In [13]:
```
# Naive Bayes
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
gnb = GaussianNB()
model = gnb.fit(train_sample, train_sample_labels)
model.score(test_sample, test_sample_labels)
preds = model.predict(test_sample)
cm = confusion_matrix(test_sample_labels, preds)
print cm
max = 0
index_i = 0
index_j = 0
for  i in xrange(0, len(cm)):
    for j in xrange(0, i):
        if cm[i, j] + cm[j, i] > max:
            max = cm[i, j] + cm[j, i]
            index_i = i
            index_j = j
errors = [i for i in xrange(0, len(test_sample)) if preds[i]
err_rate = float(len(errors))/len(preds)
print err_rate
print index_i, index_j
```

```
[[ 83   0   3   4   0   1   1   0   5   1]
 [  0 105   0   1   0   0   3   0   5   0]
 [  4   1  58  14   1   2   3   1  16   3]
 [  5   0   7  63   4   2   7   2   2   9]
 [  2   0   5   1  56   1   4   6   2  21]
 [  6   0   2  13  13  33   3   0  15   5]
 [  2   0  10   1   4   0  76   0   2   0]
 [  0   0   0   1   7   4   0  25   3  63]
 [  2   1   1   3  17   6   3   2  44  19]
 [  0   1   1   0   6   2   0   7   2  81]]
0.376
9 7
```

In [12]:
```
# SVM
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix
clf = SVC(kernel='linear')
model = clf.fit(train_sample, train_sample_labels)
model.score(test_sample, test_sample_labels)
preds = model.predict(test_sample)
cm = confusion_matrix(test_sample_labels, preds)
print cm
max = 0
index_i = 0
index_j = 0
for  i in xrange(0, len(cm)):
    for j in xrange(0, i):
        if cm[i, j] + cm[j, i] > max:
            max = cm[i, j] + cm[j, i]
            index_i = i
            index_j = j
errors = [i for i in xrange(0, len(test_sample)) if preds[i]
err_rate = float(len(errors))/len(preds)
print err_rate
print index_i, index_j
```

```
[[ 95   0   0   0   0   0   3   0   0   0]
 [  0 112   0   0   0   1   1   0   0   0]
 [  2   2  88   1   1   0   2   6   1   0]
 [  0   1   6  80   1   4   0   3   2   4]
 [  0   2   0   0  88   0   4   0   0   4]
 [  0   1   1   4   1  77   3   0   3   0]
 [  1   0   1   0   5   2  86   0   0   0]
 [  0   1   6   0   1   0   0  91   0   4]
 [  2   1   1   3   4   5   2   1  79   0]
 [  2   1   0   4  10   0   1   4   1  77]]
0.127
9 4
```

In [14]:
```
# Logistic Regression
from sklearn import linear_model
from sklearn.metrics import confusion_matrix
model = linear_model.LogisticRegression().fit(train_sample, t
model.score(test_sample, test_sample_labels)
preds = model.predict(test_sample)
cm = confusion_matrix(test_sample_labels, preds)
print cm
max = 0
index_i = 0
index_j = 0
for  i in xrange(0, len(cm)):
    for j in xrange(0, i):
        if cm[i, j] + cm[j, i] > max:
            max = cm[i, j] + cm[j, i]
            index_i = i
            index_j = j
errors = [i for i in xrange(0, len(test_sample)) if preds[i]
err_rate = float(len(errors))/len(preds)
print err_rate
print index_i, index_j
```

```
[[ 94   0   0   0   0   0   2   1   1   0]
 [  0 111   0   1   0   0   1   0   1   0]
 [  2   3  83   5   1   0   1   1   5   2]
 [  1   0   6  74   1   3   1   4   6   5]
 [  0   0   0   0  80   0   4   1   5   8]
 [  4   0   1   1   1  65   4   2   9   3]
 [  1   1   0   0   3   1  87   0   2   0]
 [  0   1   3   1   2   0   0  89   2   5]
 [  2   2   1   3   3   5   5   3  69   5]
 [  3   1   2   3   7   1   1   5   7  70]]
0.178
9 4
```

In [ ]:
```
```