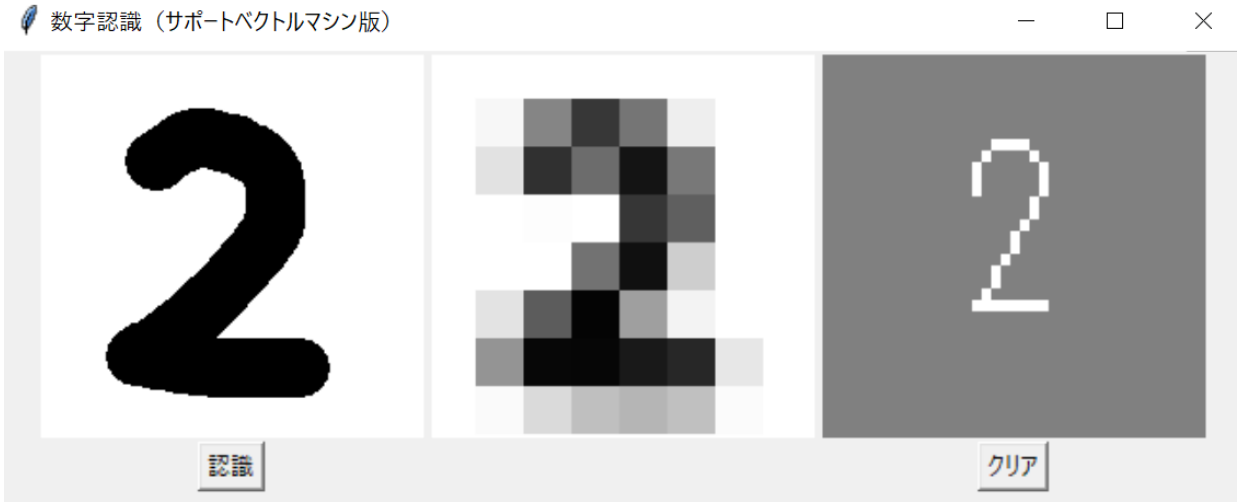
実行結果

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

```python
# -*- coding: utf-8 -*-
from sklearn import datasets, svm
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

# 分類用にサポートベクトルマシンを用意
clf = svm.SVC(C=1.0, kernel='linear')
# データに最適化
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)
```

```
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
```

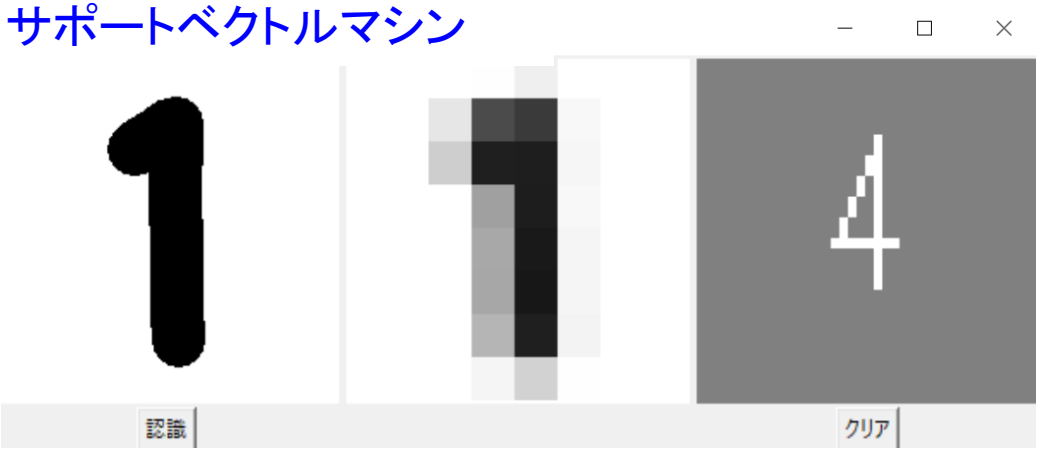
Numpy配列は0～255
濃淡は0～15の16

黒に近づくほど0に近づくように変換

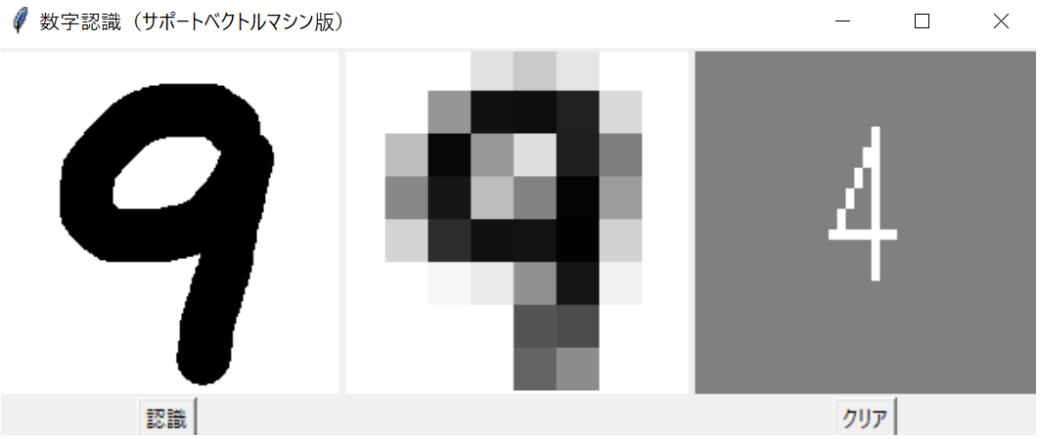
64次元のベクトルが1個のNumpy配列

```
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()
```

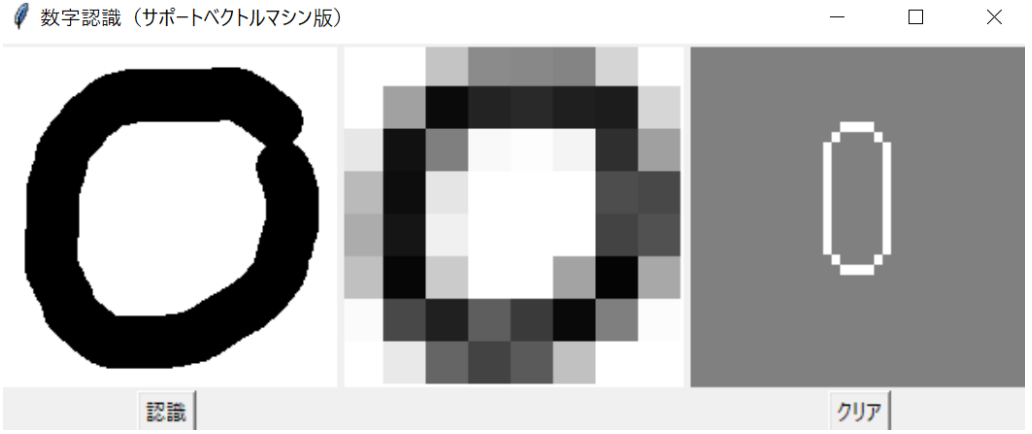
サポートベクトルマシン



×



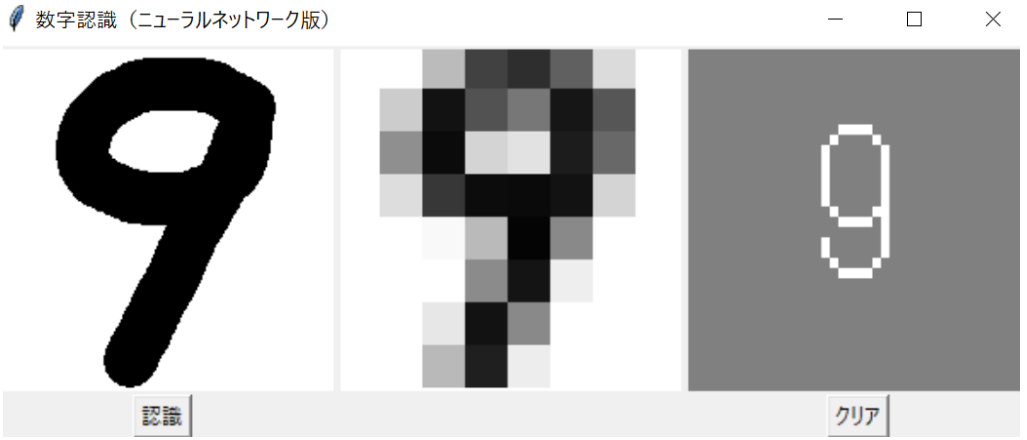
×



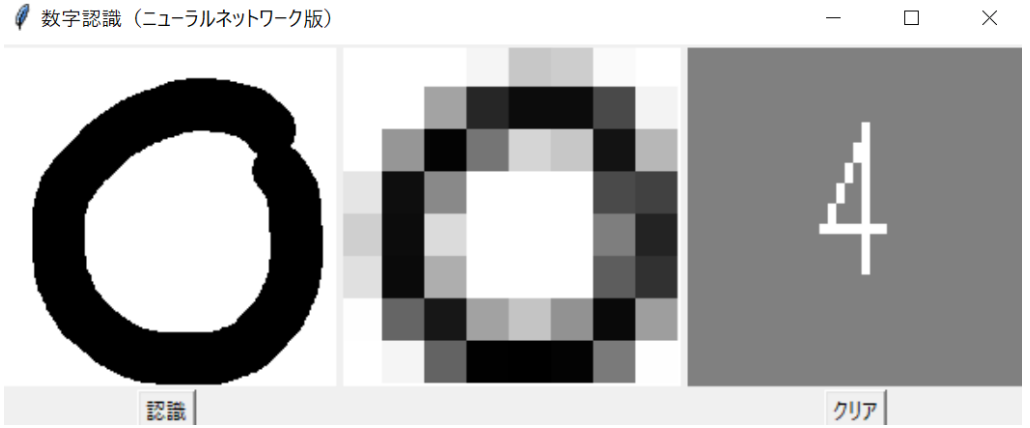
○

ニューラルネットワーク

○



○



×

```python
# -*- 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)))

```

```
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()
```