DS620 Machine Learning and Deep Learning

HOS06 Artificial Neural Networks

04/20/2021 Developed by Minh Nguyen

05/04/2021 Reviewed by Shanshan Yu

School of Technology & Computing @City University of Seattle (CityU)

Learning objectives

- Artificial Neural Networks
- Building a Neural Networks with Keras

Resources

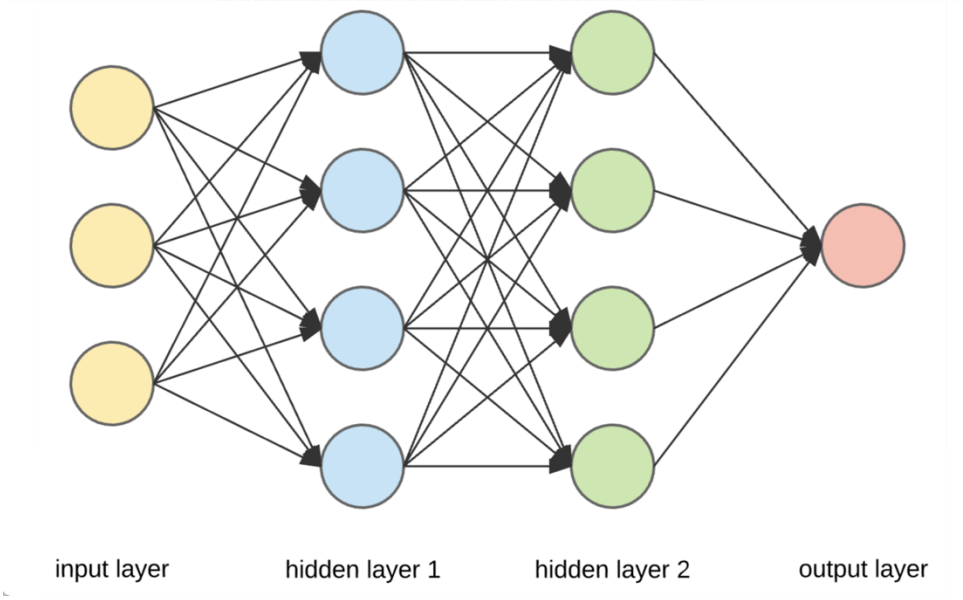
- Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, Inc.
- Ognjanovski, G. (2020, June 7). Everything you need to know about Neural Networks and Backpropagation — Machine Learning Easy and Fun. Medium. https://towardsdatascience.com/everything-you-need-to-know-about-neural-networks-and-backpropagation-machine-learning-made-easy-e5285bc2be3a
- But what is a Neural Network? | Deep learning, chapter 1. (2017, October 5). [Video]. YouTube. https://www.youtube.com/watch?v=aircAruvnKk
- Team, K. (2020, April 12). *Keras documentation: The Sequential model*. Keras. https://keras.io/guides/sequential_model/

Introduction

Taking inspiration from the structure of the human brain, the Artificial Neural Network (ANN) is the very core of deep learning. They are versatile, powerful and scalable, making them ideal to tackle large and complex Machine Learning tasks. In this week's HOS, we will get introduced to the Keras API which is a powerful, high level API that allows us to build, train and evaluate neural networks.

I. Neural Network

The neural network consists of 3 types of layers: input layer, hidden layers, and outputs layer



- Input layer — initial data for the neural network.
- Hidden layers — intermediate layers between input and output layer and place where all the computation is done. The number of hidden layers is arbitrary depending on the user.
- Output layer — produce the result for given inputs.

Each layer contains nodes (the circle), each node is connected to every node from the previous layers, this is how data travel through the neural network. Like other Machine Learning models, A Neural Network takes in training data, learns its underlining pattern then uses those patterns to predict future data. In thí HOS, we will use the MNIST handwritten digit dataset and build a neural network that can classify handwritten digit.

II. Neural Network in Keras

Preparing development environment

1. From Google Colab, create a new notebook, name it "ANN.ipynb"
2. Type the following codes to import libraries.

```
import numpy as np
import matplotlib.pyplot as plt

from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense
```

3. Import the data.

Prepare Keras dataset

```
#load dataset
(X_train_full, y_train_full), (X_test, y_test) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11493376/11490434 [==============================] - 0s 0us/step
```

```
print(X_train_full.shape)
print(y_train_full.shape)
```

```
(60000, 28, 28)
(60000,)
```

4. Let’s examine a sample to understand the properties of the input.

Viewing the data

```
X_train_full[0]
```

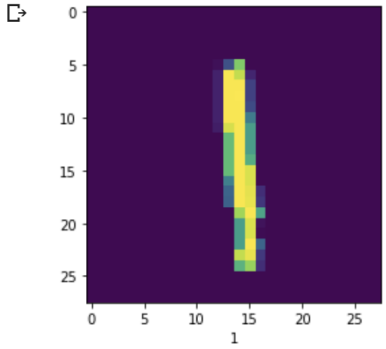
```
array([[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   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,   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,   0,   0,   0,   0,   0,   0,   0,   0,   3,
         18,  18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154, 170,
        253, 253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253, 253,
        253, 253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,  18, 219, 253, 253, 253, 253,
        253, 198, 182, 247, 241,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,  80, 156, 107, 253, 253,
        205,  11,   0,  43, 154,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,  14,   1, 154, 253,
         90,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
```

The image is represented as a 28 by 28 matrix with each cell corresponding to a pixel. The value of the cells indicates the gray scale value of each pixel.

5. Let’s visualize this as an image.

```
# Create a function that visualize the pixel and its corresponding label
def visualize_digit(index):
    plt.figure(figsize=(4,4))
    plt.imshow(X_train_full[index])
    plt.xlabel(y_train_full[index])
```

```
visualize_digit(8)
```



6. Because Gradient Descent is used to train the network, we need to Normalize the input feature. The maximum value of a cell is 255, we will divide all the input values by 255. Validation data is also required during the training process so we will also split the input data into validation set and training set.

```
# Spliting training data from validation data
# Rescale the input for Gradient Descent optimization
X_valid, X_train = X_train_full[:5000]/255.0, X_train_full[5000:]/255.0

# Reshape the output
y_valid, y_train = y_train_full[:5000], y_train_full[5000:]
```

Create an Artificial Neural Network with 2 hidden layers.

In this example, we will create a neural network by initializing a Sequential model. The Sequential model is the simplest kind of Keras model for neural networks. A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.

Next, we will add a Flatten layer which is a class of reshaping layer. Its role is to convert each input image into a 1D array. Since it's the first layer in this example, it also acts as an input layer.

The final type of layer is the Dense layer, which is a class of core layer. Its role is to carry out computation using activation function.

- Type the following code into ANN.ipynb

Create an ANN

```
model = Sequential([
    Flatten(input_shape=(28,28)), # the input shape is the size of each image
    Dense(units = 500, activation='relu'),
    Dense(units = 200, activation='relu'),
    Dense(units = 10, activation='softmax') # The number of output units is the number of output classes
])
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
dense (Dense)                (None, 500)               392500
_________________________________________________________________
dense_1 (Dense)              (None, 200)               100200
_________________________________________________________________
dense_2 (Dense)              (None, 10)                2010
=================================================================
Total params: 494,710
Trainable params: 494,710
Non-trainable params: 0
_________________________________________________________________
```

Compile and train the model.

- Once the model is created, you need to specify the optimizer and loss function that the model will use. This is done by using the compile method.

Train the ANN model

```
model.compile(loss='sparse_categorical_crossentropy', #Used when value of y is a scaler value
              optimizer="sgd", #stochastic gradient descent
              metrics = ['accuracy'])
```

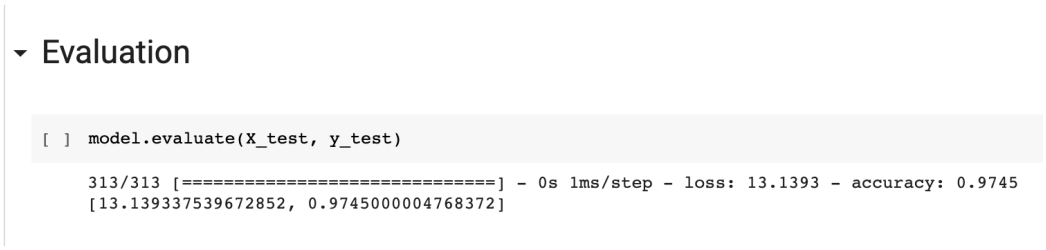
- Finally, we train the model by passing the training data and training data into the model using the fit method. The parameters epoch is the number of cycles the entire training data is passed through the network.

```
model.fit(X_train, y_train, epochs=30,
          validation_data=(X_valid, y_valid))
```

```
1719/1719 [==============================] - 8s 5ms/step - loss: 0.9644 - accuracy: 0.7681 - val_loss: 0.2972 - val_accuracy: 0.9192
Epoch 2/30
1719/1719 [==============================] - 7s 4ms/step - loss: 0.2925 - accuracy: 0.9170 - val_loss: 0.2343 - val_accuracy: 0.9342
Epoch 3/30
1719/1719 [==============================] - 7s 4ms/step - loss: 0.2350 - accuracy: 0.9346 - val_loss: 0.2048 - val_accuracy: 0.9430
Epoch 4/30
1719/1719 [==============================] - 7s 4ms/step - loss: 0.2083 - accuracy: 0.9428 - val_loss: 0.1764 - val_accuracy: 0.9498
Epoch 5/30
1719/1719 [==============================] - 7s 4ms/step - loss: 0.1753 - accuracy: 0.9497 - val_loss: 0.1533 - val_accuracy: 0.9582
Epoch 6/30
1719/1719 [==============================] - 7s 4ms/step - loss: 0.1541 - accuracy: 0.9555 - val_loss: 0.1406 - val_accuracy: 0.9614
Epoch 7/30
1719/1719 [==============================] - 7s 4ms/step - loss: 0.1367 - accuracy: 0.9619 - val_loss: 0.1273 - val_accuracy: 0.9670
Epoch 8/30
1719/1719 [==============================] - 7s 4ms/step - loss: 0.1265 - accuracy: 0.9637 - val_loss: 0.1185 - val_accuracy: 0.9690
Epoch 9/30
1719/1719 [==============================] - 7s 4ms/step - loss: 0.1060 - accuracy: 0.9709 - val_loss: 0.1095 - val_accuracy: 0.9706
Epoch 10/30
1719/1719 [==============================] - 7s 4ms/step - loss: 0.1007 - accuracy: 0.9723 - val_loss: 0.1057 - val_accuracy: 0.9724
Epoch 11/30
1719/1719 [==============================] - 7s 4ms/step - loss: 0.0962 - accuracy: 0.9736 - val_loss: 0.0991 - val_accuracy: 0.9738
Epoch 12/30
```

Evaluate testing data.

- Keras model comes with the evaluation method for testing data.



Push Your Work to GitHub

Download the notebook from Colab:

File -> ANN.ipynb

Move the downloaded file into your **Module6** working folder.

Open terminal and Navigate to the GitHub folder of this week HOS.

Make sure the assignment files on the subfolder Module6 of hos06a_YouGithubUserName folder, enter the following command to upload your work:

>>>> git add .

>>>> git commit -m “Submission for HOS06”

>>>> git push origin master