In [1]:

```
import csv
f=open("convert.csv")
```

```
---------------------------------------------------------------------------
IOError                                   Traceback (most recent call last)
<ipython-input-1-0561ccb68587> in <module>()
      1 import csv
----> 2 f=open("convert.csv")

IOError: [Errno 2] No such file or directory: 'convert.csv'
```

In [2]:

```
import os
```

In [3]:

```
directory=os.path.join("C:\Users\LOSHA1\Desktop\MNIST-dataset-in-different-formats-master\data\CSV for
mat>python convert.py")
```

In [4]:

```
for root,dirs,files in os.walk(directory):
    for file in files:
        if file.endswith(".csv"):
            f=open(file,'r')
            f.close()
```

In [5]:

```
print(f)
```

```
<built-in method f of mtrand.RandomState object at 0x0529F990>
```

In [6]:

```
import glob
import csv
```

for

In [8]:

```
for f_name in glob.glob("mnist_test.csv")
with open(f_name)as f:
    reader =csv.reader(f)
```

```
  File "<ipython-input-8-7a123ffd3230>", line 1
    for f_name in glob.glob("mnist_test.csv")
                                             ^
SyntaxError: invalid syntax
```

In [9]:

```
for f_name in glob.glob("mnist_test.csv")
with open(f_name)as f:
    reader =csv.reader(f)
```

```
  File "<ipython-input-9-ae514626a5ba>", line 1
    for f_name in glob.glob("mnist_test.csv")
                                             ^
SyntaxError: invalid syntax
```

In [11]:

```
from sklearn.datasets import fetch_mldata
mnist = fetch_mldata('MNIST original')
```

In [12]:

```
from sklearn.datasets import fetch_mldata
mnist = fetch_mldata('MNIST original')
```

In [13]:

```
X_digits, y_digits = mnist.data, mnist.target
```

In [14]:

```
print(X_digits)
```

```
[[0 0 0 ..., 0 0 0]
 [0 0 0 ..., 0 0 0]
 [0 0 0 ..., 0 0 0]
 ...,
 [0 0 0 ..., 0 0 0]
 [0 0 0 ..., 0 0 0]
 [0 0 0 ..., 0 0 0]]
```

In [15]:

```
import numpy as np
```

In [16]:

```
print "X_digits.shape:", X_digits.shape
print "Unique entries of y_digits:", np.unique(y_digits)
```

```
X_digits.shape: (70000, 784)
Unique entries of y_digits: [ 0.  1.  2.  3.  4.  5.  6.  7.  8.  9.]
```

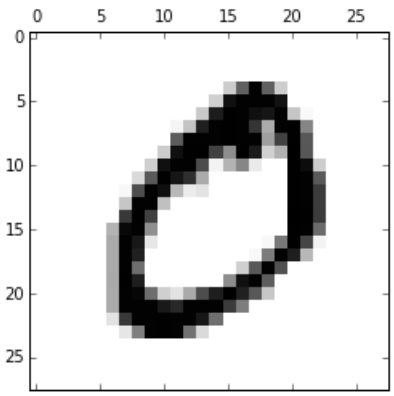
In [17]:

```
print(y_digits[0])
plt.rc("image", cmap="binary")
plt.matshow(X_digits[0].reshape(28, 28))
```

```
0.0
```

Out[17]:

```
<matplotlib.image.AxesImage at 0x9761f90>
```



In [18]:

```
zeros = X_digits[y_digits==0]  # select all the rows of X where y is zero (i.e. the zeros)
ones = X_digits[y_digits==1]   # select all the rows of X where y is one (i.e. the ones)
print "zeros.shape: ", zeros.shape
print "ones.shape: ", ones.shape
```

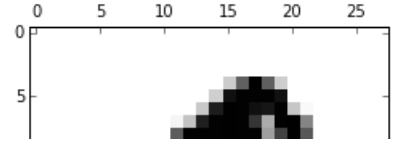
```
zeros.shape:  (6903, 784)
ones.shape:  (7877, 784)
```

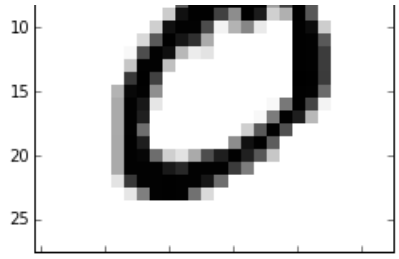
In [19]:

```
plt.matshow(zeros[0].reshape(28, 28))  # change the index of the zeros array to another number to see s
ome more zeros.
```

Out[19]:

```
<matplotlib.image.AxesImage at 0x9929d10>
```



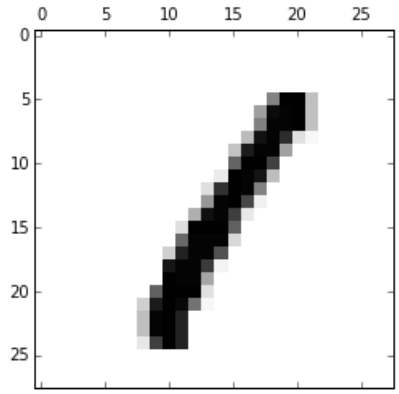


In [20]:

```
plt.matshow(ones[0].reshape(28, 28))  # change the index of the zeros array to another number to see so
me more zeros.
```

Out[20]:

```
<matplotlib.image.AxesImage at 0x995a4b0>
```



In [21]:

```
X_new = np.vstack([zeros, ones])  # this "stacks" zeros and ones vertically
print "X_new.shape: ", X_new.shape
y_new = np.hstack([np.repeat(0, zeros.shape[0]), np.repeat(1, ones.shape[0])])
print "y_new.shape: ", y_new.shape
print "y_new: ", y_new
```

```
X_new.shape:  (14780, 784)
y_new.shape:  (14780,)
y_new:  [0 0 0 ..., 1 1 1]
```

In [22]:

```
from sklearn.utils import shuffle
X_new, y_new = shuffle(X_new, y_new)
X_mnist_train = X_new[:5000]
y_mnist_train = y_new[:5000]
X_mnist_test = X_new[5000:]
y_mnist_test = y_new[5000:]
```

In [23]:

```
logreg.fit(X_mnist_train, y_mnist_train)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-23-f545086eef7d> in <module>()
----> 1 logreg.fit(X_mnist_train, y_mnist_train)

NameError: name 'logreg' is not defined
```

In [24]:

```
import matplotlib.pyplot as plt
```

In [25]:

```
from sklearn import linear_model
```

In [26]:

```
logreg=linear_model.LogisticRegression
```

In [27]:

```
logreg.fit(X_mnist_train, y_mnist_train)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-27-f545086eef7d> in <module>()
----> 1 logreg.fit(X_mnist_train, y_mnist_train)

TypeError: unbound method fit() must be called with LogisticRegression instance as first argument (got
ndarray instance instead)
```

In [28]:

```
from sklearn.linear_model import LogisticRegression
```

In [29]:

```
logreg=LogisticRegression()
```

In [30]:

```
logreg.fit(X_mnist_train, y_mnist_train)
```

Out[30]:

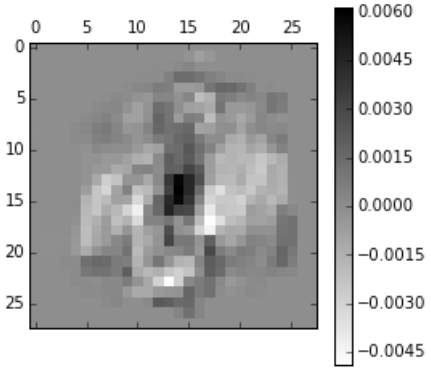
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr',
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0)
```

In [31]:

```
plt.matshow(logreg.coef_.reshape(28, 28))
plt.colorbar()
```

Out[31]:

```
<matplotlib.colorbar.Colorbar instance at 0x09C3FDF0>
```



In [32]:

```
print "Accuracy training set:", logreg.score(X_mnist_train, y_mnist_train)
print "Accuracy test set:", logreg.score(X_mnist_test, y_mnist_test)
```

```
Accuracy training set: 1.0
Accuracy test set: 0.998159509202
```

In [3]:

```
twos = X_digits[y_digits==2]  # select all the rows of X where y is zero (i.e. the zeros)
frees = X_digits[y_digits==3]   # select all the rows of X where y is one (i.e. the ones)
print "twos.shape: ", twos.shape
print "frees.shape: ", frees.shape
```

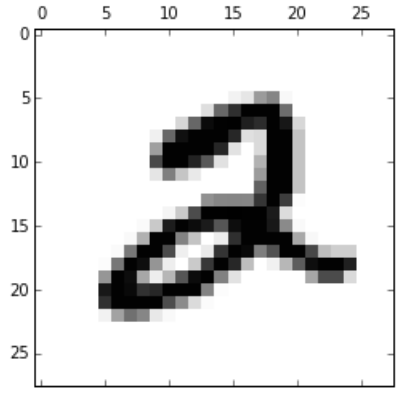
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-3-0d00f606fb4f> in <module>()
----> 1 twos = X_digits[y_digits==2]  # select all the rows of X where y is zero (i.e. the zeros)
      2 frees = X_digits[y_digits==3]   # select all the rows of X where y is one (i.e. the ones)
      3 print "twos.shape: ", twos.shape
      4 print "frees.shape: ", frees.shape

NameError: name 'X_digits' is not defined
```

In [35]:

```
plt.matshow(twos[0].reshape(28, 28))
```

Out[35]:

```
<matplotlib.image.AxesImage at 0x1732c310>
```

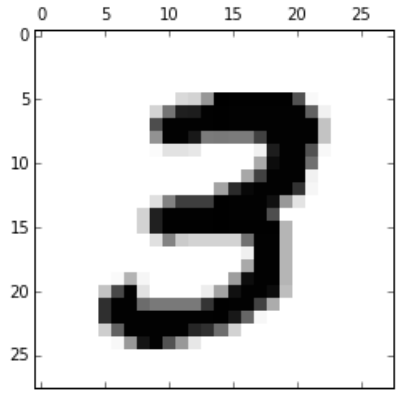


In [36]:

```
plt.matshow(frees[0].reshape(28, 28))
```

Out[36]:

```
<matplotlib.image.AxesImage at 0x172aac50>
```



In [2]:

```
X_new = np.vstack([twos, frees])  # this "stacks" zeros and ones vertically
print "X_new.shape: ", X_new.shape
y_new = np.hstack([np.repeat(2, twos.shape[0]), np.repeat(3, frees.shape[0])])
print "y_new.shape: ", y_new.shape
print "y_new: ", y_new
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-3a3211a2155d> in <module>()
----> 1 X_new = np.vstack([twos, frees])  # this "stacks" zeros and ones vertically
      2 print "X_new.shape: ", X_new.shape
      3 y_new = np.hstack([np.repeat(2, twos.shape[0]), np.repeat(3, frees.shape[0])])
      4 print "y_new.shape: ", y_new.shape
      5 print "y_new: ", y_new

NameError: name 'twos' is not defined
```

In [38]:

```
from sklearn.utils import shuffle
X_new, y_new = shuffle(X_new, y_new)
X_mnist_train = X_new[:5000]
y_mnist_train = y_new[:5000]
X_mnist_test = X_new[5000:]
y_mnist_test = y_new[5000:]
```

In [39]:

```
logreg.fit(X_mnist_train, y_mnist_train)
```

Out[39]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
```

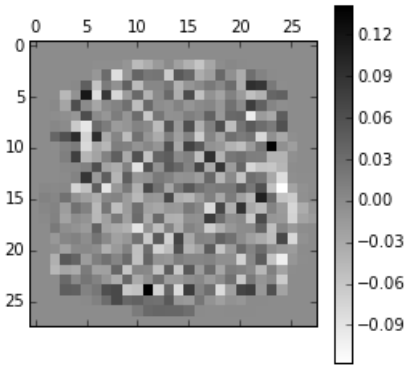
```
          intercept_scaling=1, max_iter=100, multi_class='ovr',
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0)
```

In [40]:

```
plt.matshow(logreg.coef_.reshape(28, 28))
plt.colorbar()
```

Out[40]:

```
<matplotlib.colorbar.Colorbar instance at 0x16B7CCD8>
```



In [41]:

```
print "Accuracy training set:", logreg.score(X_mnist_train, y_mnist_train)
print "Accuracy test set:", logreg.score(X_mnist_test, y_mnist_test)
```

```
Accuracy training set: 1.0
Accuracy test set: 0.949731683277
```

In [42]:

```
fours = X_digits[y_digits==4]  # select all the rows of X where y is zero (i.e. the zeros)
fives = X_digits[y_digits==5]   # select all the rows of X where y is one (i.e. the ones)
print "fours.shape: ", fours.shape
print "fives.shape: ", fives.shape
```

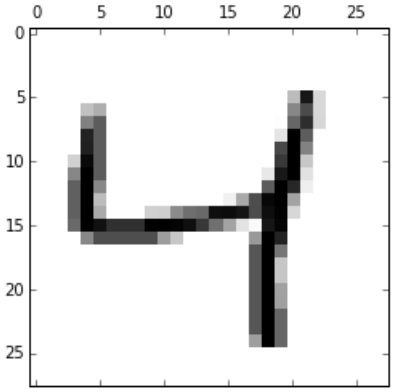
```
fours.shape:  (6824, 784)
fives.shape:  (6313, 784)
```

In [43]:

```
plt.matshow(fours[0].reshape(28, 28))
```

Out[43]:

```
<matplotlib.image.AxesImage at 0x16e798b0>
```

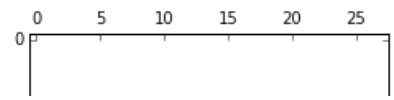


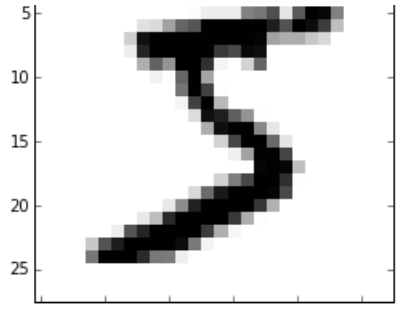
In [44]:

```
plt.matshow(fives[0].reshape(28, 28))
```

Out[44]:

```
<matplotlib.image.AxesImage at 0x16c4e950>
```





In [45]:

```
X_new = np.vstack([fours, fives])  # this "stacks" zeros and ones vertically
print "X_new.shape: ", X_new.shape
y_new = np.hstack([np.repeat(0, fours.shape[0]), np.repeat(1, fives.shape[0])])
print "y_new.shape: ", y_new.shape
print "y_new: ", y_new
```

```
X_new.shape:  (13137, 784)
y_new.shape:  (13137,)
y_new:  [0 0 0 ..., 1 1 1]
```

In [46]:

```
from sklearn.utils import shuffle
X_new, y_new = shuffle(X_new, y_new)
X_mnist_train = X_new[:5000]
y_mnist_train = y_new[:5000]
X_mnist_test = X_new[5000:]
y_mnist_test = y_new[5000:]
```

In [47]:

```
logreg.fit(X_mnist_train, y_mnist_train)
```

Out[47]:

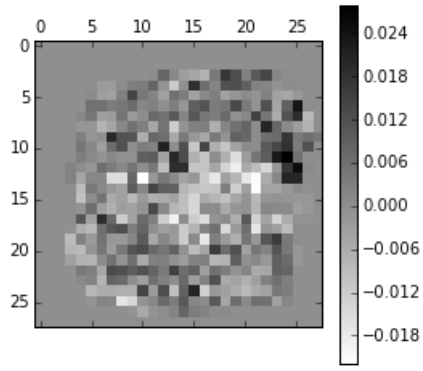
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr',
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0)
```

In [48]:

```
plt.matshow(logreg.coef_.reshape(28, 28))
plt.colorbar()
```

Out[48]:

```
<matplotlib.colorbar.Colorbar instance at 0x1754AAA8>
```



In [49]:

```
print "Accuracy training set:", logreg.score(X_mnist_train, y_mnist_train)
print "Accuracy test set:", logreg.score(X_mnist_test, y_mnist_test)
```

```
Accuracy training set: 1.0
Accuracy test set: 0.985129654664
```

In [50]:

```
sixs = X_digits[y_digits==6]  # select all the rows of X where y is zero (i.e. the zeros)
```

```
sevens = X_digits[y_digits==7]   # select all the rows of X where y is one (i.e. the ones)
print "sixs.shape: ", sixs.shape
print "sevens.shape: ", sevens.shape
```

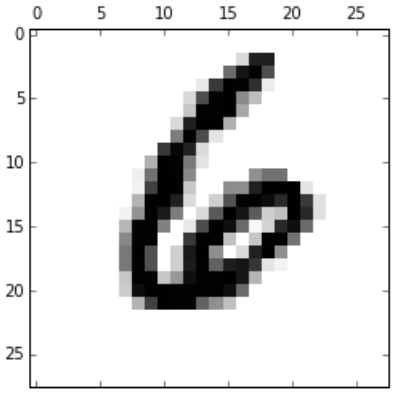
```
sixs.shape:  (6876, 784)
sevens.shape:  (7293, 784)
```

In [51]:

```
plt.matshow(sixs[0].reshape(28, 28))
```

Out[51]:

```
<matplotlib.image.AxesImage at 0x17845cb0>
```

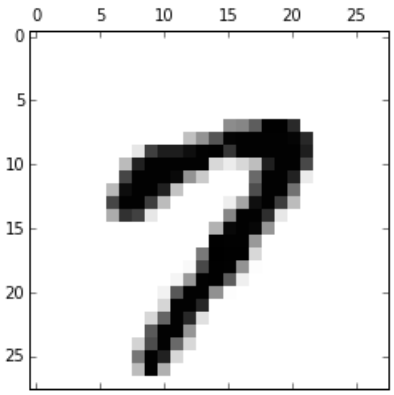


In [52]:

```
plt.matshow(sevens[0].reshape(28, 28))
```

Out[52]:

```
<matplotlib.image.AxesImage at 0x174fae30>
```



In [53]:

```
X_new = np.vstack([sixs, sevens])  # this "stacks" zeros and ones vertically
print "X_new.shape: ", X_new.shape
y_new = np.hstack([np.repeat(0, sixs.shape[0]), np.repeat(1, sevens.shape[0])])
print "y_new.shape: ", y_new.shape
print "y_new: ", y_new
```

```
X_new.shape:  (14169, 784)
y_new.shape:  (14169,)
y_new:  [0 0 0 ..., 1 1 1]
```

In [54]:

```
from sklearn.utils import shuffle
X_new, y_new = shuffle(X_new, y_new)
X_mnist_train = X_new[:5000]
y_mnist_train = y_new[:5000]
X_mnist_test = X_new[5000:]
y_mnist_test = y_new[5000:]
```

In [55]:

```
logreg.fit(X_mnist_train, y_mnist_train)
```

Out[55]:

Out[55]:

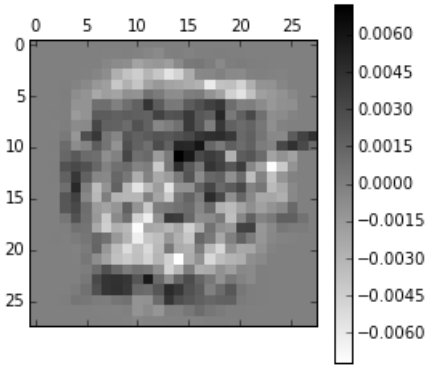
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr',
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0)
```

In [56]:

```
plt.matshow(logreg.coef_.reshape(28, 28))
plt.colorbar()
```

Out[56]:

```
<matplotlib.colorbar.Colorbar instance at 0x108D7E18>
```



In [57]:

```
print "Accuracy training set:", logreg.score(X_mnist_train, y_mnist_train)
print "Accuracy test set:", logreg.score(X_mnist_test, y_mnist_test)
```

```
Accuracy training set: 1.0
Accuracy test set: 0.998582179082
```

In [58]:

```
eigths = X_digits[y_digits==8]   # select all the rows of X where y is zero (i.e. the zeros)
nines = X_digits[y_digits==9]    # select all the rows of X where y is one (i.e. the ones)
print "eigths.shape: ", eigths.shape
print "nines.shape: ", nines.shape
```

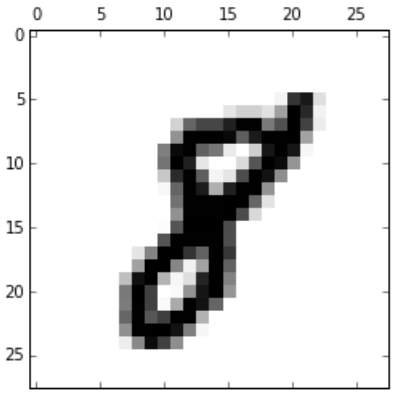
```
eigths.shape:  (6825, 784)
nines.shape:  (6958, 784)
```

In [59]:

```
plt.matshow(eigths[0].reshape(28, 28))
```

Out[59]:

```
<matplotlib.image.AxesImage at 0x16baca50>
```

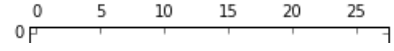


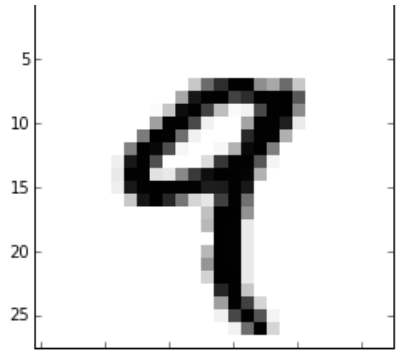
In [71]:

```
plt.matshow(nines[0].reshape(28, 28))
```

Out[71]:

```
<matplotlib.image.AxesImage at 0x10954d70>
```





In [79]:

```
X_new = np.vstack([eigths, nines])  # this "stacks" zeros and ones vertically
print "X_new.shape: ", X_new.shape
y_new = np.hstack([np.repeat(0, eigths.shape[0]), np.repeat(1, nines.shape[0])])
print "y_new.shape: ", y_new.shape
print "y_new: ", y_new
```

```
X_new.shape:  (13783, 784)
y_new.shape:  (13783,)
y_new:  [0 0 0 ..., 1 1 1]
```

In [84]:

```
from sklearn.utils import shuffle
X_new, y_new = shuffle(X_new, y_new)
X_mnist_train = X_new[:5000]
y_mnist_train = y_new[:5000]
X_mnist_test = X_new[5000:]
y_mnist_test = y_new[5000:]
```

In [85]:

```
logreg.fit(X_mnist_train, y_mnist_train)
```

Out[85]:

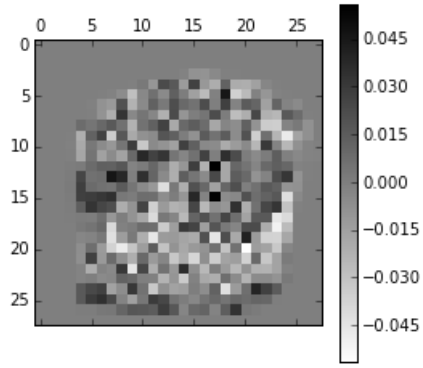
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr',
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0)
```

In [87]:

```
plt.matshow(logreg.coef_.reshape(28, 28))
plt.colorbar()
```

Out[87]:

```
<matplotlib.colorbar.Colorbar instance at 0x10D5B468>
```



In [88]:

```
print "Accuracy training set:", logreg.score(X_mnist_train, y_mnist_train)
print "Accuracy test set:", logreg.score(X_mnist_test, y_mnist_test)
```

```
Accuracy training set: 1.0
Accuracy test set: 0.973926904247
```

In [83]:

In [83]:

In [ ]: