1 サポートベクタマシンを用いたアヤメの分類

機械学習のデータセットとして広く利用されているアヤメデータセット (Iris datasets) を利用する．このデータセットは scikit-learn に付属しているため簡単に利用することができる．アヤメの特徴量として，がく片 (Sepal) と花弁 (Petal) の幅と長さの 4 種類の計測データと，アヤメの 3 種類の品種 (setosa / versicolor / virginica) で構成されるデータセットである．**図 1** はデータセットの一部を表示したものである．

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

図 1 アイリスデータセット

今回は，このデータセットをサポートベクタマシン (Support Vector Machine, SVM) と呼ぶ機械学習を用いて分類する．

1.1 必要なライブラリのインポート

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm, datasets, model_selection
```

scikit-learn ライブラリから，`svm`，`model_selection` クラスを読み込む．`svm` クラスは分類アルゴリズムのサポートベクタマシンに利用し，`model_selection` クラスはデータセットの加工に利用する．

1.2 データセットの準備

アヤメデータセットは scikit-learn ライブラリから読み込み利用することができる．

```
# アヤメデータセット
iris = datasets.load_iris()
```

データセットの内容の確認のため，データセットの内容を第 12・13 回の授業で紹介した pandas のデータフレーム形式で表示するコードを以下に示す．

```
import pandas as pd
iris_df = pd.DataFrame(iris.data, columns = iris.feature_names)
iris_df['class'] = iris.target #  0-Setosa 1-Versicolour 2-Virginica
display(iris_df.head())
```

データセットの詳細は以下のコードでも確認できる．

```
print(iris.DESCR)
```

1.3 訓練データとテストデータに分割

読み込んだデータセットを訓練データとテストデータに分割する．訓練データはモデルの構築に利用し，モデルの評価にテストデータを利用する．データセットの分割は，model_selection クラスの train_test_split() 関数を利用する．第 1 引数に説明変数の配列，第 2 引数に目的変数の配列，第 3 引数で検証データの割合を指定する．

```
# データセットを訓練用とテスト用に分割
x_train, x_test, y_train, y_test = \
model_selection.train_test_split(iris.data, iris.target, test_size = 0.1)
```

このコードを実行すると，データセットから 10% のデータがランダムに抽出されテストデータとして利用できる．「\(バックスラッシュ)」は，1 行が長い場合に改行するときに利用する．なお，以下のように train_test_split() 関数の引数に random_state=0 のようにランダムシード値を追加すると，何度実行しても同じデータをテスト用に抽出する．

```
# データセットを訓練用とテスト用に分割( ランダムシード値を固定)
x_train, x_test, y_train, y_test = \
model_selection.train_test_split(iris.data, iris.target, test_size = 0.1, random_state = 0)
```

train_test_split() 関数で生成される変数を以下にまとめる．

変数	内容
X_train	訓練用の説明 (特徴) 行列 (アヤメのがく片や花弁の幅および長さ)
X_test	テスト用の説明 (特徴) 行列 (アヤメのがく片や花弁の幅および長さ)
y_train	訓練用の目的変数 (アヤメの種類)
y_test	テスト用の目的変数 (アヤメの種類)

1.4 モデルの学習

サポートベクタマシンを使用する．データセットの学習のため，svm クラスの SVC() 関数を実行しインスタンスを生成する．fit() 関数を用いて学習データの X_train と y_train に対して学習を行う．

```
# 学習
clf = svm.SVC()
clf.fit(X_train, y_train)
```

1.5 学習したモデルの評価

学習したモデルの評価をテストデータを用いて行う．学習したモデルにテストデータ (X_test, y_test) を適用しスコアで表示する．スコアの表示は svm クラスの score() 関数を利用する．スコアの最高値は「1」であり，数値が高いほど良い結果となる．

```
# 結果の表示
print("Score: \t", clf.score(x_test, y_test))
```

学習したモデルで未知のアヤメを分類するには以下の関数を利用する．

```
clf.predic([[がく片の長さ，がく片の幅，花弁の長さ，花弁の幅]])
```

1.6 サポートベクタマシンの補足説明

1.6.1 サポートベクタマシンの概要

SVM (Support Vector Machine) は，分類や回帰問題を解くための機械学習アルゴリズムである．SVM の特徴は，データを高次元空間にマッピングし，異なるクラス間を最適に分離する超平面を見つける点である．カーネル関数（例：線形、RBF、ポリノミアル）を用いて非線形データにも対応することができる．

1.6.2 カーネル関数の種類

SVM の `SVC()` では，`kernel` 引数を指定することで，以下のようなカーネル関数を選択することができる．

- `linear`：線形カーネル
- `poly`：多項式カーネル
- `rbf`：ガウシアン RBF カーネル (デフォルト)
- `sigmoid`：シグモイドカーネル

以下に RBF カーネルを利用する場合のコード例を示す．

```
clf = svm.SVC(kernel='rbf', ← 利用するカーネルの指定
              C=1.0, ← 正則化パラメータ．大きいほど誤分類を厳しく評価
              gamma='scale') ← ガウス分布の幅を決定するパラメータ．小さいとモデルが複雑になる
```

1.6.3 評価指標の種類

スコア以外にも，以下のようなカーネル関数を選択することができる評価指標を利用することが可能である．適切な評価指標を用いることでモデルの性能を詳しく評価することが可能である．

```
# 混同行列
from sklearn.metrics import confusion_matrix
y_pred = clf.predict(X_test)
print(confusion_matrix(y_test, y_pred))
```

```
# 精度・再現率・F1 スコア
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

2 演習課題

以下の各課題の出力結果を Word でまとめ提出する．

2.1 課題 1：データの可視化

アヤメデータセットから 2 つの特徴量を選び散布図を作成する．散布図は各品種を色分けしプロットする．以下にコード例を示す．

```
# 可視化用コード例
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm, datasets, model_selection
```

```
5
6  # アヤメデータセット
7  iris = datasets.load_iris()
8  X = iris.data[:, :2]  # 最初の 2つの特徴量を使用
9  y = iris.target
10
11 # 訓練とテストに分割
12 X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.2,
       random_state=42)
13
14 # モデルの学習
15 clf = svm.SVC(kernel='linear')
16 clf.fit(X_train, y_train)
17
18 # 決定境界をプロット
19 x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
20 y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
21 xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01), np.arange(y_min, y_max, 0.01))
22
23 Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
24 Z = Z.reshape(xx.shape)
25
26 plt.contourf(xx, yy, Z, alpha=0.8)
27 plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k', marker='o')
28 plt.xlabel('Sepal length')
29 plt.ylabel('Sepal width')
30 plt.title('SVM with linear kernel')
31 plt.show()
```

2.2 課題 2：他のデータセットへの適用

scikit-learn ライブラリに付属するワインデータセットや乳がんデータセットに SVM を適用し可視化を行う．以下にワインデータセットを用いたコード例を示す．

```
1  # 可視化用コード例
2  import numpy as np
3  import matplotlib.pyplot as plt
4  from sklearn import svm, datasets, model_selection
5
6  # ワインデータセット
7  wine = datasets.load_wine()
8  X = wine.data[:, :2]  # 最初の 2つの特徴量を使用
9  y = wine.target
10
11 # 訓練とテストに分割
```

```
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.2,
    random_state=42)

# モデルの学習
clf = svm.SVC(kernel='linear')
clf.fit(X_train, y_train)

# 決定境界をプロット
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01), np.arange(y_min, y_max, 0.01))

Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, alpha=0.8)
plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k', marker='o')
plt.xlabel(wine.feature_names[0])
plt.ylabel(wine.feature_names[1])
plt.title('Wine Dataset - SVM with linear kernel')
plt.show()
```

以下に乳がんデータセットを用いたコード例を示す.

```
# 可視化用コード例
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm, datasets, model_selection

# 乳がんデータセット
cancer = datasets.load_breast_cancer()
X = cancer.data[:, :2]  # 最初の 2つの特徴量を使用
y = cancer.target

# 訓練とテストに分割
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.2,
    random_state=42)

# モデルの学習
clf = svm.SVC(kernel='rbf', gamma='scale')
clf.fit(X_train, y_train)

# 決定境界をプロット
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
```

```
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01), np.arange(y_min, y_max, 0.01))

Z = clf.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, alpha=0.8)
plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k', marker='o')
plt.xlabel(cancer.feature_names[0])
plt.ylabel(cancer.feature_names[1])
plt.title('Breast Cancer Dataset - SVM with RBF kernel')
plt.show()
```

出典・参考文献

[1] 鶴長 鎮一, “これからはじめる Python 入門講座”, 技術評論社 (2023)

[2] scikit-learn, https://scikit-learn.org/stable/

[3] TensorFlow 2 quickstart for beginners,
https://github.com/tensorflow/docs/blob/master/site/en/tutorials/quickstart/beginner.ipynb