KNN

March 3, 2019

In [8]:
```
#k-nearest neighbors
#from sklearn.utils import shuffle
from sklearn.datasets import fetch_mldata
from sklearn.model_selection import train_test_split
import pickle

mnist = pickle.load(open('mnist.pickle','rb'))
dir(mnist.train)
mnist.train.images, mnist.train.labels
mnist.train.images.shape
```

Out[8]: (55000, 784)

In [10]:
```
mnist_data = mnist.train.images[:10000]
mnist_target = mnist.train.labels[:10000]
```

In [11]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(mnist_data, mnist_target, test_si
```

In [12]:
```
from sklearn.neighbors import KNeighborsClassifier
clf = KNeighborsClassifier(3)
clf.fit(X_train, Y_train)
```

Out[12]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=None, n_neighbors=3, p=2,
           weights='uniform')
```

In [14]:
```
y_pred = clf.predict(X_test)
from sklearn.metrics import classification_report
print(classification_report(Y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       224
           1       0.95      1.00      0.97       219
           2       0.99      0.92      0.96       176
           3       0.96      0.96      0.96       203
           4       0.96      0.95      0.95       191
           5       0.95      0.95      0.95       185
```

```
           6       0.95      0.98      0.97       197
           7       0.96      0.97      0.96       185
           8       0.99      0.84      0.91       200
           9       0.93      0.96      0.94       220

   micro avg       0.96      0.95      0.96      2000
   macro avg       0.96      0.95      0.95      2000
weighted avg       0.96      0.95      0.96      2000
 samples avg       0.95      0.95      0.95      2000



d:\python\lib\site-packages\sklearn\metrics\classification.py:1143: UndefinedMetricWarning: Pre
  'precision', 'predicted', average, warn_for)


In [15]: %timeit clf.fit(X_train, Y_train)

792 ms ś 17.9 ms per loop (mean ś std. dev. of 7 runs, 1 loop each)


In [16]: %timeit clf.predict(X_test)

22.7 s ś 554 ms per loop (mean ś std. dev. of 7 runs, 1 loop each)


In [ ]:
```