Exemplos - Classificação para MNIST com SVM

Thiago Teixeira Santos
thiago.santos@embrapa.br

24 de novembro de 2017

```
In [1]: import numpy as np

        # Importa o módulo de plotagem
        import matplotlib.pyplot as plt

        # Configura a matplotlib para operar interativamente no notebook.
        # Para detalhes, execute: %matplotlib?
        %matplotlib inline

        # Configura o tamanho padrão da figura largura X altura, em polegadas
        plt.rcParams['figure.figsize'] = (10, 8)
```

1 Reconhecimento de dígitos - a famosa base MNIST

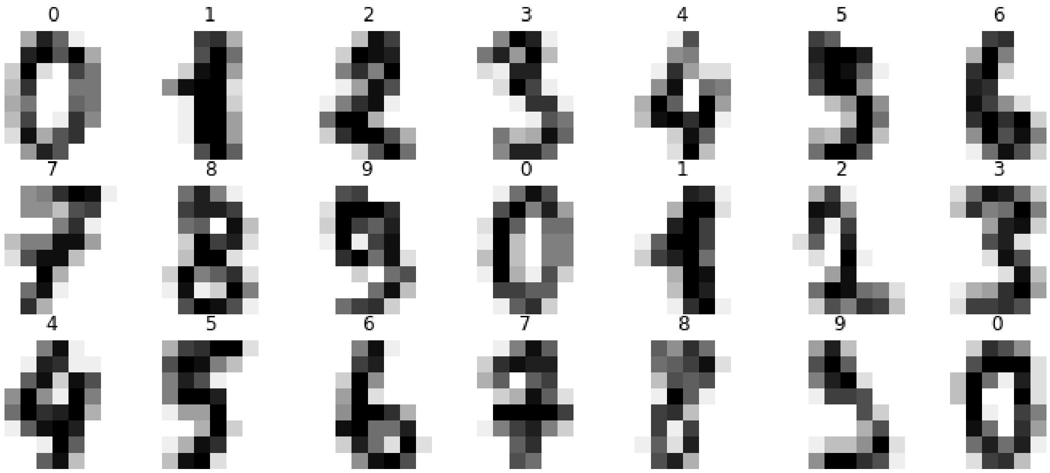
```
In [2]: # Import datasets, classifiers and performance metrics
        from sklearn import datasets, svm, metrics

        # The digits dataset
        digits = datasets.load_digits()
```

```
In [3]: print digits.DESCR
```

```
Optical Recognition of Handwritten Digits Data Set
===================================================

Notes
-----
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.

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 [4]: fig = plt.figure(figsize=(12,5))
        n_plot = 1
        for I, label in zip(digits.images, digits.target):
            plt.subplot(3, 7, n_plot)
            plt.axis('off')
            plt.imshow(I, cmap=plt.cm.gray_r, interpolation='nearest')
            plt.title('%d' % label)
            n_plot += 1
            if n_plot > 21:
                break
```



```
In [5]: from sklearn.model_selection import ShuffleSplit
        s = ShuffleSplit(1, test_size=0.30)
        data = digits.images.reshape((-1, 64))
        gs = s.split(data)

        # Obtém a divisão treinamento e teste
        train, test = gs.next()
        X_train, y_train = data[train], digits.target[train]
        X_test, y_test = data[test], digits.target[test]
```

```
In [6]: clf = svm.SVC(C=1, kernel='rbf', gamma=0.001)
        clf.fit(X_train, y_train)

        y_pred = clf.predict(X_test)
```

```
In [7]: print metrics.classification_report(y_test, y_pred)
```

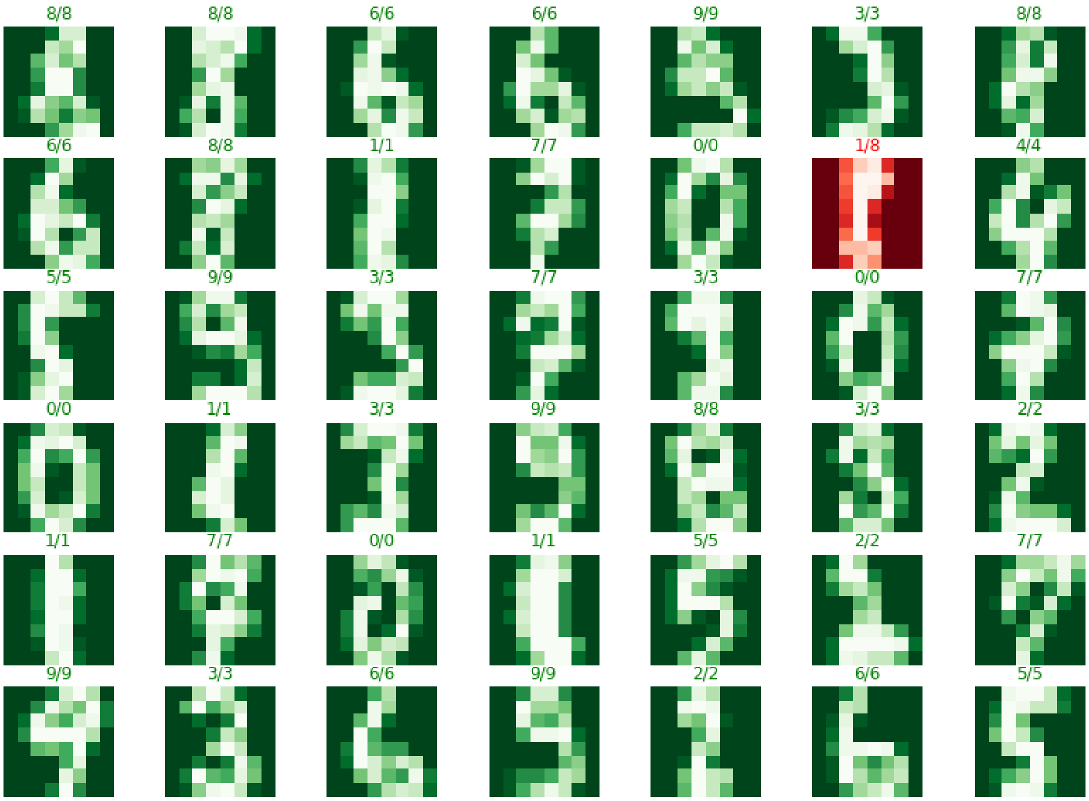
```
             precision    recall  f1-score   support

          0       1.00      1.00      1.00        45
          1       0.96      1.00      0.98        53
          2       0.98      1.00      0.99        52
          3       1.00      0.97      0.98        60
          4       1.00      1.00      1.00        61
          5       1.00      0.98      0.99        61
          6       0.98      1.00      0.99        56
          7       0.96      0.98      0.97        45
          8       1.00      0.95      0.97        61
          9       0.96      0.98      0.97        46
```

```
avg / total       0.99      0.99      0.99       540
```

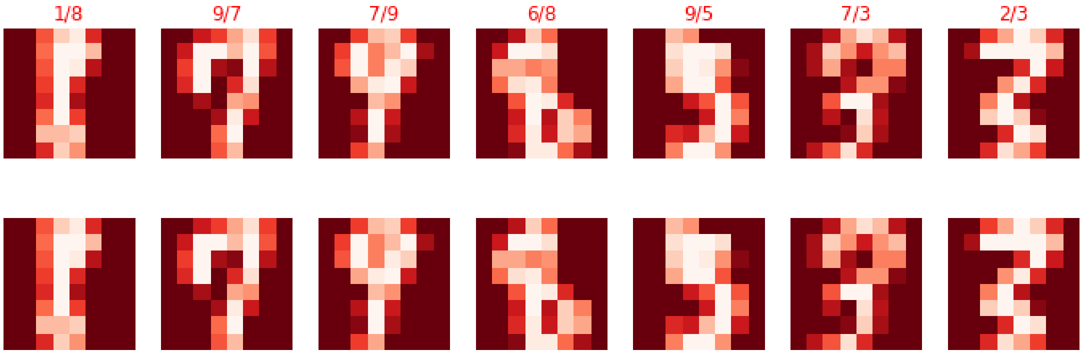
```
In [8]: print metrics.confusion_matrix(y_test, y_pred)
```

```
[[45  0  0  0  0  0  0  0  0  0]
 [ 0 53  0  0  0  0  0  0  0  0]
 [ 0  0 52  0  0  0  0  0  0  0]
 [ 0  0  1 58  0  0  0  1  0  0]
 [ 0  0  0  0 61  0  0  0  0  0]
 [ 0  0  0  0  0 60  0  0  0  1]
 [ 0  0  0  0  0  0 56  0  0  0]
 [ 0  0  0  0  0  0  0 44  0  1]
 [ 0  2  0  0  0  0  1  0 58  0]
 [ 0  0  0  0  0  0  0  1  0 45]]
```

```
In [9]: fig = plt.figure(figsize=(14,10))

        n_plot = 1
        for Xi, yi, yip in zip(digits.images[test], y_test, y_pred):
            plt.subplot(6, 7, n_plot)
            plt.axis('off')
            if yi == yip:
                plt.imshow(Xi.reshape(8,8), cmap=plt.cm.Greens_r, interpolation='nearest')
                plt.title('%d/%d' % (yip, yi), color='green')
            else:
                plt.imshow(Xi.reshape(8,8), cmap=plt.cm.Reds_r, interpolation='nearest')
                plt.title('%d/%d' % (yip, yi), color='red')

            n_plot += 1
            if n_plot > 42:
                break
```



```
In [10]: fig = plt.figure(figsize=(12,4))

         n_plot = 1
         for Xi, yi, yip in zip(digits.images[test], y_test, y_pred):
             if yi != yip:
                 plt.subplot(2, 7, n_plot)
                 plt.axis('off')
                 plt.imshow(Xi.reshape(8,8), cmap=plt.cm.Reds_r, interpolation='nearest')
                 plt.title('%d/%d' % (yip, yi), color='red')

                 plt.subplot(2, 7, n_plot+7)
                 plt.axis('off')
                 plt.imshow(Xi.reshape(8,8), cmap=plt.cm.Reds_r)

                 n_plot += 1
                 if n_plot > 7:
                     break
```


1/8
9/7
7/9
6/8
9/5
7/3
2/3