知識情報学第6回演習サンプルプログラム ex6.ipynb

- Programmed by Wu Hongle, 監修　福井健一
- Last updated: 2019/09/02
- Checked with Python 3.8.8, scikit-learn 1.0
- MIT Lisence

SVMによるBreast Cancerデータの識別

- 入れ子交差検証で最適パラメータを探索

In [1]:

```
import numpy as np
from sklearn.svm import SVC
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import StratifiedKFold, GridSearchCV
from sklearn.preprocessing import scale
```

Breast Cancerデータのロード

In [2]:

```
df = load_breast_cancer()
X = df.data
y = df.target

# z標準化
X = scale(X)
```

In [3]:

```
print(df.DESCR)
```

```
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class

    :Attribute Information:
        - radius (mean of distances from center to points on the perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
        - smoothness (local variation in radius lengths)
        - compactness (perimeter^2 / area - 1.0)
        - concavity (severity of concave portions of the contour)
        - concave points (number of concave portions of the contour)
        - symmetry
        - fractal dimension ("coastline approximation" - 1)

        The mean, standard error, and "worst" or largest (mean of the three
        worst/largest values) of these features were computed for each image,
        resulting in 30 features.  For instance, field 0 is Mean Radius, field
        10 is Radius SE, field 20 is Worst Radius.

        - class:
                - WDBC-Malignant
                - WDBC-Benign

    :Summary Statistics:

    ===================================== ====== ======
                                           Min    Max
    ===================================== ====== ======
    radius (mean):                        6.981  28.11
    texture (mean):                       9.71   39.28
    perimeter (mean):                     43.79  188.5
    area (mean):                          143.5  2501.0
    smoothness (mean):                    0.053  0.163
    compactness (mean):                   0.019  0.345
    concavity (mean):                     0.0    0.427
    concave points (mean):                0.0    0.201
    symmetry (mean):                      0.106  0.304
    fractal dimension (mean):             0.05   0.097
    radius (standard error):              0.112  2.873
    texture (standard error):             0.36   4.885
    perimeter (standard error):           0.757  21.98
    area (standard error):                6.802  542.2
    smoothness (standard error):          0.002  0.031
    compactness (standard error):         0.002  0.135
    concavity (standard error):           0.0    0.396
    concave points (standard error):      0.0    0.053
    symmetry (standard error):            0.008  0.079
    fractal dimension (standard error):   0.001  0.03
    radius (worst):                       7.93   36.04
    texture (worst):                      12.02  49.54
```

```
    perimeter (worst):                    50.41  251.2
    area (worst):                         185.2  4254.0
    smoothness (worst):                   0.071  0.223
    compactness (worst):                  0.027  1.058
    concavity (worst):                    0.0    1.252
    concave points (worst):               0.0    0.291
    symmetry (worst):                     0.156  0.664
    fractal dimension (worst):            0.055  0.208
    ===================================== ====== ======

    :Missing Attribute Values: None

    :Class Distribution: 212 - Malignant, 357 - Benign

    :Creator:  Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

    :Donor: Nick Street

    :Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
https://goo.gl/U2Uwz2

Features are computed from a digitized image of a fine needle
aspirate (FNA) of a breast mass.  They describe
characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using
Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree
Construction Via Linear Programming." Proceedings of the 4th
Midwest Artificial Intelligence and Cognitive Science Society,
pp. 97-101, 1992], a classification method which uses linear
programming to construct a decision tree.  Relevant features
were selected using an exhaustive search in the space of 1-4
features and 1-3 separating planes.

The actual linear program used to obtain the separating plane
in the 3-dimensional space is that described in:
[K. P. Bennett and O. L. Mangasarian: "Robust Linear
Programming Discrimination of Two Linearly Inseparable Sets",
Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:

ftp ftp.cs.wisc.edu
cd math-prog/cpo-dataset/machine-learn/WDBC/

|details-start|
**References**
|details-split|

- W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction
  for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on
  Electronic Imaging: Science and Technology, volume 1905, pages 861-870,
  San Jose, CA, 1993.
- O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and
  prognosis via linear programming. Operations Research, 43(4), pages 570-577,
  July-August 1995.
- W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning techniques
```

to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77 (1994) 163-171.

