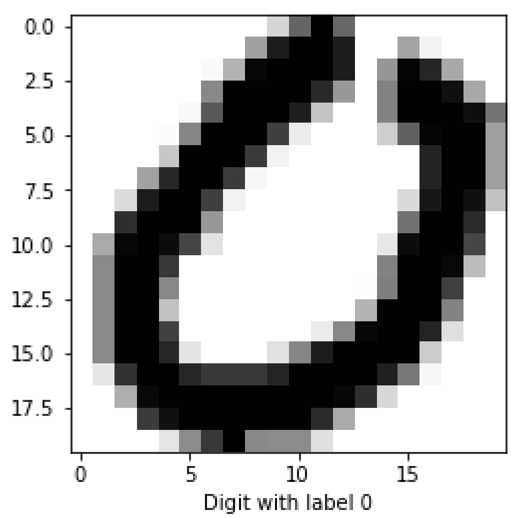
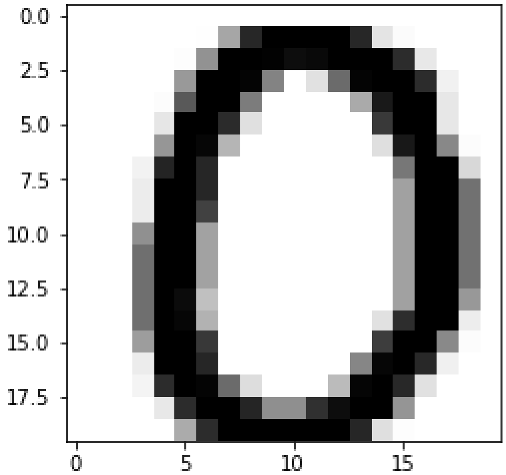
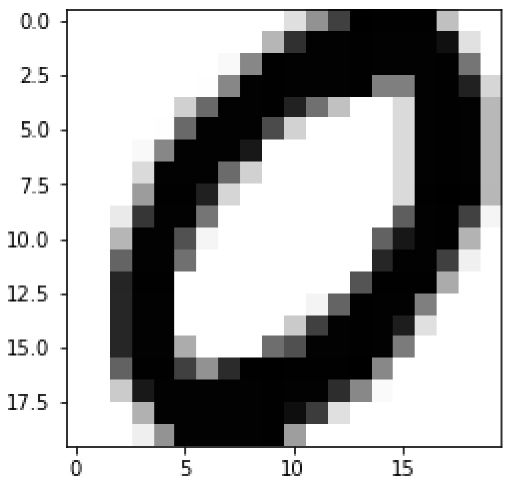
In [3]:
```
import numpy as np
import matplotlib.pyplot as plt

# Load the training dataset
train_features = np.load("train_features.npy")
train_labels = np.load("train_labels.npy").astype("int8")

n_train = train_labels.shape[0]

def visualize_digit(features, label):
    # Digits are stored as a vector of 400 pixel values. Here we
    # reshape it to a 20x20 image so we can display it.
    plt.imshow(features.reshape(20, 20), cmap="binary")
    plt.xlabel("Digit with label " + str(label))
    plt.show()

# Visualize a digit
# visualize_digit(train_features[0,:], train_labels[0])

# TODO: Plot three images with label 0 and three images with label 1

def images(a):
    counter = 0
    for i in range(20):
        if train_labels[i] == a:
            counter += 1
            visualize_digit(train_features[i,:], train_labels[i])
        if  counter==3:
            break
images(0)
images(1)
```

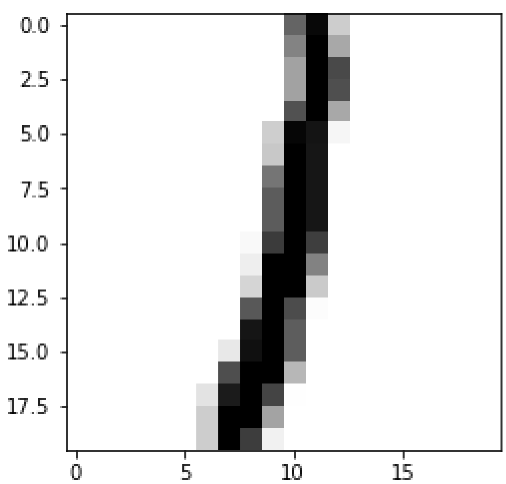




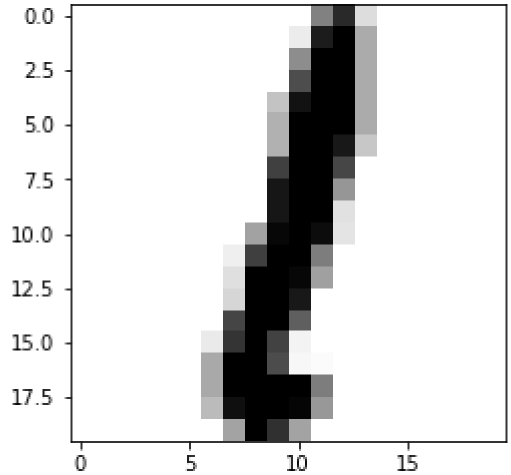
Digit with label 0



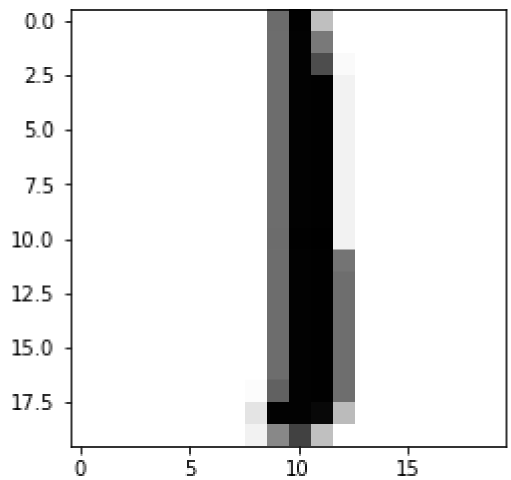
Digit with label 0



Digit with label 1



Digit with label 1



Digit with label 1

In [4]:
```
# Linear regression
from numpy.linalg import inv

# TODO: Solve the linear regression problem, regressing
# X = train_features against y = 2 * train_labels - 1

X = train_features
y = 2 * train_labels - 1

def get_w(X, y):
    XT = X.transpose()
    w = np.dot(inv(XT @ X) @ XT, y)
    return(w)

w = get_w(X, y)
res = np.dot(X, w) - y
res_2 = np.dot(res.transpose(), res)
# TODO: Report the residual error and the weight vector

print('the residual error is {} \n'.format(res_2))
print('w is a 400*1 vector: w = ')
print(w)
```

```
the residual error is 422.75079345703125

w is a 400*1 vector: w =
[ -3.30796123e-01   3.91724765e-01   1.48155391e-01  -1.60603136e-01
   1.03279680e-01  -1.96949169e-02  -1.27705634e-01   9.45962034e-03
  -1.71494633e-02  -5.67550585e-03  -4.69030812e-03  -1.12644807e-02
  -5.71013801e-03   4.59017232e-03   1.78760011e-02  -3.00972313e-02
   1.00369528e-02  -6.45812601e-02  -2.30415799e-02  -2.63008233e-02
  -1.63458556e-01   4.67080057e-01  -2.82919407e-03  -1.05642147e-01
  -1.80704594e-01   1.34412199e-01  -5.09417057e-03  -3.07268575e-02
  -6.59221411e-02   1.76010150e-02  -3.06512788e-02  -6.22963160e-03
   1.54067874e-02  -3.13660577e-02  -2.52744369e-03  -6.17406890e-03
  -1.02588534e-03   5.19412048e-02   3.95637155e-02   6.29596263e-02
  -3.88406903e-01  -2.16920286e-01  -9.80447531e-02   1.68329507e-01
  -5.70877790e-02   1.50778815e-02   2.43577361e-03   2.72967089e-02
   6.30412847e-02  -2.69233901e-02   7.73133337e-03  -1.71149727e-02
  -5.02606295e-02   8.06276128e-03  -6.38876110e-03  -1.26746073e-02
   1.55037120e-02  -1.06071420e-02  -7.69636221e-03  -4.18695137e-02
   5.26521444e-01   2.31917337e-01  -8.41784179e-02   9.90925729e-02
  -3.96229923e-02  -2.31645964e-02   6.78411871e-03  -5.28082922e-02
   2.26982012e-02  -1.84088089e-02   2.88220122e-03  -1.06377620e-02
   2.26956047e-02  -3.03869843e-02  -1.73668489e-02   2.12011971e-02
   3.33772749e-02  -3.48210931e-02  -2.71486416e-02  -6.14579022e-02
  -2.08384514e-01  -7.03627765e-02  -6.45077527e-02  -5.42105623e-02
   2.14532912e-02  -2.83413231e-02  -2.76354402e-02   1.08109191e-02
  -3.84216346e-02   2.86547896e-02  -2.90161967e-02   5.93165029e-03
  -7.47637823e-03  -9.09803435e-03  -1.19278338e-02  -1.55726075e-02
  -2.03939676e-02  -6.55515492e-03   4.85598072e-02  -3.95782813e-02
  -3.30486894e-02   1.84932053e-02   8.31140429e-02   9.48820449e-03
   1.04629993e-02   2.05513388e-02  -2.89785564e-02   1.20915063e-02
   1.13603324e-02  -1.36096030e-02   1.48535892e-02   4.62725386e-03
   3.68354190e-03   7.35396333e-03   2.30066702e-02  -1.49417445e-02
  -5.77552021e-02   2.55905166e-02  -1.23858154e-02  -3.55641991e-02
   5.60277700e-02  -7.86052197e-02  -5.50595187e-02  -3.80747020e-03
```

```
  2.05794312e-02  -8.04882869e-03   7.84287229e-04   5.85546717e-03
 -3.78696434e-02   2.48432159e-04  -4.85737249e-03   7.12568872e-03
 -6.08517416e-03  -1.06557533e-02  -4.51316051e-02   7.35598058e-03
 -5.80839813e-04  -3.72899435e-02   2.24066377e-02  -3.55857015e-02
  1.25210509e-01   3.23692858e-02   2.73668617e-02  -3.78578529e-02
 -2.84900293e-02  -2.55225599e-03  -3.19266394e-02   3.13730165e-02
  4.19985503e-02   7.49354810e-03   2.51227245e-02   6.46016747e-03
  6.20057620e-03  -2.07473338e-03   5.90182841e-03  -8.57779384e-03
  2.90097855e-03   2.03040242e-02  -8.59597176e-02  -8.97464901e-03
 -5.60657494e-02   5.42648807e-02  -1.96553599e-02   5.73925935e-02
 -2.16799136e-02   7.91462883e-03  -5.28785214e-03  -7.76379257e-02
 -5.07036373e-02  -2.74826195e-02   6.07776642e-02   4.88356575e-02
 -2.40392610e-02  -4.98383641e-02  -4.93458658e-03  -4.80975434e-02
 -4.14481685e-02  -6.00590520e-02  -1.63088292e-02  -5.77998385e-02
 -1.04835004e-01  -5.49009591e-02   1.14539508e-02  -4.62886021e-02
  9.94826853e-03  -2.29425728e-03  -5.04532829e-02   6.33369684e-02
  8.83274525e-03   1.28797181e-02   5.98827526e-02   5.08481637e-02
  5.55638745e-02   3.24129872e-02   1.85606256e-03  -6.51602447e-03
  2.07820237e-02   4.16170210e-02   5.11917062e-02   8.70737806e-03
  3.71086597e-03  -3.43105271e-02  -3.77525166e-02   2.70951204e-02
 -5.89128584e-03  -1.95766538e-02   9.65390354e-03  -7.81856477e-02
  2.13212445e-02  -1.35685146e-01   4.39425915e-01  -8.18718001e-02
  5.43916002e-02  -3.66787612e-02  -5.72940633e-02   4.75731492e-02
 -8.32982268e-03  -7.52136633e-02  -2.67496258e-02  -4.92984504e-02
  3.49877179e-02   5.02439737e-02   6.05673790e-02  -3.81416082e-02
  2.50256062e-03  -1.11072585e-02  -6.17939904e-02  -2.07912438e-02
  7.77739659e-03   9.29733291e-02   3.28987777e-01  -5.16273789e-02
 -1.47197694e-02  -8.32061023e-02  -7.43252039e-02  -7.29229301e-02
 -4.12035249e-02   3.09913158e-02   1.29102431e-02  -2.84672678e-02
 -3.92790325e-03  -1.00521073e-01  -3.35057825e-02   1.55609846e-03
 -7.21260905e-04  -2.80618668e-02   5.95020130e-03   3.56642902e-02
  1.08798593e-02   1.07424021e-01   4.42860126e-02  -1.00261532e-03
  1.39009580e-03  -8.00528973e-02  -3.36570404e-02  -7.74169713e-03
 -8.94960016e-03  -2.00045556e-02   2.30861567e-02  -7.37616941e-02
  8.12910497e-03   4.37760875e-02  -3.59041095e-02  -3.68333161e-02
 -6.17185645e-02  -3.72650661e-02  -5.80597967e-02  -1.79026444e-02
  5.28743528e-02   7.61481002e-04   4.68757376e-02   3.64771038e-02
  3.71705405e-02   3.71804237e-02   3.30805629e-02  -3.52992415e-02
 -8.89341757e-02  -2.42443606e-02  -7.89628774e-02   5.99945411e-02
 -8.14004987e-02  -3.61125022e-02  -1.25761181e-02  -2.42130980e-02
  5.14357537e-03   1.78405661e-02   2.22027320e-02   2.82184780e-03
  3.35302949e-02  -4.93126549e-02   2.01935694e-02   3.94619480e-02
 -2.18398906e-02  -9.06860158e-02  -4.52998281e-03  -2.45162938e-02
  1.84685104e-02  -1.53115913e-02  -1.20573044e-02  -1.18375078e-01
  2.88001671e-02  -1.90547481e-02  -5.65437078e-02   1.42882764e-02
 -2.23247074e-02  -8.45870189e-03  -2.25760341e-02   2.53214911e-02
 -3.29714268e-02   2.33236756e-02   1.05693061e-02   1.18442606e-02
  3.81864794e-02   2.81872228e-02  -2.45874375e-03   5.40287420e-03
 -4.95716929e-04   1.93904787e-02   5.34845442e-02  -5.50350510e-02
 -1.27018005e-01  -1.13497362e-01   7.43942559e-02  -6.41146451e-02
 -2.47328319e-02   1.66992620e-02   2.09305733e-02   1.21586993e-02
  4.45517376e-02  -1.40924025e-02   1.58408955e-02   2.01890692e-02
 -1.64979994e-02   1.85777508e-02   2.12734863e-02   7.97321647e-03
 -3.30512077e-02  -1.31743565e-01  -5.20438924e-02  -6.18480146e-03
  4.09973860e-02   6.37193769e-02   1.58209503e-02   8.63967761e-02
  4.42898162e-02  -2.31197551e-02  -8.82010534e-03  -3.39905694e-02
  1.49171129e-02   4.80811819e-02   1.25215128e-02  -2.58592516e-03
```

```
 3.72425020e-02  -2.24012323e-03   5.62974922e-02   9.65552405e-03
 1.01853073e-01   8.83490518e-02  -4.26429659e-02   2.76069760e-01
 5.16162813e-02   5.58910817e-02   1.50369890e-02  -7.86101259e-03
-1.79310963e-02   5.78273460e-03  -1.39468256e-02  -2.94634756e-02
-7.18816891e-02  -6.41089454e-02   2.71457061e-03  -2.76246220e-02
-3.69601846e-02   7.89900869e-03   7.59387240e-02  -8.44904035e-03
 1.20694041e-02   1.71198979e-01  -3.02480102e-01   2.80249119e-03
-2.34687328e-03  -3.16523015e-02   3.73742916e-03  -3.21977921e-02
 1.20266862e-02  -4.40902263e-03   1.04535855e-02   3.74992751e-03
 1.62230078e-02  -3.36626545e-03  -4.06874008e-02  -1.61136314e-02
-6.13355637e-03  -3.09129804e-02  -5.01953661e-02   1.58628821e-02
-1.27862543e-01   1.68132693e-01  -2.77890831e-01   4.19037044e-02]
```

In [5]:

```python
# Load the test dataset
# It is good practice to do this after the training has been
# completed to make sure that no training happens on the test
# set!

from numpy import logical_xor, logical_not
test_features = np.load("test_features.npy")
test_labels = np.load("test_labels.npy").astype("int8")

n_test = test_labels.shape[0]

# TODO: Implement the classification rule and evaluate it
# on the training and test set

def correct_rate(features, labels, n, w, rule):
    predict = (np.dot(features, w) > rule)
    correct = logical_not(logical_xor(predict, labels))
    return float(sum(correct))/n


print('the correct pertentage by this model in the training \n \
        set is {} \n'.format(correct_rate(train_features, train_labels, n_tr

print('the correct pertentage by this model in the test \n \
        set is {} \n'.format(correct_rate(test_features, test_labels, n_test
```

```
the correct pertentage by this model in the training
         set is 0.9975909833949926

the correct pertentage by this model in the test
         set is 0.9981087470449173
```

In [6]:
```
# TODO: Try regressing against a vector with 0 for class 0
# and 1 for class 1

w2 = get_w(X, train_labels)
print('0 for 0, 1 for 1 model: the correct pertentage by this model in the t
        set is {} \n'.format(correct_rate(train_features, train_labels, n_tr

print('0 for 0, 1 for 1 model: the correct pertentage by this model in the t
        set is {} \n'.format(correct_rate(test_features, test_labels, n_test
```

```
0 for 0, 1 for 1 model: the correct pertentage by this model in the train
ing
         set is 0.9896756431213972

0 for 0, 1 for 1 model: the correct pertentage by this model in the test
         set is 0.9914893617021276
```

In [7]:
```
# TODO: Form a new feature matrix with a column of ones added
# and do both regressions with that matrix

train_features_1 = np.hstack((train_features, np.ones((n_train, 1))))
test_features_1 = np.hstack((test_features, np.ones((n_test, 1))))

w3 = get_w(train_features_1, 2*train_labels-1)
print('train with bias column: the correct pertentage by this model in the t
        set is {} \n'.format(correct_rate(train_features_1, train_labels, n_

print('train with bias column: the correct pertentage by this model in the t
        set is {} \n'.format(correct_rate(test_features_1, test_labels, n_te


w4 = get_w(train_features_1, train_labels)
print('train with bias column & 0 for 0 1 for 1: the correct pertentage by t
        set is {} \n'.format(correct_rate(train_features_1, train_labels, n_

print('train with bias column & 0 for 0 1 for 1: the correct pertentage by t
        set is {} \n'.format(correct_rate(test_features_1, test_labels, n_te
```

```
train with bias column: the correct pertentage by this model in the train
ing
         set is 0.9941495311021251

train with bias column: the correct pertentage by this model in the test
         set is 0.9962174940898345

train with bias column & 0 for 0 1 for 1: the correct pertentage by this
model in the training
         set is 0.9941495311021251

train with bias column & 0 for 0 1 for 1: the correct pertentage by this
model in the test
         set is 0.9962174940898345
```

In [8]:
```
# Logistic Regression

# You can also compare against how well logistic regression is doing.
# We will learn more about logistic regression later in the course.

import sklearn.linear_model

lr = sklearn.linear_model.LogisticRegression()
lr.fit(X, train_labels)

test_error_lr = 1.0 * sum(lr.predict(test_features) != test_labels) / n_test
print(1-test_error_lr)
```

```
0.999527186761
```

In [ ]: