Homework 3 Part 2

This is a group (final project group) assignment

In this problem, you will working with the final project's training data. Be sure to download the data files from Canvas:

1. "data_train.npy"
2. "t_train.npy"

In this assignment you are allowed to use any libraries you want.

Computing Resources

- Use our HiPerGator cluster **"eel5840"**.
- The training data (data and labels) has been uploaded to the "share" folder. You may read directly from that folder.

Training Data Set

The training data set is the same for every team in this course and it contains 70% randomly selected samples collected by all individuals in this course.

You should expect to receive the blind *easy test set* in the same format (numpy arrays with similar dimensions).

Let's load the training data:

In [1]:
```
import numpy as np
import numpy.random as npr
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('bmh')
```

In [2]:
```
# Labels Integer Encoding

labels_names = ['x','square root','plus sign','negative sign',
                'equal','percent','partial','product','pi','summation']
```

In [3]:
```
# Loading Data
data_train = np.load('data_train.npy')
labels_train = np.load('t_train.npy')

print(data_train.shape, labels_train.shape)
```

```
(90000, 9032) (9032,)
```

We can see from the plot below that each class' representation in the training data is well-balanced:

In [ ]:
```
# Counting number samples per class
vals, counts = np.unique(labels_train, return_counts=True)

plt.bar(vals, counts)
plt.xticks(range(10),range(10))
plt.xlabel('Classes',size=20)
plt.ylabel('# Samples per Class', size=20)
plt.title('Training Data (Total = '+str(data_train.shape[1])+' samples)',size=15);
```

Let's now display some examples of the training samples for each one of the 10 classes:

In [ ]:
```
# Displaying some examples per class

for i in range(0,10):
    rnd_sample = npr.permutation(np.where(labels_train==i)[0])
    fig=plt.figure(figsize=(15,15))
    for j in range(25):
        fig.add_subplot(5,5,j+1)
        plt.imshow(data_train[:,rnd_sample[j]].reshape((300,300)), cmap='gray')
        ll = int(labels_train[rnd_sample[j]])
        plt.axis('off');plt.title(str(labels_names[ll])+' ('+str(ll)+')',size=15)
    plt.show()
    print('\n\n')
```

Note that you can **invert the color**:

In [ ]:
```
data_train = 255-data_train
```

In [ ]:
```
# # Displaying some examples per class

# for i in range(0,10):
#     rnd_sample = npr.permutation(np.where(labels_train==i)[0])
#     fig=plt.figure(figsize=(15,15))
#     for j in range(25):
#         fig.add_subplot(5,5,j+1)
#         plt.imshow(data_train[:,rnd_sample[j]].reshape((300,300)), cmap='gray')
#         ll = int(labels_train[rnd_sample[j]])
#         plt.axis('off');plt.title(str(labels_names[ll])+' ('+str(ll)+')',size=15)
#     plt.show()
#     print('\n\n')
```

Objectives

The goal of this assignment is provide a set of structured tasks to get you started on the final project for EEL5840.

By the end of this assignment, you will:

1. Define a set of ground rules to work within your group.
2. Discuss and identify challenges imposed by the data collection step.
3. Design a preliminary plan for a standard Machine Learning pipeline (data collection - feature extraction - training the model - evaluation).
4. Apply off-the-shelf libraries to train an algorithm and perform hyperparameter tuning with cross-validation.
5. Discuss your choice for the baseline model.
6. Make a concrete plan for the next steps of the project.

Notes

This assignment does not expect algorithms to obtain accuracy scores near 90% in your validation set (in fact, it may be considerably smaller than that).

Solutions will not be posted as this assignment is preliminary work that may be used directly towards your final project. But you will receive feedback on your work and discussions.

Problem 1 (6 points)

As a group, answer the following questions:

1. Individually create **at least 4 ground rules** for working in a group. Share your answers with your team members, and together, come up with a consensus set of ground rules for working in (your) group. Submit **at least 4 ground rules** that all team members have agreed to follow when working together in the final project.

1.) All communications will occur over Microsoft Teams.

2.) All members will be present for all group meetings.

3.) All members will contribute towards the final project in approximately equal roles.

4.) All members will communicate with one another respectfully and promptly.

5.) Work hard play hard.

Problem 2 (6.5 points)

Analyze the data by performing image visualizations and any other appropriate data processing techniques. Provide a discussion about the training dataset. Are there are classes that appear to

be challenging? Are they any images that are mislabeled? Are classes balanced? Are there are pre-processing strategies would you apply prior to feature extraction?

After inspection of the training set images, it is evident there are numerous misclassified images, improperly oriented images, and images that contain no symbol. As such, the images were manually relabled by the group, rotated when necessary, or simply discarded from the training set. The two sets of symbols that are most difficult to classify are x and + since their appearances can be quite similar under the proper rotation, and the product symbol and pi which differ essentially only in size. Overall, the classes should be approximately balanced since the same number of images were to be obtained for each symbol type.

==========

Example

If you wish to relabel images (recommended), here's an example using visualization:

In [ ]:
```
# # Helper function for data visualization
import math

def class_visualization(label, X, t, class_names, figsize=(15,150)):

    # Index locations where class label stored in data_train
    sample_idx = np.where(t==label)[0]

    #Number of samples in class label
    N = len(sample_idx)

    # settings for grid plot
    grid_rows = math.ceil(N/10)
    grid_columns = 10

    # Plotting all images
    plt.figure(figsize=figsize)
    for i in range(N):
        plt.subplot(grid_rows, grid_columns, i+1)
        plt.imshow(X[:,sample_idx[i]].reshape(300,300), cmap='gray')
        plt.axis('off')
        plt.title(class_names[label]+'('+str(label)+'): '+str(sample_idx[i]));

class_visualization(0, data_train, labels_train, labels_names)
```

If there are any images that appear small and I can't distinguish, I'll plot them individually for better inspection:

In [ ]:
```
## View individual images
idx = 70
plt.imshow(data_train[:,idx].reshape(300,300), cmap='gray')
plt.axis('off')
plt.title(labels_names[0]+'(0): '+str(idx));
```

In [ ]:
```
#Sample code to rotate images.
#For np.rot90: k = 1 (default) rotates 90 deg counter clockwise while k = 3 rotates 90

rotate_this1 = data_train[:,70]
rotated_image1 = np.rot90(rotate_this1.reshape(300,300))
data_train[:,70] = rotated_image1.reshape(90000)
plt.imshow(data_train[:,70].reshape(300,300),cmap='gray')
```

In [ ]:
```
# # Images identified as misclassified by visualization
idx_to_correct = np.array([495, 2099, 2862, 2976, 3357, 3575, 3628, 3838, 4186, 4631,
labels_correct = np.array([  1,   8,    1,    8,    6,    8,   -1,    1,    4,    4,
                            -1,    3,    6,    1,   -1,    1,    5,    3,   -1])

# # Update labels for class label
labels_train[idx_to_correct] = labels_correct

# # Visualize data to make sure
class_visualization(0, data_train, labels_train, labels_names)
```

In [ ]:
```
## Nathan's misclassified images
labels_train[2271] = 3; labels_train[2274] = 10; labels_train[2286] = 3
labels_train[2310] = 4; labels_train[2310] = 4; labels_train[2319] = 10
labels_train[2324] = 3; labels_train[2327] = 10; labels_train[2359] = 10
labels_train[2406] = 10; labels_train[2440] = 10; labels_train[2483] = 8
labels_train[2497] = 3; labels_train[2505] = 10; labels_train[2524] = 10
labels_train[2530] = 10; labels_train[2536] = 10; labels_train[2546] = 5
labels_train[2552] = 10; labels_train[2553] = 0; labels_train[2583] = 2
labels_train[2588] = 10; labels_train[2605] = 0; labels_train[2608] = 10
labels_train[2671] = 10; labels_train[2711] = 10; labels_train[2722] = 9
labels_train[2724] = 8; labels_train[2737] = 10; labels_train[2769] = 8
labels_train[2790] = 8; labels_train[2803] = 10; labels_train[2812] = 10
labels_train[2815] = 9; labels_train[2852] = 0; labels_train[2931] = 3
labels_train[2976] = 8; labels_train[2987] = 5; labels_train[2991] = 5
labels_train[3159] = 8; labels_train[3161] = 10; labels_train[3214] = 10
labels_train[3254] = 0; labels_train[3261] = 10; labels_train[3214] = 10
labels_train[3254] = 0; labels_train[3261] = 10; labels_train[3284] = 6
labels_train[3357] = 6; labels_train[3424] = 9; labels_train[3464] = 2
labels_train[3487] = 10; labels_train[3532] = 10; labels_train[3547] = 10
labels_train[3575] = 7; labels_train[3587] = 10; labels_train[3615] = 9
labels_train[3618] = 6; labels_train[3628] = 10; labels_train[3695] = 10
labels_train[3712] = 3; labels_train[3721] = 2; labels_train[3725] = 3
labels_train[3732] = 7; labels_train[3795] = 2; labels_train[3836] = 6
labels_train[3838] = 1; labels_train[3842] = 10; labels_train[3961] = 10
labels_train[4016] = 7; labels_train[4067] = 10; labels_train[4073] = 10
labels_train[4114] = 5; labels_train[4146] = 5; labels_train[4161] = 9
labels_train[4167] = 8; labels_train[4174] = 1; labels_train[4186] = 4
labels_train[4190] = 10; labels_train[4215] = 2; labels_train[4234] = 10
labels_train[4277] = 0; labels_train[4284] = 10; labels_train[4288] = 10
labels_train[4297] = 4; labels_train[4362] = 10; labels_train[4365] = 10
labels_train[4368] = 10; labels_train[4372] = 10; labels_train[4386] = 10
labels_train[4398] = 10; labels_train[4400] = 9; labels_train[4409] = 4
labels_train[4458] = 5; labels_train[4471] = 10; labels_train[4488] = 10
```

In [ ]:
```
# ## Nathan's images to be rotated
# ## For np.rot90: k = 1 (default) rotates 90 deg counter clockwise while k = 3 rotate
rotation_index = [2266, 2336, 2425, 2505, 2514, 2522, 2524, 2551, 2631, 2645, 2665,
```

```
                  2668, 2669, 2714, 2721, 2731, 2758, 2801, 2821, 2838, 2870, 2885,
                  2901, 2903, 2918, 2925, 2934, 2941, 2953, 2956, 2967, 2974, 2977,
                  3017, 3050, 3052, 3073, 3097, 3116, 3151, 3169, 3194, 3209, 3212,
                  3255, 3306, 3309, 3317, 3393, 3475, 3492, 3511, 3561, 3762, 3774,
                  3793, 3802, 3821, 3825, 3828, 3845, 3861, 3862, 3877, 3890, 3929,
                  3938, 3943, 3945, 3981, 4011, 4014, 4074, 4096, 4098, 4107, 4140,
                  4144, 4158, 4165, 4170, 4179, 4189, 4222, 4312, 4313, 4323, 4330,
                  4359, 4364, 4401, 4412, 4492]

rotation_index2 = [4284]
rotation_index3 = [2530, 2815, 3144, 3172, 3615, 3645, 3836, 4161, 4461]

# ## Rotate by 90 deg.
for i in range(len(rotation_index)):
    rotate_this = data_train[:,rotation_index[i]]
    rotated_image = np.rot90(rotate_this.reshape(300,300), k=3)
    data_train[:,rotation_index[i]] = rotated_image.reshape(90000)

# ## Rotate by 180 deg.
for i in range(len(rotation_index2)):
    rotate_this = data_train[:,rotation_index2[i]]
    rotated_image = np.rot90(rotate_this.reshape(300,300), k=2)
    data_train[:,rotation_index2[i]] = rotated_image.reshape(90000)

# ## Rotate by -90 deg.
for i in range(len(rotation_index3)):
    rotate_this = data_train[:,rotation_index3[i]]
    rotated_image = np.rot90(rotate_this.reshape(300,300), k=1)
    data_train[:,rotation_index3[i]] = rotated_image.reshape(90000)
```

In [ ]:

```
## Amit's misclassified images
labels_train[4532] = 2; labels_train[4554] = 5; labels_train[4590] = 5;
labels_train[4657] = 8; labels_train[4687] = 2; labels_train[4714] = 7;
labels_train[4752] = 2; labels_train[4794] = 10; labels_train[4829] = 10;
labels_train[4840] = 8; labels_train[4866] = 7; labels_train[4892] = 10;
labels_train[4927] = 6; labels_train[5002] = 9; labels_train[5011] = 7;
labels_train[5030] = 1; labels_train[5041] = 8; labels_train[5048] = 5;
labels_train[5106] = 10; labels_train[5114] = 10; labels_train[5135] = 10;
labels_train[5142] = 2; labels_train[5149] = 10; labels_train[5169] = 2;
labels_train[5191] = 8; labels_train[5241] = 8; labels_train[5342] = 3;
labels_train[5373] = 8; labels_train[5389] = 8; labels_train[5448] = 8;
labels_train[5499] = 7; labels_train[5528] = 5; labels_train[5536] = 4;
labels_train[5563] = 3; labels_train[5614] = 10; labels_train[5650] = 10;
labels_train[5667] = 8; labels_train[5718] = 6; labels_train[5725] = 6;
labels_train[5730] = 1; labels_train[5770] = 10; labels_train[5791] = 10;
labels_train[5792] = 2; labels_train[5797] = 10; labels_train[5801] = 10;
labels_train[5850] = 7; labels_train[5864] = 10; labels_train[5886] = 10;
labels_train[5896] = 10; labels_train[5899] = 2; labels_train[5992] = 2;
labels_train[6001] = 10; labels_train[6067] = 10; labels_train[6074] = 10;
labels_train[6082] = 9; labels_train[6103] = 1; labels_train[6109] = 1;
labels_train[6116] = 7; labels_train[6174] = 8; labels_train[6216] = 10;
labels_train[6221] = 10; labels_train[6242] = 8; labels_train[6266] = 10;
labels_train[6318] = 10; labels_train[6326] = 10; labels_train[6328] = 3;
labels_train[6338] = 5; labels_train[6339] = 0; labels_train[6354] = 4;
labels_train[6369] = 7; labels_train[6373] = 4; labels_train[6382] = 2;
labels_train[6399] = 10; labels_train[6434] = 3; labels_train[6471] = 6;
labels_train[6490] = 1; labels_train[6505] = 8; labels_train[6572] = 10;
```

```
labels_train[6609] = 6; labels_train[6616] = 10; labels_train[6639] = 7;
labels_train[6662] = 10; labels_train[6716] = 9; labels_train[6722] = 8;
labels_train[6736] = 1;
```

In [ ]:
```
# ## Amit's images to be rotated.
# ## For np.rot90: k = 1 (default) rotates 90 deg counter clockwise while k = 3 rotate
rotation_index = [4573, 4601, 4626, 4627, 4644, 4658, 4691, 4846, 4853, 4918, 5001, 50
                  5037, 5057, 5083, 5272, 5325, 5341, 5348, 5361, 5405, 5441, 5540, 556
                  5597, 5599, 5601, 5606, 5656, 5659, 5668, 5734, 5749, 5795, 5917, 592
                  5922, 5995, 6020, 6021, 6048, 6057, 6090, 6118, 6138, 6193, 6147, 617
                  6203, 6249, 6293, 6345, 6347, 6348, 6362, 6414, 6416, 6463, 6501, 650
                  6515, 6588, 6640, 6641, 6646, 6650, 6668, 6704, 6764]

rotation_index2 = [5505, 5084, 5910]
rotation_index3 = [6473, 6543]

# ## Rotate by 90 deg.
for i in range(len(rotation_index)):
    rotate_this = data_train[:,rotation_index[i]]
    rotated_image = np.rot90(rotate_this.reshape(300,300), k=3)
    data_train[:,rotation_index[i]] = rotated_image.reshape(90000)

# ## Rotate by 180 deg.
for i in range(len(rotation_index2)):
    rotate_this = data_train[:,rotation_index2[i]]
    rotated_image = np.rot90(rotate_this.reshape(300,300), k=2)
    data_train[:,rotation_index2[i]] = rotated_image.reshape(90000)

# ## Rotate by -90 deg.
for i in range(len(rotation_index3)):
    rotate_this = data_train[:,rotation_index3[i]]
    rotated_image = np.rot90(rotate_this.reshape(300,300), k=1)
    data_train[:,rotation_index3[i]] = rotated_image.reshape(90000)
```

