In [18]:
```
import mltools.cluster as cluster
import mltools as ml
import numpy as np
import matplotlib.pyplot as plt
import mltools.dtree as dtree
from scipy import linalg
from sklearn import svm
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import StratifiedShuffleSplit
from sklearn.ensemble import BaggingClassifier
from sklearn.multiclass import OneVsOneClassifier
```

In [19]:
```
X = np.genfromtxt("data/X_train.txt", delimiter=None)
Y = np.genfromtxt("data/Y_train.txt", delimiter=None)
X, Y = ml.shuffleData(X=X, Y=Y)
X, _ = ml.rescale(X)

# #[Xtr, Ytr, Xte, Yte] = ml.splitData(X,Y, 0.05)
# clf = svm.SVC(kernel='rbf')
# clf.fit(X[:20000,:], Y[:20000])
# print np.mean(Y[180000:]==clf.predict(X[180000:]))
```

In [ ]:

```
# from sklearn.model_selection import ShuffleSplit
# rs = ShuffleSplit(n_splits=1, test_size=0.25, random_state=3)
# print rs.split(X[:20000])


# n_estimators_range = [1, 10, 100, 1000]

mse = []
# for n_estimators in n_estimators_range:
clf = BaggingClassifier(svm.SVC(kernel="rbf", C=10, gamma=0.01), max_samples
# clf = svm.SVC(kernel="rbf", C=1000000, gamma=gamma)
clf.fit(X[:150000, :], Y[:150000])
YteHat = clf.predict(X[150000:])
YtrHat = YteHat = clf.predict(X[:150000])
mse.append(np.mean(Y[150000:]==YteHat))
print mse

# plt.plot(n_estimators_range, mse)
# plt.show()

# n_estimators = 2000
# clf = svm.SVC()
# clf.fit(X[:10000,:], Y[:10000])

#clf = BaggingClassifier(svm.SVC(kernel="rbf", C=4.0, gamma=1000), max_sampl

# mse = []
# training_error= []
# cc = np.logspace(-3,6,10)
# for c in cc:
#     clf = svm.SVC(kernel="rbf", C=c, gamma = 0.1)
#     clf.fit(X[:1000], Y[:1000])
#     print "Data is fit."
#     Ytest = clf.predict(X[199000:])
#     yte = Y[199000:]
#     mse.append(np.mean(Ytest != yte))
#     training_error.append(np.mean(Y[:1000] != clf.predict(X[:1000])))
#     print mse
#     print training_error
#     print np.sum(Ytest == np.ones((Ytest.shape)))
#     print np.sum(yte == np.ones((yte.shape)))
#     print

# print("The best parameters are %s with a score of %0.2f"
#       % (grid.best_params_, grid.best_score_))

# Xtest = np.genfromtxt('data/X_test.txt')
# Ytest = clf.predict(Xtest)
# np.savetxt('svm-c-1-g-3.txt',
# np.vstack( (np.arange(len(Ytest)) , Ytest) ).T, '%d, %.2f',header='ID,Prob
```

In [ ]:
```
print training_error
print mse
plt.semilogx(cc, training_error, label="training error")
plt.semilogx(cc, mse, label = "validation error")
plt.legend()
plt.title('Parameter C variations for RBF SVM ')
plt.ylabel('Error')
plt.xlabel('C')
plt.savefig('rbf-c-variation-4')
plt.show()
```

In [23]:
```
np.mean(Y[:150000]==clf.predict(X[:150000]))
```

Out[23]:
```
0.67556666666666665
```

In [24]:
```
print np.sum(Y[150000:] == np.ones((Ytest.shape)))
print np.sum(Y[:150000] == np.ones((yte.shape)))
```

```
0
0

C:\Anaconda2\lib\site-packages\ipykernel\__main__.py:1: DeprecationWarnin
g: elementwise == comparison failed; this will raise an error in the futu
re.
  if __name__ == '__main__':
C:\Anaconda2\lib\site-packages\ipykernel\__main__.py:2: DeprecationWarnin
g: elementwise == comparison failed; this will raise an error in the futu
re.
  from ipykernel import kernelapp as app
```