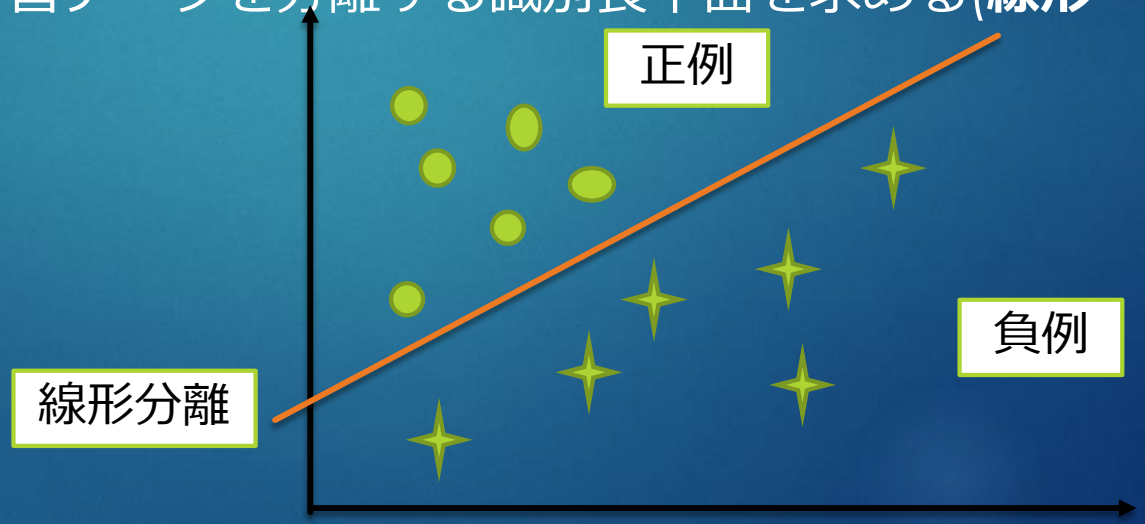
SVM(サポートベクターマシーン)

GROUP2 19K0016 高橋湧汰

Chapter1:SVMの概略

Support Vector Machineとは？

- クラスを明確に分ける境界線を引くための手法の事。
- 教師あり学習で2クラス(正例,負例)の識別を行う。
  - ・事前に学習データと正解ラベルが与えられ、それに基づいて学習を行う。
  - ・未知のデータに対しても分類が行える。
- SVMでは、特徴空間上で学習データを分離する識別長平面を求める(**線形分離**)



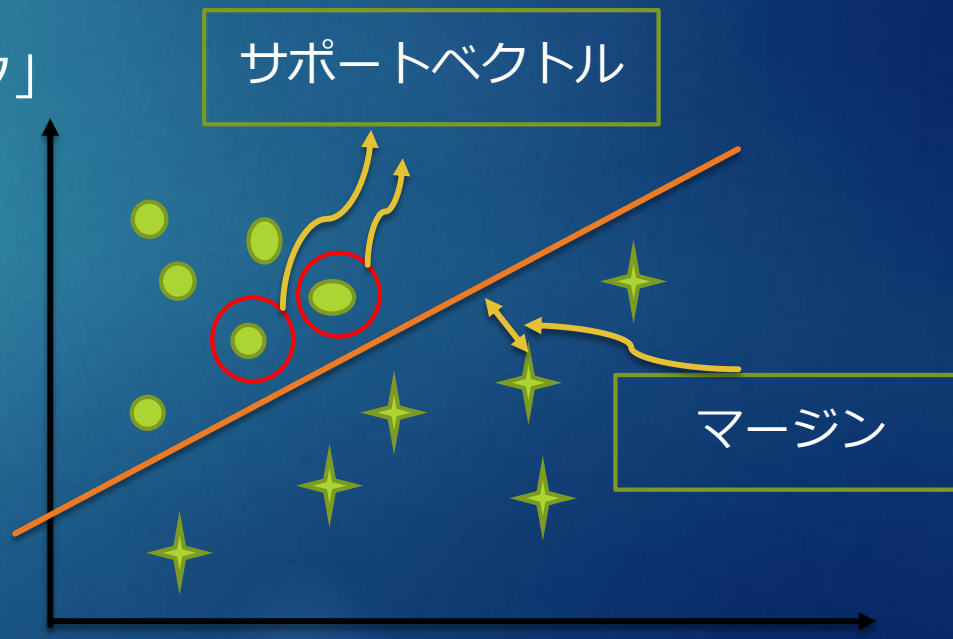
考え方

- 正しい部分類基準を見つけるために、マージンの最大化という考え方を使う。
- マージン(margin)：判定する境界線(直線)とデータの距離
- サポートベクトル：直線と最も近くにあるデータ

