよい。実際にアップロードされているかどうかはセル内で ls コマンドを使って確認できる。

Colab についての基本的な使用方法の説明は以下にある。

```
https://colab.research.google.com/notebooks/basic_features_overview.ipynb
```

Colab 上で入力した命令はすべて自動で ipynb ファイル（notebook）として Google のクラウド上に保存されており、次に Colab のページにアクセスした時に候補として表示される。

matplotlib.pyplot を使ってグラフを描くサンプルコードの説明は以下にある。

```
https://colab.research.google.com/notebooks/charts.ipynb
```

Colab における TPU や GPU の使用についての説明は以下にある。

```
https://colab.research.google.com/notebooks/gpu.ipynb
```

Google Drive のマウント

Google Colaboratory はしばらく操作しないとアップロードしたデータは消えてしまうため、データや計算結果を Google Drive に置き、それをマウントするという方法も可能である。もし Google アカウントを持っていない場合はまずそれを作成する。その後、セル上で以下を実行することでマウントできる。

```
from google.colab import drive
drive.mount('/content/gdrive')
```

Go to this URL in a browser: という文字列の左に URL が表示されるが、そちらにアクセスし、どのアカウントの Google Drive をマウントするかを選択すると、認証用のコードが表示される。それをコピーして Colab 側で Enter your authorization code:というメッセージの下の欄に貼り付ける。

これによって Colab 上で/content/gdrive の下に Google Drive がマウントされる。

pyenv と pip による Python 環境の構築

自分の PC を持ち込む場合、以下のようにローカルマシン上で Python 環境を構築できる。

Linux あるいは Mac の場合：

まずは端末を開く。これには Ctrl と Alt と t をこの順で指を離さずに押す。

pyenv を使うとひとつのマシン上で python の環境を複数作成できる。pyenv は以下のように端末のシェルから git を使ってインストールする。

```
git clone https://github.com/yyuu/pyenv.git ~/.pyenv
```

これによって異なる種類やバージョンのライブラリをインストールした複数の環境を構築できるため、ライブラリへの依存やバージョンを考慮したアプリケーション開発に役立つ。

インストール後、~/.bashrc 等の設定ファイルを atom などのエディタで開く。

```
atom ~/.bashrc &
```

末尾に以下を追加する。

```
export PYENV_ROOT="$HOME/.pyenv"
export PATH="$PYENV_ROOT/bin:$PATH"
eval "$(pyenv init -)"
```

その後、シェルから以下を実行する。

```
source ~/.bashrc
```

以下のように pyenv で Python 環境を構築する。あまり新しいバージョンを入れると TensorFlow 等のフレームワークが対応していないことがあるため、この演習では anaconda3-5.3.1 を使用する。

```
pyenv install anaconda3-5.3.1
pyenv rehash
pyenv global anaconda3-5.3.1
```

上記では pyenv install によって Python をインストールし、pyenv rehash でリハッシュし、pyenv global によってそのバージョンの Python を使用することを選択している。
pyenv を使うと複数のバージョンの Python をインストールし、pyenv global によってそれらの間で切り替えられる。上記コマンド実行後、実際に切り替わっているか、以下で確認せよ。

```
pyenv versions
```

上記で system と python 3...の 2 行が表示され、python 3... の左側に「*」が出ていればよい。

続いて pip を使い、各種パッケージを以下のようにインストールする。

```
pip install jupyter numpy pandas scipy scikit-learn
```

インストールには数分程度掛かることがある。その場合、以下の「IPython と Jupyter」という章に書かれている Python のコードは端末上で「python」と入力して改行することで起動される Python インタープリタでも実行できるので、先にそちらを試してみてもよい。Python インタープリタから抜けるには quit()と入力する。

インストール時に容量オーバーが出た場合は du コマンドを使って大きなファイルを探し、消すとよい。たとえば~/lec/java ディレクトリにあるファイルサイズの大きい 20 件のファイルならびにディレクトリを表示させるのは以下のようにする。

```
du -a ~/lec/java | sort -n -r | head - 20
```

とりあえず消せるファイルとして、Firefox のキャッシュ（過去に見たページを保存してあるもの）がある。これは rm コマンドで消してもよいが、慣れていない場合は間違ったファイルを消してしまうことがあるので、以下のように mv コマンドでディレクトリ.cache/mozilla を old.mozilla という名前に変えてからホームディレクトリで右クリックして削除し、その後ゴミ箱を空にすること。

```
mv ~/.cache/mozilla ~/old.mozilla
```

Windows の場合：

以下のページから Anaconda の Windows 版をダウンロードしてインストールすると、スタートメニューから Jupyter Notebook を起動できるようになる。

```
https://www.continuum.io/downloads
```

IPython と Jupyter

Python のインタープリタである IPython は開発用に広く使用されている。IPython をブラウザ上で動かすものが Jupyter Notebook であり、Google Colab は Google が自社のサーバ上で提供している Jupyter Notebook である。

本節（Ipytho と Jupyter）は自分の PC に前節の Python 環境を構築しなかった場合は飛ばしてよい。

自分の PC で Jupyter Notebook を動かすには Anaconda のインストール後、シェル上（端末上）で以下のコマンドを入力することで、ブラウザからアクセスできる Jupyter Notebook が起動する。

```
jupyter notebook &
```

ブラウザに新しいタブが開かなくても、以下の URL にアクセスしてみると起動していることがある。

```
http://localhost:8888
```

ブラウザ上の Jupyter Notebook の画面の右上にある「New」ボタンをクリックし、プルダウンメニューから「Python3」を選択すると、IPython インタープリタが開始するので、ここに Python のコマンドを入力していける。

また、Jupyter Notebook の Home 画面の右上の「New」ボタンをクリックして表示される「Text File」を選択すると、テキストファイルが作成できるので、これを使って Python のコードをファイルに保存することができる。ファイル名は untitled.txt として左上に表示されているが、クリックすることで変えられる。保存は File→Save で行う。

Jupyter Notebook からプログラムを実行するにはコマンド run を使う。たとえば squares.py というプログラムは以下のように実行できる。

```
run squares.py
```

コード作成と実行

本節以降、Google Colab あるいは自分の PC の Jupyter Notebook 上で順次コマンドを打ち込む形で進めていくとよい。実行には Shift+Enter（Shift を押しながら Enter）を押すこと。

以下のコードを入力した上で、Shift を押しながら Enter を押してみよ。なお、range(0,9)は 0 以上 9 未満の範囲の整数を表す。また、%は整数の割り算で余りを求める演算子である。また、str は数値を文字列に変換する関数である。文字列同士は演算子「+」によって結合できる。

```
for x in range(0,9):
  if x % 2 == 0:
    print(str(x) + "^2 = " + str(x * x))
```

すでに実行したコードのまとまり（セル）を再編集し、Shift+Enter を押すことでふたたび実行させることも可能である。

リストとタプル

Python でデータの集合を扱うために頻出するデータ構造としてリストとタプルがある。

リストを作るには角括弧で括る。

```
animals = ['dog', 'cat', 'mouse', 'gorilla']
```

タプルを作るには丸括弧で括る。

```
summer = ('sun', 'beach', 'wave', 'sea', 'sky')
```

なお、Python では文字列を括るのにシングルクオート（'）とダブルクオーツ（"）の両方が使える。どちらでも同じように使える。文字列の中に'が現れる場合は"、逆に"が現れる場合は'、というように使い変えることができる。

リストやタプルの要素を取り出すにはインデックスを角括弧内で指定する。

```
animals[0]
summer[1]
```

リストのリストやタプルのタプルを作ることも可能である。

```
combined = [animals, summer];
```

リストから部分リストを、タプルから部分タプルを切り出すのには「スライス」と呼ばれるオフセット指定を使用する。スライスは変数名の後に角括弧を付け 3 つの数字をコンマで区切って指定知る。3 つの数字はそれぞれ開始位置、終了位置、ステップ幅の順で指定する。ステップ幅は省略できる。また、終了位置の 1 つ前までが切り出される。

```
summer[1:3]
```

なお、文字列からの切り出しもスライスを使って行える。たとえば str = "Hello World!"の時、str[4:11:2]は（0 から数え始めるので 4+1 で）5 文字目から（12 より 1 文字前までである）11 文字目までを 2 文字ごとに取り出した部分文字列、すなわち oWrd になる。

```
str = "Hello World!"
print(str[4:11:2])
```

指定された値を持つ要素のインデックスを求めるのには index を使う。

```
animals.index('mouse')
```

リストとタプルの違いとして、以下が挙げられる。

・リストは値を後から書き換えられるが、タプルは値を書き換えることができない。

・タプルはリストより消費スペースが少ない。

・関数の引数はタプルとして渡される。

ファイル読み込み

データをファイルに読み書きするには open 関数を使ってファイルを開く。たとえば data.txt というファイルからデータを読み込み、各行を出力させるには以下のようにする。事前に Google Drive 上に data.txt というファイルをアップロードし（自分の PC 上で行う場合は端末上で Jupyter Notebook を起動したのと同じディレクトリに作成し）、cd コマンドを使ってそのファイルが存在するディレクトリに移動しておくこと。ファイルが存在するかどうかは ls コマンドによって確認できる。

```
f = open('data.txt', 'r')
for line in f:
    print(line)
f.close()
```

ファイルに書き出す場合は write メソッドを使う。

```
f = open('message.txt', 'w')
f.write('hello!')
f.close()
```

データセット

UCI Machine Learning Repository では統計学で古くから使われてきた定評あるデータセットが数多く公開されている。

http://archive.ics.uci.edu/ml/index.php

また、Kaggle ではアカウントを作成することで近年作成された多様なデータセットがダウンロード可能である。

https://www.kaggle.com/

いずれもデータセットのダウンロードはすべて無料である。また、データの多くは CSV 形式などのテキストファイルとしてダウンロードできる。

CSV ファイルの読み込み

以下のページでは統計学で有名な iris データセットが得られる。

http://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data

上記ページの全体を iris.data という名前で保存し、以下のように走らせることで、各サンプルの品種名を表示できる。split(',')はコンマ（,）をセパレータとして文字列を分割し、リストを作成するメソッドである。elems[4]によってリスト elems の 5 番目の要素を取得している。

```
f = open('iris.data', 'r')
for line in f:
    elems = line.split(',')
    if len(elems) > 1:
        print(elems[4])
```

```
f.close()
```

文字列を整数に変換するには関数 int、浮動小数点数に変換するには float を使用する。たとえば上記の iris データセットで花弁の長さ（petal length）である第 3 属性の平均値を求めるには以下のようにすればよい。"{0:.2f}".format()は小数点以下 2 桁までを表示させるというフォーマット指定である。

```
f = open('iris.data', 'r')
totalPetalLength = 0
sampleNum = 0
for line in f:
    elems = line.split(',')
    if len(elems) > 1:
        totalPetalLength += float(elems[2])
        sampleNum += 1
print('average petal length is ', "{:.2f}".format(totalPetalLength /
sampleNum))
f.close()
```

辞書

Python の辞書は他の言語でハッシュや連想配列と呼ばれるデータ構造に相当する。キーと値をコロンで区切った組をコンマで区切って並べ、全体を中括弧で括ることで作成できる。

```
capitals = {
     'UK' : 'London',
     'France' : 'Paris',
     'Germany' : 'Berlin'
     }
```

角括弧を使うことでキーを使って値を求められる。

```
print(capitals['France'])
```

また、以下のように追加も行える。

```
capitals['Italy'] = 'Rome'
```

辞書から特定の要素の削除を行うには del を使う。

```
del capitals['UK']
```

辞書が持つキーの一覧をリストで取り出すにはメソッド keys、値の一覧を取り出すにはメソッド values が使える。

キーを使ってソートされた一覧を取り出すには組み込み関数 sorted を使うことができる。

関数

Python で関数を定義するには予約語 def を使う。def で始まる行の末尾の「:」を忘れないように注意すること。以下は階乗を求める関数を定義する例である。

```
def factorial(n):
     f = 1
     for i in range(1,n+1):
          f = f * i
     return f
```

上記の関数定義を Shift+Enter で実行した後、以下を入力して Shift+Enter を押す。これは 100 の階乗（1×2×3×...×98×99×100）という数を求めるということである。

```
factorial(100)
```

すると大きな桁数を持つ数が返ってくる。このように Python では C 言語や Java などと異なり、任意の桁数の整数を扱える 。

numpy

Python で Matlab と同様に行列や多次元配列を操作できるようにするライブラリが numpy である。numpy によって提供される array というデータ構造が Matlab の行列や多次元配列に相当する。

numpy を使用するにあたっては、それを import しておく必要がある。また、毎回 numpy と書くのは長いため、省略で np と書いて numpy という意味を表せるよう、as np を最後に付け加える。すなわち以下のコマンドをプログラムの冒頭に書く。

```
import numpy as np
```

numpy の基本的なデータ構造は配列（array）であり、たとえば以下のように作成できる。

```
a = np.array([2, 3, 5, 7, 11, 13, 17, 19])
```

配列は Matlab の多次元配列と同様、任意の階数を持つことができる。階数 1 の配列がベクトル、階数 2 の配列が行列であり、それ以上は高階テンソルと呼ばれる。

reshape を使うと数列を任意の形に並べ替えることができる。たとえば配列 a を 4 行 2 列の行列にするには以下のようにする。

```
b = a.reshape([4,2])
```

配列の成分はリストやタプルと同様、角括弧で指定できる。スライスも使用できる。数学と異なり、次元の指定を 0 から数え始めることに注意すること。つまり一番上の行は第 0 行であり、一番左の列は第 0 列である。

```
b[0,1]
b[2:,:]
```

配列の次元を求めるには shape プロパティを使う。行列の場合、戻り値を構成する 2 つの数字は行数と列数である。

```
b.shape
```

成分がすべて 0 である配列を作るには zeros()、成分がすべて 1 である配列を作る ones()、成分を 0 から 1 までの一様分布からのサンプリングで得られる配列を作る random()などがある。引数としてタプルを与える必要があるので、丸括弧が二重になることに注意する。

```
c = np.ones((3,2,3))
d = np.random.random((3,4,2))
```

行列の場合、行と列を入れ替える操作を転置（transpose）と呼ぶ。numpy の配列では行、列、深さ、等々、任意の方向の間の入れ替えを transpose メソッドによって行える。引数は入れ替え前の次元である。

```
transposed = d.transpose([1,0,2])
```

2 つのベクトルの内積は dot によって計算できる。arange は range の配列版であり、第一引数以上、第二引数未満の範囲の数字が並んだ配列を作る関数である。

```
e = np.arange(2,5);
f = np.arange(3,6);
ip = np.dot(e,f)
print(ip)
```

Pandas

Pandas は Python で表形式のデータを扱うためのライブラリである。数値だけでなく文字列なども含むデータを扱う際に力を発揮する。numpy の配列には数値しか入れられないが、Pandas のデータ構造であるデータフレーム（DataFrame）には任意のデータを入れられる。

Pandas を使うためにはまず import を行う。

```
import pandas as pd
```

データフレームを作るには関数 DataFrame を使用する。また、データフレームにはインデックス名（属性名）とカラム名（列名）を付けられる。

```
df = pd.DataFrame([["11/1","11/2","11/3"],
['Tue','Wed','Thu'],['Regular','Regular','Holiday']])
df.index = ["Date", "Day of the Week", "Type"]
df.columns = ["Day 1", "Day 2", "Day 3"]
```

単に df と入力するとデータフレームの中身を確認できる。

CSV 形式での書き出しは以下のように行える。

```
df.to_csv("calendar.csv")
```

読み込みは以下で行える。（iris.data というファイルが同じディレクトリに存在する必要がある）

```
iris = pd.read_csv("iris.data")
```

角括弧で指定することで特定の列を取り出せる。

```
df['Day 2']
```

インデックス（属性名）と カラム名（列名）を指定してデータを取り出すには ix を使う。

```
df.ix['Date', 'Day 2']
```

番号で指定することも可能である。

```
df.ix[2,0]
```

Matplotlib

Python でグラフを描くのに広く使用されているライブラリが Matplotlib である。ランダムに配置された 50 個の点からなる散布図を描く例は以下である。

```
import numpy as np
import matplotlib.pyplot as plt
N = 50
x = np.random.rand(N)
y = np.random.rand(N)
colors = np.random.rand(N)
area = np.pi * (15 * np.random.rand(N))**2  # 0 to 15 point radiuses
plt.scatter(x, y, s=area, c=colors, alpha=0.5)
plt.show()
```

なお、Jupyter Notebook を使用する場合、上記を実行する前に事前に以下のコマンドを打ち込んでおくと、図が別ウィンドウではなく現在のブラウザ内に埋め込まれて表示されるようになるので試してみること。

```
%matplotlib inline
```

線形分離

linear_separation.py は直線によって 2 次元空間を分割するサンプルである。これを利用し、以下の問題に答えよ。

練習問題 1：
新規データ（テストデータ）に対し、その特徴量の値（模擬レポート成績と模擬テスト成績）に応じて合否を予測（合格と予測するなら 1、不合格と予測するなら 0 を出力）するプログラムを作成せよ。

練習問題 2：
前問のプログラムで予測値の計算をベクトルの内積を使った形に変更せよ。これには numpy の関数 matmul が使える。numpy 配列 a と b の内積は numpy.matmul(a,b) で求まる。matmul は matrix multiplication の略である。なお、import numpy as np という形でインポートした場合は numpy.matmul ではなく np.matmul と書くことになる。

練習問題 3：

新規データの真のクラス（合否）と予測結果を比較し、平均精度（正解だった割合）を求めるプログラムを作成せよ。

パーセプトロン

perceptron_surface.py はパーセプトロンの出力を 3 次元サーフェスプロットとして描いている。

練習問題 4：
perceptron_surface.py において、活性（シグモイド関数の引数）の計算をベクトルの内積 np.mutmul を使った形に変更せよ。np.meshgrid の戻り値は行列（すなわち 2 次元配列）のペア g0 と g1 なので、それらをペアリングした要素を取り出すのにリスト内包表記と zip を使う。得られたペアを e0 と e1 とすると、それにバイアス項のための 1 を繋げたベクトルは np.array([e0, e1, 1]))で作れる。結果は 2 次元リストになるので、これを np.array() によって 2 次元配列に変換したものを activation とすればよい。

練習問題 5：
前問のプログラムで重みベクトルを定数倍した時に出力の形状がどう変わるかを確認せよ。分離直線についてはどうか。

練習問題 6：
新規データ（テストデータ）に対し、その特徴量の値（模擬レポート成績と模擬テスト成績）に応じて合格確率を予測として出力するプログラムを作成せよ。

activation = data × おもみ
かくりつ値 = sigmoid(activation)

パーセプトロンによる多値分類

練習問題 7：
ソフトマックス関数を実装し、ベクトルを入力してその出力がどうなるかを確認せよ。
preceptron_surface.py にある sigmoid 関数を参考にするとよい。

練習問題 8：
行列 W = np.array([[5,3],[2,-1],[4,2]])とベクトル x = np.array([3,2])の積を np.matmul によって求め、それが手計算で行った結果と一致することを確認せよ。

練習問題 9：
行列の積とソフトマックス関数を使うことで、パーセプトロンによる多値分類を実行せよ。重み行列としては前問で作成した W = np.array([[5,3],[2,-1],[4,2]]) を使用せよ。

$$\begin{pmatrix} 5 & 3 \\ 2 & -1 \\ 4 & 2 \end{pmatrix}\begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 21 \\ 4 \\ 16 \end{pmatrix}$$

多層ニューラルネットワーク

練習問題 10：
ReLU を関数として実装せよ。

練習問題 11：
中間層を持つ多層ニューラルネットワークを実装せよ。活性化関数として ReLU を、出力関数としてはソフトマックス関数を使用せよ。

練習問題 12：
任意の重みを持つ多層ニューラルネットワークによって iris データに対してラベル（品種）の予測を行い、それを正解ラベルと比較することで平均精度を求めよ。

scikit-learn

scikit-learn は機械学習や統計に関するライブラリである。シェルから以下のようにインストールする。-U オプションを付けるのはすでに Anaconda の一部としてインストールされている scikit-learn をアンインストールし、最新のバージョンを入れるためである。容量オーバーでインストールできなかった場合は Anaconda に入っている scikit-learn を使えばよいが、機能は最新ではない。

```
pip install -U scikit-learn
```

以下のページに scikit-learn を使う多数のサンプルコードが公開されているので、調べてみるとよい。

```
http://scikit-learn.org/stable/auto_examples/index.html#general-examples
```

クラスタリング （clustering）

クラスタリングとは観測されたデータに対し、自然な分け方を見つけるタスクである。分類や回帰と異なり、クラスタリングでは独立変数と従属変数の区別は存在しない。そのため教師なし学習に分類される。

有名な Iris データセットに対してクラスタリングを行う例を示す。UCI Machine Learning Repository の以下の URL から Data Folder というリンクを辿り、iris.data をダウンロードしてくる。iris.names はデータに対する説明である。

```
http://archive.ics.uci.edu/ml/datasets/Iris
```

k-means 法によるクラスタリングがどのように行われるかを視覚的に示しているのが kmeans.ipynb である。観測値である各標本は小さい丸で、クラスタ中心は赤で縁取られた大きな丸で表されている。標本やクラスタの座標は属性（sepal length と petal width）の値を表している。

k-means 法ではクラスタ位置をランダムに定めた後（関数 initializeClusters が相当）、以下の操作が交互に行われる。

1. 各標本について、そこからもっとも近い重心を持つクラスタへの割り当てを行う。（関数 asignToClosest が相当）。
2. 各クラスタについて、それに属している標本点の重心に移動させる。（関数 updateClusterCenters が相当）。

これによって標本は近いもの同士がひとつのクラスタにまとめられていくようになる。

以下のサンプルコード clusIris.py では iris データの持つ 2 つの属性を targetAttr によって選び出し、クラスタ数 5 でクラスタリングを行っている。

実際のクラスタリングは clus_labels = kres.fit_predict(X) で行われている。これによって clus_labels は標本数次元のベクトルとなり、その成分は標本がどのクラスタに属しているかを示すラベルである。

得られた各クラスタはマーカーの色によって区別している。c=clus_labels と書いていることで各マーカーが clus_labels の成分の値に応じた色で塗られる。

また、各クラスタの重心の座標を clusCent に格納し、赤く大きなマーカーとして描いている。

ここでは属性 2 つだけを使って結果を視覚化しているが、クラスタリングにはすべての属性が利用されている。

線形回帰（linear regression）

独立変数と従属変数の値が共に知られている多数の標本（訓練データ）をもとに、独立変数から従属変数の値を求める（予測する）関数を学習するタスクを回帰と呼ぶ。線形回帰は回帰に対するもっともシンプルな方法のひとつであり、学習される関数が一次関数となるものである。一次関数であるため、データから求められる（学習される）のは各独立変数に対する重みと定数項（バイアス項）である。

GlobalTemperatures.csv は気温が過去 250 年間でどのように変化してきたかを示すデータであり、Kaggle の Climate Change データセットに含まれている。
https://www.kaggle.com/berkeleyearth/climate-change-earth-surface-temperature-data

linreg_temperature.py はそれに対して線形回帰を行っている。データ自体を表す標本の分布、回帰による内挿（interpolation）、回帰による内挿と外挿（extrapolation）の結果を図示している。

誤差二乗和（MSE）

予測値の入った predicted と実測値（正解値）の入った y の差（誤差）の分布をヒストグラム（h = plt.hist）を使ってプロットしてみる。

```
error = np.array(y - predicted)
bins = np.linspace(error.min(), error.max(), 20)
h = plt.hist(error, bins)
```

なお、bins はヒストグラムを描くために使われるビン（スロット）の区切りを並べたベクトルである。np.linspace を使うことで、error の最小値から最大値を 100 分割し、それをヒストグラムのビンの区切りとしている。

誤差の大きさは誤差二乗和（二乗誤差, mean squared error）を使って評価されることが多い。これは以下のように求められる。

```
mse = (error ** 2).mean()
```

練習問題 13：
GlobalTemperatures.csv に対する線形回帰について誤差二乗和を求めてみよ。

numpy の sum を使う

多項式回帰（polynomial regression）

UCI Machine Learning データセットにある abalone（アワビ）データセットに対して多項式回帰を行う例を示す。polyAbalone.py では abalone データのうち、長さ（length）から全体重量（whole weight）を予測する二次の多項式回帰を行っている。

練習問題 14：
abalone データに対し、長さから全体重量の予測を線形回帰と 2 次の多項式回帰で行い、誤差二乗和を線形回帰の場合と比較せよ。

ガウシアンプロセス回帰（Gaussian process regression）

ガウシアンプロセス回帰（GP 回帰）はカーネル法のひとつであり、独立変数の値がテストデータに近い訓練データの従属変数の値を使って予測を行うというものである。GP 回帰を使うと訓練データ周辺を通って結んでいく曲線が得られる。

gpAirQuality.py は UCI ML Repository にある Air Quality Data に対し、気温の変化を曲線で補完して描くものである。1 時間ごとのデータが入っているため、それを独立変数とし、気温を従属変数としている。

ニューラルネットワークによる回帰

練習問題 15：
2 値分類を行う（1 層の）パーセプトロンからその出力関数であるシグモイド関数を取り除くと、浮動小数点数（実数）を出力するようになるため、回帰に使用できる。そのようなニューラルネットワークを表す関数を作成せよ。

練習問題 16：
Kaggle の GlobalTemperatures.csv について、前問で作成した回帰器を使って年から気温の予測を行うようにし、それと正解値の間の差の二乗の平均を求めることで誤差を評価せよ。重み（回帰直線の傾きと切片）はデータの散布図を見ながら適当に決めるとよい。

勾配降下法

練習問題 17：
誤差二乗和を損失関数とした勾配降下法によって前問のプログラムのパラメータである重み行列を更新し、損失がどのように変わっていくかを確認せよ。

クロスエントロピー

練習問題 18：
多値分類を行うパーセプトロンについて、ソフトマックス関数に基づくクロスエントロピーの勾配を求める関数を作成せよ。

練習問題 19：
前問のニューラルネットワークについて、ソフトマックス関数に基づくクロスエントロピーを損失関数として計算する関数を作成せよ。

練習問題 18：
多値分類を行うパーセプトロンについて、ソフトマックス関数に基づくクロスエントロピーの勾配を求める関数を作成せよ。

練習問題 19：
前問のニューラルネットワークについて、ソフトマックス関数に基づくクロスエントロピーを損失関数として計算する関数を作成せよ。

練習問題 20：
前問のニューラルネットワークについて、勾配降下法に基づく重み行列の更新を行うようにせよ。iris データについての学習に伴い、損失関数と精度がどのように変化していくかを視覚化せよ。

練習問題 21：
多層ニュラルネットワークのパラメータを誤差逆伝播法によって学習し、精度の変化を求めよ。