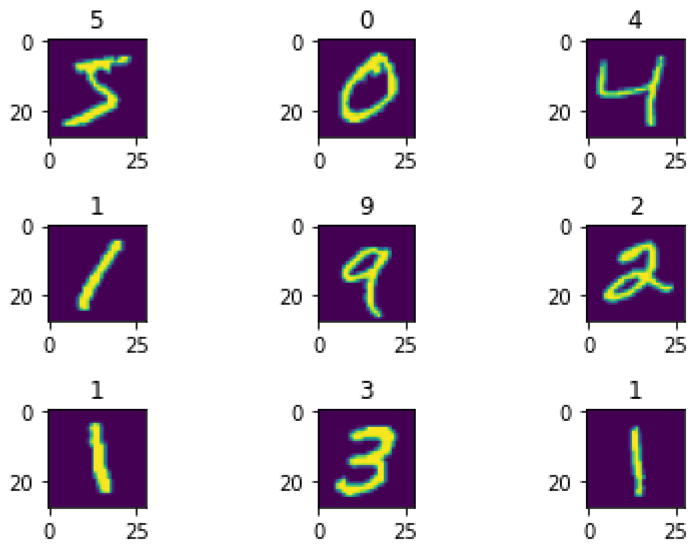
In [1]:
```
#common packages we basically always need
import numpy as np
import matplotlib.pyplot as plt
import time
```

In [2]:
```
#load the MNIST dataset with binary pixel values
import scipy.io as sio
data = sio.loadmat('mnist.mat')
print(data)



Xtrain, Xtest = data['trainX'].astype(float), data['testX'].astype(float)
ytrain, ytest = data['trainY'][0], data['testY'][0]



#pull and plot some samples
for k in range(9):
    plot_data = Xtrain[k,:]
    plot_data = np.reshape(plot_data,(28,28))
    plot_label = ytrain[k]
    plt.subplot(3,3,k+1)
    plt.imshow(plot_data)
    plt.title(plot_label)

plt.tight_layout()

def get_small_dataset(X,y,m):
    return X[:m,:],y[:m]


num_labels = len(np.unique(ytrain))
num_feats = Xtrain.shape[1]
```

```
{'__header__': b'MATLAB 5.0 MAT-file Platform: posix, Created on: Wed Oct 18 19:00:09 20
17', '__version__': '1.0', '__globals__': [], 'testX': array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=uint8), 'testY': array([[7, 2, 1, ..., 4, 5, 6]],
dtype=uint8), 'trainY': array([[5, 0, 4, ..., 5, 6, 8]], dtype=uint8), 'trainX': array
([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=uint8)}
```



In [3]:
```
def get_dist(Xtrain,zquery):
    ## for i in range(len(Xtrain)):

    result = []
    for k in range(len(Xtrain)):
        Xtrain2 = Xtrain[k]
        x = abs(Xtrain2 - zquery)
        val = np.linalg.norm(x, ord=2)**2
        result.append(val)

    return result

print(get_dist(Xtrain,Xtrain[0,:])[0])
print(get_dist(Xtrain,Xtest[0,:])[10])
print(get_dist(Xtrain,Xtest[10,:])[50])
```

```
0.0
6069461.999999999
5661744.000000001
```

In [4]:
```
import scipy.stats as ss

m = 100
K = 3

class Element:
  def __init__(self, distance, value):
    self.distance = distance
    self.value = value

Xtrain_small, ytrain_small = get_small_dataset(Xtrain,ytrain,m)

def pred(zquery,Xtrain,ytrain, K):
    array = get_dist(Xtrain, zquery)

    mainArray = []
    for i in range(len(array)):
        element = Element(array[i], ytrain[i])
        mainArray.append(element)

    mainArray.sort(key=lambda x: x.distance, reverse=False)

    count = [0,0,0,0,0,0,0,0,0,0]
    result = 0

    index = -1
    for i in range(K):
        element = mainArray[i]
        if (result < count[element.value] + 1):
            index = element.value
            result = count[element.value] + 1
        count[element.value] = count[element.value] + 1
        ## print(count, index)



    return index

ytest_pred = ytest + 0
for k in range(Xtest.shape[0]):
    z = Xtest[k,:]
    ytest_pred[k] = pred(z,Xtrain_small, ytrain_small, K)


print(ytest_pred[:20])
print(ytest[:20])
```

```
[7 2 1 0 4 1 9 4 6 9 0 9 9 0 1 9 7 7 3 4]
[7 2 1 0 4 1 4 9 5 9 0 6 9 0 1 5 9 7 3 4]
```

In [7]:
```
def get_accuracy(ytest, ypred):
    total = len(ytest)
    correct = 0
    for i in range(len(ytest)):
        if ytest[i] == ypred[i]:
            correct = correct + 1
    return correct / total

get_accuracy(ytest,ytest_pred)
```

Out[7]:
```
0.6794
```

In [9]:
```
import time
for m in [100,1000, 2500]:
    Xtrain_small, ytrain_small = get_small_dataset(Xtrain,ytrain,m)
    for K in [1,3,5]:

        start = time.time()
        ytest_pred = ytest + 0
        for k in range(Xtest.shape[0]):
            z = Xtest[k,:]
            ytest_pred[k] = pred(z,Xtrain_small, ytrain_small, K)

        print(m,K,get_accuracy(ytest,ytest_pred), time.time()-start)
```

```
100 1 0.6794 10.875805854797363
100 3 0.6694 10.729932069778442
100 5 0.6426 10.848812103271484
1000 1 0.869 111.03866696357727
1000 3 0.872 111.45527601242065
1000 5 0.8635 109.75263381004333
2500 1 0.9136 298.9400300979614
2500 3 0.9187 324.6318910121918
2500 5 0.9157 311.4742362499237
```

In [ ]:

In [ ]:

In [ ]:

```
In [ ]:
```