＝＞「どちらかに分けにくいデータ」

- よって、直線とデータの距離(マージン)を大きくして

誤判定を防げばよい。



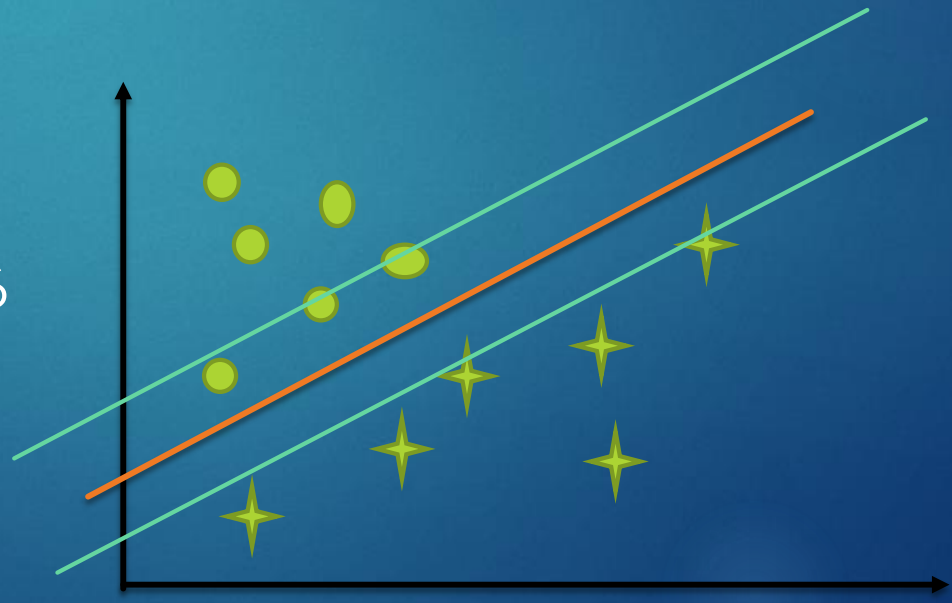
数学的に考えると

- 直線$ax + by + c = 0$(今後は**識別関数**と呼ぶとする)において、これに平行で、サポートベクトルを通る式をそれぞれ以下のように置く：

$$\begin{cases} ax_i + by_i + c = 1 : \text{正例} \\ ax_i + by_i + c = -1 : \text{負例} \end{cases}$$ (各データの番号をiとおいた)

- この時マージンdは、
- $d = \frac{|ax_i+by_i+c|}{\sqrt{(a^2+b^2)}} = \frac{1}{\sqrt{(a^2+b^2)}}$
- このdを最大化(a^2, b^2を最小化)する

ことが方針



目的関数：$\frac{1}{2}(a^2+b^2)$を最小化するa,b,c

制約条件：$\begin{cases} \mathrm{ax_i}+by_i+c\geq 1：正例 \\ \mathrm{ax_i}+by_i+c\leq -1：負例 \end{cases}$ $(i\,\epsilon\{1,2,3,\ldots,n\})$

正解ラベルと双対問題の導入

- 正解ラベルを$t_i (i \epsilon \{1,2,3, \dots, n\})$と置くと、先程の制約条件は

$$\begin{cases} \mathrm{t_i}(\mathrm{ax_i} + by_i + c) \geq 1 : \text{正例} \\ \mathrm{t_i}(\mathrm{ax_i} + by_i + c) \leq -1 : \text{負例} \end{cases} (i \, \epsilon \{1,2,3, \dots, n\})$$

となる。

- 双対問題：最適化を行うような問題を扱う際に、制約条件を用いて別の扱いやすい問題へと置き換えて解くやり方のこと。主問題と双対問題のいずれか一方が最適解を持つなら、もう一方も最適解を持ち、主問題の最小値と双対問題の最大値は一致すると定義されている。
- 解決したい問題(目的関数,制約条件)を主問題、置き換えた後の問題を補問題と言う。
- 今回は補問題としてラグランジュの未定乗数法を使う。

ラグランジュの未定乗数法：

$L(x, y, \lambda)$：ラグランジュ関数

$= f(x, y) - \lambda g(x, y)$を作ると$(\alpha, \beta)$が極限を与える

$\rightarrow (\alpha, \beta)$は$\dfrac{\partial L}{\partial x} = \dfrac{\partial L}{\partial y} = \dfrac{\partial L}{\partial \lambda} = 0$の解)

- ラグランジュ関数を使うと、先程の目的関数は以下の様に書き直せる：

$L=\frac{1}{2}(a^2+b^2)+\lambda_1\{1-t_1(ax_1+by_1+c)\}\lambda_2\{1-t_2(ax_2+by_2+c)\}+\cdots+\lambda_i\{1-(ax_i+by_i+c)\}$を最小化するa,b,c

かつ

$\lambda_1,\lambda_2,\ldots,\lambda_i \geq 0$の式について、その最大値を求める。

- この式を具体的に解いてみる。

$$\frac{\partial L}{\partial a}=a-\lambda_1 t_1 x_1-\lambda_2 t_2 x_2-\cdots-\lambda_i\, t_i x_i=0$$
$$\frac{\partial L}{\partial b}=b-\lambda_1 t_1 x_1-\lambda_2 t_2 x_2-\cdots-\lambda_i\, t_i x_i=0$$
$$\frac{\partial L}{\partial c}=-\lambda_1 t_1 x_1-\lambda_2 t_2 x_2-\cdots-\lambda_i\, t_i x_i=0$$

つまり、

$$\begin{cases} a=\lambda_1 t_1 x_1+\lambda_2 t_2 x_2+\cdots+\lambda_i\, t_i x_i \\ b=\lambda_1 t_1 x_1+\lambda_2 t_2 x_2+\cdots+\lambda_i\, t_i x_i \end{cases} \ldots[1]$$

$$t_1\lambda_1+t_2\lambda_2+\cdots+t_i\lambda_i=0 \ldots[2]$$

- [1]をL=の式に代入すると、

$$L$$
$$= \frac{1}{2}(a^2 + b^2) + \lambda_1\{1 - t_i(ax_1 + by_1 + c)\} + \lambda_2\{1 - t_2(1x_2 + by_2 + c)\} + \cdots$$
$$+ \lambda_i\{1 - t_i(1x_i + by_i + c)\}$$
$$= \frac{1}{2}(a^2+b^2) - a(\lambda_1 t_1 x_1 + \lambda_2 t_2 x_2 + \cdots + \lambda_i t_i x_i) - b(\lambda_1 t_1 y_1 + \lambda_2 t_2 y_2 + \cdots$$
$$+ \lambda_i t_i y_i) = \frac{1}{2}(a^2 + b^2) - a(a) - b(b) + (\lambda_1 + \lambda_2 + \cdots + \lambda_i)$$

整理すると、

$$L = -\frac{1}{2}(a^2 + b^2) + (\lambda_1 + \lambda_2 + \cdots + \lambda_i)$$

さらに、a^2, b^2について、[1]を代入すると

$$a^2 = (\lambda_1 t_1 x_1 + \lambda_2 t_2 x_2 + \cdots + \lambda_i\, t_i x_i)^2$$
$$= \lambda_1\lambda_1 t_1 t_1 x_1 x_1 + \lambda_1\lambda_2 t_1 t_2 x_1 x_2 + \cdots + \lambda_i\lambda_i t_i t_i x_i x_i$$
$$b^2 = (\lambda_1 t_1 y_1 + \lambda_2 t_2 y_2 + \cdots + \lambda_i\, t_i y_i)^2$$
$$= \lambda_1\lambda_1 t_1 t_1 y_1 y_1 + \lambda_1\lambda_2 t_1 t_2 y_1 y_2 + \cdots + \lambda_i\lambda_i t_i t_i y_i y_i$$

式としてあまり綺麗なものではないが、データを入力する際には便利である。

- 故に補問題は以下のようになる：

$$\begin{cases} a = \lambda_1 t_1 x_1 + \lambda_2 t_2 x_2 + \cdots + \lambda_i\, t_i x_i \\ b = \lambda_1 t_1 x_1 + \lambda_2 t_2 x_2 + \cdots + \lambda_i\, t_i x_i \end{cases}$$ の条件の下、

$$L = \frac{1}{2}\{\lambda_1\lambda_1 t_1 t_1 (x_1 x_1 + y_1 y_1) + \lambda_1\lambda_2 t_1 t_2 (x_1 x_2 + y_1 y_2) + \cdots + \lambda_i\lambda_i t_i t_i (x_i x_i + y_i y_i)$$

$$+(\lambda_1 + \lambda_2 + .. + \lambda_i)\ (\lambda_1, \lambda_2, \ldots, \lambda_i \geq 0)$$

となるLの最大値を求めればよい。

Chapter2:Excelで動かして学ぶ

具体例

- 下の表は、男性A,B,Cと女性D,E,Fを対象に、製品X,Yの好感度x,yを調べたものである。この表とExcelを用いて、男女を区別するx,yの識別関数を求めよ。

No.	**Name**	Popularity Rating		**Sex**
		x	y	
1	A	0	0	m.
2	B	0	1	m.
3	C	1	1	m.
4	D	1	0	fe.
5	E	2	0	fe.
6	F	2	1	fe.

解説

- まず男性を負例、女性を正例と置くことにする。そして男性に所属するデータの要素には-1を、女性は1とすると以下の表の様にまとめられる。

	No	Name	x	y	Sex	P or N
Negative	1	A	0	0	m.	-1
	2	B	0	1	m.	-1
	3	C	1	1	m.	-1
Positive	4	D	1	0	fe.	1
	5	E	2	0	fe.	1
	6	F	3	1	fe.	1

- 先程の表を使い式で表すと、

$$\begin{cases} (\mathrm{ax_i}+by_i+c) \geq 1：正例(female) \\ (\mathrm{ax_i}+by_i+c) \leq -1：負例(male) \end{cases} (i \in \{1,2,3,\ldots,6\})$$

- ここで正解ラベルtを導入し、正例に対して1,負例に対して-1とすると

$$t_i(ax_i+by_i+c) \geq 1$$

とまとめられる。

ここからExcelでこの問題を解いていく。またラグランジュの未定乗数法で出てきたλをμとする。今回は$\mu_1 = 1.000, \mu_2 = 4.000, \mu_3 = 1.000, \mu_4 = 4.000, \mu_5 = 2.000, \mu_6 = 1.000$とおく。

		μ1	μ2	μ3	μ4	μ5	μ6
		1.000	4.000	1.000	4.000	2.000	1.000
μ1	1.000	1.000	4.000	1.000	4.000	2.000	1.000
μ2	4.000	4.000	16.000	4.000	16.000	8.000	4.000
μ3	1.000	1.000	4.000	1.000	4.000	2.000	1.000
μ4	4.000	4.000	16.000	4.000	16.000	8.000	4.000
μ5	2.000	2.000	8.000	2.000	8.000	4.000	2.000
μ6	1.000	1.000	4.000	1.000	4.000	2.000	1.000

- 右の表より、Lの右辺は、

No	1	2	3	4	5	6
1	0.000	0.000	0.000	0.000	0.000	0.000
2	0.000	16.000	4.000	0.000	0.000	-4.000
3	0.000	4.000	2.000	-4.000	-4.000	-3.000
4	0.000	0.000	-4.000	16.000	16.000	8.000
5	0.000	0.000	-4.000	16.000	16.000	8.000
6	0.000	-4.000	-3.000	8.000	8.000	5.000

のようになる。この時Lの値は、

(目的関数)	
L	-35.5

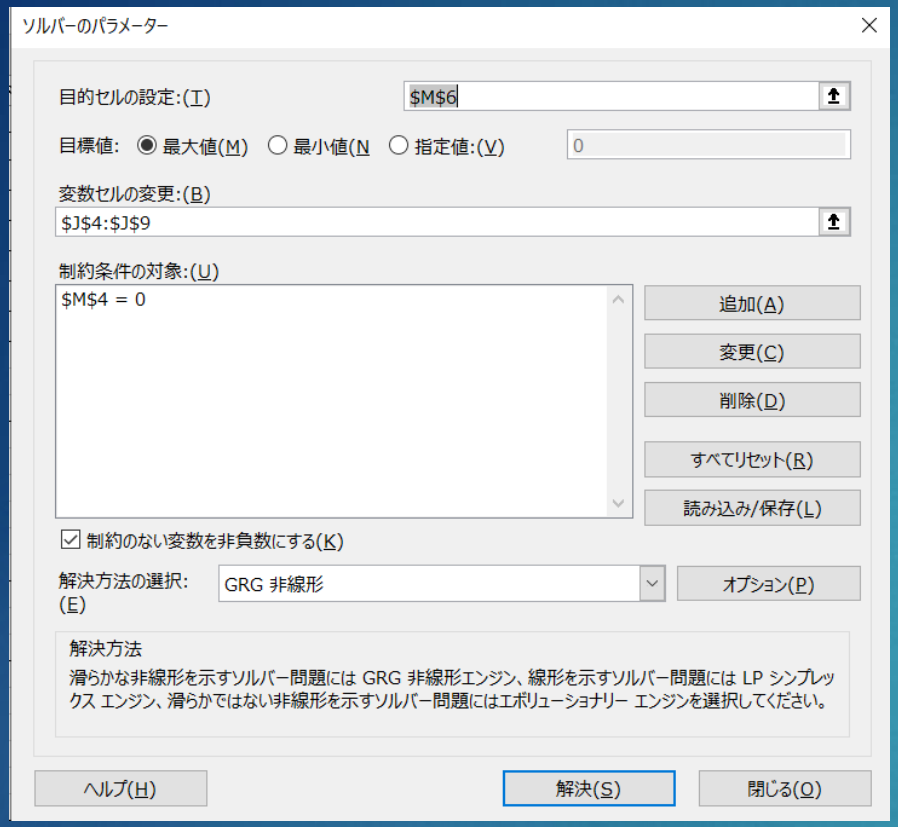
となっている。

		x1	x2	x3	x4	x5	x6
		0	0	1	1	2	2
x1	0	0	0	0	0	0	0
x2	0	0	0	0	0	0	0
x3	1	0	0	1	1	2	2
x4	1	0	0	1	1	2	2
x5	2	0	0	2	2	4	4
x6	2	0	0	2	2	4	4

		y1	y2	y3	y4	y5	y6
		0	1	1	0	0	1
y1	0	0	0	0	0	0	0
y2	1	0	1	1	0	0	1
y3	1	0	1	1	0	0	1
y4	0	0	0	0	0	0	0
y5	0	0	0	0	0	0	0
y6	1	0	1	1	0	0	1

		t1	t2	t3	t4	t5	t6
		-1	-1	-1	1	1	1
t1	-1	1	1	1	-1	-1	-1
t2	-1	1	1	1	-1	-1	-1
t3	-1	1	1	1	-1	-1	-1
t4	1	-1	-1	-1	1	1	1
t5	1	-1	-1	-1	1	1	1
t6	1	-1	-1	-1	1	1	1

- この時、ソルバーを用いて最適解を求めると、



条件	
$\Sigma t_i\mu_i$	0.000
(目的関数)	
L	4
a	b
2.000	-2.000

μ	値
μ_1	1.647
μ_2	0.000
μ_3	2.353
μ_4	3.647
μ_5	0.000
μ_6	0.353

- これよりLが最大の時、L=4,a=2,b=-2となり、μは以上のとおり。つまり、

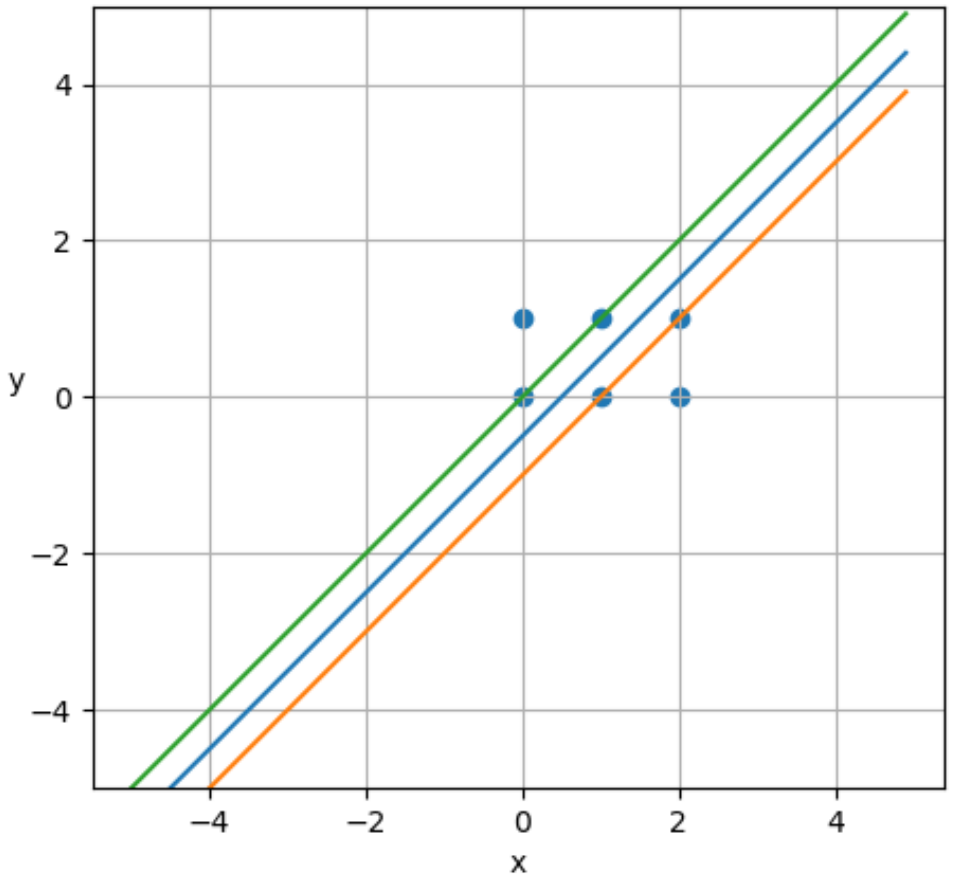
$$\begin{cases} a = \mu_1 t_1 x_1 + \mu_2 t_2 x_2 + \cdots \mu_6 t_6 x_6 = 2 \\ b = \mu_1 t_1 x_1 + \mu_2 t_2 x_2 + \cdots + \mu_6 t_6 x_6 = -2 \end{cases}$$

- この時

$$\begin{cases} c = 1 - ax_i - by_i : \text{正例} \\ c = -1 - ax_i - by_i : \text{負例} \end{cases}$$

なので、c=1が求められる。

- 故に求める識別関数は$2x - 2y - 1 = 0$
- 図示すると右の通り



SVMを使用した実装例

Pythonのscikit learnのデータセットを用いた演習

SVMの応用例

- テキスト分類：文章をあらかじめ用意したカテゴリに分類することができる。主にスパムメールの検出などに使われる。
- 数字認識：手書きの数字の画像のピクセル値をデータに、それを0~9などの数字カテゴリに分類する。郵便番号の追跡などに使われる。
- 顔検出：画像のピクセルの値から、その画像に顔が含まれているか否か、含まれていれば事前に設定したカテゴリの中で誰であるかを分類する。
- 今回はPythonにおける数字認識を実装する。
- バージョンは3.7.2, IDEはJupyter Notebook,使用するモジュールはmatplotlib(可視化のためのもの),sklearn(機械学習のモジュール)

ハードマージンSVMとソフトマージンSVM

- これまでに説明・実装した方法は、クラスが完全に線形分離可能でありマージン最大化を完全に満たすような長平面でのみ適用可能な手法で「ハードマージンSVM」と呼ばれる。
- なので、線形分離不可能なデータの学習をしたい場合「ソフトマージンSVM」を使用する。そこで制約条件を

・マージンが最大である必要がないようにする

・超平面で分離に失敗するデータがあっても許容する

緩めることで対応する→パラメータC(どれだけ誤分類を許容するかのパラメータ)の導入

データセットの読み込みと可視化

```
#Scikit learnのライブラリに含まれているサンプルデータをロード
from sklearn import datasets
import matplotlib.pyplot as plt #可視化用
digits=datasets.load_digits()
images_and_labels=list(zip(digits.images,digits.target))
for index,(image,label) in enumerate(images_and_labels[:10]): #0～9までの読み込み
    plt.subplot(2,5,index+1)
    plt.imshow(image,cmap=plt.cm.gray_r,interpolation='nearest')
    plt.axis('off')
    plt.title('Training: %i' % label)
plt.show()
```

データセットには何が入っているのか？

```
from sklearn import datasets
digits=datasets.load_digits()
dir(digits)
>>>['DESCR', 'data', 'feature_names', 'frame', 'images', 'target',
'target_names']
```

DESCR	データセットの説明文
data	画像データ(特徴量：訓練とテストデータ)
feature_names	画像データの名前
frame	None
images	画像データを8x8にしたもの
target	画像データに対応する数字
target_names	targetデータの名前

訓練データとテストデータの用意

- 今回は読み込んだデータセットの2/3を訓練データ,1/3をテストデータとする。

画像データ
digits.data

X_train:訓練データ

X_test:テストデータ

教師データ
digits.target

y_train:教師データ

y_test:教師データ

```
from sklearn import datasets,svm
digits=datasets.load_digits()
n_train=len(digits.data)*2//3#データの2/3の個数
X_train=digits.data[:n_train]#dataの前半2/3
y_train=digits.target[:n_train]#targetの前半2/3
X_test=digits.data[n_train:]#dataの後半1/3
y_test=digits.target[n_train:]#targetの後半1/3
#print([d.shape for d in [X_train,y_train,X_test,y_test]])
```

```
from sklearn import datasets,svm
digits=datasets.load_digits()
n_train=len(digits.data)*2//3#データの2/3の個数
X_train=digits.data[:n_train]#dataの前半2/3
y_train=digits.target[:n_train]#targetの前半2/3
X_test=digits.data[n_train:]#dataの後半1/3
y_test=digits.target[n_train:]#targetの後半1/3
#print([d.shape for d in [X_train,y_train,X_test,y_test]])
clf=svm.SVC(gamma=0.001,C=100.0)
"""gamma:この値が大きいほど協会が複雑になる
   C:どれだけ誤分類を許すかのパラメータ(値が大きいほど厳しい)
"""
#SVMの学習
clf.fit(X_train,y_train)#学習
print(clf.score(X_test,y_test))#正答率？
>>>0.968280467445742 9
predicted=clf.predict(X_test)#分類結果の取り出し
(y_test != predicted).sum() #誤検出した総数
>>>19
```

学習結果の評価レポート

- Sklearnのmetrics.confusion_matrix()では、各数字ごとに正解した数とどの数字と読み間違えたかチェックできる。

Precision:適合率

recall:再現率

fl-score:F値(再現率と適合率の調和平均)

support:個数

```
from sklearn import metrics as met
print(met.classification_report(y_test,predicted))
print(met.confusion_matrix(y_test,predicted))
>>> precision    recall  f1-score   support

           0       1.00      0.98      0.99        59
           1       0.97      1.00      0.98        62
           2       1.00      0.98      0.99        60
           3       0.96      0.85      0.91        62
           4       0.98      0.95      0.97        62
           5       0.95      0.98      0.97        59
           6       0.98      0.98      0.98        61
           7       0.98      1.00      0.99        61
           8       0.90      0.98      0.94        55
           9       0.95      0.97      0.96        58

    accuracy                           0.97       599
   macro avg       0.97      0.97      0.97       599
weighted avg       0.97      0.97      0.97       599
```

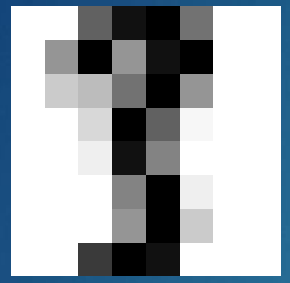
```
[[58  0  0  0  1  0  0  0  0  0]#0は誤検出なし
 [ 0 62  0  0  0  0  0  0  0  0]#1も誤検出なし
 [ 0  0 59  1  0  0  0  0  0  0]#2は1個だけ3と誤検出
 [ 0  0  0 53  0  2  0  1  6  0]#3を5・8と誤検出
 [ 0  0  0  0 59  0  0  0  0  3]#4を9と誤検出
 [ 0  0  0  0  0 58  1  0  0  0]#5を6と誤検出
 [ 0  1  0  0  0  0 60  0  0  0]#6を0と誤検出
 [ 0  0  0  0  0  0  0 61  0  0]#7を8と誤検出
 [ 0  1  0  0  0  0  0  0 54  0]#8を1と誤検出
 [ 0  0  0  1  0  1  0  0  0 56]]#9を3・5と誤検出
```

画像で確認(可視化)

- 前スライドよりテストデータの400番目周辺の誤検出が目立つので、可視化して確かめてみる。

```
#誤検出が目立った部分の可視化
imgs_yt_preds=list(zip(digits.images[n_train:],y_train,predicted))
for index,(image,y_t,pred) in enumerate(imgs_yt_preds[404:416]):
    plt.subplot(3,4,index+1)#3x4で表示
    plt.axis("off")
    plt.tight_layout()
    plt.imshow(image,cmap="Greys",interpolation="nearest")
    plt.title(f"t:{y_t} pre:{pred}",fontsize=15)
plt.show()
```

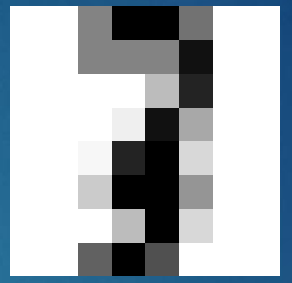
t:8 pre:8



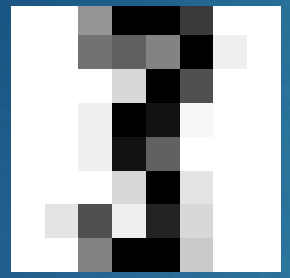
t:9 pre:3

t:0 pre:7

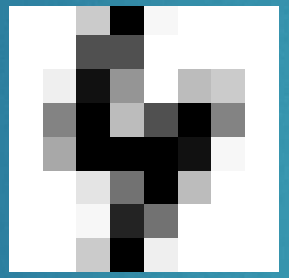
t:1 pre:7



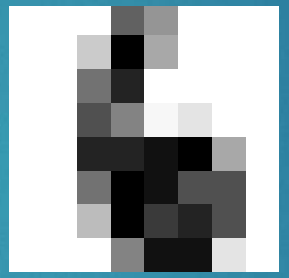
t:2 pre:3



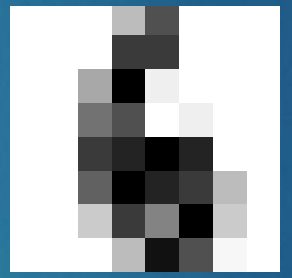
t:3 pre:4



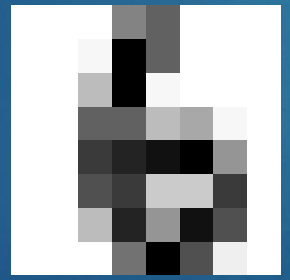
t:4 pre:6



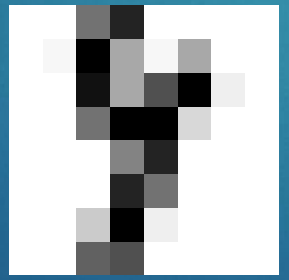
t:5 pre:6



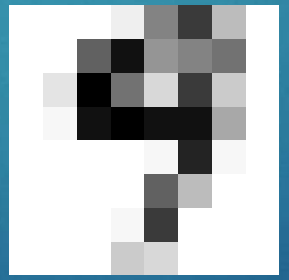
t:6 pre:6



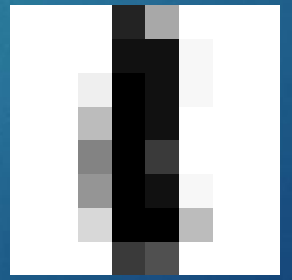
t:7 pre:9



t:8 pre:9



t:9 pre:1



Chapter4:まとめ

- SVMの概要
  - ・線形SVMの理論
  - ・数学的な実装ー双対問題とラグランジュ未定乗数法
- Excelによる具体例

- Pythonによる手書き文字認識
  - ・sklearnモジュールによるSVM,datasetのインポート
  - ・matplotlibによる可視化
- まとめ

付録

Pythonのコード一覧と参考文献

```
#識別関数と座標の図示
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(-5, 5, 0.1)
y =x-0.5
y1=x-1
y2=x
plt.ylim([-5, 5])
plt.gca().set_aspect('equal',
adjustable='box')
plt.xlabel('x')
plt.ylabel('y',  rotation=0)
plt.grid()
plt.plot(x, y)
plt.plot(x,y1)
plt.plot(x,y2)

data=[[0,0],[0,1],[1,1],[1,0],[2,0],[2,1]]
X,Y=zip(*data)
plt.scatter(X,Y)
plt.show()
```

```
#Scikit learnのライブラリに含まれているサンプルデータをロード
from sklearn import datasets,svm
import matplotlib.pyplot as plt #可視化用
digits=datasets.load_digits()
images_and_labels=list(zip(digits.images,digits.target))
for index,(image,label) in
enumerate(images_and_labels[:10]): #0～9までの読み込み
    plt.subplot(2,5,index+1)

plt.imshow(image,cmap=plt.cm.gray_r,interpolation='nea
rest')
    plt.axis('off')
    plt.title('Training: %i' % label)
plt.show()
```

```
n_train=len(digits.data)*2//3#データの2/3の個数
X_train=digits.data[:n_train]#dataの前半2/3
y_train=digits.target[:n_train]#targetの前半2/3
X_test=digits.data[n_train:]#dataの後半1/3
y_test=digits.target[n_train:]#targetの後半1/3
#print([d.shape for d in [X_train,y_train,X_test,y_test]])
clf=svm.SVC(gamma=0.001,C=100.0)
"""gamma:この値が大きいほど協会が複雑になる
   C:どれだけ誤分類を許すかのパラメータ(値が大きいほど厳しい)
"""
#SVMの学習
clf.fit(X_train,y_train)
print(clf.score(X_test,y_test))#正答率？
predicted=clf.predict(X_test)#分類結果の取り出し
(y_test != predicted).sum() #誤検出した総数
#学習させた結果
from sklearn import metrics as met
print(met.classification_report(y_test,predicted))
print(met.confusion_matrix(y_test,predicted))
```

```
#誤検出が目立った部分の可視化
imgs_yt_preds=list(zip(digits.images[n_train:],y_train,predicted))
for index,(image,y_t,pred) in enumerate(imgs_yt_preds[404:416]):
    plt.subplot(3,4,index+1)#3x4で表示
    plt.axis("off")
    plt.tight_layout()
    plt.imshow(image,cmap="Greys",interpolation="nearest")
    plt.title(f"t:{y_t} pre:{pred}",fontsize=15,color="red")
plt.show()
```

参考文献一覧

- http://www.kana-lab.c.titech.ac.jp/lecture/lec_2018_osaka/note_03-svm.pdf
- http://ibisforest.org/index.php?F%E5%80%A4
- https://www.slideshare.net/mknh1122/svm-13623887
- https://qiita.com/yhyhyhjp/items/ebda34f46369b7d3ac8e
- https://aiacademy.jp/media/?p=248
- https://logics-of-blue.com/svm-concept/
- Excelでわかる機械学習超入門涌井 良幸 (著), 涌井 貞美 (著)技術評論社 (講義資料第4章.pdf)
- 詳細!Python3入門ノート 大重美幸(著) ソーテックス社