# ○[AI自習帳 2]　アメンボの「手書き数字(文字)」を AI プログラムで識別する　amenbo the 3rd

2018.10.07

◆構成要素別プログラム; AI プログラム(例)

本稿では、アメンボが手書した「数字(文字)」を、「scikit-learn」を利用した AI 手法により
判読(識別)する Pyhton プログラム例を解説します。
一見すると、為替の予測手法と無関係に見えますが、実は AI 手法としては共通であり、
読者が、本稿を核(ベース)として為替予測に必要な AI 手法を身に着ける一助になると考えます。
(Python コード(文法)の解説はしません、WEB 上に豊富にある情報を参照ください)

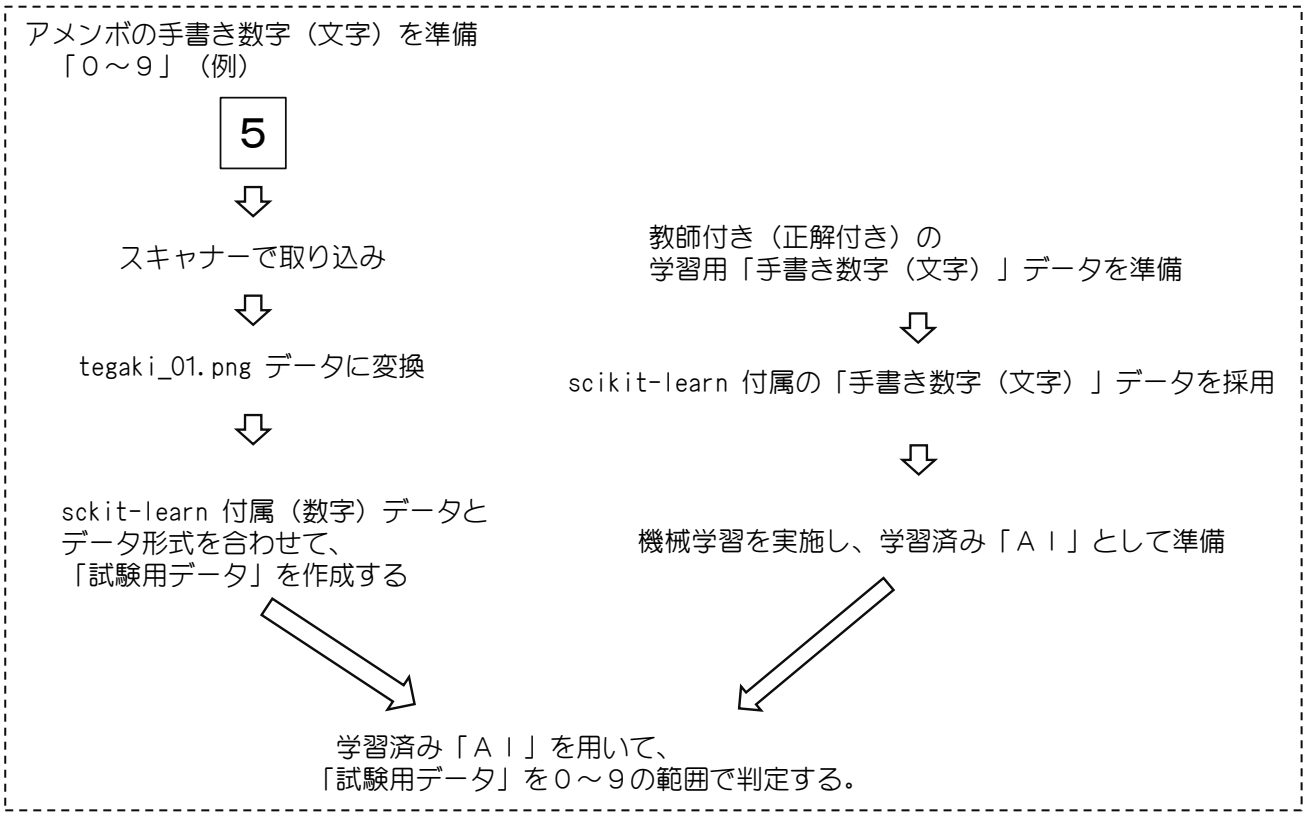
※本稿は、『[自習 3]「手書き数字」の、Python プログラムによる処理例』とセットで読んでください。
本稿ではアメンボの「手書き数字(文字)」データを、「scikit-learn」付属の「手書き数字(文字)」を
学習用データとして利用し、とサラットと比較(推測)していますが、「アメンボのデータ」を
「学習用データ」と比較可能にするには、どの様に整形していけば良いのかが判らず苦労しました。
(上記の最も混乱する部分が理解できる様に、[別稿 3]では可成り詳細にデータで解説しています)

目次



## 1. 本稿で実現すること

※アメンボの手書き数字「tegaki_01.png」データを、下記の手順で判定します。



## 2. Python 用モジュール追加

※先ず、Python 側の準備が必要となります、下記に従ってライブラリをインストールしてください。

本稿で使う Python 用のライブラリは、

| | | |
|---|---|---|
| 「numpy」 | ； | 数値計算ライブラリ |
| 「scikit-learn」 | ； | AI 用「機械学習ライブラリ」 |
| 「Pillow」 | ； | 画像処理ライブラリ |

です。

※ただし、py コード上で呼出すときは、

| | | |
|---|---|---|
| numpy | ⇒ | import numpy |
| scikit-learn | ⇒ | from sklearn import ** |
| Pillow | ⇒ | from PIL import ** |

と記述します。

※下記の「pip」(または pip3)コマンドを使いインストールします。

```
pip install ライブラリ名
```

例; pip install scikit-learn

インストールした結果は、「pip list」で確認出来ます。

※機械学習用ライブラリとしては「TesorFlow」も有名ですが、アメンボは未だ使った事がありません。

## 3. データの準備

※機械学習用データと、識別(ターゲット)用データを準備します。

(1)学習用の「手書き文字データ」の準備

・scikit-learn に付属している「手書き文字データ・セット」を利用します。

py コード内の、「from sklearn import datasets」により使用可能となります。

(2)アメンボの手書き「文字データ」の準備

・アメンボは、紙に書いた文字をスキャナーで取り込んで、「tegaki_01.png」として保存しました。

このデータは下記「tegaki_hantei_SVC_02.py」と同じフォルダー内に置きます。

## 4. Python コード

※アメンボが引っかかった内容を「コメント」としてコード中に埋め込みました、
　また、実行には不要ではあるのですが、試してみたコードも「＃;コメント」で一部を残してあります。
　（やや、コードが読みずらくなりましたが！、きっと参考になると思います）

[tegaki_hantei_SVC_02.py]　••　本稿に添付

```
# -*- coding: utf-8 -*-
"""
Created on Thu May 17 23:32:34 2018
[tegaki_hantei_SVC_02.py]
@author: kenken
"""
#-----------------------------------
#from sklearn import datasets,cross_validation,svm,metrics#「cross_validation」は 0.20 から対応がなくなった！
#確認;今回入れた「scikit-learn」は「0.19.1」だ！けど、同じエラーが出た
from sklearn import datasets,model_selection,svm, metrics
import numpy as np
from PIL import Image
#from matplotlib import pylab as plt
#-----------------------------------
print("step1")
# scikit-learn 付属の「手書き数字(文字)データセット」を準備
digits=datasets.load_digits()
#　テスト用「手書き数字」画像の取込みと加工
image=Image.open("tegaki_01.png").convert('L')
image=image.resize((8,8),Image.LANCZOS)
#image=image.resize((8,8),Image.ANTIALIAS)
img=np.asarray(image,dtype=float)
img=np.floor(16-16*(img/256))
img=img.flatten()
#　学習用データの準備
data_train=digits.data
label_train=digits.target
# テストデータを設定
data_test=img
label_test=[5]

print("step2")
#
print("digits.data[5]=",data_train[5])
print("data_test= ",data_test)

print("digits.data[5]=",digits.data[5].shape)
print("data_test= ",data_test.shape)
```

・アメンボの手書き数字（文字）データを scikit-learn 付属の「手書き数字データセット」と同じデータ形式にするための処理。
・上記「データ形式」の詳細は、別紙［自習３］を参照ください。

←「学習用」データとは、
　scikit-learn 附属の手書き数字データセット

←「テスト用」データとは、
　アメンボの手書き数字のこと。

←［５］番目のデータを書出し
←「テスト用」データを書出し

←「data[5]」と「data_test」行列の大きさを書出し

```
#   機械学習の実施
clf=svm.SVC(gamma=0.001)
clf.fit(data_train,label_train)
#

print("step3")

print("---------------予測結果----------------")
#predict=clf.predict(data_test.reshape(1,-1))
# 「data_test.reshape(1,-1)」により、2次元配列データの「1行分」の形式にしておく
predict=clf.predict(data_test.reshape(1,-1))
print("テストラベル",label_test)
print("解析結果",predict)

print("------------データの確認-----------------------")
print("data_test.reshape(1,-1)=",data_test.reshape(1,-1))
print("------------最後に！-----------------------")
#   正解率を計算
ac_score=metrics.accuracy_score(label_test,predict)
print("正解率",ac_score*100,"%")
print("ようやく完了")


'''
--------------  記録(コメント化)  -----------------
#predict=clf.predict(data_test)
#上記コードでの出現エラー;
 Expected 2D array, got 1D array instead:
array=[0. 0. 2. ... 0. 0. 0.].
Reshape your data either using array.reshape(-1, 1)
 if your data has a single feature or array.reshape(1, -1)
 if it contains a single sample.
！！data_test は1個(シングル・データ)のために、この処理が必要なのか！！？？

♦参考(重要);
data_train[5]
Out[28]: array([ 0.,  0., 12., ..., 10.,  0.,  0.])

data_test
Out[29]: array([0., 0., 2., ..., 0., 0., 0.])

data_test.reshape(1,-1)
Out[30]: array([[0., 0., 2., ..., 0., 0., 0.]])
2次元配列になってる！
もしかして、data_train[5]は、2次元配列データの内の、1行分データなので、
テストデータも、2次元配列データの1つ、と言う形式にしておく必要があるのでは？！
••••••••
この記法は，shape に (1, -1) や (-1, 1) を指定すると，それぞれ 2 次元の横ベクトルや縦ベクトル
```

←SVC（サポート・ベクタ・マシン）アルゴリズムを
使って（教師付き）機会学習を実施、
「clf」が学習済みＡＩとなる。

←学習済みのＡＩにより、
テスト・データ（data_test)を判定する。

←正解率を計算
「label_test]に正解を入れてある場合に
計算可能。

```
を簡便に作ることができます.
In [40]: np.arange(6).reshape((1, -1))
Out[40]: array([[0, 1, 2, 3, 4, 5]])
In [41]: np.arange(6).reshape((-1, 1))
Out[41]:
array([[0],
       [1],
       [2],
       [3],
       [4],
       [5]])
-------------------------------------------------
'''
```

### 5. 実行結果

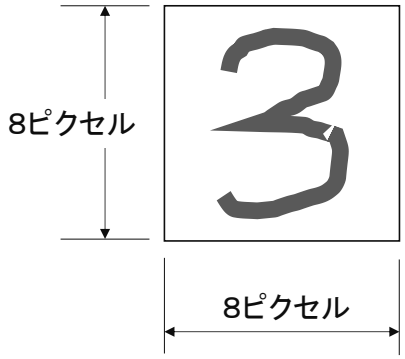
コンソールに出力された内容（例）を示します；

```
step1
step2
digits.data[5]= [ 0.  0. 12. ... 10.  0.  0.]
data_test=  [0. 0. 2. ... 0. 0. 0.]
digits.data[5]= (64,)
data_test=  (64,)
step3
--------------予測結果---------------
テストラベル [5]
解析結果 [5]
-----------データの確認-----------------------
data_test.reshape(1,-1)= [[0. 0. 2. ... 0. 0. 0.]]
-----------最後に！-----------------------
正解率 100.0 %
ようやく完了
```

＜補足＞　‥　詳細は別紙[自習 3]を参照ください

１．本稿では、手書き数字（文字）の「教師付き学習用データ」として、
「scikit-learn」付属の「手書き数字データ」を使用しています。

（１）「scikit-learn」付属の「手書き数字データ（digit)」のイメージ




・　８×８　ピクセル＝６４　ピクセル
・　色深度；　４ ビット（０～１６）
０＝黒
１＝白
０～１６範囲のグレースケールで表現

（２）また、このデータ（digit）の格納形式は、

１つの「手書き数字（文字）データ」＝［６４行、１列］

です！

２．アメンボの「手書き数字」データは、
紙上に書いた数字を「スキャナー」で読み取ったものを使っています。

以　上