4. Multiclass SVM

In [1]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import time
import gzip
import sys
import os
import copy
import numpy as np
import pandas as pd
import pickle
import string
import operator
import bz2
import random
from scipy import stats
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import KMeans
from sklearn.neighbors import BallTree
from sklearn import metrics
from sklearn.preprocessing import scale
from pylab import rcParams
from struct import unpack
from scipy.stats import multivariate_normal
from matplotlib.pyplot import figure
from sklearn.linear_model import Perceptron
from sklearn.svm import LinearSVC
from sklearn.model_selection import GridSearchCV

if sys.version_info[0] == 2:
    from urllib import urlretrieve
else:
    from urllib.request import urlretrieve

if not sys.warnoptions:
    import warnings
    warnings.simplefilter("ignore")
```

(a) Load in the MNIST data: a training set of 60,000 points and a separate test set of 10,000 points.

This will set x to a 60000 x 784 array where each row corresponds to an image, and y to a length-60000 array where each entry is a label (0-9). There is also a routine to display images: use displaychar(x[0]) to show the first data point, for instance.

In [2]:
```
def download(filename, source = 'http://yann.lecun.com/exdb/mnist/index.html'):
    print("Downloading %s" % filename)
    urlretrieve(source + filename, filename)

def load_mnist_images(filename):
    if not os.path.exists(filename):
        download(filename)
    # Read the inputs in Yann LeCun's binary format.
    with gzip.open(filename, 'rb') as f:
```

```
        data = np.frombuffer(f.read(), np.uint8, offset = 16)
    data = data.reshape(-1,784)
    return data / np.float32(256)

def load_mnist_labels(filename):
    if not os.path.exists(filename):
        download(filename)
    with gzip.open(filename, 'rb') as f:
        data = np.frombuffer(f.read(), np.uint8, offset = 8)
    return data

def displaychar(image):
    plt.imshow(np.reshape(image, (28,28)), cmap=plt.cm.gray)
    plt.axis('off')
    plt.show()
```

In [3]:
```
# Load the training data set
train_data = load_mnist_images('train-images-idx3-ubyte.gz')
train_labels = load_mnist_labels('train-labels-idx1-ubyte.gz')

print('Shape of the train data:\n', train_data.shape)
print('\nShape of the train labels:\n', train_labels.shape)
```

```
Shape of the train data:
 (60000, 784)

Shape of the train labels:
 (60000,)
```

In [4]:
```
# Load the testing data set
test_data = load_mnist_images('t10k-images-idx3-ubyte.gz')
test_labels = load_mnist_labels('t10k-labels-idx1-ubyte.gz')

print('Shape of the test data:\n', test_data.shape)
print('\nShape of the test labels:\n', test_labels.shape)
```

```
Shape of the test data:
 (10000, 784)

Shape of the test labels:
 (10000,)
```

(b) Learn a linear SVM classifier using sklearn.svm.LinearSVC. You will need to see loss='hinge'. How can you choose a suitable value of C? Explain your methodology.

In [5]:
```
clf_train = LinearSVC(C = 1.0, loss = 'hinge', random_state = 42, tol = 1e-05)
clf_train.fit(train_data, train_labels)
```

Out[5]:
```
LinearSVC(loss='hinge', random_state=42, tol=1e-05)
```

In [6]:
```
clf_train.score(train_data, train_labels)
```

Out[6]:
```
0.9294666666666667
```

In [7]:
```
# choose a suitable value of C
parameters = {'C':[1e-5, 0.01, 0.1, 1, 10, 100, 1000]}
clf_c = LinearSVC(loss = 'hinge', random_state = 42, tol = 1e-05)
clf_c = GridSearchCV(clf_c, parameters)
clf_c.fit(train_data, train_labels)
sorted(clf_c.cv_results_.keys())
# print best parameter after tuning
print('The best C is: ',clf_c.best_params_)

# print how our model looks after hyper-parameter tuning
print(clf_c.best_estimator_)
```

```
The best C is:  {'C': 1}
LinearSVC(C=1, loss='hinge', random_state=42, tol=1e-05)
```

(c) Report the final test error. Is this data linearly separable?

In [8]:
```
clf = LinearSVC(C = 1, loss = 'hinge', random_state = 42, tol = 1e-05)
clf.fit(train_data, train_labels)
score = clf.score(test_data, test_labels)
error = 1 - clf.score(test_data, test_labels)
score, error
```

Out[8]:
```
(0.922, 0.07799999999999996)
```

In [9]:
```
print('The test error is:', error)
```

```
The test error is: 0.07799999999999996
```

In [12]:
```
print('Since this is not converged and the dataset has more than 2 target values
```

```
Since this is not converged and the dataset has more than 2 target values, the d
ataset is non-linearly separable.
```

In [ ]:
```

```