レポート提出票

科目名: 情報工学実験3

実験課題名: 課題2 パターン認識

実施日: 2021年 6月 17日

学籍番号: 4619055

氏名: 辰川力駆

共同実験者:

© 2020-21 東京理科大学 工学部 情報工学科

1　はじめに

パターン認識とは、入力データをあらかじめ定めていた複数のクラスの 1 つに対応させる処理のことをいう。分類を行うクラスの種類に応じて、パターン認識は大きく、「分類問題」と「予測問題」に分けることができる。また、分類問題、予測問題の両方に関して、「教師あり学習」と「教師なし学習」の 2 つが存在しており、本実験では、入力データと出力の組みが与えられているデータを利用したパターン認識である「教師あり学習」について理解する。また、「教師あり学習」の代表的な手法であるロジスティック回帰とサポートベクトルマシンについて実装する。

2　レポート課題

2.1　課題 1

課題内容

講義のロジスティック回帰モデルでは、全ての変数を用いたが、モデルに組み込む変数を選択し最適なモデルを選択することができる。変数選択手法に関してどれか一つ調べる。

手順

1. web で変数選択手法について検索する。
2. その変数選択手法に関して Python で実装する。
3. このプログラムを実行する。

結果

変数選択手法として、Wrapper Method について調べた。機械学習モデルを使用して特徴量の組み合わせを評価することで、変数間の関係を探し出し、それぞれのモデルに最適な特徴量の組み合わせを探し出す手法である。

それについて実装したのがソースコード 1 である。結果を比較すると以下の表 1 のようになる。

表 1: ロジスティク回帰モデルの変数選択をした場合としていない場合

	変数選択なし (すべて使う)	変数選択あり
accuracy	0.78125	0.78125
汎化誤差	0.21875	0.21875
precision	0.72727	0.73810
f1 score	0.64646	0.63918

考察

表 1 より、変数選択なしより変数選択ありのほうが適合率が少しだけ高かった。逆に f1 score は僅かに変数選択なしのほうが高かった。変数選択を実装した場合としない場合でほとんど変わらなかったことから、変数選択を実装しなくても全ての変数を用いた場合で十分にロジスティック回帰で良いモデルを作成できると考える。

2.2 課題 2

課題内容

本実験では線形カーネル関数を用いたサポートベクトルマシンの手法を適用した。課題 2 ではそれ以外のカーネル関数を用いたサポートベクトルマシンについて調査し、実装する。

手順

1. web でカーネル関数を用いたサポートベクトルマシンについて検索する。
2. そのサポートベクトルマシンに関して Python で実装する。
3. このプログラムを実行する。

結果

線形カーネル関数以外のサポートベクトルマシンとして、RBF カーネルについて調べた。コスト関数 C とカーネル関数の γ を用いる方法であり、γ の値が小さいほど単純な決定境界となり、大きいほど複雑な決定境界となる。

それについて実装したのがソースコード 2 である。結果を比較すると以下の表 2 のようになる。

$$\begin{aligned} \gamma &= 0.01 \\ C &= 1.02 \end{aligned}$$

表 2: 線形カーネルと RBF カーネルの結果

	線形カーネル	RBF カーネル
accuracy	0.78125	0.74375
汎化誤差	0.21875	0.25625
precision	0.75	0.70588
f1 score	0.63158	0.53933

考察

RBF カーネルは、

$$k(x, x') = \exp(-\gamma||x - x'||^2)$$

という式で定義される。

表 2 より、accuracy、precision、f1 score のどれにおいても線形カーネル関数を用いたサポートベクトルマシンの方が性能が高いと分かる。なので、今回のデータに関していえば線形カーネル関数を用いたサポートベクトルマシンを実装するほうが良いと考える。

2.3 課題 3

課題内容

sklearn.dataset の中にあるワインの品質データ (Wine recognition dataset) を用いて、今回の実験の手法 (ロジスティック、SVM) の中から 2 値分類を行う。

手順

1. ロジスティクと SVM に関して Python で実装する。
2. このプログラムを実行する。

結果

ワインの品質データに関して、(0,1) の組合せで、ロジステック回帰分析と線形カーネルのサポートベクトルマシンを考えた。それについて実装したのがソースコード 3 である。結果を比較すると以下の表 3 のようになる。

表 3: ロジステック回帰モデルとサポートベクトルマシンの結果

	ロジステック	線形 SVM
accuracy	0.94872	1.0
汎化誤差	0.05128	0.0
precision	0.95	1.0
f1 score	0.95	1.0

考察

これらを比較すると、線形カーネルのサポートベクトルマシンの方が accuracy、precision、f1 score において良い性能だと分かる。しかし、ロジステック回帰の方も良い精度なので、どちらも有力な性能であると考える。

2.4　課題 4

課題内容

課題 3 のワインの品質データに対して 3 値分類を行う。手法についての調査し説明する。

手順

1. web で 3 値分類について検索する。
2. その 3 値分類の方法について Python で実装する。
3. このプログラムを実行する。

結果

3 値分析の中で決定木について調べた。それについて実装したのがソースコード 4 である。結果は正確度が 0.94444 となった。

考察

決定木分析は、段階的にデータを分割していき、木のような分析結果を出力するものなので実装しやすいと考える。また、結果からこの学習モデルは、性能が高いものであると分かる。

3　感想

先週の感想でも述べたが、自分的には動画を 1 つにまとめてくれる方が見やすかった。良かった点としては、前回よりは端的にまとめていたので見返すことがしやすかった。あまり良くなかった点としては、学校に行くのに動画を見るだけなのはあまり良くないと考える。

参考文献

[1] 特徴量選択のまとめ - Qiita

https://qiita.com/shimopino/items/5fee7504c7acf044a521

最終閲覧日 2021/06/20

[2] 変数選択 (Feature Selection) の実装と改善の確認 - 学習する天然ニューラルネット

https://aotamasaki.hatenablog.com/entry/2018/04/27/222536

最終閲覧日 2021/06/20

[3] 【SVM】rbf カーネルのハイパーパラメータをグリッドサーチとベイズ最適化で探す【iris データセット】

https://kenyu-life.com/2019/08/24/rbf_svm_gs/

最終閲覧日 2021/06/20

[4] 3 種類のワインの科学的特徴を行ってみる-決定木 - AI 人工知能テクノロジー

https://newtechnologylifestyle.net/3%E7%A8%AE%E9%A1%9E%E3%81%AE%E3%83%AF%E3%82%A4%E3%83%B3%E3%81%AE%E7%A7%91%E5%AD%A6%E7%9A%84%E7%89%B9%E5%BE%B4%E3%82%92%E8%A1%8C%E3%81%A3%E3%81%A6%E3%81%BF%E3%82%8B-%E6%B1%BA%E5%AE%9A%E6%9C%A8/

最終閲覧日 2021/06/20

A 付録

ソースコード 1: レポート課題 1

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,
    recall_score, f1_score
from sklearn.metrics import roc_curve, auc
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFECV

pima = pd.read_csv("data/pima.csv")

X = pima.iloc[:,0:7]
Y =pima["type"].map({'No': 0, 'Yes': 1})

select = RFECV(LogisticRegression(), cv=10, scoring='average_precision')
select.fit(X, Y)
mask = select.support_
X_selected = X.iloc[:,mask]

X_tr, X_te, Y_tr, Y_te = train_test_split(X_selected, Y, test_size=0.3, random_state
    =4619055)

lr = LogisticRegression(C=np.inf,solver='newton-cg')
lr.fit(X_tr, Y_tr)

Y_pred = lr.predict(X_te)

print('confusion matrix = \n', confusion_matrix(y_true=Y_te, y_pred=Y_pred))
print('accuracy = ', accuracy_score(y_true=Y_te, y_pred=Y_pred))
print('汎化誤差 = ', 1-accuracy_score(y_true=Y_te, y_pred=Y_pred))
print('precision = ', precision_score(y_true=Y_te, y_pred=Y_pred))
print('f1 score = ', f1_score(y_true=Y_te, y_pred=Y_pred))

Y_score = lr.predict_proba(X_te)[:, 1]
fpr, tpr, thresholds = roc_curve(y_true=Y_te, y_score=Y_score)

plt.plot(fpr, tpr, label='roc curve (area = %0.3f)' % auc(fpr, tpr))
plt.plot([0, 1], [0, 1], linestyle='--', label='random')
plt.plot([0, 0, 1], [0, 1, 1], linestyle='--', label='ideal')
plt.legend()
plt.xlabel('false positive rate')
plt.ylabel('true positive rate')
```

ソースコード 2: レポート課題 2

```
import pandas as pd
import numpy as np
```

```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,
    recall_score, f1_score

pima = pd.read_csv("data/pima.csv")
X = pima.iloc[:,0:7]
Y =pima["type"].map({'No': 0, 'Yes': 1})
X_tr, X_te, Y_tr, Y_te = train_test_split(X, Y, test_size=0.3, random_state=4619055)

model = SVC(C=1 ,kernel='linear', random_state=4619055)
model.fit(X_tr, Y_tr)
Y_pred = model.predict(X_te)

print('confusion matrix = \n', confusion_matrix(y_true=Y_te, y_pred=Y_pred))
print('accuracy = ', accuracy_score(y_true=Y_te, y_pred=Y_pred))
print('汎化誤差 = ', 1-accuracy_score(y_true=Y_te, y_pred=Y_pred))
print('precision = ', precision_score(y_true=Y_te, y_pred=Y_pred))
print('f1 score = ', f1_score(y_true=Y_te, y_pred=Y_pred))

best_score = float(0.0)
best_param_gamma = 0.0
best_param_C = 0.0

scores = pd.DataFrame()
for gamma in np.linspace(0.01, 10, 100):
    for C in np.linspace(0.01, 10, 100):
        svm = SVC(kernel = 'rbf', gamma=gamma, C=C, random_state=4619055)
        svm.fit(X_tr, Y_tr)
        scores = scores.append(
                {
                    'gamma': gamma,
                    'C': C,
                    'accuracy': svm.score(X_te, Y_te)
                },
                ignore_index=True)

        if best_score < svm.score(X_te, Y_te):
            best_score = svm.score(X_te, Y_te)
            best_param_gamma = gamma
            best_param_C = C

print("ベストスコア：", round(best_score,2))
print('g:%s c:%s' %(round(best_param_gamma,2),round(best_param_C,2)))

gamma = best_param_gamma
C = best_param_C

svm = SVC(kernel = 'rbf', gamma=gamma, C=C, random_state=4619055)
```

```
svm.fit(X_tr, Y_tr)
Y_pred = svm.predict(X_te)
print('confusion␣matrix␣=␣\n', confusion_matrix(y_true=Y_te, y_pred=Y_pred))
print('accuracy␣=␣', accuracy_score(y_true=Y_te, y_pred=Y_pred))
print('汎化誤差␣=␣', 1−accuracy_score(y_true=Y_te, y_pred=Y_pred))
print('precision␣=␣', precision_score(y_true=Y_te, y_pred=Y_pred))
print('f1␣score␣=␣', f1_score(y_true=Y_te, y_pred=Y_pred))
```

ソースコード 3: レポート課題 3

```
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,
    recall_score, f1_score
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_wine

wine = load_wine()
data = pd.DataFrame(data=wine.data, columns=wine.feature_names)
data['category'] = wine.target
data = data.query("category␣==␣0␣|␣category␣==␣1")
X = data[data.columns[data.columns != 'category']]
Y = data['category']

X_tr, X_te, Y_tr, Y_te = train_test_split(X, Y, test_size=0.3, random_state=4619055)

lr = LogisticRegression(C=np.inf, solver='newton-cg')

lr.fit(X_tr, Y_tr)
Y_pred = lr.predict(X_te)
print('confusion␣matrix␣=␣\n', confusion_matrix(y_true=Y_te, y_pred=Y_pred))
print('accuracy␣=␣', accuracy_score(y_true=Y_te, y_pred=Y_pred))
print('汎化誤差␣=␣', 1−accuracy_score(y_true=Y_te, y_pred=Y_pred))
print('precision␣=␣', precision_score(y_true=Y_te, y_pred=Y_pred))
print('f1␣score␣=␣', f1_score(y_true=Y_te, y_pred=Y_pred))

model = SVC(C=1 ,kernel='linear', random_state=4619055)

model.fit(X_tr, Y_tr)
Y_pred = model.predict(X_te)
print('confusion␣matrix␣=␣\n', confusion_matrix(y_true=Y_te, y_pred=Y_pred))
print('accuracy␣=␣', accuracy_score(y_true=Y_te, y_pred=Y_pred))
print('汎化誤差␣=␣', 1−accuracy_score(y_true=Y_te, y_pred=Y_pred))
print('precision␣=␣', precision_score(y_true=Y_te, y_pred=Y_pred))
print('f1␣score␣=␣', f1_score(y_true=Y_te, y_pred=Y_pred))
```

ソースコード 4: レポート課題 4

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_graphviz
from IPython.display import Image
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_wine
from sklearn.metrics import confusion_matrix

data= load_wine()
dataX = pd.DataFrame(data=data.data,columns=data.feature_names)
dataY = pd.DataFrame(data=data.target)
dataY = dataY.rename(columns={0: 'class'})
X_train, X_test, Y_train, Y_test = train_test_split(dataX, dataY, test_size=0.3,
    random_state=4619055)
clf = DecisionTreeClassifier()
clf.fit(X_train,Y_train)
df = pd.DataFrame(clf.predict_proba(X_test))
df = df.rename(columns={0: 'class_0',1: 'class_1',2: 'class_2'})
df = pd.DataFrame(clf.predict(X_test))
df = df.rename(columns={0: '判定'})
df = pd.DataFrame(confusion_matrix(Y_test,clf.predict(X_test).reshape(-1,1)))
df = df.rename(columns={0: 'class_0',1: 'class_1',2: 'class_2'}, index={0: '
    class_0',1: 'class_1',2: 'class_2'})
print(df)
clf.score(X_test,Y_test)
```