Tests_and_validation

January 30, 2019

In [2]:
```
from sklearn.datasets import load_digits
digits = load_digits()
print(digits.DESCR)
x = digits.data
y = digits.target
```

```
.. _digits_dataset:

Optical recognition of handwritten digits dataset
--------------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 5620
    :Number of Attributes: 64
    :Attribute Information: 8x8 image of integer pixels in the range 0..16.
    :Missing Attribute Values: None
    :Creator: E. Alpaydin (alpaydin '@' boun.edu.tr)
    :Date: July; 1998

This is a copy of the test set of the UCI ML hand-written digits datasets
http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits

The data set contains images of hand-written digits: 10 classes where
each class refers to a digit.

Preprocessing programs made available by NIST were used to extract
normalized bitmaps of handwritten digits from a preprinted form. From a
total of 43 people, 30 contributed to the training set and different 13
to the test set. 32x32 bitmaps are divided into nonoverlapping blocks of
4x4 and the number of on pixels are counted in each block. This generates
an input matrix of 8x8 where each element is an integer in the range
0..16. This reduces dimensionality and gives invariance to small
distortions.

For info on NIST preprocessing routines, see M. D. Garris, J. L. Blue, G.
T. Candela, D. L. Dimmick, J. Geist, P. J. Grother, S. A. Janet, and C.
L. Wilson, NIST Form-Based Handprint Recognition System, NISTIR 5469,
```

```
1994.

.. topic:: References

  - C. Kaynak (1995) Methods of Combining Multiple Classifiers and Their
    Applications to Handwritten Digit Recognition, MSc Thesis, Institute of
    Graduate Studies in Science and Engineering, Bogazici University.
  - E. Alpaydin, C. Kaynak (1998) Cascading Classifiers, Kybernetika.
  - Ken Tang and Ponnuthurai N. Suganthan and Xi Yao and A. Kai Qin.
    Linear dimensionalityreduction using relevance weighted LDA. School of
    Electrical and Electronic Engineering Nanyang Technological University.
    2005.
  - Claudio Gentile. A New Approximate Maximal Margin Classification
    Algorithm. NIPS. 2000.
```

```
In [3]: x[0]
```

```
Out[3]: array([ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.,  0.,  0., 13., 15., 10.,
               15.,  5.,  0.,  0.,  3., 15.,  2.,  0., 11.,  8.,  0.,  0.,  4.,
               12.,  0.,  0.,  8.,  8.,  0.,  0.,  5.,  8.,  0.,  0.,  9.,  8.,
                0.,  0.,  4., 11.,  0.,  1., 12.,  7.,  0.,  0.,  2., 14.,  5.,
               10., 12.,  0.,  0.,  0.,  0.,  6., 13., 10.,  0.,  0.,  0.])
```

```
In [10]: # hypotheses
         from sklearn import svm
         h1 = svm.LinearSVC(C=1.0)
         h2 = svm.SVC(kernel='rbf', degree=3, gamma=0.001, C=1.0)# classifier with a radial ba
         h3 = svm.SVC(kernel='poly', degree=3,gamma='auto', C=1.0) # third degree polynomial c
```

```
In [6]: h1.fit(x, y)
        print(h1.score(x, y)) # the effectiveness of the algorithm, tested for the same data a
```

```
0.9905397885364496
```

```
d:\python\lib\site-packages\sklearn\svm\base.py:931: ConvergenceWarning: Liblinear failed to c
  "the number of iterations.", ConvergenceWarning)
```

```
In [7]: h2.fit(x, y)
        print(h2.score(x, y)) # the effectiveness of the algorithm, tested for the same data a
```

```
0.9988870339454646
```

```
In [11]: h3.fit(x, y)
         print(h3.score(x, y)) # the effectiveness of the algorithm, tested for the same data
```

```
1.0
```

```
In [14]: from sklearn.model_selection import train_test_split
         random_state = 1
         X_train, X_test, Y_train, Y_test = train_test_split(x,y,
                                                             test_size=0.30,
                                                             random_state = random_state) # 30

         h1.fit(X_train, Y_train)
         print(h1.score(X_test, Y_test))

         h2.fit(X_train, Y_train)
         print(h2.score(X_test, Y_test))

         h3.fit(X_train, Y_train)
         print(h3.score(X_test, Y_test))
```

```
0.9555555555555556
0.9907407407407407
0.9851851851851852
```

```
d:\python\lib\site-packages\sklearn\svm\base.py:931: ConvergenceWarning: Liblinear failed to co
  "the number of iterations.", ConvergenceWarning)
```

```
In [17]: from sklearn.model_selection import train_test_split
         random_state = 1
         X_train, X_validation_test, Y_train, Y_validation_test = train_test_split(x,y,
                                                                                   test_size=0.40
                                                                                   random_state =


         X_validation, X_test, Y_validation, Y_test = train_test_split(X_validation_test,Y_vali
                                                                       test_size=0.50,
                                                                       random_state = random_sta

         for hypothesis in [h1, h2, h3]:
             hypothesis.fit(X_train, Y_train)
             print ("%s\n -> average effectiveness at the validation stage = %s" %
                    (hypothesis, hypothesis.score(X_validation, Y_validation)))
             print ("%s\n -> average effectiveness at the test stage = %s" %
                    (hypothesis, hypothesis.score(X_test, Y_test)))
```

```
LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
     intercept_scaling=1, loss='squared_hinge', max_iter=1000,
     multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
     verbose=0)
 -> average effectiveness at the validation stage = 0.947075208913649
LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
     intercept_scaling=1, loss='squared_hinge', max_iter=1000,
     multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
     verbose=0)
```

```
 -> average effectiveness at the test stage = 0.9388888888888889
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
 -> average effectiveness at the validation stage = 0.9916434540389972
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
 -> average effectiveness at the test stage = 0.9777777777777777
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='poly',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
 -> average effectiveness at the validation stage = 0.9888579387186629
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='poly',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
 -> average effectiveness at the test stage = 0.975


d:\python\lib\site-packages\sklearn\svm\base.py:931: ConvergenceWarning: Liblinear failed to co
  "the number of iterations.", ConvergenceWarning)
```

In [ ]: