

実行結果

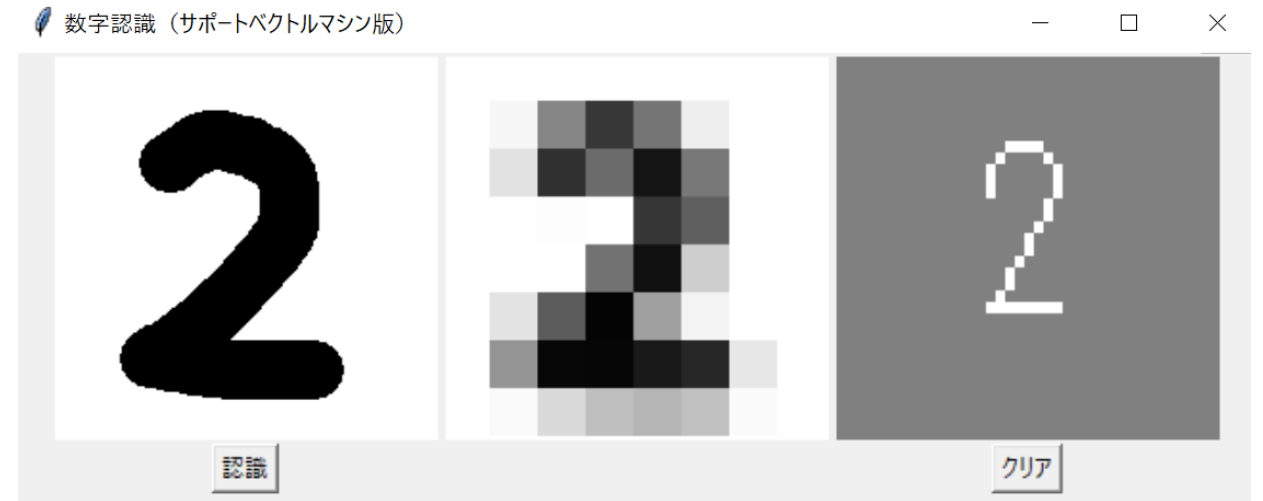
①Runするとこの画面が表示



②マウスで「2」を描く



③中: 手書きデータを濃淡データに換算
右: 認識して描いた文字



PIL(Pillow): 画像処理ライブラリー

画像読み込み機能を追加する

```
1 # -*- coding: utf-8 -*-
2 from sklearn import datasets, svm
3 import numpy as np
4 import PIL
5 from PIL import Image, ImageTk, ImageDraw, ImageFilter
6 try:
7     import Tkinter as tk
8 except ImportError:
9     import tkinter as tk # for Python 3
10
11 # 手書き数字のデータをロードし、変数digitsに格納
12 digits = datasets.load_digits()
13
14 # 特徴量のセットを変数Xに、ターゲットを変数yに格納
15 X = digits.data
16 y = digits.target
17
18 # 分類用にサポートベクトルマシンを用意
19 clf = svm.SVC(C=1.0, kernel='linear')
20 # データに最適化
21 clf.fit(X, y)
22
23 ##### 以下はGUIアプリケーション用の内容
24
25 class Application(tk.Frame):
26     # 初期化用関数
27     def __init__(self, master=None):
28         tk.Frame.__init__(self, master)
29         self.pack()
30         self.w = 200 # 一つの描画領域の幅
31         self.h = 200 # 一つの描画領域の高さ
32         self.d = 30 # マウスで描画する際の一点の直径
33         self.r = 20 # 旧バージョン用のフィルタ半径
34         self.create_widgets()
```

```
36 # 様々なGUI部品を構築
37 def create_widgets(self):
38     w = self.w
39     h = self.h
40     # マウスで描画するための領域
41     self.canvas = tk.Canvas(self, width=w, height=h, bg='white')
42     # self.canvasを左端に配置
43     self.canvas.grid(row=0, column=0)
44     # 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
45     self.canvas.bind('<Button1-Motion>', self.draw_digit)
46
47     # 認識用に用いる画像(self.canvasと共通化不能)
48     self.img = Image.new('L', (w, h), color=255)
49     # self.img上に描画するために必要なdraw
50     self.draw = ImageDraw.Draw(self.img)
51
52     # 認識用に self.img を加工して表示するキャンバス
53     self.recog_canvas = tk.Canvas(self, width=w, height=h, bg='white')
54     # self.recog_canvas上に描画するための画像を生成して関連付け
55     self.recog_img = tk.PhotoImage(width=w, height=h)
56     self.recog_canvas.create_image((w/2, h/2), image=self.recog_img, state='normal')
57     self.recog_canvas.image = self.recog_img
58     # self.canvasを中央に配置
59     self.recog_canvas.grid(row=0, column=1)
60
61     # 結果表示用のキャンバス
62     self.digit_canvas = tk.Canvas(self, width=w, height=h, bg='gray')
63     # self.digit_canvasを右端に配置
64     self.digit_canvas.grid(row=0, column=2)
65
66     # 認識ボタン
67     self.recog_btn = tk.Button(self, text='認識', command=self.recog)
68     # 認識ボタンを下段左に配置
69     self.recog_btn.grid(row=1, column=0)
70     # クリアボタン
71     self.clear_btn = tk.Button(self, text='クリア', command=self.clear)
72     # クリアボタンを下段右に配置
73     self.clear_btn.grid(row=1, column=2)
74
```

```

75 # 認識ボタンが押されたときに実行される関数
76 def recog(self):
77     w = self.w
78     h = self.h
79     # 描画されたイメージを8x8のrecog_imgに縮小
80     try:
81         s = Image.PILLOW_VERSION
82     except AttributeError:
83         s = PIL.__version__
84     majorVersion = int(s[0])
85     if majorVersion >= 3: # for Stretch
86         recog_img = self.img
87         recog_img = recog_img.resize(size=(8, 8), resample=Image.BICUBIC)
88     else: # for Jessie
89         recog_img = self.img.filter(ImageFilter.GaussianBlur(radius=self.r))
90         recog_img = recog_img.resize(size=(8, 8), resample=Image.BILINEAR)
91
92     # (8, 8)のrecog_imgを描画用に拡大
93     if majorVersion >= 7:
94         recog_img_large = recog_img.resize(size=(w, h), resample=0)
95     else:
96         recog_img_large = recog_img.resize(size=(w, h))
97     self.recog_img = ImageTk.PhotoImage(image=recog_img_large)
98     self.recog_canvas.create_image((w/2, h/2), image=self.recog_img, state='normal')
99     self.recog_canvas.image = self.recog_img
100
101     # 8x8の画像を機械学習用に加工
102     Ximg = np.asarray(recog_img, dtype=int)
103     # 最大値を16に
104     Ximg = 16*Ximg/255.0
105     # 整数に丸める
106     Ximg = Ximg.astype(int)
107     # 白黒反転
108     Ximg = 16-Ximg
109     # 機械学習用データの完成
110     X = np.array([Ximg.flatten()])
111     # 最適化済みのサポートベクトルマシンから予測を取得
112     y = clf.predict(X)
113     # 予測結果を右の領域の表示
114     self.digit_canvas.create_text(w/2, h/2, text = '{0:d}'.format(y[0]), fill='white', font = ('FixedSys', int(w/2)))
115

```

Numpy配列は0～255
濃淡は0～15の16

黒に近づくほど0に近づくように変換

64次元のベクトルが1個のNumpy配列

```
116 # クリアボタンが押されたときに実行される関数
117 def clear(self):
118     w = self.w
119     h = self.h
120     # 左の描画領域をクリア
121     self.canvas.delete('all')
122     self.draw.rectangle(xy=[0, 0, w, h],
123                         outline='white', fill='white')
124     # 中央の認識領域をクリア
125     self.recog_canvas.delete('all')
126     self.recog_img = tk.PhotoImage(width=w, height=h)
127     self.recog_canvas.create_image((w/2, h/2), image=self.recog_img, state='normal')
128     self.recog_canvas.image = self.recog_img
129     # 右の結果表示領域をクリア
130     self.digit_canvas.delete('all')
131
132 # 左側の描画領域でマウスが動いたときに呼ばれる関数
133 def draw_digit(self, event):
134     d = self.d
135     x = event.x
136     y = event.y
137     # 描画領域に黒丸を描画
138     id=self.canvas.create_oval(x-d/2, y-d/2, x+d/2, y+d/2)
139     self.canvas.itemconfigure(id, fill='black')
140     # 認識用の画像に黒丸を描画
141     self.draw.ellipse(xy=[x-d/2, y-d/2, x+d/2, y+d/2], fill=0, outline=0)
142
143 root = tk.Tk()
144 app = Application(master=root)
145 app.master.title('数字認識 (サポートベクトルマシン版)')
146 app.mainloop()
```

サポートベクトルマシン



数字認識（サポートベクトルマシン版）



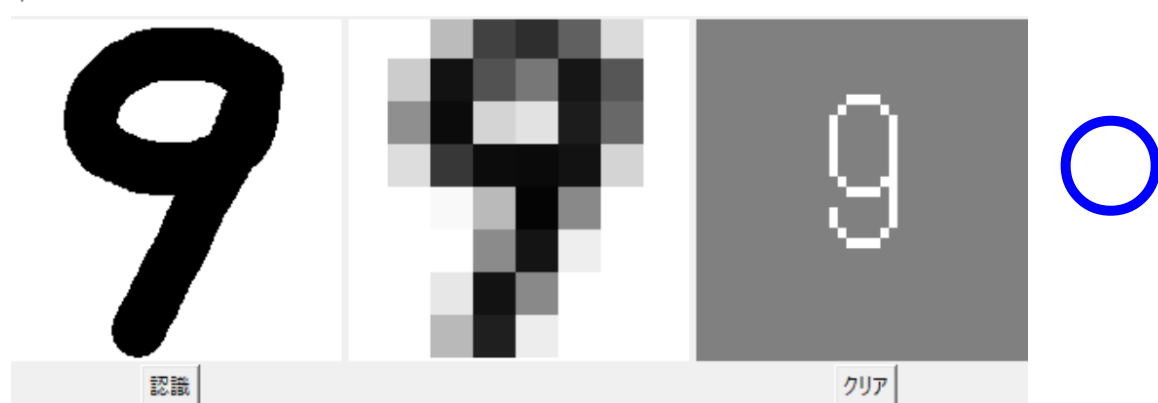
数字認識（サポートベクトルマシン版）



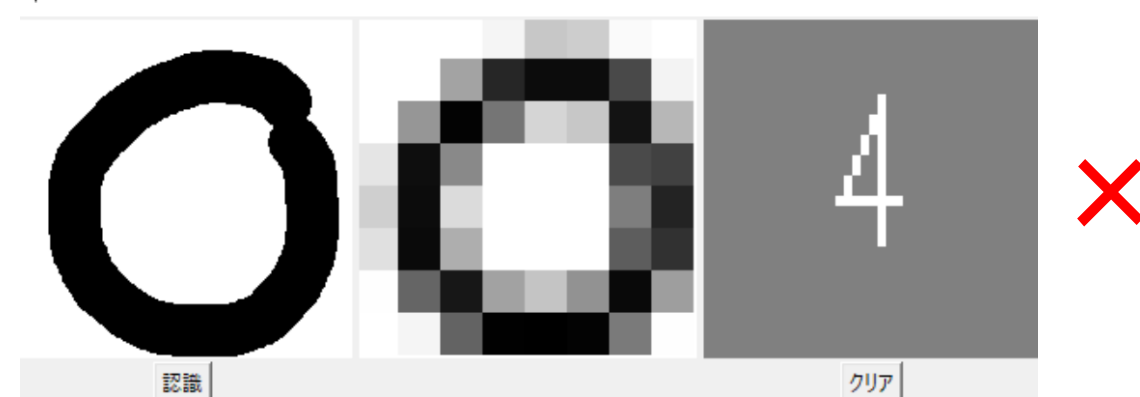
ニューラルネットワーク



数字認識（ニューラルネットワーク版）



数字認識（ニューラルネットワーク版）



```
1 # -*- coding: utf-8 -*-
2 from sklearn import datasets
3 from sklearn.neural_network import MLPClassifier
4 import numpy as np
5 import PIL
6 from PIL import Image, ImageTk, ImageDraw, ImageFilter
7 try:
8     import Tkinter as tk
9 except ImportError:
10     import tkinter as tk # for Python 3
11
12 # 手書き数字のデータをロードし、変数digitsに格納
13 digits = datasets.load_digits()
14
15 # 特徴量のセットを変数Xに、ターゲットを変数yに格納
16 X = digits.data
17 y = digits.target
18
19 # 分類用に多層ニューラルネットワークを用意
20 # 毎回異なる乱数を利用
21 clf = MLPClassifier(hidden_layer_sizes=(100, ), max_iter=1000, tol=0.0001, random_state=None)
22 # ニューラルネットワークの学習
23 print('学習中...')
24 clf.fit(X, y)
25
26 ##### 以下はGUIアプリケーション用の内容
27
28 class Application(tk.Frame):
29     # 初期化関数
30     def __init__(self, master=None):
31         tk.Frame.__init__(self, master)
32         self.pack()
33         self.w = 200 # 一つの描画領域の幅
34         self.h = 200 # 一つの描画領域の高さ
35         self.d = 30 # マウスで描画する際の一点の直径
36         self.r = 20 # 旧バージョン用のフィルタ半径
37         self.create_widgets()
38
```



```
39 # 様々なGUI部品を構築
40 def create_widgets(self):
41     w = self.w
42     h = self.h
43     # マウスで描画するための領域
44     self.canvas = tk.Canvas(self, width=w, height=h, bg='white')
45     # self.canvasを左端に配置
46     self.canvas.grid(row=0, column=0)
47     # 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
48     self.canvas.bind('<Button1-Motion>', self.draw_digit)
49
50     # 認識用に用いる画像(self.canvasと共通化不能)
51     self.img = Image.new('L', (w, h), color=255)
52     # self.img上に描画するために必要なdraw
53     self.draw = ImageDraw.Draw(self.img)
54
55     # 認識用に self.img を加工して表示するキャンバス
56     self.recog_canvas = tk.Canvas(self, width=w, height=h, bg='white')
57     # self.recog_canvas上に描画するための画像を生成して関連付け
58     self.recog_img = tk.PhotoImage(width=w, height=h)
59     self.recog_canvas.create_image((w/2, h/2), image=self.recog_img, state='normal')
60     self.recog_canvas.image = self.recog_img
61     # self.canvasを中央に配置
62     self.recog_canvas.grid(row=0, column=1)
63
64     # 結果表示用のキャンバス
65     self.digit_canvas = tk.Canvas(self, width=w, height=h, bg='gray')
66     # self.digit_canvasを右端に配置
67     self.digit_canvas.grid(row=0, column=2)
68
69     # 認識ボタン
70     self.recog_btn = tk.Button(self, text='認識', command=self.recog)
71     # 認識ボタンを下段左に配置
72     self.recog_btn.grid(row=1, column=0)
73     # クリアボタン
74     self.clear_btn = tk.Button(self, text='クリア', command=self.clear)
75     # クリアボタンを下段右に配置
76     self.clear_btn.grid(row=1, column=2)
77
```

```

78 # 認識ボタンが押されたときに実行される関数
79 def recog(self):
80     w = self.w
81     h = self.h
82     # 描画されたイメージを8x8のrecog_imgに縮小
83     try:
84         s = Image.PILLOW_VERSION
85     except AttributeError:
86         s = PIL.__version__
87     majorVersion = int(s[0])
88     if majorVersion>=3: # for Stretch
89         recog_img = self.img
90         recog_img = recog_img.resize(size=(8, 8), resample=Image.BICUBIC)
91     else: # for Jessie
92         recog_img = self.img.filter(ImageFilter.GaussianBlur(radius=self.r))
93         recog_img = recog_img.resize(size=(8, 8), resample=Image.BILINEAR)
94
95     # (8, 8)のrecog_imgを描画用に拡大
96     if majorVersion>=7:
97         recog_img_large = recog_img.resize(size=(w, h), resample=0)
98     else:
99         recog_img_large = recog_img.resize(size=(w, h))
100     self.recog_img = ImageTk.PhotoImage(image=recog_img_large)
101     self.recog_canvas.create_image((w/2, h/2), image=self.recog_img, state='normal')
102     self.recog_canvas.image = self.recog_img
103
104     # 8x8の画像を機械学習用に加工
105     Ximg = np.asarray(recog_img, dtype=int)
106     # 最大値を16に
107     Ximg = 16*Ximg/255.0
108     # 整数に丸める
109     Ximg = Ximg.astype(int)
110     # 白黒反転
111     Ximg = 16-Ximg
112     # 機械学習用データの完成
113     X = np.array([Ximg.flatten()])
114     # 学習済みのニューラルネットワークから予測を取得
115     y = clf.predict(X)
116     # 予測結果を右の領域の表示
117     self.digit_canvas.create_text(w/2, h/2, text = '{0:d}'.format(y[0]), fill='white', font = ('FixedSys', int(w/2)))
118

```



```

119 # クリアボタンが押されたときに実行される関数
120 def clear(self):
121     w = self.w
122     h = self.h
123     # 左の描画領域をクリア
124     self.canvas.delete('all')
125     self.draw.rectangle(xy=[0, 0, w, h],
126                         outline='white', fill='white')
127     # 中央の認識領域をクリア
128     self.recog_canvas.delete('all')
129     self.recog_img = tk.PhotoImage(width=w, height=h)
130     self.recog_canvas.create_image((w/2, h/2), image=self.recog_img, state='normal')
131     self.recog_canvas.image = self.recog_img
132     # 右の結果表示領域をクリア
133     self.digit_canvas.delete('all')
134
135 # 左側の描画領域でマウスが動いたときに呼ばれる関数
136 def draw_digit(self, event):
137     d = self.d
138     x = event.x
139     y = event.y
140     # 描画領域に黒丸を描画
141     id=self.canvas.create_oval(x-d/2, y-d/2, x+d/2, y+d/2)
142     self.canvas.itemconfigure(id, fill='black')
143     # 認識用の画像に黒丸を描画
144     self.draw.ellipse(xy=[x-d/2, y-d/2, x+d/2, y+d/2], fill=0, outline=0)
145
146 root = tk.Tk()
147 app = Application(master=root)
148 app.master.title('数字認識 (ニューラルネットワーク版)')
149 app.mainloop()

```