**データベース処理とデータ分析**

**１０． 分類**

概要 Abstract
今日の授業では、機械学習による分類を行う．データの分類能力を行う能力を持つ分類器の作成である．

SVM kernel

Nearest Neighbor

Random Forests

演習 (Exercises)
今日の授業で使用するデータセット（データセット名: Iris）の出典
出典：http://archive.ics.uci.edu/ml/datasets/Iris

R. A. Fisher
Department of Statistics
University of Cambridge

 (1) Linear SVMによる機械学習

機械学習を行う．まずはLinear SVM を用いる．Linear SVM のオブジェクトを作成し，classifier という名前のオブジェクトに格納する．このとき，正則化パラメータCを1.0に設定し，カーネルには'linear'を指定する．分類対象データとしてIrisデータセットを使用し，がく片の長さと幅の2特徴量を用いて学習を行う．

Linear SVMの分類器を使用して予測を行う．動作確認として，2次元の特徴ベクトル[[5.0, 3.5]]および[[6.5, 3.0]]に対する分類予測を実行する．予測結果が出力され，エラーメッセージが出なければ正常に動作していることを確認できる．

import numpy as np

import sklearn.svm

import matplotlib.pyplot as plt

from sklearn import datasets

import pandas as pd

# Irisデータセットの読み込みとデータフレームの作成

iris = datasets.load\_iris()

d = pd.DataFrame(data=iris.data, columns=iris.feature\_names)

d['species'] = iris.target

# SVMパラメータの設定とモデルの学習

C = 1.0 # 正則化パラメータ

classifier = sklearn.svm.SVC(kernel='linear', C=C).fit(d.iloc[:, [0, 1]], d['species'])

# 分類器の動作確認

print(classifier.predict([[5.0, 3.5]]))

print(classifier.predict([[6.5, 3.0]]))

(2) 機械学習の結果を表示

機械学習の結果を表示するために，まず、描画のためのメッシュグリッドを作成する．このメッシュグリッドの各点に対してSVMによる分類予測を行い，分類結果を表示する．次に，元のIrisデータセットの特徴量（がく片の長さと幅）を散布図として重ねて描画する．

import numpy as np

import sklearn.svm

import matplotlib.pyplot as plt

from sklearn import datasets

import pandas as pd

# Irisデータセットの読み込みとデータフレームの作成

iris = datasets.load\_iris()

d = pd.DataFrame(data=iris.data, columns=iris.feature\_names)

d['species'] = iris.target

# SVMパラメータの設定とモデルの学習

C = 1.0 # 正則化パラメータ

classifier = sklearn.svm.SVC(kernel='linear', C=C).fit(d.iloc[:, [0, 1]], d['species'])

# 決定境界の可視化のためのメッシュグリッド作成

m = 0.2 # マージン

xmin, xmax = d.iloc[:, 0].min() - m, d.iloc[:, 0].max() + m

ymin, ymax = d.iloc[:, 1].min() - m, d.iloc[:, 1].max() + m

h = 0.01 # グリッド解像度

xx, yy = np.meshgrid(np.arange(xmin, xmax, h), np.arange(ymin, ymax, h))

# 決定境界の計算と描画

Z = classifier.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

plt.pcolor(xx, yy, Z, alpha=0.3)

plt.scatter(d.iloc[:, 0], d.iloc[:, 1], c=d['species'], cmap='Set1')

plt.title('Linear SVM on Iris Data')

plt.scatter(d['sepal length (cm)'], d['sepal width (cm)'], c=d['species'], cmap='Set1')

plt.title('Original Iris Data')

plt.show()

(3) 3種類のSVMカーネルを用いた分類結果を比較する．カーネルには以下を使用する：

* Linear kernel SVM（線形カーネル）
* RBF kernel SVM（RBFカーネル，gamma=0.7）
* Polynomial kernel SVM（次数3の多項式カーネル）

すべてのモデルにおいて，正則化パラメータCを1.0に設定する．Irisデータセットのがく片の長さと幅を特徴量として学習を行う．可視化のために，データの範囲に0.2のマージンを加えた領域を0.01間隔のメッシュグリッドに分割する．各SVMモデルでの分類結果を描画する．品種に応じて'Set1'カラーマップで色分けされる．

# 3種類のカーネルを用いたSVMの比較

import numpy as np

import sklearn.svm

import matplotlib.pyplot as plt

from sklearn import datasets

import pandas as pd

# Irisデータセットの読み込みとデータフレームの作成

iris = datasets.load\_iris()

d = pd.DataFrame(data=iris.data, columns=iris.feature\_names)

d['species'] = iris.target

# SVMパラメータの設定とモデルの学習

C = 1.0 # 正則化パラメータ

# 3種類のSVMモデルの学習と可視化

classifiers = [

 ('Linear SVM', sklearn.svm.SVC(kernel='linear', C=C)),

 ('RBF SVM', sklearn.svm.SVC(kernel='rbf', C=C, gamma=0.7)),

 ('Polynomial SVM', sklearn.svm.SVC(kernel='poly', C=C, degree=3))

]

m = 0.2 # マージン

xmin, xmax = d.iloc[:, 0].min() - m, d.iloc[:, 0].max() + m

ymin, ymax = d.iloc[:, 1].min() - m, d.iloc[:, 1].max() + m

h = 0.01 # グリッド解像度

xx, yy = np.meshgrid(np.arange(xmin, xmax, h), np.arange(ymin, ymax, h))

plt.figure(figsize=(15, 5))

for i, (name, clf) in enumerate(classifiers, 1):

 clf.fit(d.iloc[:, [0, 1]], d['species'])

 Z = clf.predict(np.c\_[xx.ravel(), yy.ravel()])

 Z = Z.reshape(xx.shape)

 plt.subplot(1, 3, i)

 plt.pcolor(xx, yy, Z, alpha=0.3)

 plt.scatter(d.iloc[:, 0], d.iloc[:, 1], c=d['species'], cmap='Set1')

 plt.title(f'{name} on Iris Data')

plt.show()

(4) k-最近傍法（k-NN）を用いた分類

k-最近傍法（k-NN）を用いた分類を行う。k（参照する近傍点の数）の値を1，3，5と変えて，その影響を比較する．Irisデータセットのがく片の長さと幅を特徴量として使用する．可視化のために，データの範囲に0.2のマージンを加えた領域を0.01間隔のメッシュグリッドに分割する．各k-NNモデルでの分類結果を描画する．品種に応じて'Set1'カラーマップで色分けされる．

# k-最近傍法による分類と可視化

import numpy as np

import sklearn.neighbors

import matplotlib.pyplot as plt

from sklearn import datasets

import pandas as pd

# Irisデータセットの読み込みとデータフレームの作成

iris = datasets.load\_iris()

d = pd.DataFrame(data=iris.data, columns=iris.feature\_names)

d['species'] = iris.target

# メッシュグリッドの作成

m = 0.2 # マージン

xmin, xmax = d.iloc[:, 0].min() - m, d.iloc[:, 0].max() + m

ymin, ymax = d.iloc[:, 1].min() - m, d.iloc[:, 1].max() + m

h = 0.01 # グリッド解像度

xx, yy = np.meshgrid(np.arange(xmin, xmax, h), np.arange(ymin, ymax, h))

# 3種類のk-NNモデルの学習と可視化

classifiers = [

 ('1-Nearest Neighbor', sklearn.neighbors.KNeighborsClassifier(n\_neighbors=1)),

 ('3-Nearest Neighbor', sklearn.neighbors.KNeighborsClassifier(n\_neighbors=3)),

 ('5-Nearest Neighbor', sklearn.neighbors.KNeighborsClassifier(n\_neighbors=5))

]

plt.figure(figsize=(15, 5))

for i, (name, clf) in enumerate(classifiers, 1):

 clf.fit(d.iloc[:, [0, 1]], d['species'])

 Z = clf.predict(np.c\_[xx.ravel(), yy.ravel()])

 Z = Z.reshape(xx.shape)

 plt.subplot(1, 3, i)

 plt.pcolor(xx, yy, Z, alpha=0.3)

 plt.scatter(d.iloc[:, 0], d.iloc[:, 1], c=d['species'], cmap='Set1')

 plt.title(f'{name} on Iris Data')

plt.show()

(5) ランダムフォレストを用いた分類

ランダムフォレストを用いた分類を行う．sklearn.ensembleモジュールのRandomForestClassifierを使用し，デフォルトのパラメータ設定でモデルを学習する．Irisデータセットのがく片の長さと幅を特徴量として使用する．可視化のために，データの範囲に0.2のマージンを加えた領域を0.01間隔のメッシュグリッドに分割する．ランダムフォレストでの分類結果を描画する．品種に応じて'Set1'カラーマップで色分けされる．さらに，学習したモデルの特徴量重要度（feature\_importances\_）を算出し，がく片の長さと幅それぞれの分類への寄与度を表示する．

# ランダムフォレストによる分類と可視化

import numpy as np

import sklearn.ensemble

import matplotlib.pyplot as plt

from sklearn import datasets

import pandas as pd

# Irisデータセットの読み込みとデータフレームの作成

iris = datasets.load\_iris()

d = pd.DataFrame(data=iris.data, columns=iris.feature\_names)

d['species'] = iris.target

# メッシュグリッドの作成

m = 0.2 # マージン

xmin, xmax = d.iloc[:, 0].min() - m, d.iloc[:, 0].max() + m

ymin, ymax = d.iloc[:, 1].min() - m, d.iloc[:, 1].max() + m

h = 0.01 # グリッド解像度

xx, yy = np.meshgrid(np.arange(xmin, xmax, h), np.arange(ymin, ymax, h))

# ランダムフォレストモデルの学習と可視化

classifier7 = sklearn.ensemble.RandomForestClassifier().fit(d.iloc[:, [0, 1]], d['species'])

Z = classifier7.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

# 決定境界とデータ点の可視化

plt.pcolor(xx, yy, Z, alpha=0.3)

plt.scatter(d.iloc[:, 0], d.iloc[:, 1], c=d['species'], cmap='Set1')

plt.title('Random Forest on Iris Data')

plt.show()

# 特徴量の重要度を表示

print('特徴量の重要度:', classifier7.feature\_importances\_)