Practice Lab: Neural Networks for Handwritten Digit Recognition, Multiclass

In this exercise, we use a neural network to recognize the hand-written digits 0-9.

In [1]:
```
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.activations import linear, relu, sigmoid
import matplotlib.pyplot as plt
```

softmax function

In [3]:
```
# implement softmax function

def my_softmax(z):
    """ Softmax converts a vector of values to a probability distribution.
    Args:
      z (ndarray (N,))  : input data, N features
    Returns:
      a (ndarray (N,))  : softmax of z
    """
    z_sum = sum(np.exp(z))
    a = np.exp(z)/z_sum

    return a
```

Neural Networks

In this exercise, we use a neural network to recognize ten handwritten digits, 0-9.

In [7]:
```
# load dataset
X, y = load_data(...)
```

```
(5000, 400)
```

In [8]:
```
print ('The first element of X is: ', X[0])
```

```
The first element of X is:  [ 0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e
+00  0.00e+00
  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00
  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00
  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00
  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00
  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00
  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00
  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00
  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00
  0.00e+00  0.00e+00  0.00e+00  0.00e+00  8.56e-06  1.94e-06 -7.37e-04
 -8.13e-03 -1.86e-02 -1.87e-02 -1.88e-02 -1.91e-02 -1.64e-02 -3.78e-03
  3.30e-04  1.28e-05  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00
  0.00e+00  0.00e+00  1.16e-04  1.20e-04 -1.40e-02 -2.85e-02  8.04e-02
  2.67e-01  2.74e-01  2.79e-01  2.74e-01  2.25e-01  2.78e-02 -7.06e-03
  2.35e-04  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00
  1.28e-17 -3.26e-04 -1.39e-02  8.16e-02  3.83e-01  8.58e-01  1.00e+00
  9.70e-01  9.31e-01  1.00e+00  9.64e-01  4.49e-01 -5.60e-03 -3.78e-03
  0.00e+00  0.00e+00  0.00e+00  0.00e+00  5.11e-06  4.36e-04 -3.96e-03
 -2.69e-02  1.01e-01  6.42e-01  1.03e+00  8.51e-01  5.43e-01  3.43e-01
  2.69e-01  6.68e-01  1.01e+00  9.04e-01  1.04e-01 -1.66e-02  0.00e+00
  0.00e+00  0.00e+00  0.00e+00  2.60e-05 -3.11e-03  7.52e-03  1.78e-01
  7.93e-01  9.66e-01  4.63e-01  6.92e-02 -3.64e-03 -4.12e-02 -5.02e-02
  1.56e-01  9.02e-01  1.05e+00  1.51e-01 -2.16e-02  0.00e+00  0.00e+00
  0.00e+00  5.87e-05 -6.41e-04 -3.23e-02  2.78e-01  9.37e-01  1.04e+00
  5.98e-01 -3.59e-03 -2.17e-02 -4.81e-03  6.17e-05 -1.24e-02  1.55e-01
  9.15e-01  9.20e-01  1.09e-01 -1.71e-02  0.00e+00  0.00e+00  1.56e-04
 -4.28e-04 -2.51e-02  1.31e-01  7.82e-01  1.03e+00  7.57e-01  2.85e-01
  4.87e-03 -3.19e-03  0.00e+00  8.36e-04 -3.71e-02  4.53e-01  1.03e+00
  5.39e-01 -2.44e-03 -4.80e-03  0.00e+00  0.00e+00 -7.04e-04 -1.27e-02
  1.62e-01  7.80e-01  1.04e+00  8.04e-01  1.61e-01 -1.38e-02  2.15e-03
 -2.13e-04  2.04e-04 -6.86e-03  4.32e-04  7.21e-01  8.48e-01  1.51e-01
 -2.28e-02  1.99e-04  0.00e+00  0.00e+00 -9.40e-03  3.75e-02  6.94e-01
  1.03e+00  1.02e+00  8.80e-01  3.92e-01 -1.74e-02 -1.20e-04  5.55e-05
 -2.24e-03 -2.76e-02  3.69e-01  9.36e-01  4.59e-01 -4.25e-02  1.17e-03
  1.89e-05  0.00e+00  0.00e+00 -1.94e-02  1.30e-01  9.80e-01  9.42e-01
  7.75e-01  8.74e-01  2.13e-01 -1.72e-02  0.00e+00  1.10e-03 -2.62e-02
  1.23e-01  8.31e-01  7.27e-01  5.24e-02 -6.19e-03  0.00e+00  0.00e+00
  0.00e+00  0.00e+00 -9.37e-03  3.68e-02  6.99e-01  1.00e+00  6.06e-01
  3.27e-01 -3.22e-02 -4.83e-02 -4.34e-02 -5.75e-02  9.56e-02  7.27e-01
  6.95e-01  1.47e-01 -1.20e-02 -3.03e-04  0.00e+00  0.00e+00  0.00e+00
  0.00e+00 -6.77e-04 -6.51e-03  1.17e-01  4.22e-01  9.93e-01  8.82e-01
  7.46e-01  7.24e-01  7.23e-01  7.20e-01  8.45e-01  8.32e-01  6.89e-02
 -2.78e-02  3.59e-04  7.15e-05  0.00e+00  0.00e+00  0.00e+00  0.00e+00
  1.53e-04  3.17e-04 -2.29e-02 -4.14e-03  3.87e-01  5.05e-01  7.75e-01
  9.90e-01  1.01e+00  1.01e+00  7.38e-01  2.15e-01 -2.70e-02  1.33e-03
  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00
  0.00e+00  2.36e-04 -2.26e-03 -2.52e-02 -3.74e-02  6.62e-02  2.91e-01
  3.23e-01  3.06e-01  8.76e-02 -2.51e-02  2.37e-04  0.00e+00  0.00e+00
  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00
  0.00e+00  0.00e+00  6.21e-18  6.73e-04 -1.13e-02 -3.55e-02 -3.88e-02
 -3.71e-02 -1.34e-02  9.91e-04  4.89e-05  0.00e+00  0.00e+00  0.00e+00
  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00
  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00
  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00
  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00
  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00
  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00  0.00e+00
  0.00e+00]
```

In [9]:
```
print ('The first element of y is: ', y[0,0])
print ('The last element of y is: ', y[-1,0])
```

```
The first element of y is:  0
The last element of y is:  9
```

In [10]:
```
print ('The shape of X is: ' + str(X.shape))
print ('The shape of y is: ' + str(y.shape))
```

```
The shape of X is: (5000, 400)
The shape of y is: (5000, 1)
```

Tensorflow Model Implementation

In [13]:
```
# using Keras Sequential model and dense layer with ReLU activation to construct three
tf.random.set_seed(1234) # for consistent results
model = Sequential(
    [
        ### START CODE HERE ###
        tf.keras.Input(shape=(400,)),
        Dense(25,activation ='relu',name ="L1"),
        Dense(15,activation ='relu',name ="L2"),
        Dense(10,activation ='linear',name ="L3")
        ### END CODE HERE ###
    ], name = "my_model"
)
```

In [14]:
```
model.summary()
```

```
Model: "my_model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 L1 (Dense)                  (None, 25)                10025

 L2 (Dense)                  (None, 15)                390

 L3 (Dense)                  (None, 10)                160

=================================================================
Total params: 10,575
Trainable params: 10,575
Non-trainable params: 0
_________________________________________________________________
```

In [16]:
```
[layer1, layer2, layer3] = model.layers
```

In [17]:
```
#### Examine Weights shapes
W1,b1 = layer1.get_weights()
W2,b2 = layer2.get_weights()
W3,b3 = layer3.get_weights()
print(f"W1 shape = {W1.shape}, b1 shape = {b1.shape}")
print(f"W2 shape = {W2.shape}, b2 shape = {b2.shape}")
print(f"W3 shape = {W3.shape}, b3 shape = {b3.shape}")
```

```
W1 shape = (400, 25), b1 shape = (25,)
W2 shape = (25, 15), b2 shape = (15,)
W3 shape = (15, 10), b3 shape = (10,)
```

In [ ]:
```
## model fitting
model.compile(
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
)

history = model.fit(
    X,y,
    epochs=100
)
```

In [21]:
```
# Prediction
image_of_two = X[1015]
display_digit(image_of_two)

prediction = model.predict(image_of_two.reshape(1,400))  # prediction

print(f" predicting a Two: \n{prediction}")
print(f" Largest Prediction index: {np.argmax(prediction)}") ## predicted result is di
```

```
Canvas(toolbar=Toolbar(toolitems=[('Home', 'Reset original view', 'home', 'home'),
('Back', 'Back to previous …
 predicting a Two:
[[-16.35  -4.51   3.47  -5.38 -29.91 -17.83 -22.78 -11.9  -14.67 -18.07]]
 Largest Prediction index: 2
```

In [22]:
```
prediction_p = tf.nn.softmax(prediction)

print(f" predicting a Two. Probability vector: \n{prediction_p}")
print(f"Total of predictions: {np.sum(prediction_p):0.3f}")
```

```
 predicting a Two. Probability vector:
[[2.46e-09 3.43e-04 1.00e+00 1.43e-04 3.18e-15 5.61e-10 3.98e-12 2.12e-07
  1.32e-08 4.40e-10]]
Total of predictions: 1.000
```

In [24]:
```
yhat = np.argmax(prediction_p)

print(f"np.argmax(prediction_p): {yhat}")
```

```
np.argmax(prediction_p): 2
```

In [ ]: