In [29]:
```
#Practical-6 : Classification of Images from Not-MNIST dataset using SVM Class
ifier
#Shubham S Kale
import os
import struct
import numpy as np

import csv
import random
import math
import operator


def read(dataset = "training", path = "."):
    if dataset is "training":
        fname_img = os.path.join(path, 'train-images')
        fname_lbl = os.path.join(path, 'train-labels')
    elif dataset is "testing":
        fname_img = os.path.join(path, 'test-images')
        fname_lbl = os.path.join(path, 'test-labels')
    else:
        raise ValueError, "dataset must be 'testing' or 'training'"

    # Load everything in some numpy arrays
    with open(fname_lbl, 'rb') as flbl:
        magic, num = struct.unpack(">II", flbl.read(8))
        lbl = np.fromfile(flbl, dtype=np.int8)

    with open(fname_img, 'rb') as fimg:
        magic, num, rows, cols = struct.unpack(">IIII", fimg.read(16))
        img = np.fromfile(fimg, dtype=np.uint8).reshape(len(lbl), rows, cols)

    get_img = lambda idx: (lbl[idx], img[idx])

    # Create an iterator which returns each image in turn
    for i in xrange(len(lbl)):
        yield get_img(i)

def show(image):
    from matplotlib import pyplot
    import matplotlib as mpl
    fig = pyplot.figure()
    ax = fig.add_subplot(1,1,1)
    imgplot = ax.imshow(image, cmap=mpl.cm.Greys)
    imgplot.set_interpolation('nearest')
    ax.xaxis.set_ticks_position('top')
    ax.yaxis.set_ticks_position('left')
    pyplot.show()
```

In [30]:
```
training_data = list(read(dataset='training', path='.'))
testing_data = list(read(dataset='testing', path='.'))

print len(training_data)
print len(testing_data)
```

```
60000
10000
```

In [31]:
```
tr_dt = np.zeros(shape=(60000,784))
tr_lbl = np.zeros(shape=(60000,1))
ts_dt = np.zeros(shape=(10000,784))
ts_lbl = np.zeros(shape=(10000,1))
```

In [32]:
```
for i in xrange(len(training_data)):
    label, pixels = training_data[i]
    tr_dt[i,:]  = pixels.reshape((1,784))
    tr_lbl[i,:] = label
for i in xrange(len(testing_data)):
    label, pixels = testing_data[i]
    ts_dt[i,:]  = pixels.reshape((1,784))
    ts_lbl[i,:] = label
```

In [44]:
```
import random

num_of_samples_for_training = 5000
num_of_samples_for_testing  = 250


indices_train = random.sample(range(0, 59999), num_of_samples_for_training)
traindata   = tr_dt[indices_train,:]
trainlabel = tr_lbl[indices_train,:]


indices_test = random.sample(range(0, 9999), num_of_samples_for_testing)
testdata   = ts_dt[indices_test,:]
testlabel = ts_lbl[indices_test,:]
```

In [45]:
```
print 'trainset   = ', traindata.shape
print 'testset    = ', testdata.shape
print 'trainlabel = ', trainlabel.shape
print 'testlabel  = ', testlabel.shape
```

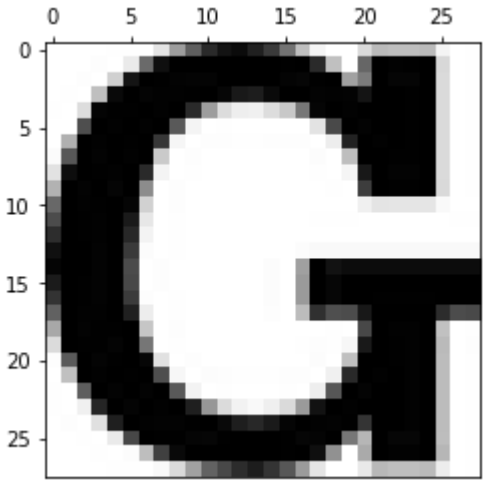
```
trainset   =  (5000L, 784L)
testset    =  (250L, 784L)
trainlabel =  (5000L, 1L)
testlabel  =  (250L, 1L)
```

In [46]:
```
trainingSet = np.concatenate((traindata, trainlabel), axis=1)
testSet = np.concatenate((testdata, testlabel), axis=1)
print trainingSet.shape
print testSet.shape
```

```
(5000L, 785L)
(250L, 785L)
```

In [47]:
```
label, pixels = training_data[500]
print(label)
print(pixels.shape)
show(pixels)
```

```
6
(28L, 28L)
```



In [48]:
```
from sklearn import svm

sup_vec= svm.SVC()

print('SVM score: %f' % sup_vec.fit(traindata, trainlabel.ravel()).score(testd
ata, testlabel.ravel()))
```

```
SVM score: 0.116000
```

In [ ]:
```

```