Problem 3 (24 Points)

Problem description

So far, we have worked with ~2 dimensional problems with 2-3 classes. Most often in ML, there are many more explanatory variables and classes than this. In this problem, you'll be training logistic regression models on a database of grayscale images of hand-drawn digits, using SciKit-Learn. Now there are 400 (20x20) input features and 10 classes (digits 0-9).

As usual, you can use any code from previous problems.

Summary of deliverables

- OvR model accuracy on training data
- OvR model accuracy on testing data
- Multinomial model accuracy on training data
- Multinomial model accuracy on testing data

Imports and Utility Functions:

In [12]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression

def visualize(xdata, index, title=""):
    image = xdata[index,:].reshape(20,20).T
    plt.figure()
    plt.imshow(image,cmap = "binary")
    plt.axis("off")
    plt.title(title)
    plt.show()
```

Load data

The following cell loads in training and testing data into the following variables:

- `x_train` : 4000x400 array of input features, used for training
- `y_train` : Array of ground-truth classes for each point in `x_train`
- `x_test` : 1000x400 array of input features, used for testing
- `y_test` : Array of ground-truth classes for each point in `x_test`

You can visualize a digit with the `visualize(x_data, index)` function.

In [13]:
```
x_train = np.load("data/w3-hw3-train_x.npy")
y_train = np.load("data/w3-hw3-train_y.npy")
x_test = np.load("data/w3-hw3-test_x.npy")
y_test = np.load("data/w3-hw3-test_y.npy")

visualize(x_train,1234)
```

Logistic Regression Models

Use sklearn's `LogisticRegression` to fit a multinomial logistic regression model on the training data. You may need to increase the `max_iter` argument for the model to converge.

Train 2 models: one using the One-vs-Rest method, and another that minimizes multinomial loss. You can do these by setting the `multi_class` argument to "ovr" and "multinomial", respectively.

More information: [https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html (https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)

In [20]:

```
# YOUR CODE GOES HERE (sklearn models)

OVR_model = LogisticRegression(multi_class="ovr",max_iter=10000)
OVR_model.fit(x_train,y_train)
OVR_train =  OVR_model.predict(x_train)
OVR_test = OVR_model.predict(x_test)

MULTI_model = LogisticRegression(multi_class="multinomial",max_iter=10000)
MULTI_model.fit(x_train,y_train)
MULTI_train =  OVR_model.predict(x_train)
MULTI_test = OVR_model.predict(x_test)
```

```
C:\Users\zsqu4\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2
kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\linear_model\_l
ogistic.py:1256: FutureWarning: 'multi_class' was deprecated in version 1.5 and w
ill be removed in 1.7. Use OneVsRestClassifier(LogisticRegression(..)) instead. L
eave it to its default value to avoid this warning.
  warnings.warn(
C:\Users\zsqu4\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.11_qbz5n2
kfra8p0\LocalCache\local-packages\Python311\site-packages\sklearn\linear_model\_l
ogistic.py:1247: FutureWarning: 'multi_class' was deprecated in version 1.5 and w
ill be removed in 1.7. From then on, it will always use 'multinomial'. Leave it t
o its default value to avoid this warning.
  warnings.warn(
```

Accuracy

Compute and print the accuracy of each model on the training and testing sets as a percent.

In [21]:

```
# YOUR CODE GOES HERE (print the 4 requested accuracy values)
OVR_train_accuracy = np.sum(OVR_train == y_train)/len(y_train)*100
OVR_test_accuracy = np.sum(OVR_test == y_test)/len(y_test)*100
MULTI_train_accuracy = np.sum(MULTI_train == y_train)/len(y_train)*100
MULTI_test_accuracy = np.sum(MULTI_test == y_test)/len(y_test)*100


print("OVR Train Accuracy:",OVR_train_accuracy, r"%")
print("OVR Test Accuracy:", OVR_test_accuracy, r"%")
print("Multinomial Train Accuracy:",MULTI_train_accuracy, r"%")
print("Multinomial Test Accuracy:", MULTI_test_accuracy, r"%")
```

```
OVR Train Accuracy: 94.675 %
OVR Test Accuracy: 90.8 %
Multinomial Train Accuracy: 94.675 %
Multinomial Test Accuracy: 90.8 %
```

In [ ]:

```

```