In [ ]:
```
## Archit's misclassified images.
labels_train[32] = 4; labels_train[46] = 10; labels_train[54] = 6;
labels_train[61] = 10; labels_train[70] = 7; labels_train[85] = 7;
labels_train[92] = 3; labels_train[116] = 10; labels_train[119] = 4;
labels_train[120] = 7; labels_train[161] = 10; labels_train[162] = 9;
labels_train[177] = 1; labels_train[178] = 8; labels_train[181] = 3;
labels_train[188] = 0; labels_train[212] = 3;  labels_train[312] = 5;
labels_train[316] = 7; labels_train[358] = 0;  labels_train[383] = 4;
labels_train[411] = 9; labels_train[427] = 8; labels_train[441] = 1;
labels_train[444] = 9; labels_train[456] = 10; labels_train[469] = 10;
labels_train[485] = 10; labels_train[496] = 10; labels_train[502] = 10;
labels_train[510] = 3; labels_train[524] = 0; labels_train[540] = 6;
labels_train[563] = 6; labels_train[582] = 4; labels_train[586] = 9;
labels_train[594] = 4; labels_train[617] = 10; labels_train[648] = 6;
labels_train[706] = 10; labels_train[722] = 0; labels_train[748] = 1;
labels_train[750] = 10; labels_train[752] = 9; labels_train[759] = 9;
labels_train[790] = 0; labels_train[832] = 10; labels_train[964] = 9;
labels_train[966] = 0; labels_train[1023] = 9; labels_train[1039] = 10;
labels_train[1047] = 4; labels_train[1091] = 3; labels_train[1115] = 4;
labels_train[1142] = 9; labels_train[1179] = 0; labels_train[1248] = 1;
labels_train[1267] = 9; labels_train[1321] = 9; labels_train[1341] = 9;
labels_train[1371] = 4; labels_train[1390] = 6; labels_train[1424] = 3;
labels_train[1454] = 10; labels_train[1504] = 6; labels_train[1506] = 3;
```

```
labels_train[1522] = 4; labels_train[1524] = 1; labels_train[1532] = 1;
labels_train[1538] = 0; labels_train[1601] = 9; labels_train[1708] = 6;
labels_train[1773] = 1; labels_train[1786] = 10; labels_train[1830] = 10;
labels_train[1835] = 8; labels_train[1842] = 0; labels_train[1851] = 6;
labels_train[1878] = 7; labels_train[1945] = 10; labels_train[1957] = 3;
labels_train[2000] = 9; labels_train[2033] = 3; labels_train[2047] = 9;
labels_train[2051] = 4; labels_train[2099] = 7; labels_train[2148] = 4;
labels_train[2172] = 10; labels_train[2187] = 6; labels_train[2202] = 10;
labels_train[2204] = 4; labels_train[2233] = 0; labels_train[2236] = 9;
labels_train[2256] = 10; labels_train[2266] = 7; labels_train[2286] = 3;
labels_train[2324] = 3; labels_train[2327] = 10; labels_train[2336] = 6;
labels_train[2406] = 10; labels_train[2457] = 4; labels_train[2497] = 3;
labels_train[2505] = 0; labels_train[2514] = 0; labels_train[2522] = 1;
labels_train[2524] = 7; labels_train[2551] = 1; labels_train[2553] = 1;
```

In [ ]:

```
# ## Archit's images to be rotated.
# ## For np.rot90: k = 1 (default) rotates 90 deg counter clockwise while k = 3 rotate
rotation_index = [70, 85, 92, 119, 120, 177, 178, 181, 188, 212, 316, 383, 411, 427,
                  441, 540, 563, 582, 586, 594, 748, 752, 759, 1023, 1047, 1091, 1115,
                  1142, 1179, 1248, 1267, 1321, 1341, 1371, 1390, 1424, 1504, 1506, 152
                  1524, 1532, 1538, 1601, 1773, 1835, 1842, 1851, 1878, 1957, 2000, 203
                  2047, 2051, 2148, 2204, 2266, 2336, 2457, 2505, 2514, 2522, 2524, 255

rotation_index3 = [648, 722, 964, 1708, 2187, 2236]

# ## Rotate by 90 deg.
for i in range(len(rotation_index)):
    rotate_this = data_train[:,rotation_index[i]]
    rotated_image = np.rot90(rotate_this.reshape(300,300), k=3)
    data_train[:,rotation_index[i]] = rotated_image.reshape(90000)

 ## Rotate by -90 deg.
for i in range(len(rotation_index3)):
    rotate_this = data_train[:,rotation_index3[i]]
    rotated_image = np.rot90(rotate_this.reshape(300,300), k=1)
    data_train[:,rotation_index3[i]] = rotated_image.reshape(90000)
```

In [ ]:

```
## Humberto's misclassified images
labels_train[6777] = 8; labels_train[6780] = 8; labels_train[6791] = 10;
labels_train[6800] = 10; labels_train[6801] = 8; labels_train[6912] = 10;
labels_train[6928] = 8; labels_train[6959] = 1; labels_train[6963] = 10;
labels_train[7000] = 10; labels_train[7094] = 1; labels_train[7207] = 10;
labels_train[7224] = 10; labels_train[7331] = 8; labels_train[7336] = 10;
labels_train[7356] = 5; labels_train[7360] = 2; labels_train[7370] = 10;
labels_train[7377] = 8; labels_train[7418] = 10; labels_train[7447] = 10;
labels_train[7451] = 8; labels_train[7461] = 3; labels_train[7494] = 10;
labels_train[7499] = 8; labels_train[7507] = 10; labels_train[7512] = 10;
labels_train[7522] = 2; labels_train[7557] = 6; labels_train[7600] = 6;
labels_train[7602] = 3; labels_train[7604] = 6; labels_train[7606] = 10;
labels_train[7618] = 3; labels_train[7619] = 10; labels_train[7622] = 6;
labels_train[7662] = 1; labels_train[7675] = 10; labels_train[7695] = 5;
labels_train[7697] = 10; labels_train[7704] = 0; labels_train[7716] = 10;
labels_train[7765] = 10; labels_train[7774] = 7; labels_train[7786] = 0;
labels_train[7793] = 0; labels_train[7806] = 1; labels_train[7846] = 5;
labels_train[7855] = 10; labels_train[7961] = 10; labels_train[8033] = 7;
labels_train[8058] = 0; labels_train[8070] = 7; labels_train[8089] = 10;
labels_train[8095] = 8; labels_train[8119] = 8; labels_train[8157] = 1;
```

```
labels_train[8200] = 0; labels_train[8278] = 6; labels_train[8280] = 4;
labels_train[8293] = 0; labels_train[8299] = 0; labels_train[8413] = 8;
labels_train[8438] = 5; labels_train[8506] = 9; labels_train[8556] = 3;
labels_train[8559] = 8; labels_train[8615] = 0; labels_train[8625] = 5;
labels_train[8630] = 2; labels_train[8652] = 10; labels_train[8676] = 0;
labels_train[8686] = 3; labels_train[8747] = 10; labels_train[8787] = 5;
labels_train[8792] = 2; labels_train[8796] = 8; labels_train[8806] = 9;
labels_train[8835] = 10; labels_train[8974] = 10; labels_train[8948] = 2;
labels_train[8990] = 4; labels_train[9018] = 10;
```

In [ ]:
```
## Humberto's images to be rotated.
## For np.rot90: k = 1 (default) rotates 90 deg counter clockwise while k = 3 rotates
rotation_index = [6863, 6864, 6885, 6923, 6936, 6939, 6940, 6947, 6977, 6990, 6993,
                  7025, 7027, 7045, 7052, 7056, 7057, 7064, 7092, 7154, 7194, 7279,
                  7308, 7346, 7361, 7364, 7375, 7403, 7476, 7498, 7556, 7640, 7645,
                  7647, 7656, 7687, 7702, 7729, 7762, 7765, 7803, 7902, 7942, 7944,
                  8032, 8082, 8110, 8126, 8130, 8131, 8156, 8182, 8190, 8219, 8220,
                  8223, 8251, 8264, 8286, 8344, 8348, 8455, 8361, 8380, 8404, 8417,
                  8427, 8470, 8475, 8493, 8497, 8548, 8585, 8622, 8633, 8667, 8678,
                  8705, 8739, 8743, 8823, 8839, 8863, 8899, 8900, 8953]

rotation_index3 = [6816, 6830, 6956, 7095, 7570, 7745, 7894, 7896, 8305, 8310, 8506,
                  8647, 8731, 9026]

## Rotate by 90 deg.
for i in range(len(rotation_index)):
    rotate_this = data_train[:,rotation_index[i]]
    rotated_image = np.rot90(rotate_this.reshape(300,300), k=3)
    data_train[:,rotation_index[i]] = rotated_image.reshape(90000)

## Rotate by -90 deg.
for i in range(len(rotation_index3)):
    rotate_this = data_train[:,rotation_index3[i]]
    rotated_image = np.rot90(rotate_this.reshape(300,300), k=1)
    data_train[:,rotation_index3[i]] = rotated_image.reshape(90000)
```

In [ ]:
```
# Displaying some examples per class
for j in np.arange(1500,4515,1):
    plt.imshow(data_train[:,j].reshape((300,300)), cmap='gray')
    label_idx = int(labels_train[j])
    plt.axis('off');plt.title('Label ' + str(label_idx) + " " + labels_names[label_idx
    plt.show()
print('\n\n')
```

Once you fix all mislabeled samples, you can save the updated file:

In [ ]:
```
np.save("labels_train_corrected.npy", labels_train)
np.save("data_train_correct.npy", data_train)
```

==========

In [ ]:
```
data_train_t = np.load('data_train_correct.npy')
labels_train_t = np.load('labels_train_corrected.npy')
```

```
print(data_train_t.shape, labels_train_t.shape)
```

In [ ]:
```
# Counting number samples per class
vals, counts = np.unique(labels_train_t, return_counts=True)

plt.bar(vals, counts)
plt.xticks(range(10),range(10))
plt.xlabel('Classes',size=20)
plt.ylabel('# Samples per Class', size=20)
plt.title('Training Data (Total = '+str(data_train_t.shape[1])+' samples)',size=15);
```

Problem 3 (35 points)

Design and train at least 3 classifiers on the training data (examples include: KNN, SVM, Naive Bayes, FLDA, etc.). **You can work with the data as is, or you may transform the data in any way you see fit** (gray-color, PCA features or any other feature extraction technique, morphological preprocessing, etc.).

You are allowed to use *any* libraries, this includes `scikit-learn`, `tensorflow` or `pytorch`.

In your implementation, you should follow the standard procedures:

1. Preprocess the data, this includes feature scaling.
2. Split data into training and test sets. Make sure you use the same training set in all classifiers (recommendation: fix `random_state` in `train_test_split` function).
3. Perform k-fold cross-validation to determine the best set of parameters for each classifier. Consider using the `GridSearchCV` or `RandomizedSearchCV` approach for experimental design. See an example in Lecture 21.
4. Report performance results in both training and test sets.
5. Include a paragraph that compares the performance of all classifiers, and steps you will take to improve upon these preliminary results.

All classifiers should be compared in terms of their performance, using the following evaluation metrics:

1. Overall accuracy, and
2. Confusion matrices

Note: You may include a paragraph description about your experience in using the HiPerGator. If you encounter any issues, please describe them.

In [4]:
```
import os
import numpy as np
import cv2

#classifiers
```

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.preprocessing import MinMaxScaler

if __name__ == "__main__":

    D = 100

    X = np.load('data_train_correct.npy').T
    t = np.load('labels_train_corrected.npy')
    X = np.array([ cv2.resize(x.reshape(300,300),(D,D)).reshape(D*D) for x in X ])

    X_train, X_test, Y_train, Y_test = train_test_split(X, t, test_size=0.33)

    # Training and Running LDA
    print("----------LDA TRAINING----------")
    lda = LinearDiscriminantAnalysis()
    lda.fit(X_train, Y_train)
    Y_train_pred = lda.predict(X_train)
    Y_test_pred = lda.predict(X_test)

    print("LDA RESULTS")
    print("TRAINING RESULTS")

    print(classification_report(Y_train,Y_train_pred))

    print("TESTING RESULTS")
    print(classification_report(Y_test,Y_test_pred))

    print("CONFUSION MATRIX FOR LDA")
    print("Train_matrix")
    print(confusion_matrix(Y_train, Y_train_pred))
    print("Test_matrix")
    print(confusion_matrix(Y_test, Y_test_pred))
```

```
----------LDA TRAINING----------
LDA RESULTS
TRAINING RESULTS
              precision    recall  f1-score   support

         0.0       0.99      0.99      0.99       576
         1.0       0.99      0.98      0.99       597
         2.0       0.97      0.98      0.97       613
         3.0       0.99      0.97      0.98       625
         4.0       0.99      0.98      0.99       612
         5.0       0.98      0.99      0.98       601
         6.0       0.97      0.98      0.97       625
         7.0       0.98      0.99      0.98       594
         8.0       0.98      0.99      0.98       616
         9.0       0.98      0.99      0.98       592

    accuracy                           0.98      6051
   macro avg       0.98      0.98      0.98      6051
weighted avg       0.98      0.98      0.98      6051

TESTING RESULTS
              precision    recall  f1-score   support

         0.0       0.14      0.14      0.14       327
         1.0       0.20      0.17      0.18       306
         2.0       0.29      0.19      0.23       290
         3.0       0.17      0.27      0.21       279
         4.0       0.12      0.23      0.16       291
         5.0       0.13      0.08      0.10       302
         6.0       0.16      0.10      0.12       279
         7.0       0.18      0.13      0.15       309
         8.0       0.11      0.10      0.10       287
         9.0       0.18      0.20      0.19       311

    accuracy                           0.16      2981
   macro avg       0.17      0.16      0.16      2981
weighted avg       0.17      0.16      0.16      2981

CONFUSION MATRIX FOR LDA
Train_matrix
[[568   0   2   0   0   2   0   1   1   2]
 [  0 587   0   0   1   2   4   2   1   0]
 [  0   1 599   4   0   0   5   2   2   0]
 [  3   0   5 608   2   1   2   0   3   1]
 [  2   0   3   2 597   0   3   1   1   3]
 [  1   0   1   0   0 592   2   1   2   2]
 [  0   2   2   2   0   1 615   1   1   1]
 [  0   0   1   1   0   0   3 586   1   2]
 [  2   0   1   0   0   2   1   2 607   1]
 [  0   0   2   0   0   3   2   0   1 584]]
Test_matrix
[[47 19 20 43 78 25 21 20 28 26]
 [41 53 13 47 44 16 12 20 26 34]
 [22 24 55 43 46 13 18 15 28 26]
 [20 18 20 75 40 15 17 16 26 32]
 [29 24 10 46 68 14 16 25 31 28]
 [39 26 11 42 52 24 16 25 30 37]
```

```
[29 30 14 34 61 18 27 18 23 25]
[31 21 19 40 61 18 17 41 27 34]
[38 27 16 35 54 22 13 22 29 31]
[45 27 12 28 62 25 12 20 19 61]]
```

In this project we used three models for classification of the hand written hand symbols.

SVM :

Training accuracy: 0.999841822208162

Test accuracy: 0.46494464944649444

KNN:

Test accuracy:0.46

k value: 1 Accuracy: 0.4554122265844083

k value: 2 Accuracy: 0.4178351093662367

k value: 3 Accuracy: 0.42344363432417276

k value: 4 Accuracy: 0.4195176668536175

k value: 5 Accuracy: 0.41615255187885586

k value: 6 Accuracy: 0.4094223219293326

k value: 7 Accuracy: 0.4038137969713965

k value: 8 Accuracy: 0.3841839596186203

k value: 9 Accuracy: 0.3869882220975883

LDA:

Training accuracy:0.98

Test accuracy:0.16

The Confudion matrix is given in each coding section which has been uploaded in the files names KNN.ipynb and SVM_classifier_HW3_part2.ipynb.

Analysing the train accuracy and test accuracy of the three classification algorithms we came to a conclusion that SVM gives the better classification result both in train and test accruracy. While the train accuracy for LDA is very high but the test accuracy is very low.As for the KNN its very close to SVM in classification.

Although these classification algorithms are good but rather then using this algorithm for the classification task we will be using CNN for this task. We aleady had the the codes for VGG-16 CNN but were not able to train it yet because while training on the Jupyter notebook it keeps

throughing error of memory sortage from the GPU. We will try to solve this problem on HyperGator. Thus we will come to a conclusion after training the CNN model and getting its accuracy.

As for the Algorithms now we will try to tune the hyperparameters such that it can produce better result for the test data. Overfitting may be a cause for the vast difference between test and train accuracy. We will try to reduce overfitting using regularization,cross-validation,etc.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Submit your Solution

Confirm that you've successfully completed the assignment.

Along with the Notebook, include a PDF of the notebook with your solutions.

`add` and `commit` the final version of your work, and `push` your code to your GitHub repository.

Submit the URL of your GitHub Repository as your assignment submission on Canvas.
