Sklearn

Визуализация данных

In [1]:
```
from sklearn import datasets

import numpy as np
```

In [2]:
```
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

Загрузка выборки

In [3]:
```
digits  =  datasets.load_digits()
```

In [5]:
```
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 (http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Hand
written+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 o
f
    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.
```

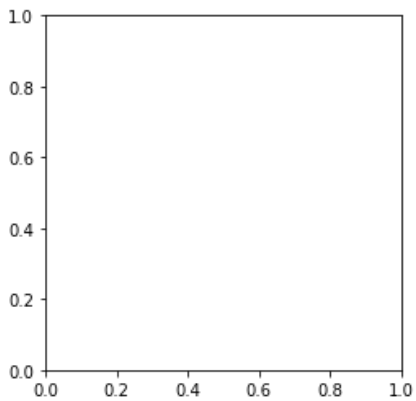
In [6]:
```
print 'target:', digits.target[0]
print 'features: \n', digits.data[0]
print 'number of features:', len(digits.data[0])
```

```
target: 0
features:
[  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.]
number of features: 64
```

Визуализация объектов выборки

In [7]:
```
#не будет работать: Invalid dimensions for image data
pylab.imshow(digits.data[0])
```

```
TypeErrorTraceback (most recent call last)
<ipython-input-7-b65189543ec9> in <module>()
      1 #не будет работать: Invalid dimensions for image data
----> 2 pylab.imshow(digits.data[0])

/home/sigor/anaconda2/lib/python2.7/site-packages/matplotlib/pyplot.pyc in
imshow(X, cmap, norm, aspect, interpolation, alpha, vmin, vmax, origin, ex
tent, shape, filternorm, filterrad, imlim, resample, url, hold, data, **kw
args)
   3155                         filternorm=filternorm, filterrad=filterrad
,
   3156                         imlim=imlim, resample=resample, url=url, d
ata=data,
-> 3157                         **kwargs)
   3158     finally:
   3159         ax._hold = washold

/home/sigor/anaconda2/lib/python2.7/site-packages/matplotlib/__init__.pyc
in inner(ax, *args, **kwargs)
   1896                     warnings.warn(msg % (label_namer, func.__name_
_),
   1897                                   RuntimeWarning, stacklevel=2)
-> 1898             return func(ax, *args, **kwargs)
   1899         pre_doc = inner.__doc__
   1900         if pre_doc is None:

/home/sigor/anaconda2/lib/python2.7/site-packages/matplotlib/axes/_axes.py
c in imshow(self, X, cmap, norm, aspect, interpolation, alpha, vmin, vmax,
origin, extent, shape, filternorm, filterrad, imlim, resample, url, **kwar
gs)
   5122                                 resample=resample, **kwargs)
   5123
-> 5124         im.set_data(X)
   5125         im.set_alpha(alpha)
   5126         if im.get_clip_path() is None:

/home/sigor/anaconda2/lib/python2.7/site-packages/matplotlib/image.pyc in
set_data(self, A)
    598         if (self._A.ndim not in (2, 3) or
    599                 (self._A.ndim == 3 and self._A.shape[-1] not in (3
, 4))):
--> 600             raise TypeError("Invalid dimensions for image data")
    601
    602         self._imcache = None

TypeError: Invalid dimensions for image data
```



In [8]:
```
digits.data[0].shape
```

Out[8]: (64,)

In [9]:
```
print digits.data[0].reshape(8,8)
```

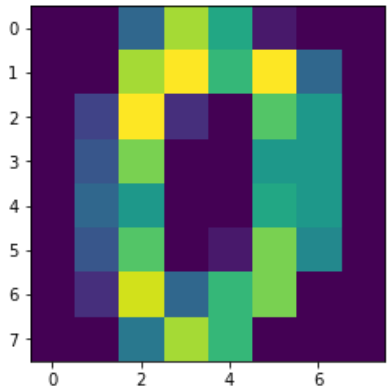
```
[[  0.   0.   5.  13.   9.   1.   0.   0.]
 [  0.   0.  13.  15.  10.  15.   5.   0.]
 [  0.   3.  15.   2.   0.  11.   8.   0.]
 [  0.   4.  12.   0.   0.   8.   8.   0.]
 [  0.   5.   8.   0.   0.   9.   8.   0.]
 [  0.   4.  11.   0.   1.  12.   7.   0.]
 [  0.   2.  14.   5.  10.  12.   0.   0.]
 [  0.   0.   6.  13.  10.   0.   0.   0.]]
```

In [10]:
```
digits.data[0].reshape(8,8).shape
```

Out[10]: (8, 8)

In [11]:
```
pylab.imshow(digits.data[0].reshape(8,8))
```

Out[11]: <matplotlib.image.AxesImage at 0x7f30259ac1d0>



In [12]:
```
print digits.keys()
```

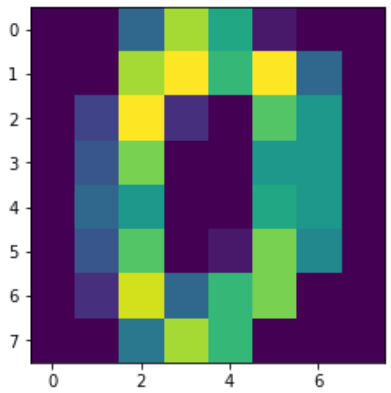
```
['images', 'data', 'target_names', 'DESCR', 'target']
```

In [13]:
```
print digits.images[0]
```

```
[[  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.]]
```

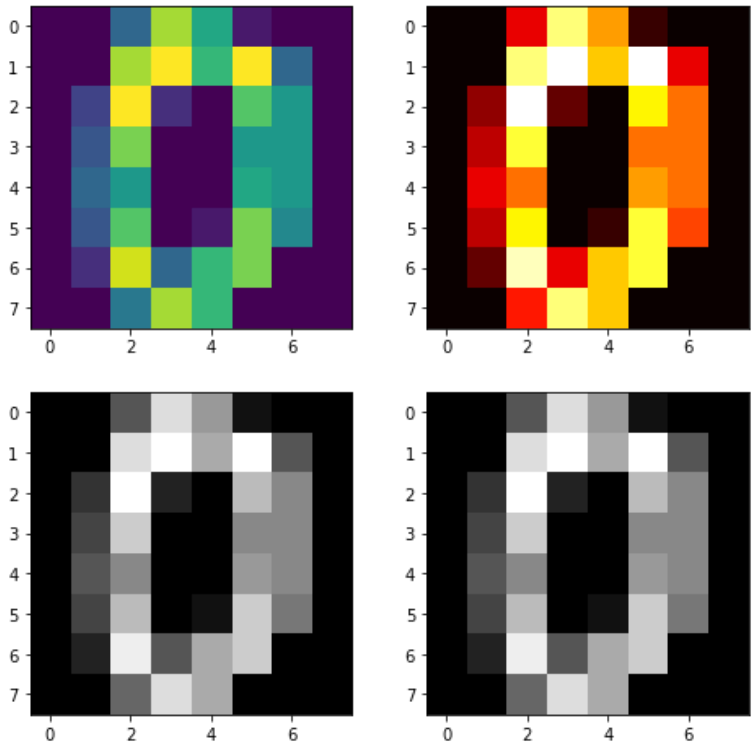
In [14]:
```
pylab.imshow(digits.images[0])
```

Out[14]: <matplotlib.image.AxesImage at 0x7f3025978410>

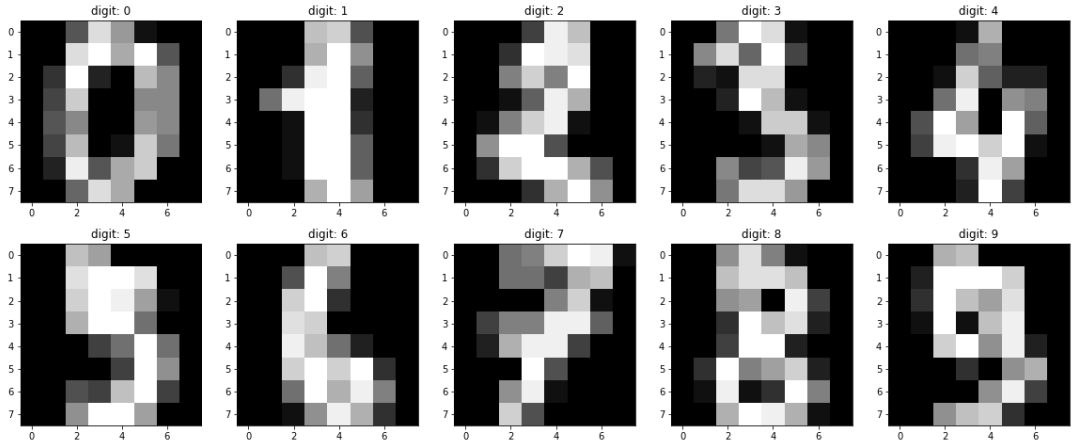


In [15]:
```
pyplot.figure(figsize(8, 8))

pyplot.subplot(2, 2, 1)
pylab.imshow(digits.images[0])

pyplot.subplot(2, 2, 2)
pylab.imshow(digits.images[0], cmap = 'hot')

pyplot.subplot(2, 2, 3)
pylab.imshow(digits.images[0], cmap = 'gray')

pyplot.subplot(2, 2, 4)
pylab.imshow(digits.images[0], cmap = 'gray', interpolation = 'nearest')
```

Out[15]: <matplotlib.figure.Figure at 0x7f30257a7e50>



In [16]:
```
pyplot.figure(figsize(20, 8))

for plot_number, plot in enumerate(digits.images[:10]):
    pyplot.subplot(2, 5, plot_number + 1)
    pylab.imshow(plot, cmap = 'gray')
    pylab.title('digit: ' + str(digits.target[plot_number]))
```



Уменьшение размерности

In [17]:
```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report

from collections import  Counter
```

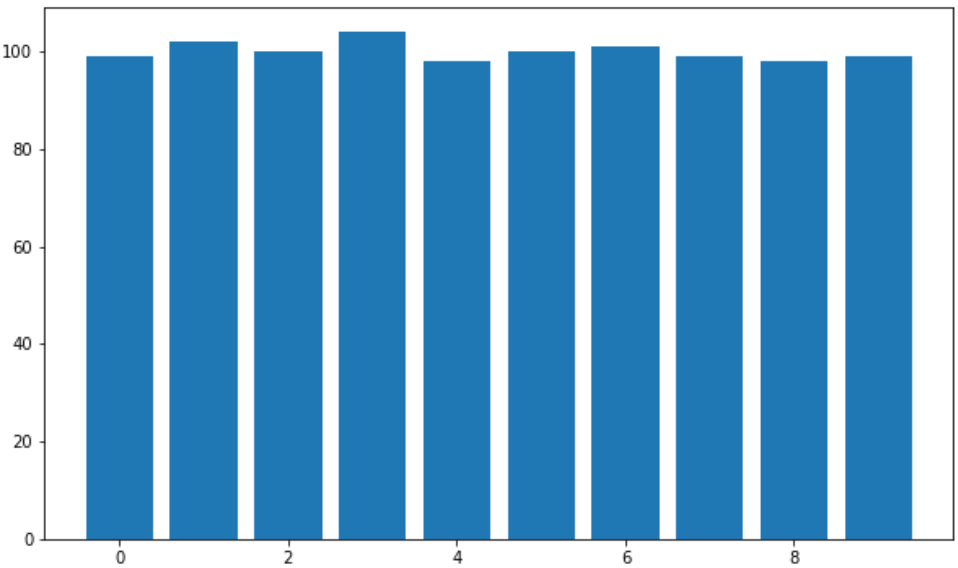
In [18]:
```
data = digits.data[:1000]
labels = digits.target[:1000]
```

In [19]:
```
print Counter(labels)
```

```
Counter({3: 104, 1: 102, 6: 101, 2: 100, 5: 100, 0: 99, 7: 99, 9: 99, 4: 9
8, 8: 98})
```

In [20]:
```
pylab.figure(figsize = (10, 6))
pylab.bar(Counter(labels).keys(), Counter(labels).values())
```

Out[20]: <Container object of 10 artists>



In [22]:
```
classifier = KNeighborsClassifier()
```

In [23]:
```
classifier.fit(data, labels)
```

Out[23]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=5, p=2,
           weights='uniform')
```

In [24]:
```
print classification_report(classifier.predict(data), labels)
```

```
             precision    recall  f1-score   support

          0       1.00      1.00      1.00        99
          1       1.00      0.97      0.99       105
          2       1.00      1.00      1.00       100
          3       1.00      0.98      0.99       106
          4       1.00      1.00      1.00        98
          5       0.99      1.00      0.99        99
          6       1.00      1.00      1.00       101
          7       0.99      0.99      0.99        99
          8       0.97      0.99      0.98        96
          9       0.96      0.98      0.97        97

avg / total       0.99      0.99      0.99      1000
```

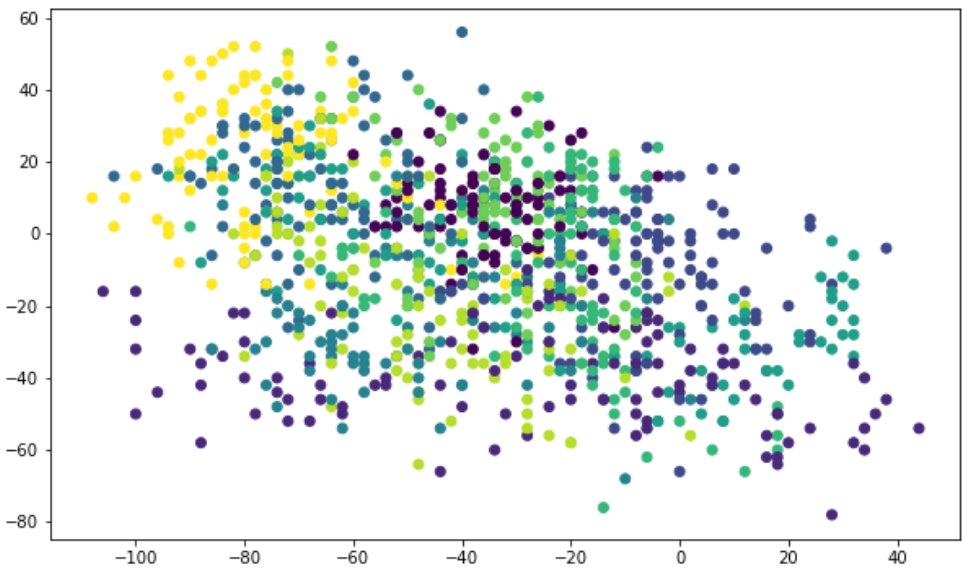
Random projection

In [25]:
```
from sklearn import random_projection
```

In [26]:
```
projection = random_projection.SparseRandomProjection(n_components = 2, ran
data_2d_rp = projection.fit_transform(data)
```

In [27]:
```
pylab.figure(figsize=(10, 6))
pylab.scatter(data_2d_rp[:, 0], data_2d_rp[:, 1], c = labels)
```

Out[27]:
```
<matplotlib.collections.PathCollection at 0x7f3023c39590>
```



In [28]:
```
classifier.fit(data_2d_rp, labels)
print classification_report(classifier.predict(data_2d_rp), labels)
```

```
             precision    recall  f1-score   support

          0       0.74      0.47      0.58       154
          1       0.75      0.58      0.65       131
          2       0.67      0.59      0.63       113
          3       0.71      0.51      0.60       144
          4       0.38      0.56      0.45        66
          5       0.42      0.53      0.47        79
          6       0.52      0.57      0.55        93
          7       0.35      0.51      0.42        69
          8       0.37      0.58      0.45        62
          9       0.62      0.69      0.65        89

avg / total       0.60      0.55      0.56      1000
```

PCA

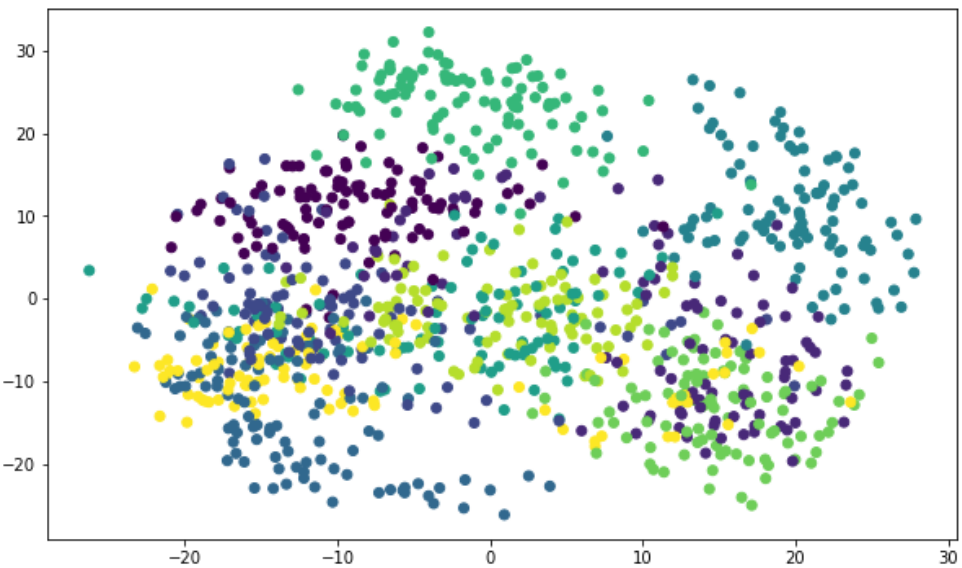
In [29]:
```
from sklearn.decomposition import RandomizedPCA
```

In [30]:
```
pca = RandomizedPCA(n_components = 2, random_state = 0)
data_2d_pca = pca.fit_transform(data)
```

```
/home/sigor/anaconda2/lib/python2.7/site-packages/sklearn/utils/deprecatio
n.py:58: DeprecationWarning: Class RandomizedPCA is deprecated; Randomized
PCA was deprecated in 0.18 and will be removed in 0.20. Use PCA(svd_solver
='randomized') instead. The new implementation DOES NOT store whiten ``com
ponents_``. Apply transform to get them.
  warnings.warn(msg, category=DeprecationWarning)
```

In [31]:
```
pylab.figure(figsize = (10, 6))
pylab.scatter(data_2d_pca[:, 0], data_2d_pca[:, 1], c = labels)
```

Out[31]:
```
<matplotlib.collections.PathCollection at 0x7f301bb89ad0>
```



In [32]:
```
classifier.fit(data_2d_pca, labels)
print classification_report(classifier.predict(data_2d_pca), labels)
```

```
             precision    recall  f1-score   support

          0       0.83      0.73      0.77       113
          1       0.56      0.54      0.55       105
          2       0.59      0.56      0.58       105
          3       0.77      0.79      0.78       101
          4       0.95      0.93      0.94       100
          5       0.56      0.54      0.55       104
          6       0.92      0.93      0.93       100
          7       0.75      0.71      0.73       104
          8       0.62      0.66      0.64        92
          9       0.52      0.67      0.58        76

avg / total       0.71      0.71      0.71      1000
```

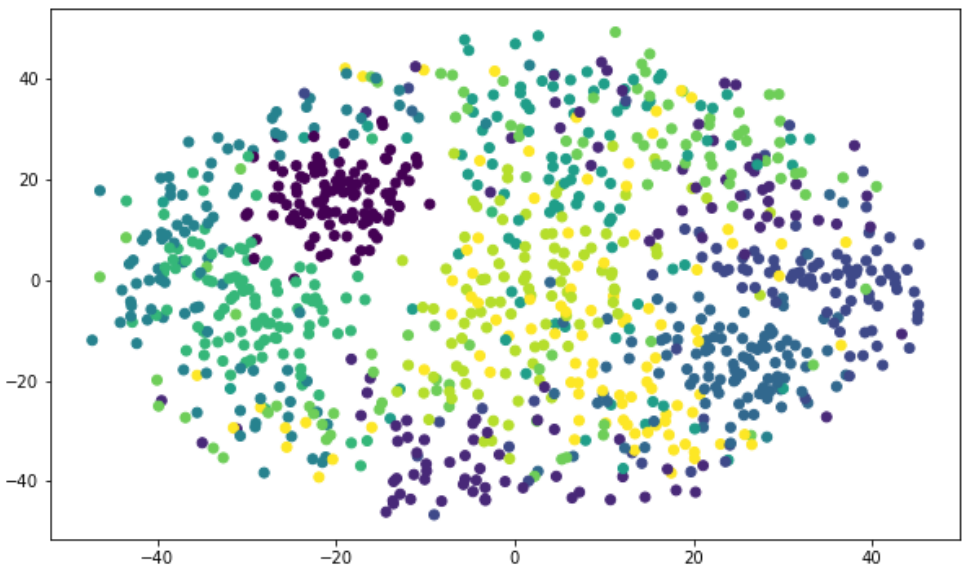
MDS

In [33]:
```
from sklearn import manifold
```

In [37]:
```
mds = manifold.MDS(n_components = 2, n_init = 1, max_iter = 100)
data_2d_mds = mds.fit_transform(data)
```

In [38]:
```
pylab.figure(figsize=(10, 6))
pylab.scatter(data_2d_mds[:, 0], data_2d_mds[:, 1], c = labels)
```

Out[38]: <matplotlib.collections.PathCollection at 0x7f3019f2bb50>



In [39]:
```
classifier.fit(data_2d_mds, labels)
print classification_report(classifier.predict(data_2d_mds), labels)
```

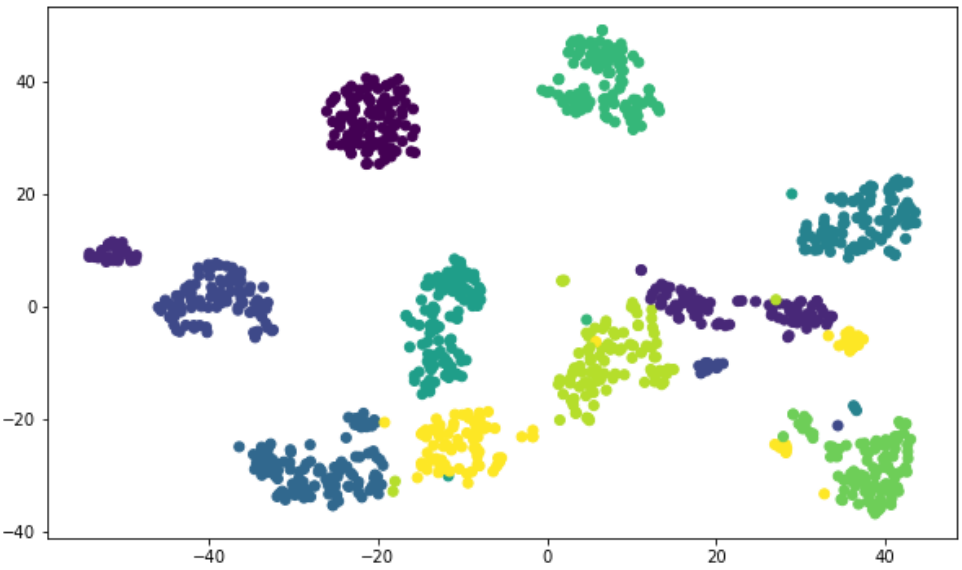
```
             precision    recall  f1-score   support

          0       0.96      0.90      0.93       106
          1       0.66      0.58      0.62       115
          2       0.76      0.75      0.75       102
          3       0.83      0.83      0.83       104
          4       0.77      0.68      0.72       111
          5       0.68      0.60      0.64       113
          6       0.83      0.84      0.84       100
          7       0.60      0.63      0.61        94
          8       0.62      0.59      0.61       103
          9       0.34      0.65      0.45        52

avg / total       0.72      0.70      0.71      1000
```

t- SNE

In [40]:
```
tsne = manifold.TSNE(n_components = 2, init = 'pca', random_state = 0)
data_2d_tsne = tsne.fit_transform(data)
```

In [41]:
```
pylab.figure(figsize = (10, 6))
pylab.scatter(data_2d_tsne[:, 0], data_2d_tsne[:, 1], c = labels)
```

Out[41]: <matplotlib.collections.PathCollection at 0x7f301a6c2150>



In [42]:
```
classifier.fit(data_2d_tsne, labels)
print classification_report(classifier.predict(data_2d_tsne), labels)
```

```
             precision    recall  f1-score   support

          0       1.00      1.00      1.00        99
          1       1.00      0.99      1.00       103
          2       0.99      1.00      0.99        99
          3       1.00      0.97      0.99       107
          4       1.00      0.99      0.99        99
          5       0.98      1.00      0.99        98
          6       0.99      1.00      1.00       100
          7       0.99      0.98      0.98       100
          8       0.97      0.98      0.97        97
          9       0.97      0.98      0.97        98

avg / total       0.99      0.99      0.99      1000
```

In [ ]: