In [1]:
```
%pylab inline
import pylab
from sklearn.datasets import fetch_mldata
DATA_PATH = '~/data'
mnist = fetch_mldata('MNIST original', data_home=DATA_PATH)
```

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

In [2]:
```
train = mnist.data[:60000]
test = mnist.data[60000:]
```

In [3]:
```
test_sample = test[::100]
```

In [7]:
```
train_labels = mnist.target[:60000]
test_labels = mnist.target[60000:]
test_labels_sample = test_labels[::100]
```

In [8]:
```
%%time
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=4, algorithm='brute').fit(tra
in, train_labels)
```

```
CPU times: user 6.2 ms, sys: 2.75 ms, total: 8.95 ms
Wall time: 7.87 ms
```

In [9]:
```
%%time
model.score(test_sample, test_labels_sample)
```

```
CPU times: user 775 ms, sys: 303 ms, total: 1.08 s
Wall time: 850 ms
```

Out[9]:
```
0.96999999999999997
```

In [11]:
```
preds = model.predict(test_sample)
errors = [i for i in range(0, len(test_sample)) if preds[i] != test_la
bels_sample[i]]
err_rate = float(len(errors))/len(preds)
print (err_rate)
```

```
0.03
```

In [13]:
```
##########----------------Question 3 Start
########## Generating confusion matrix
##########

from sklearn.metrics import confusion_matrix
confusion_matrix(preds, test_labels_sample)
```

Out[13]:
```
array([[10,  0,  0,  0,  0,  0,  0,  0,  0,  0],
       [ 0, 12,  0,  0,  0,  0,  0,  1,  0,  0],
       [ 0,  0, 10,  0,  0,  0,  0,  0,  0,  0],
       [ 0,  0,  0, 10,  0,  1,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  9,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  8,  0,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  9,  0,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0, 10,  0,  0],
       [ 0,  0,  0,  0,  0,  0,  0,  0,  9,  0],
       [ 0,  0,  0,  0,  1,  0,  0,  0,  0, 10]])
```

In [14]:
```
train = mnist.data[:60000]
train_labels = mnist.target[:60000]

test = mnist.data[60000:]
test_labels = mnist.target[60000:]

# For traning
train_sample = train[::100]
train_sample_labels = train_labels[::100]
# for testing
test_sample = test[::10]
test_sample_labels = test_labels[::10]
```

In [15]:
```
############ Naive Bayes Model


from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
modelGnb = gnb.fit(train_sample, train_sample_labels)
```

In [16]:
```
%%time
modelGnb.score(test_sample, test_sample_labels)
```

```
CPU times: user 82.8 ms, sys: 64.5 ms, total: 147 ms
Wall time: 159 ms
```

Out[16]:
```
0.624
```

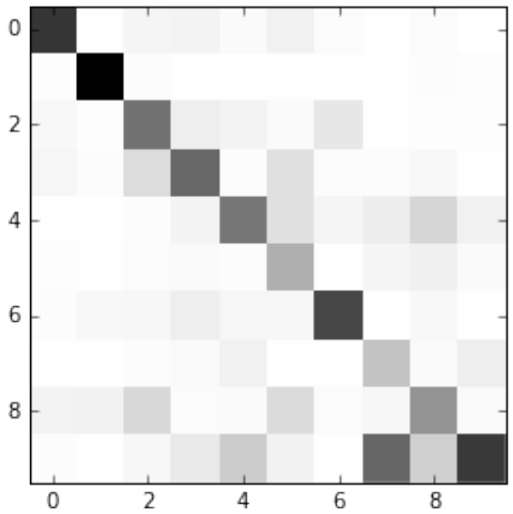
In [17]:
```
######## Naive Bayes Model Part 1 :- Error in the Naive Bayes Model.


preds = modelGnb.predict(test_sample)
errors = [i for i in range(0, len(test_sample)) if preds[i] != test_sa
mple_labels[i]]
err_rate = float(len(errors))/len(preds)
print (err_rate)
```

```
0.376
```

In [18]:
```
######### Naive Bayes Model Part 2 :- Confusion Matrix of Naive Bayes
Model
mat1 = confusion_matrix(preds, test_sample_labels)
mat1
```

Out[18]:
```
array([[ 83,   0,   4,   5,   2,   6,   2,   0,   2,   0],
       [  0, 105,   1,   0,   0,   0,   0,   0,   1,   1],
       [  3,   0,  58,   7,   5,   2,  10,   0,   1,   1],
       [  4,   1,  14,  63,   1,  13,   1,   1,   3,   0],
       [  0,   0,   1,   4,  56,  13,   4,   7,  17,   6],
       [  1,   0,   2,   2,   1,  33,   0,   4,   6,   2],
       [  1,   3,   3,   7,   4,   3,  76,   0,   3,   0],
       [  0,   0,   1,   2,   6,   0,   0,  25,   2,   7],
       [  5,   5,  16,   2,   2,  15,   2,   3,  44,   2],
       [  1,   0,   3,   9,  21,   5,   0,  63,  19,  81]])
```

In [20]:
```
######### Naive Bayes Model Part3 :- pairs of digits most frequently c
onfused with each other
######### while using Naive Bayes Model are:- top 3 :-  (63) 7 and 9,
(21) 4 and 9, (19) 8 and 9

import matplotlib.pyplot as plt
plt.imshow(mat1, cmap='binary', interpolation='None')
plt.show()
```



In [21]:
```
########### SVM (Linear Kernel)...
########### Importing and Setting up


from sklearn import svm
clf = svm.SVC(kernel='linear')
modelsvm = clf.fit(train_sample, train_sample_labels)
```

In [22]:
```
%%time
modelsvm.score(test_sample, test_sample_labels)
```

```
CPU times: user 388 ms, sys: 6.73 ms, total: 395 ms
Wall time: 420 ms
```

Out[22]:
```
0.873
```

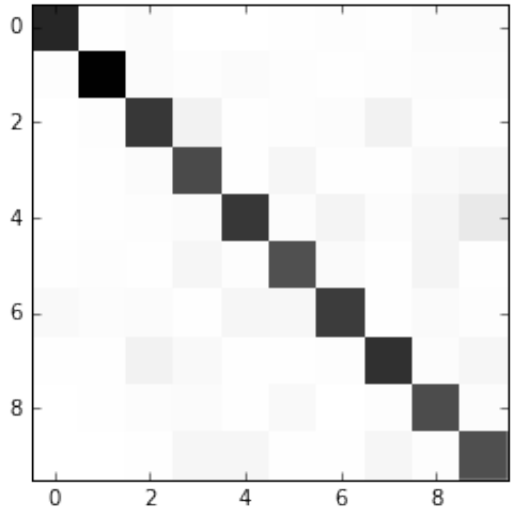
In [23]:
```
############ SVM (Linear Kernel) Part 1:- Error Rate of SVM


preds = modelsvm.predict(test_sample)
errors = [i for i in range(0, len(test_sample)) if preds[i] != test_sa
mple_labels[i]]
err_rate = float(len(errors))/len(preds)
print (err_rate)
```

```
0.127
```

In [24]:
```
############# SVM (Linear Kernel) Part 2 :- Confusion Matrix of SVM


mat2 = confusion_matrix(preds, test_sample_labels)
mat2
```

Out[24]:
```
array([[ 95,   0,   2,   0,   0,   0,   1,   0,   2,   2],
       [  0, 112,   2,   1,   2,   1,   0,   1,   1,   1],
       [  0,   0,  88,   6,   0,   1,   1,   6,   1,   0],
       [  0,   0,   1,  80,   0,   4,   0,   0,   3,   4],
       [  0,   0,   1,   1,  88,   1,   5,   1,   4,  10],
       [  0,   1,   0,   4,   0,  77,   2,   0,   5,   0],
       [  3,   1,   2,   0,   4,   3,  86,   0,   2,   1],
       [  0,   0,   6,   3,   0,   0,   0,  91,   1,   4],
       [  0,   0,   1,   2,   0,   3,   0,   0,  79,   1],
       [  0,   0,   0,   4,   4,   0,   0,   4,   0,  77]])
```

In [25]:
```
######### SVM (Linear Kernel) Part3 :- pairs of digits most frequently
confused with each other are :-
######### while using SVM are  top 3 :-  (10) 4 and 9, (6) 2 and 3, (6
) 3 and 7


import matplotlib.pyplot as plt
plt.imshow(mat2, cmap='binary', interpolation='None')
plt.show()
```



In [26]:
```
##########-------------Logistic Regression Model----
######### Importing and setting up


from sklearn import linear_model
logistic = linear_model.LogisticRegression()
logistic.fit(train_sample, train_sample_labels)
```

Out[26]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_interce
pt=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jo
bs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0
.0001,
          verbose=0, warm_start=False)
```

In [27]:
```
%%time
logistic.score(test_sample, test_sample_labels)
```

```
CPU times: user 7.57 ms, sys: 4.55 ms, total: 12.1 ms
Wall time: 7.39 ms
```

Out[27]:
```
0.82199999999999995
```

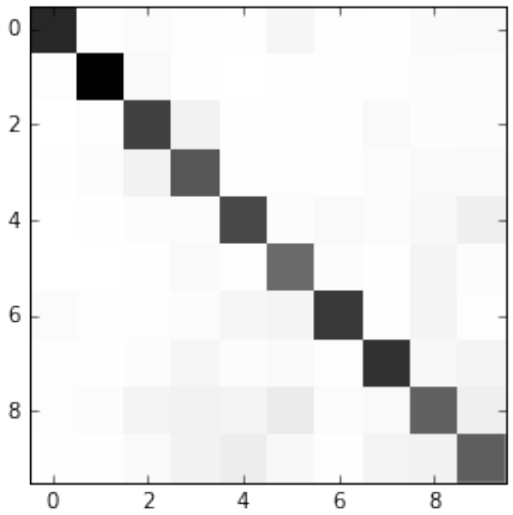
In [28]:
```
############ Logistic Regression Model Part 1:- Error Rate of Logistic
Regression


preds = logistic.predict(test_sample)
errors = [i for i in range(0, len(test_sample)) if preds[i] != test_sa
mple_labels[i]]
err_rate = float(len(errors))/len(preds)
print (err_rate)
```

```
0.178
```

In [30]:
```
############# Logistic Regression Model Part 2 :- Confusion Matrix of
Logistic Regression


mat3 = confusion_matrix(preds, test_sample_labels)
mat3
```

Out[30]:
```
array([[ 94,   0,   2,   1,   0,   4,   1,   0,   2,   3],
       [  0, 111,   3,   0,   0,   0,   1,   1,   2,   1],
       [  0,   0,  83,   6,   0,   1,   0,   3,   1,   2],
       [  0,   1,   5,  74,   0,   1,   0,   1,   3,   3],
       [  0,   0,   1,   1,  80,   1,   3,   2,   3,   7],
       [  0,   0,   0,   3,   0,  65,   1,   0,   5,   1],
       [  2,   1,   1,   1,   4,   4,  87,   0,   5,   1],
       [  1,   0,   1,   4,   1,   2,   0,  89,   3,   5],
       [  1,   1,   5,   6,   5,   9,   2,   2,  69,   7],
       [  0,   0,   2,   5,   8,   3,   0,   5,   5,  70]])
```

In [31]:
```
######### Logistic Regression Model Part3 :- pairs of digits most frequently
######### confused with each other while using Logistic Regression are
#########   top 3 :-  (8) 4 and 9, (7) 4 and 9, (7) 8 and 9, (6) 2 and 3, (5) 8 and 6



import matplotlib.pyplot as plt
plt.imshow(mat3, cmap='binary', interpolation='None')
plt.show()
```



In [ ]: