# hw1

February 8, 2016

In [1]:
```
%matplotlib inline
```

```
/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/site-packages/matplotlib/font_manager.py
  warnings.warn('Matplotlib is building the font cache using fc-list. This may take a moment.')
```

In [2]:
```
import scipy.io as sio
import python_helper as ph
import random as rand
from sklearn import svm, metrics
import pylab as plt
```

In [3]:
```
digits = sio.loadmat("data/digit-dataset/train.mat")
N = (len(digits['train_labels']))
population_list = [_ for _ in range(N)]
validation_set = rand.sample(population_list, 10000)
training_set = list(set(population_list) - set(validation_set))
true_labels = [digits["train_labels"][_][0] for _ in validation_set]
```

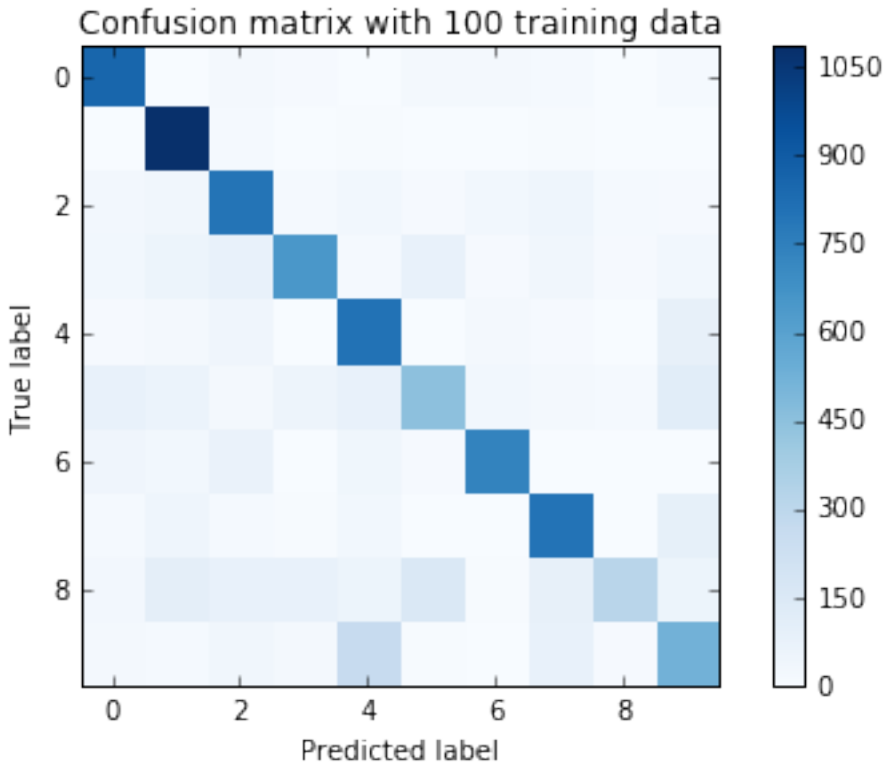
In [4]:
```
def train_and_validation(i):
    rand_indexes = rand.sample(training_set, i)
    sample_labels = []
    sample_images = []
    for j in rand_indexes:
        sample_labels.append(digits["train_labels"][j][0])
        train_image = []
        for m in range(28):
            for n in range(28):
                train_image.append(digits["train_images"][m][n][j])
        sample_images.append(train_image)
    svc = svm.SVC(kernel="linear").fit(sample_images, sample_labels)

    validation_images = []
    for k in validation_set:
        validation_image = []
        for m in range(28):
            for n in range(28):
                validation_image.append(digits["train_images"][m][n][k])
        validation_images.append(validation_image)
    predict_labels=svc.predict(validation_images)
    return predict_labels
```

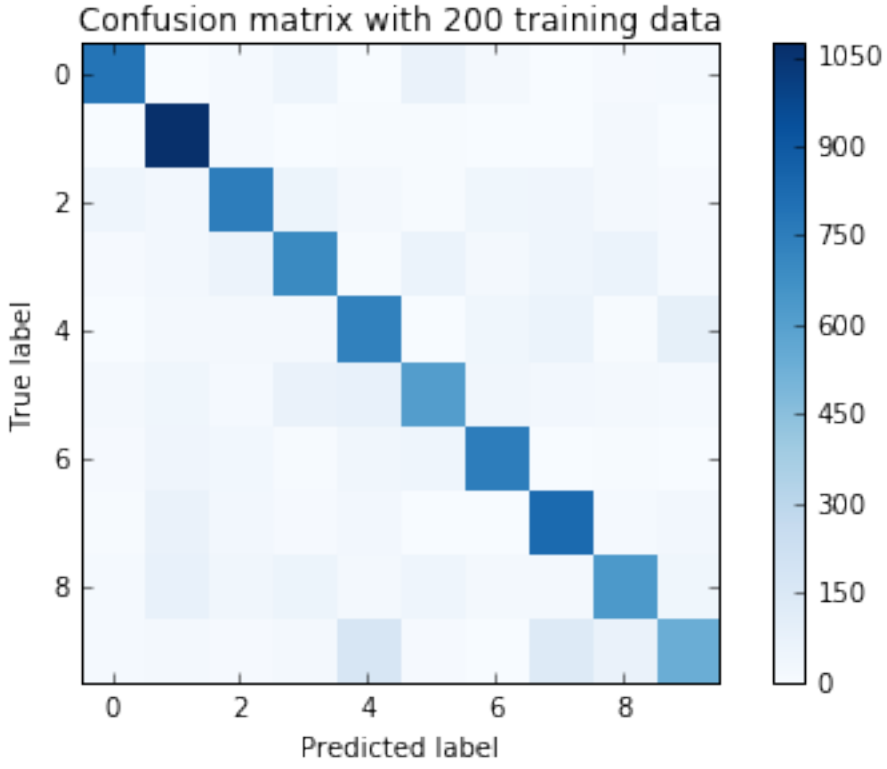
In [5]:
```
##### Problem 1
nums = [100,200,500,1000,2000,5000,10000]
predicted_lables = [train_and_validation(n) for n in nums]
```

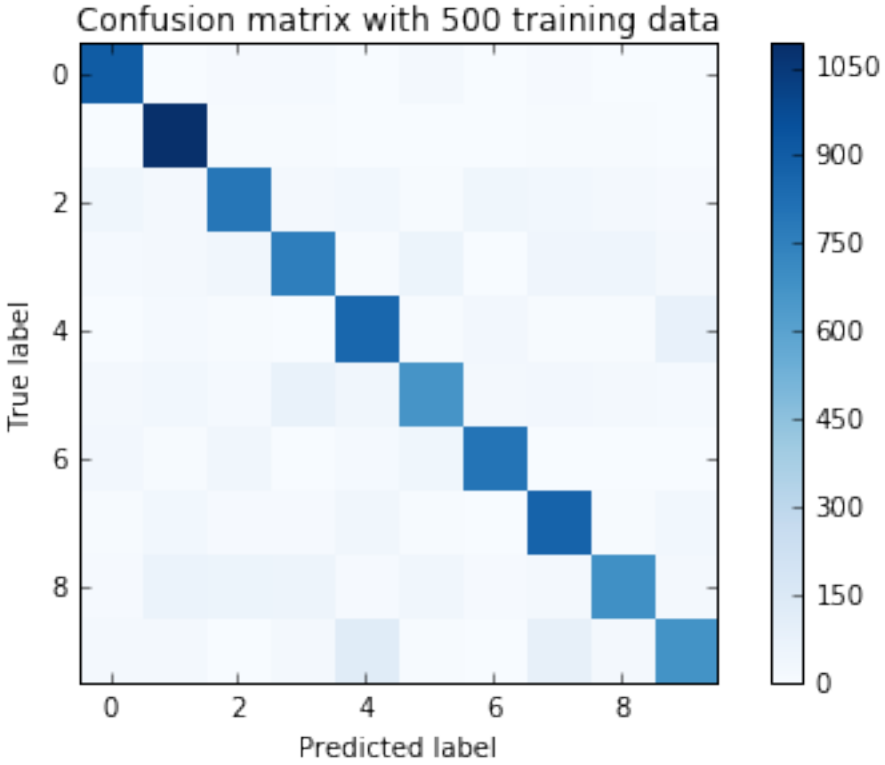
```
error_rate = [ph.benchmark(predicted_lables[i],true_labels) for i in range(len(nums))]
plt.plot(nums, error_rate)
plt.xlabel("Number of Training Data")
plt.ylabel('Error Rate')
plt.title('Number of Training Data vs. Error Rate')
```

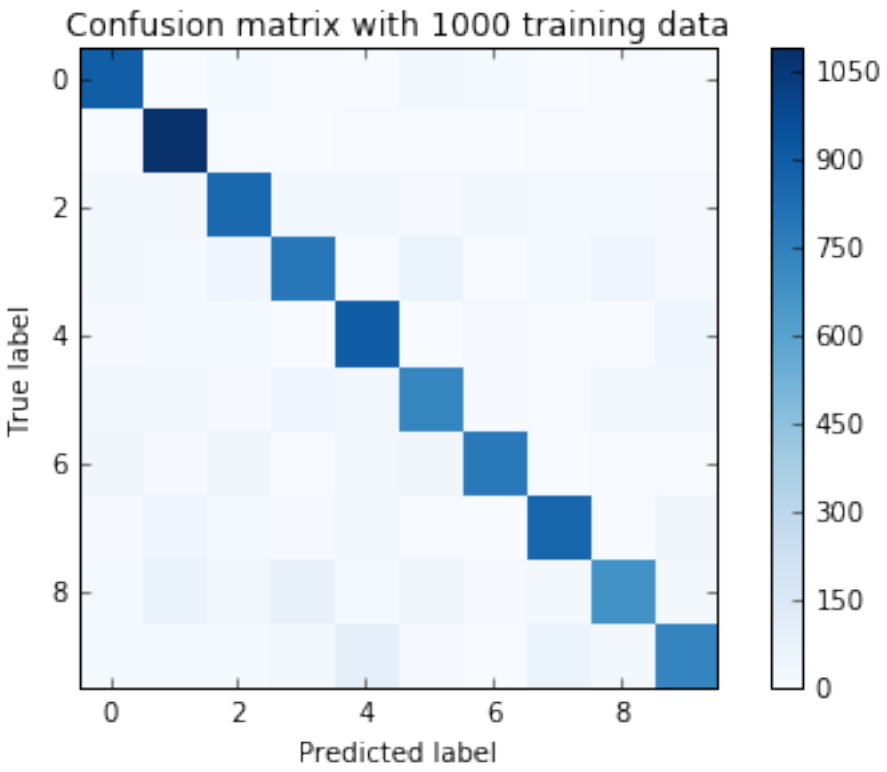
Out[5]: <matplotlib.text.Text at 0x11c3ff390>

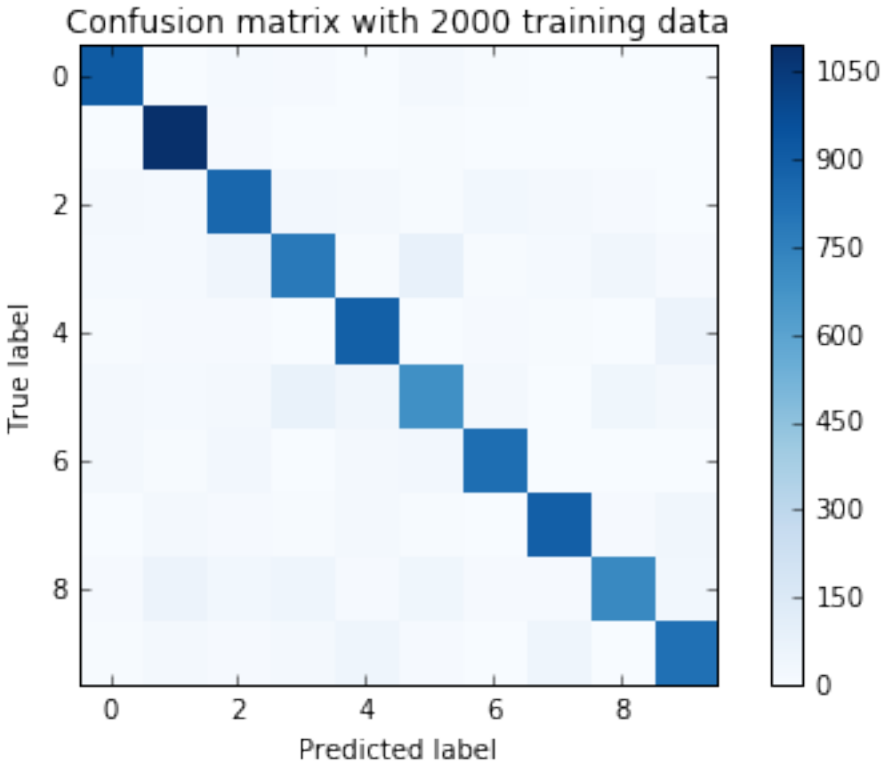

In [7]:
```
##### Problem 2
def plot_confusion_matrix(cm, title='Confusion matrix', cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')


# Compute confusion matrix
for i in range(len(nums)):
    cm = metrics.confusion_matrix(true_labels, predicted_lables[i])
    # np.set_printoptions(precision=2)
    plt.figure()
    plot_confusion_matrix(cm, title = "Confusion matrix with "+str(nums[i])+" training data")
```

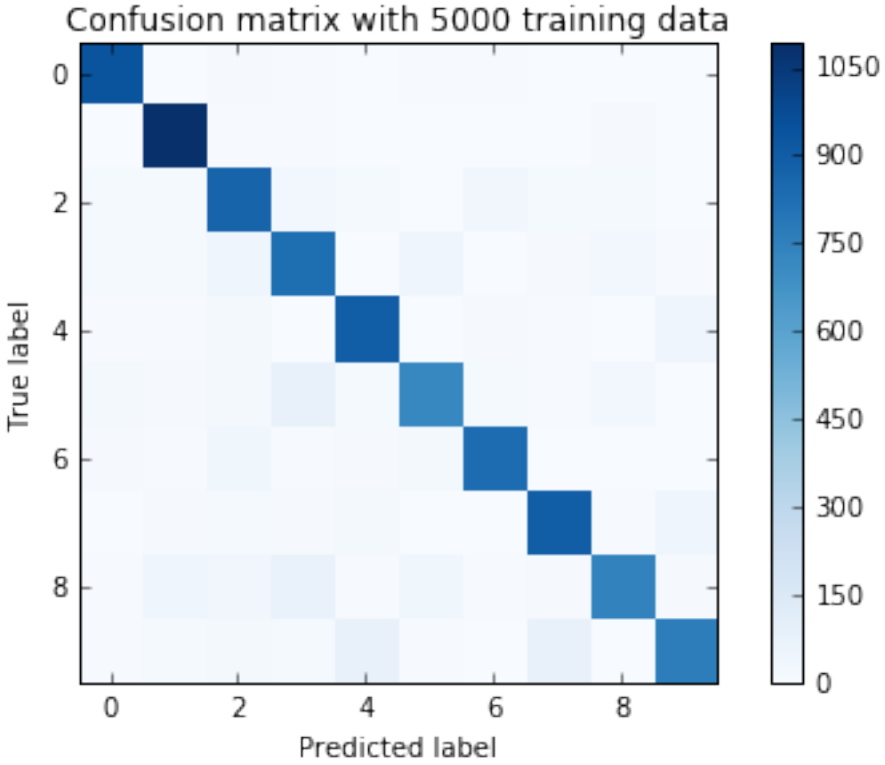

Confusion matrix with 100 training data
True label
Predicted label
0
2
4
6
8
1050
900
750
600
450
300
150
0

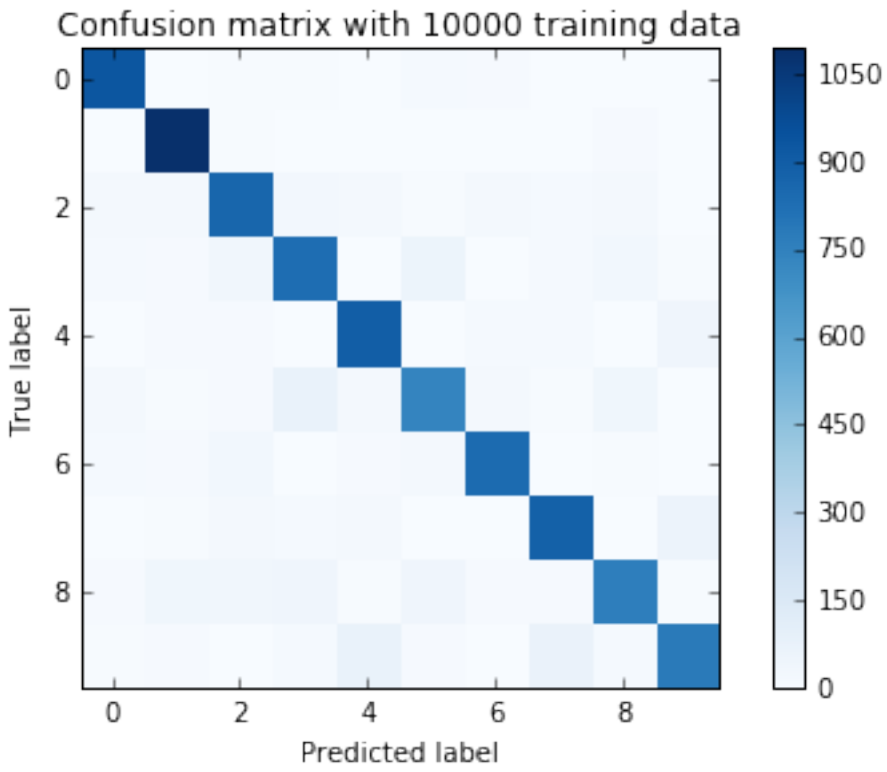

Confusion matrix with 200 training data
True label
Predicted label
0
2
4
6
8
1050
900
750
600
450
300
150
0


Confusion matrix with 500 training data
True label
Predicted label
0
2
4
6
8
1050
900
750
600
450
300
150
0


Confusion matrix with 1000 training data
True label
Predicted label
0
2
4
6
8
1050
900
750
600
450
300
150
0

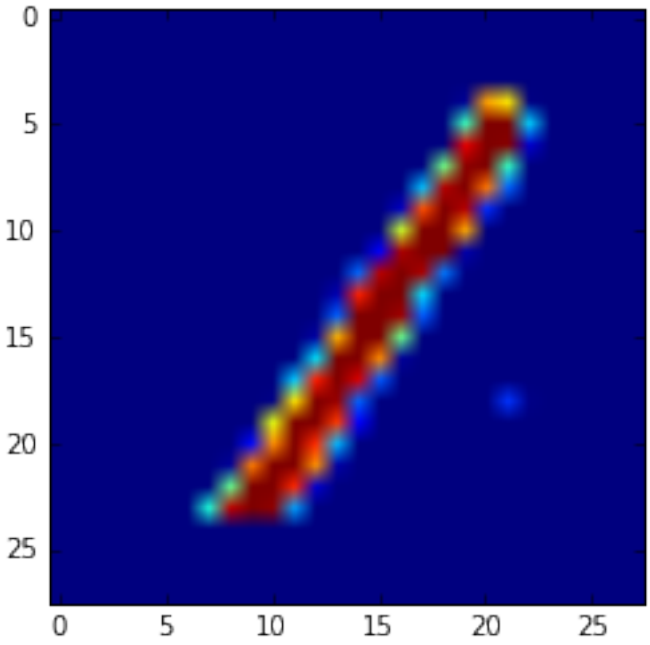

Confusion matrix with 2000 training data
True label
Predicted label
0
2
4
6
8
1050
900
750
600
450
300
150
0


Confusion matrix with 5000 training data
True label
Predicted label
0
2
4
6
8
1050
900
750
600
450
300
150
0




```
In [9]: train_image = []
        for m in range(28):
            row = []
            for n in range(28):
                row.append(digits["train_images"][m][n][10000])
            train_image.append(row)
        plt.imshow(train_image)
```

```
Out[9]: <matplotlib.image.AxesImage at 0x10d8eee10>
```




In [ ]: