

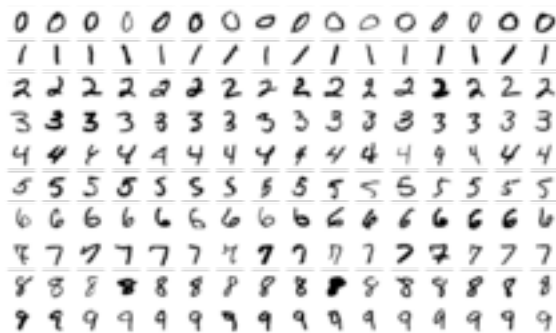
BU.330.760 Deep Learning with Unstructured Data

Lab 4. Keras and Deep Feedforward Network

Learning Goal: practice using Keras package to train a multilayer perceptron model on MNIST dataset

Required Skills: knowledge on feedforward network and Keras

1. We will use Keras (<https://keras.io>) in lab 4, which is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano.
2. If you are using Google CoLab, you do not need to install anything for lab4 since all the required packages are already available on Google CoLab.
3. We will use the MNIST dataset for this lab. The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems. The testing dataset was taken from American high school students. The black and white images from NIST were normalized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels. See sample images below:



(https://en.wikipedia.org/wiki/MNIST_database)

4. In Jupyter notebook, or your own Python development environment, build the following scripts
 - a. Import required packages.

```
import keras
from keras.datasets import mnist
from tensorflow.python.keras.layers import Dense, Dropout
from tensorflow.python.keras import Sequential
import matplotlib.pyplot as plt
from PIL import Image
```
 - b. First, let's plot some MNIST instances. We will load the data, and create a grid of 3x3 images.

```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
for i in range(0, 9):
    plt.subplot(330 + 1 + i)
    plt.imshow(Image.fromarray(x_train[i]))
plt.show()
```

- c. Since the dataset has digits 0-9, there will be 10 classes. We will use a batch size of 128, and 20 training epochs.

```
batch_size = 128
num_classes = 10
epochs = 20
```

- d. The database contains 60,000 training images and 10,000 testing images, each of 28x28 pixel. We will normalize from the vectors from 0-255 to range 0-1.

```
x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')
```

- e. Then convert class vectors to binary class matrices.

```
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

- f. We can build the model now.

```
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))
model.summary()
```

- g. We will use cross-entropy for the loss function, and stochastic gradient descent as the optimization procedure.

```
model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])
```

- h. Now let's train the model, and test.

```

history = model.fit(x_train, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

```

- i. Finally, let's plot the accuracy curve as well as loss curve of the training process.

```

#Plot the Loss Curves
plt.figure(figsize=[8,6])
plt.plot(history.history['loss'],'r',linewidth=3.0)
plt.plot(history.history['val_loss'],'b',linewidth=3.0)
plt.legend(['Training loss', 'Validation Loss'],fontsize=18)
plt.xlabel('Epochs ',fontsize=16)
plt.ylabel('Loss',fontsize=16)
plt.title('Loss Curves',fontsize=16)
#Plot the Accuracy Curves
plt.figure(figsize=[8,6])
plt.plot(history.history['accuracy'],'r',linewidth=3.0)
plt.plot(history.history['val_accuracy'],'b',linewidth=3.0)
plt.legend(['Training Accuracy', 'Validation Accuracy'],fontsize=18)
plt.xlabel('Epochs ',fontsize=16)
plt.ylabel('Accuracy',fontsize=16)
plt.title('Accuracy Curves',fontsize=16)

```

5. Assignment question:

Draw the computation graph for the following function: $f(a, b, c, d, e) = \frac{1}{(1+(a^b+c^d) \times e)^2}$

Compute the gradient of the function with respect to its inputs at $(a, b, c, d, e) = (1, 1, 1, 1, 1)$ (refer to Lecture 3)

Submission:

Finish all the steps in the lab, save it to lab4.ipynb file with all the outputs.

Complete the assignment question. You can either use drawing and equations in word, or handwrite and then scan or take a picture using your phone. You can either save it as a pdf or an image, such as assignment4.pdf, assignment4.png or assignment4.jpg.

Your submission should include

1. lab4.ipynb, or lab4.py together with all the outputs.
2. assignment4.pdf, or assignment4.png, or assignment4.jpg.

Due: Feb 22nd 11:30am EST

Reference:

<https://www.learnopencv.com/image-classification-using-feedforward-neural-network-in-keras/>

IDS 576: Advanced Predictive Models and Applications for Business Analytics Assignment 1,
Theja Tulabandhula, 2017, University of Illinois at Chicago

<https://www.freecodecamp.org/news/install-tensorflow-and-keras-using-anaconda-navigator-without-command-line/>