03ml

May 21, 2023

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
D_train = np.loadtxt('sample_data/mnist_train_small.csv', delimiter=',')
X_train = D_train[:, 1:]
y_train = D_train[:, 0].astype(int)
```

[3]:
```
D_test = np.loadtxt('sample_data/mnist_test.csv', delimiter=',')
X_test = D_test[:, 1:]
y_test = D_test[:, 0].astype(int)
```

0.1 k-Nearest Neighbor Classification

[4]:
```
from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n_neighbors=7, metric='euclidean')
model.fit(X_train, y_train)
y_predicted = model.predict(X_train)
```

[5]:
```
from sklearn.metrics import accuracy_score

acc_train = accuracy_score(y_train, y_predicted)
print(f"{acc_train:.4f}")
```

```
0.9703
```

[6]:
```
y_predicted = model.predict(X_test)
acc_test = accuracy_score(y_test, y_predicted)
print(f"{acc_test:.4f}")
```

```
0.9572
```

0.2 Tuning hyperparameters

[ ]:
```
from sklearn.model_selection import GridSearchCV

parameters = {'n_neighbors':[3, 5, 7], 'metric':['manhattan', 'euclidean']}
knn = KNeighborsClassifier()
gcv = GridSearchCV(knn, parameters, cv=5, verbose=2)
```

```
gcv.fit(X_train, y_train)
```

[12]:
```
from sklearn.experimental import enable_halving_search_cv  # noqa
from sklearn.model_selection import HalvingGridSearchCV

parameters = {'n_neighbors':[3, 5, 7], 'metric':['manhattan', 'euclidean']}
knn = KNeighborsClassifier()
g = HalvingGridSearchCV(knn, parameters, cv=5, factor=2,
                        aggressive_elimination=False, verbose=2)
g.fit(X_train, y_train)
```

```
n_iterations: 3
n_required_iterations: 3
n_possible_iterations: 3
min_resources_: 5000
max_resources_: 20000
aggressive_elimination: False
factor: 2
----------
iter: 0
n_candidates: 6
n_resources: 5000
Fitting 5 folds for each of 6 candidates, totalling 30 fits
[CV] END ...metric=manhattan, n_neighbors=3; total time=   3.8s
[CV] END ...metric=manhattan, n_neighbors=3; total time=   3.8s
[CV] END ...metric=manhattan, n_neighbors=3; total time=   4.0s
[CV] END ...metric=manhattan, n_neighbors=3; total time=   3.8s
[CV] END ...metric=manhattan, n_neighbors=3; total time=   5.3s
[CV] END ...metric=manhattan, n_neighbors=5; total time=   3.7s
[CV] END ...metric=manhattan, n_neighbors=5; total time=   4.2s
[CV] END ...metric=manhattan, n_neighbors=5; total time=   3.7s
[CV] END ...metric=manhattan, n_neighbors=5; total time=   3.7s
[CV] END ...metric=manhattan, n_neighbors=5; total time=   5.1s
[CV] END ...metric=manhattan, n_neighbors=7; total time=   3.7s
[CV] END ...metric=manhattan, n_neighbors=7; total time=   5.3s
[CV] END ...metric=manhattan, n_neighbors=7; total time=   3.8s
[CV] END ...metric=manhattan, n_neighbors=7; total time=   4.7s
[CV] END ...metric=manhattan, n_neighbors=7; total time=   3.8s
[CV] END ...metric=euclidean, n_neighbors=3; total time=   0.3s
[CV] END ...metric=euclidean, n_neighbors=3; total time=   0.3s
[CV] END ...metric=euclidean, n_neighbors=3; total time=   0.3s
[CV] END ...metric=euclidean, n_neighbors=3; total time=   0.3s
[CV] END ...metric=euclidean, n_neighbors=3; total time=   0.3s
[CV] END ...metric=euclidean, n_neighbors=5; total time=   0.3s
[CV] END ...metric=euclidean, n_neighbors=5; total time=   0.3s
[CV] END ...metric=euclidean, n_neighbors=5; total time=   0.3s
[CV] END ...metric=euclidean, n_neighbors=5; total time=   0.7s
[CV] END ...metric=euclidean, n_neighbors=5; total time=   0.4s
```

```
[CV] END ...metric=euclidean, n_neighbors=7; total time=   0.3s
[CV] END ...metric=euclidean, n_neighbors=7; total time=   0.3s
[CV] END ...metric=euclidean, n_neighbors=7; total time=   0.3s
[CV] END ...metric=euclidean, n_neighbors=7; total time=   0.3s
[CV] END ...metric=euclidean, n_neighbors=7; total time=   0.3s
----------
iter: 1
n_candidates: 3
n_resources: 10000
Fitting 5 folds for each of 3 candidates, totalling 15 fits
[CV] END ...metric=euclidean, n_neighbors=5; total time=   1.0s
[CV] END ...metric=euclidean, n_neighbors=5; total time=   1.0s
[CV] END ...metric=euclidean, n_neighbors=5; total time=   1.0s
[CV] END ...metric=euclidean, n_neighbors=5; total time=   1.1s
[CV] END ...metric=euclidean, n_neighbors=5; total time=   1.0s
[CV] END ...metric=euclidean, n_neighbors=7; total time=   1.0s
[CV] END ...metric=euclidean, n_neighbors=7; total time=   1.6s
[CV] END ...metric=euclidean, n_neighbors=7; total time=   1.0s
[CV] END ...metric=euclidean, n_neighbors=7; total time=   1.0s
[CV] END ...metric=euclidean, n_neighbors=7; total time=   1.0s
[CV] END ...metric=euclidean, n_neighbors=3; total time=   1.0s
[CV] END ...metric=euclidean, n_neighbors=3; total time=   2.0s
[CV] END ...metric=euclidean, n_neighbors=3; total time=   1.0s
[CV] END ...metric=euclidean, n_neighbors=3; total time=   1.0s
[CV] END ...metric=euclidean, n_neighbors=3; total time=   1.0s
----------
iter: 2
n_candidates: 2
n_resources: 20000
Fitting 5 folds for each of 2 candidates, totalling 10 fits
[CV] END ...metric=euclidean, n_neighbors=7; total time=   3.7s
[CV] END ...metric=euclidean, n_neighbors=7; total time=   4.4s
[CV] END ...metric=euclidean, n_neighbors=7; total time=   3.6s
[CV] END ...metric=euclidean, n_neighbors=7; total time=   4.7s
[CV] END ...metric=euclidean, n_neighbors=7; total time=   3.6s
[CV] END ...metric=euclidean, n_neighbors=3; total time=   4.8s
[CV] END ...metric=euclidean, n_neighbors=3; total time=   3.8s
[CV] END ...metric=euclidean, n_neighbors=3; total time=   4.6s
[CV] END ...metric=euclidean, n_neighbors=3; total time=   3.6s
[CV] END ...metric=euclidean, n_neighbors=3; total time=   5.0s
```

[12]:
```
HalvingGridSearchCV(estimator=KNeighborsClassifier(), factor=2,
                    param_grid={'metric': ['manhattan', 'euclidean'],
                                'n_neighbors': [3, 5, 7]},
                    verbose=2)
```

[13]:
```
g.best_estimator_
```

[13]: KNeighborsClassifier(metric='euclidean', n_neighbors=3)