# -\*- coding: utf-8 -\*-

from sklearn import datasets

from sklearn.neural\_network import MLPClassifier

import numpy as np

import PIL

from PIL import Image, ImageTk, ImageDraw, ImageFilter

try:

 import Tkinter as tk

except ImportError:

 import tkinter as tk # for Python 3

# 手書き数字のデータをロードし、変数digitsに格納

digits = datasets.load\_digits()

# 特徴量のセットを変数Xに、ターゲットを変数yに格納

X = digits.data

y = digits.target

# 分類用に多層ニューラルネットワークを用意

# 毎回異なる乱数を利用

clf = MLPClassifier(hidden\_layer\_sizes=(100, ), max\_iter=1000, tol=0.0001, random\_state=None)

# ニューラルネットワークの学習

print('学習中…')

clf.fit(X, y)

##### 以下はGUIアプリケーション用の内容

class Application(tk.Frame):

 # 初期化用関数

 def \_\_init\_\_(self, master=None):

 tk.Frame.\_\_init\_\_(self, master)

 self.pack()

 self.w = 200 # 一つの描画領域の幅

 self.h = 200 # 一つの描画領域の高さ

 self.d = 30 # マウスで描画する際の一点の直径

 self.r = 20 # 旧バージョン用のフィルタ半径

 self.create\_widgets()

 # 様々なGUI部品を構築

 def create\_widgets(self):

 w = self.w

 h = self.h

 # マウスで描画するための領域

 self.canvas = tk.Canvas(self, width=w, height=h, bg='white')

 # self.canvasを左端に配置

 self.canvas.grid(row=0, column=0)

 # self.canvasでマウスが動いた際にself.draw\_digitが実行されるよう設定

 self.canvas.bind('<Button1-Motion>', self.draw\_digit)

 # 認識用に用いる画像(self.canvasと共通化不能)

 self.img = Image.new('L', (w, h), color=255)

 # self.img上に描画するために必要なdraw

 self.draw = ImageDraw.Draw(self.img)

 # 認識用に self.img を加工して表示するキャンバス

 self.recog\_canvas = tk.Canvas(self, width=w, height=h, bg='white')

 # self.recog\_canvas上に描画するための画像を生成して関連付け

 self.recog\_img = tk.PhotoImage(width=w, height=h)

 self.recog\_canvas.create\_image((w/2,h/2), image=self.recog\_img, state='normal')

 self.recog\_canvas.image = self.recog\_img

 # self.canvasを中央に配置

 self.recog\_canvas.grid(row=0, column=1)

 # 結果表示用のキャンバス

 self.digit\_canvas = tk.Canvas(self, width=w, height=h, bg='gray')

 # self.digit\_canvasを右端に配置

 self.digit\_canvas.grid(row=0, column=2)

 # 認識ボタン

 self.recog\_btn = tk.Button(self, text='認識', command=self.recog)

 # 認識ボタンを下段左に配置

 self.recog\_btn.grid(row=1, column=0)

 # クリアボタン

 self.clear\_btn = tk.Button(self, text='クリア', command=self.clear)

 # クリアボタンを下段右に配置

 self.clear\_btn.grid(row=1, column=2)

 # 認識ボタンが押されたときに実行される関数

 def recog(self):

 w = self.w

 h = self.h

 # 描画されたイメージを8x8のrecog\_imgに縮小

 try:

 s = Image.PILLOW\_VERSION

 except AttributeError:

 s = PIL.\_\_version\_\_

 majorVersion = int(s[0])

 if majorVersion>=3: # for Stretch

 recog\_img = self.img

 recog\_img = recog\_img.resize(size=(8,8), resample=Image.BICUBIC)

 else: # for Jessie

 recog\_img = self.img.filter(ImageFilter.GaussianBlur(radius=self.r))

 recog\_img = recog\_img.resize(size=(8,8), resample=Image.BILINEAR)

 # (8,8)のrecog\_imgを描画用に拡大

 if majorVersion>=7:

 recog\_img\_large = recog\_img.resize(size=(w, h), resample=0)

 else:

 recog\_img\_large = recog\_img.resize(size=(w, h))

 self.recog\_img = ImageTk.PhotoImage(image=recog\_img\_large)

 self.recog\_canvas.create\_image((w/2, h/2), image=self.recog\_img, state='normal')

 self.recog\_canvas.image = self.recog\_img

 # 8x8の画像を機械学習用に加工

 Ximg = np.asarray(recog\_img, dtype=int)

 # 最大値を16に

 Ximg = 16\*Ximg/255.0

 # 整数に丸める

 Ximg = Ximg.astype(int)

 # 白黒反転

 Ximg = 16-Ximg

 # 機械学習用データの完成

 X = np.array([Ximg.flatten()])

 # 学習済みのニューラルネットワークから予測を取得

 y = clf.predict(X)

 # 予測結果を右の領域の表示

 self.digit\_canvas.create\_text(w/2, h/2, text = '{0:d}'.format(y[0]), fill='white', font = ('FixedSys', int(w/2)))

 # クリアボタンが押されたときに実行される関数

 def clear(self):

 w = self.w

 h = self.h

 # 左の描画領域をクリア

 self.canvas.delete('all')

 self.draw.rectangle(xy=[0,0,w,h],

 outline='white', fill='white')

 # 中央の認識領域をクリア

 self.recog\_canvas.delete('all')

 self.recog\_img = tk.PhotoImage(width=w, height=h)

 self.recog\_canvas.create\_image((w/2,h/2), image=self.recog\_img, state='normal')

 self.recog\_canvas.image = self.recog\_img

 # 右の結果表示領域をクリア

 self.digit\_canvas.delete('all')

 # 左側の描画領域でマウスが動いたときに呼ばれる関数

 def draw\_digit(self,event):

 d = self.d

 x = event.x

 y = event.y

 # 描画領域に黒丸を描画

 id=self.canvas.create\_oval(x-d/2, y-d/2, x+d/2, y+d/2)

 self.canvas.itemconfigure(id, fill='black')

 # 認識用の画像に黒丸を描画

 self.draw.ellipse(xy=[x-d/2, y-d/2, x+d/2, y+d/2], fill=0, outline=0)

root = tk.Tk()

app = Application(master=root)

app.master.title('数字認識（ニューラルネットワーク版）')

app.mainloop()