CS289A_HW01_MNIST

January 30, 2017

In [16]:
```
%load_ext autoreload
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

In [17]:
```
%autoreload 2
```

In [18]:
```
import numpy as np
import HW01_utils as utils
import trainfunctions as tf
```

In [46]:
```
_LOCAL_PATH = r"C:\Users\Mitch\Documents\Cal\2 - 2017 Spring\COMPSCI 289A
_DATA_PATH = "Data\hw01_data"

_DATA_DIR = _LOCAL_PATH + "\\" + _DATA_PATH
trainpath = r"mnist\train.mat"
```

In [20]:
```
valsetsize = 10000
samples = np.array([100, 200, 500, 1000, 2000, 5000, 10000])
hyperparams = np.logspace(-8,4,num=20)
```

In [21]:
```
# Load MNIST training data
mnist = utils.loaddata(trainpath,_DATA_DIR,'trainX')

# Shuffle data before splitting
np.random.shuffle(mnist)
```

In [22]:
```
trainset,valset = utils.partition(valsetsize,mnist)
trainsetarrays,trainsetlabels = utils.separatelabels(trainset)
valsetarrays,valsetlabels = utils.separatelabels(valset)
```

In [23]:
```
Accs = np.empty((len(samples),len(hyperparams)))
i = 0 # sample index counter
for nsamples in samples:
    print(nsamples,'samples')
    j = 0 # hyperparameter index counter
    for hp in hyperparams:
        acc = tf.TrainAndScoreNsamples(trainsetarrays[:nsamples],trainsetl
```

```
                print('\tC =',hp,'\tAccuracy:',acc)
                Accs[i,j] = acc
                j+=1
            i+=1
        print(Accs)
```

```
100 samples
        C = 1e-08         Accuracy: 0.1119
        C = 4.28133239872e-08         Accuracy: 0.2301
        C = 1.83298071083e-07         Accuracy: 0.6438
        C = 7.84759970351e-07         Accuracy: 0.7169
        C = 3.35981828628e-06         Accuracy: 0.7133
        C = 1.43844988829e-05         Accuracy: 0.7133
        C = 6.15848211066e-05         Accuracy: 0.7133
        C = 0.000263665089873         Accuracy: 0.7133
        C = 0.00112883789168         Accuracy: 0.7133
        C = 0.00483293023857         Accuracy: 0.7133
        C = 0.0206913808111         Accuracy: 0.7133
        C = 0.088586679041         Accuracy: 0.7133
        C = 0.379269019073         Accuracy: 0.7133
        C = 1.62377673919         Accuracy: 0.7133
        C = 6.95192796178         Accuracy: 0.7133
        C = 29.7635144163         Accuracy: 0.7133
        C = 127.42749857         Accuracy: 0.7133
        C = 545.559478117         Accuracy: 0.7133
        C = 2335.72146909         Accuracy: 0.7133
        C = 10000.0         Accuracy: 0.7133
200 samples
        C = 1e-08         Accuracy: 0.0963
        C = 4.28133239872e-08         Accuracy: 0.425
        C = 1.83298071083e-07         Accuracy: 0.7747
        C = 7.84759970351e-07         Accuracy: 0.8005
        C = 3.35981828628e-06         Accuracy: 0.7947
        C = 1.43844988829e-05         Accuracy: 0.7947
        C = 6.15848211066e-05         Accuracy: 0.7947
        C = 0.000263665089873         Accuracy: 0.7947
        C = 0.00112883789168         Accuracy: 0.7947
        C = 0.00483293023857         Accuracy: 0.7947
        C = 0.0206913808111         Accuracy: 0.7947
        C = 0.088586679041         Accuracy: 0.7947
        C = 0.379269019073         Accuracy: 0.7947
        C = 1.62377673919         Accuracy: 0.7947
        C = 6.95192796178         Accuracy: 0.7947
        C = 29.7635144163         Accuracy: 0.7947
        C = 127.42749857         Accuracy: 0.7947
        C = 545.559478117         Accuracy: 0.7947
        C = 2335.72146909         Accuracy: 0.7947
        C = 10000.0         Accuracy: 0.7947
```

```
500 samples
        C = 1e-08         Accuracy: 0.3171
        C = 4.28133239872e-08         Accuracy: 0.7527
        C = 1.83298071083e-07         Accuracy: 0.865
        C = 7.84759970351e-07         Accuracy: 0.8635
        C = 3.35981828628e-06         Accuracy: 0.8552
        C = 1.43844988829e-05         Accuracy: 0.8548
        C = 6.15848211066e-05         Accuracy: 0.8548
        C = 0.000263665089873         Accuracy: 0.8548
        C = 0.00112883789168         Accuracy: 0.8548
        C = 0.00483293023857         Accuracy: 0.8548
        C = 0.0206913808111         Accuracy: 0.8548
        C = 0.088586679041         Accuracy: 0.8548
        C = 0.379269019073         Accuracy: 0.8548
        C = 1.62377673919         Accuracy: 0.8548
        C = 6.95192796178         Accuracy: 0.8548
        C = 29.7635144163         Accuracy: 0.8548
        C = 127.42749857         Accuracy: 0.8548
        C = 545.559478117         Accuracy: 0.8548
        C = 2335.72146909         Accuracy: 0.8548
        C = 10000.0         Accuracy: 0.8548
1000 samples
        C = 1e-08         Accuracy: 0.5782
        C = 4.28133239872e-08         Accuracy: 0.85
        C = 1.83298071083e-07         Accuracy: 0.8867
        C = 7.84759970351e-07         Accuracy: 0.8909
        C = 3.35981828628e-06         Accuracy: 0.8815
        C = 1.43844988829e-05         Accuracy: 0.881
        C = 6.15848211066e-05         Accuracy: 0.881
        C = 0.000263665089873         Accuracy: 0.881
        C = 0.00112883789168         Accuracy: 0.881
        C = 0.00483293023857         Accuracy: 0.881
        C = 0.0206913808111         Accuracy: 0.881
        C = 0.088586679041         Accuracy: 0.881
        C = 0.379269019073         Accuracy: 0.881
        C = 1.62377673919         Accuracy: 0.881
        C = 6.95192796178         Accuracy: 0.881
        C = 29.7635144163         Accuracy: 0.881
        C = 127.42749857         Accuracy: 0.881
        C = 545.559478117         Accuracy: 0.881
        C = 2335.72146909         Accuracy: 0.881
        C = 10000.0         Accuracy: 0.881
2000 samples
        C = 1e-08         Accuracy: 0.7981
        C = 4.28133239872e-08         Accuracy: 0.8824
        C = 1.83298071083e-07         Accuracy: 0.9043
        C = 7.84759970351e-07         Accuracy: 0.9086
        C = 3.35981828628e-06         Accuracy: 0.895
```

```
        C = 1.43844988829e-05         Accuracy: 0.8926
        C = 6.15848211066e-05         Accuracy: 0.8926
        C = 0.000263665089873         Accuracy: 0.8926
        C = 0.00112883789168         Accuracy: 0.8926
        C = 0.00483293023857         Accuracy: 0.8926
        C = 0.0206913808111         Accuracy: 0.8926
        C = 0.088586679041         Accuracy: 0.8926
        C = 0.379269019073         Accuracy: 0.8926
        C = 1.62377673919         Accuracy: 0.8926
        C = 6.95192796178         Accuracy: 0.8926
        C = 29.7635144163         Accuracy: 0.8926
        C = 127.42749857         Accuracy: 0.8926
        C = 545.559478117         Accuracy: 0.8926
        C = 2335.72146909         Accuracy: 0.8926
        C = 10000.0         Accuracy: 0.8926
5000 samples
        C = 1e-08         Accuracy: 0.8721
        C = 4.28133239872e-08         Accuracy: 0.9038
        C = 1.83298071083e-07         Accuracy: 0.9183
        C = 7.84759970351e-07         Accuracy: 0.9202
        C = 3.35981828628e-06         Accuracy: 0.9135
        C = 1.43844988829e-05         Accuracy: 0.9041
        C = 6.15848211066e-05         Accuracy: 0.903
        C = 0.000263665089873         Accuracy: 0.903
        C = 0.00112883789168         Accuracy: 0.903
        C = 0.00483293023857         Accuracy: 0.903
        C = 0.0206913808111         Accuracy: 0.903
        C = 0.088586679041         Accuracy: 0.903
        C = 0.379269019073         Accuracy: 0.903
        C = 1.62377673919         Accuracy: 0.903
        C = 6.95192796178         Accuracy: 0.903
        C = 29.7635144163         Accuracy: 0.903
        C = 127.42749857         Accuracy: 0.903
        C = 545.559478117         Accuracy: 0.903
        C = 2335.72146909         Accuracy: 0.903
        C = 10000.0         Accuracy: 0.903
10000 samples
        C = 1e-08         Accuracy: 0.8935
        C = 4.28133239872e-08         Accuracy: 0.9147
        C = 1.83298071083e-07         Accuracy: 0.9255
        C = 7.84759970351e-07         Accuracy: 0.9298
        C = 3.35981828628e-06         Accuracy: 0.9229
        C = 1.43844988829e-05         Accuracy: 0.9118
        C = 6.15848211066e-05         Accuracy: 0.9071
        C = 0.000263665089873         Accuracy: 0.906
        C = 0.00112883789168         Accuracy: 0.906
        C = 0.00483293023857         Accuracy: 0.906
        C = 0.0206913808111         Accuracy: 0.906
```

```
        C = 0.088586679041         Accuracy: 0.906
        C = 0.379269019073         Accuracy: 0.906
        C = 1.62377673919          Accuracy: 0.906
        C = 6.95192796178          Accuracy: 0.906
        C = 29.7635144163          Accuracy: 0.906
        C = 127.42749857          Accuracy: 0.906
        C = 545.559478117          Accuracy: 0.906
        C = 2335.72146909          Accuracy: 0.906
        C = 10000.0          Accuracy: 0.906
[[ 0.1119  0.2301  0.6438  0.7169  0.7133  0.7133  0.7133  0.7133  0.7133
   0.7133  0.7133  0.7133  0.7133  0.7133  0.7133  0.7133  0.7133  0.7133
   0.7133  0.7133]
 [ 0.0963  0.425   0.7747  0.8005  0.7947  0.7947  0.7947  0.7947  0.7947
   0.7947  0.7947  0.7947  0.7947  0.7947  0.7947  0.7947  0.7947  0.7947
   0.7947  0.7947]
 [ 0.3171  0.7527  0.865   0.8635  0.8552  0.8548  0.8548  0.8548  0.8548
   0.8548  0.8548  0.8548  0.8548  0.8548  0.8548  0.8548  0.8548  0.8548
   0.8548  0.8548]
 [ 0.5782  0.85    0.8867  0.8909  0.8815  0.881   0.881   0.881   0.881
   0.881   0.881   0.881   0.881   0.881   0.881   0.881   0.881   0.881
   0.881   0.881 ]
 [ 0.7981  0.8824  0.9043  0.9086  0.895   0.8926  0.8926  0.8926  0.8926
   0.8926  0.8926  0.8926  0.8926  0.8926  0.8926  0.8926  0.8926  0.8926
   0.8926  0.8926]
 [ 0.8721  0.9038  0.9183  0.9202  0.9135  0.9041  0.903   0.903   0.903
   0.903   0.903   0.903   0.903   0.903   0.903   0.903   0.903   0.903
   0.903   0.903 ]
 [ 0.8935  0.9147  0.9255  0.9298  0.9229  0.9118  0.9071  0.906   0.906
   0.906   0.906   0.906   0.906   0.906   0.906   0.906   0.906   0.906
   0.906   0.906 ]]
```

In [28]:
```
# Find the index of the maximum value in the accuracies table
maxindex = np.array([int(len(Accs)*np.argmax(Accs)/(len(Accs.flatten()))),
print('The index of the maximum accuracy ('+str(Accs[maxindex[0],maxindex[

besthp = hyperparams[maxindex[1]]
bestns = samples[maxindex[0]]
# Determine which sample count-hyperparameter combination this corresponds
print('This corresponds to a hyperparameter of C = '+ str(besthp) + ' when
```

```
The index of the maximum accuracy (0.9298) is: [6 3]
This corresponds to a hyperparameter of C = 7.84759970351e-07 when training on 1000
```

In [29]:
```
besthp = 7.84759970351e-07
bestns = 10000
```

In [30]:
```
# Load test data
testpath = r"mnist\test.mat"
```

```
        mnist_test = utils.loaddata(testpath,_DATA_DIR,'testX')
        predictions = tf.TrainAndPredictNsamples(trainsetarrays[:bestns],trainsetl
```

In [27]:
```
IDs = np.arange(len(predictions))
numpycsv = np.c_[IDs,predictions]
np.savetxt(_LOCAL_PATH+r'\MNIST_testpredictions.csv',numpycsv,fmt='%i',del
```

In [31]:
```
from matplotlib import pyplot as plt
```

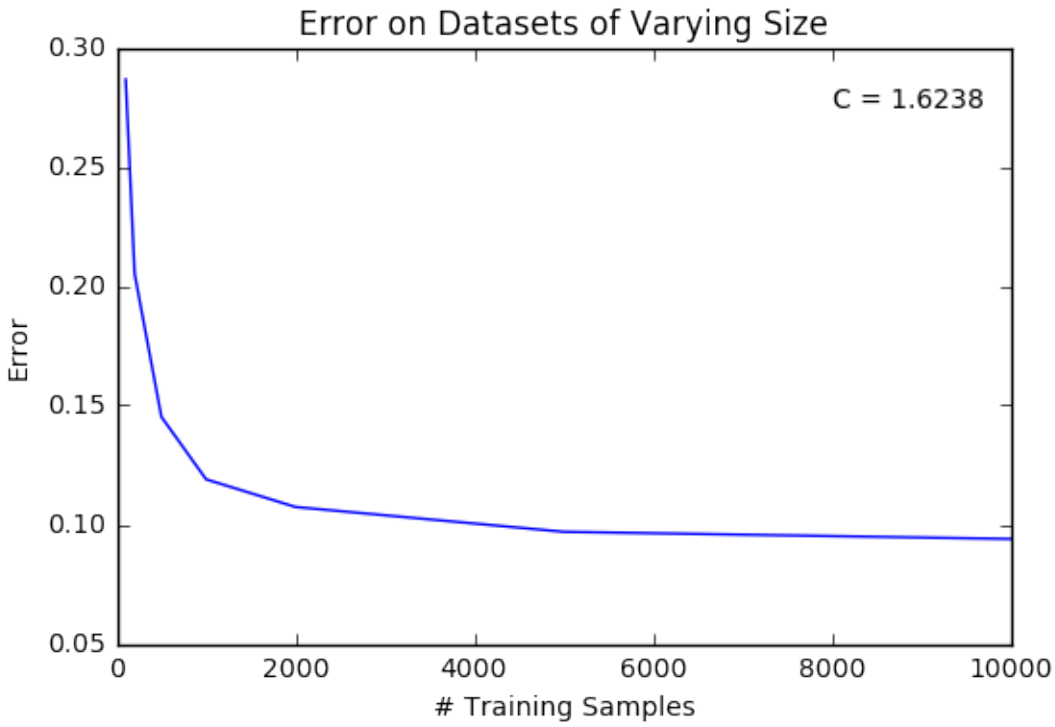
In [32]:
```
hpC1 = 13
```

In [33]:
```
errors = np.ones_like(Accs[:,hpC1])-Accs[:,hpC1]
```

In [42]:
```
fig = plt.figure()
plt.plot(samples,errors)
plt.title('Error on Datasets of Varying Size')
plt.xlabel('# Training Samples')
plt.ylabel('Error')
plt.text(8000,0.275,'C = '+str(round(hyperparams[hpC1],4)))
```

Out[42]:
```
<matplotlib.text.Text at 0x23b99f887f0>
```

In [43]:
```
plt.show()
```



In [47]:
```
fig.savefig(_LOCAL_PATH+r'\Figures\MNIST_SampleAcc.jpg')
```

```
In [ ]: # Export data to csv files for report
        np.savetxt(_LOCAL_PATH+r'\MNIST_Accuracies.csv',Accs,fmt='%f',delimiter=',
        np.savetxt(_LOCAL_PATH+r'\MNIST_hyperparams.csv',hyperparams,fmt='%.8f',del
```