|details-end|

入れ子交差検証でハイパーパラメータを最適化

- 【課題1】探索するパラメータにカーネル関数の追加や範囲を変更して最適パラメータを探してみましょう
    - グリッドサーチパラメータリストの書き方は下記を参照
        - https://scikit-learn.org/stable/modules/grid_search.html#grid-search
    - SVCの可能なパラメータリストは下記を参照
        - https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC
- 【課題2】Optunaを利用してハイパーパラメータを最適化し，グリッドサーチと比較してみましょう.
    - Optuna: https://optuna.org
    - 使い方は，Code Exmaplesを参照
    - グリッドサーチ同様に入れ子の交差検証を用いること
    - optunaでパラメータの生成範囲指定は下記を参照（関数 suggest_***）
        - https://optuna.readthedocs.io/en/stable/reference/generated/optuna.trial.Tria
- 【課題3】最適なカーネル関数およびハイパーパラメータ，そこから分かるデータの特徴について考察してみましょう.

In [4]:
```
# 外側ループのための交差検証用データ生成インスタンス
kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=1)

acc_trn_list = []  #外側ループのfold毎の学習データに対するaccuracy格納用
acc_tst_list = []  #外側ループのfold毎のテストデータに対するaccuracy格納用

# グリッドサーチのパラメータリスト
parameters = {'gamma':[0.01, 0.02, 0.05, 0.1, 0,2, 1, 10, 100]}
# 内側ループでグリッドサーチを行う交差検証インスタンス
gs = GridSearchCV(SVC(), parameters, cv=2)

k=0
for train_itr, test_itr in kfold.split(X, y):
    # 内側ループのグリッドサーチ
    gs.fit(X[train_itr], y[train_itr])
    print('Fold #{:2d}; Best Parameter: {}, Accuracy: {:.3f}'\
        .format(k+1,gs.best_params_,gs.best_score_))
    acc_trn_list.append(gs.score(X[train_itr],y[train_itr]))
    acc_tst_list.append(gs.score(X[test_itr],y[test_itr]))
    k=k+1
```

```
Fold # 1; Best Parameter: {'gamma': 0.02}, Accuracy: 0.969
Fold # 2; Best Parameter: {'gamma': 0.02}, Accuracy: 0.963
Fold # 3; Best Parameter: {'gamma': 0.02}, Accuracy: 0.967
Fold # 4; Best Parameter: {'gamma': 0.02}, Accuracy: 0.961
Fold # 5; Best Parameter: {'gamma': 0.02}, Accuracy: 0.967
Fold # 6; Best Parameter: {'gamma': 0.02}, Accuracy: 0.973
Fold # 7; Best Parameter: {'gamma': 0.02}, Accuracy: 0.961
Fold # 8; Best Parameter: {'gamma': 0.02}, Accuracy: 0.963
Fold # 9; Best Parameter: {'gamma': 0.02}, Accuracy: 0.955
Fold #10; Best Parameter: {'gamma': 0.05}, Accuracy: 0.959
```

平均Accuracy

In [5]:

```
print('Training data: %1.3f' % np.mean(acc_trn_list))
print('Test data: %1.3f' % np.mean(acc_tst_list))
```

```
Training data: 0.986
Test data: 0.974
```

【課題1】探索するパラメータにカーネル関数の追加や範囲を変更して最適パラメータを探してみましょう

In [18]:

```
# 外側ループのための交差検証用データ生成インスタンス
kfold = StratifiedKFold(n_splits=10, shuffle=True, random_state=1)

acc_trn_list = []  #外側ループのfold毎の学習データに対するaccuracy格納用
acc_tst_list = []  #外側ループのfold毎のテストデータに対するaccuracy格納用

# グリッドサーチのパラメータリスト
parameters = {
    'kernel': ['linear', 'poly', 'rbf'],  # 線形カーネル、多項式カーネル、RBFカ
    'C': [0.01, 0.1, 1, 10, 100],  # ハイパーパラメータC
    'degree': [2, 3, 4],  # 多項式カーネルの次数
    'gamma': [0.01, 0.02, 0.05, 0.1, 0.2, 1, 10, 100]  # RBFカーネルと多項式カー
}

# 内側ループでグリッドサーチを行う交差検証インスタンス
gs = GridSearchCV(SVC(), parameters, cv=2)

k=0
for train_itr, test_itr in kfold.split(X, y):
    # 内側ループのグリッドサーチ
    gs.fit(X[train_itr], y[train_itr])
    print('Fold #{:2d}; Best Parameter: {}, Accuracy: {:.3f}'\
        .format(k+1,gs.best_params_,gs.best_score_))
    acc_trn_list.append(gs.score(X[train_itr],y[train_itr]))
    acc_tst_list.append(gs.score(X[test_itr],y[test_itr]))
    k=k+1

print('Training data: %1.3f' % np.mean(acc_trn_list))
print('Test data: %1.3f' % np.mean(acc_tst_list))
```

```
Fold # 1; Best Parameter: {'C': 10, 'degree': 2, 'gamma': 0.02, 'kernel': 'rb
f'}, Accuracy: 0.979
Fold # 2; Best Parameter: {'C': 0.1, 'degree': 2, 'gamma': 0.01, 'kernel': 'line
ar'}, Accuracy: 0.971
Fold # 3; Best Parameter: {'C': 0.1, 'degree': 2, 'gamma': 0.01, 'kernel': 'line
ar'}, Accuracy: 0.973
Fold # 4; Best Parameter: {'C': 1, 'degree': 2, 'gamma': 0.01, 'kernel': 'linea
r'}, Accuracy: 0.973
Fold # 5; Best Parameter: {'C': 0.1, 'degree': 2, 'gamma': 0.01, 'kernel': 'line
ar'}, Accuracy: 0.973
Fold # 6; Best Parameter: {'C': 0.1, 'degree': 2, 'gamma': 0.01, 'kernel': 'line
ar'}, Accuracy: 0.979
Fold # 7; Best Parameter: {'C': 10, 'degree': 2, 'gamma': 0.01, 'kernel': 'rb
f'}, Accuracy: 0.979
Fold # 8; Best Parameter: {'C': 0.1, 'degree': 2, 'gamma': 0.01, 'kernel': 'line
ar'}, Accuracy: 0.975
Fold # 9; Best Parameter: {'C': 10, 'degree': 2, 'gamma': 0.01, 'kernel': 'rb
f'}, Accuracy: 0.975
Fold #10; Best Parameter: {'C': 10, 'degree': 2, 'gamma': 0.02, 'kernel': 'rb
f'}, Accuracy: 0.975
Training data: 0.987
Test data: 0.977
```

【課題2】Optunaを利用してハイパーパラメータを最適化し，グリッドサーチと比較してみましょう．

In [26]:
```
import optuna
from sklearn.model_selection import cross_val_score
```

In [32]:
```
def objective(trial):
    #  パラメータ
    C = trial.suggest_float('C', 1e-5, 1e5, log=True)
    kernel = trial.suggest_categorical('kernel', ['linear', 'poly', 'rbf'])
    degree = trial.suggest_int('degree', 2, 4)
    gamma = trial.suggest_float('gamma', 1e-5, 1e5, log=True)

    model = SVC(C=C, kernel=kernel, degree=degree, gamma=gamma)

    score = cross_val_score(model, X_train, y_train, cv=2)
    return score

acc_trn_list = []  #外側ループのfold毎の学習データに対するaccuracy格納用
acc_tst_list = []  #外側ループのfold毎のテストデータに対するaccuracy格納用
k=0
for train_idx, test_idx in kfold.split(X, y):
    X_train, X_test = X[train_idx], X[test_idx]
    y_train, y_test = y[train_idx], y[test_idx]

    study = optuna.create_study(direction='maximize')
    study.optimize(objective, n_trials=5)

    print(study.best_params)
    best_params = study.best_params

    C = best_params['C']
    kernel = best_params['kernel']
    degree = best_params['degree']
```

```
    gamma = best_params['gamma']

    model = SVC(C=C, kernel=kernel, degree=degree, gamma=gamma)
    model.fit(X_train, y_train)
    acc_trn_list.append(model.score(X_train, y_train))
    acc_tst_list.append(model.score(X_test, y_test))
    k=k+1


print('Training data: %1.3f' % np.mean(acc_trn_list))
print('Test data: %1.3f' % np.mean(acc_tst_list))
```

```
[I 2023-11-11 12:01:34,178] A new study created in memory with name: no-name-d76
9b912-3d24-43dc-9d3c-485ceea3c599
[W 2023-11-11 12:01:34,248] Trial 0 failed with parameters: {'C': 24752.29679769
1597, 'kernel': 'rbf', 'degree': 4, 'gamma': 440.8261362691303} because of the f
ollowing error: The value array([0.62890625, 0.62890625]) could not be cast to f
loat.
[W 2023-11-11 12:01:34,250] Trial 0 failed with value array([0.62890625, 0.62890
625]).
[W 2023-11-11 12:01:34,286] Trial 1 failed with parameters: {'C': 1.898726541369
6324e-05, 'kernel': 'poly', 'degree': 4, 'gamma': 0.002682243627946421} because
of the following error: The value array([0.62890625, 0.62890625]) could not be c
ast to float.
[W 2023-11-11 12:01:34,287] Trial 1 failed with value array([0.62890625, 0.62890
625]).
[W 2023-11-11 12:01:34,299] Trial 2 failed with parameters: {'C': 0.005397518348
408045, 'kernel': 'linear', 'degree': 3, 'gamma': 6630.147793078563} because of
the following error: The value array([0.9375    , 0.96484375]) could not be cast
to float.
[W 2023-11-11 12:01:34,300] Trial 2 failed with value array([0.9375    , 0.96484
375]).
[W 2023-11-11 12:01:34,310] Trial 3 failed with parameters: {'C': 0.300445875811
0252, 'kernel': 'linear', 'degree': 4, 'gamma': 665.6846757632077} because of th
e following error: The value array([0.9765625 , 0.97265625]) could not be cast t
o float.
[W 2023-11-11 12:01:34,312] Trial 3 failed with value array([0.9765625 , 0.97265
625]).
[W 2023-11-11 12:01:34,333] Trial 4 failed with parameters: {'C': 6217.775769076
074, 'kernel': 'poly', 'degree': 4, 'gamma': 8550.388825286664} because of the f
ollowing error: The value array([0.76171875, 0.74609375]) could not be cast to f
loat.
[W 2023-11-11 12:01:34,335] Trial 4 failed with value array([0.76171875, 0.74609
375]).
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
c:\Users\kio\zemi\Knowledge_Informatics\ex6.ipynb Cell 14 line 2
     <a href='vscode-notebook-cell:/c%3A/Users/kio/zemi/Knowledge_Informatics/ex6.ipynb#X20sZmlsZQ%3D%3D?line=19'>20</a> study = optuna.create_study(direction='maximize')
     <a href='vscode-notebook-cell:/c%3A/Users/kio/zemi/Knowledge_Informatics/ex6.ipynb#X20sZmlsZQ%3D%3D?line=20'>21</a> study.optimize(objective, n_trials=5)
---> <a href='vscode-notebook-cell:/c%3A/Users/kio/zemi/Knowledge_Informatics/ex6.ipynb#X20sZmlsZQ%3D%3D?line=22'>23</a> print(study.best_params)
     <a href='vscode-notebook-cell:/c%3A/Users/kio/zemi/Knowledge_Informatics/ex6.ipynb#X20sZmlsZQ%3D%3D?line=23'>24</a> best_params = study.best_params
     <a href='vscode-notebook-cell:/c%3A/Users/kio/zemi/Knowledge_Informatics/ex6.ipynb#X20sZmlsZQ%3D%3D?line=25'>26</a> C = best_params['C']

File c:\Users\kio\zemi\Knowledge_Informatics\.venv\lib\site-packages\optuna\study\study.py:114, in Study.best_params(self)
    102 @property
    103 def best_params(self) -> dict[str, Any]:
    104     """Return parameters of the best trial in the study.
    105 
    106     .. note::
   (...)
    111 
    112     """
--> 114     return self.best_trial.params

File c:\Users\kio\zemi\Knowledge_Informatics\.venv\lib\site-packages\optuna\study\study.py:157, in Study.best_trial(self)
    151 if self._is_multi_objective():
    152     raise RuntimeError(
    153         "A single best trial cannot be retrieved from a multi-objective study. Consider "
    154         "using Study.best_trials to retrieve a list containing the best trials."
    155     )
--> 157 return copy.deepcopy(self._storage.get_best_trial(self._study_id))

File c:\Users\kio\zemi\Knowledge_Informatics\.venv\lib\site-packages\optuna\storages\_in_memory.py:234, in InMemoryStorage.get_best_trial(self, study_id)
    231 best_trial_id = self._studies[study_id].best_trial_id
    233 if best_trial_id is None:
--> 234     raise ValueError("No trials are completed yet.")
    235 elif len(self._studies[study_id].directions) > 1:
    236     raise RuntimeError(
    237         "Best trial can be obtained only for single-objective optimization."
    238     )

ValueError: No trials are completed yet.
```

【課題3】最適なカーネル関数およびハイパーパラメータ，そこから分かるデータの特徴について考察してみましょう．

In [ ]: