CS4287 Neural Computing

Lab 3: Week 5, S1 AY 24/25

Introducing TensorFlow

Preliminaries

Create an account in Google Colab

https://colab.research.google.com

Open a new file in Colab– a new Jupyter Notebook

- ➔ Jupyter Notebooks is an web-based interactive python development environment
- ➔ Colab provides a runtime with GPU support, useful for matrix computation.
- ➔ A matrix is a tensor!
- ➔ Tensorflow is a framework and APIs for deep learning

Basics of Jupyter Notebook:

- Type in a block consisting of 1 or more statements
- hit the "Run Cell"
- Hit "+Code" to enter a new block
- Hit "+Text" to enter comments

An example of a block could be L1-L3 in the Exercise 1 program

Exercise 1: A Neural Network in Keras and Tensorflow for MNIST

Based on: *Antonio Gulli, Amita Kapoor, and Sujit Pal. Deep Learning with TensorFlow 2 and Keras, 2nd Edition. Packt> Press. 2019*

Prog1 implements a neural network in TensorFlow 2.0 for the MNIST dataset

The code uses the Keras API which is a layer of abstraction on top of TensorFlow

The dataset is split into X_train for updating the weights, and X_test for accessing performance.

Data is converted into float32 to use 32 bit precision and normalised in the range [0..1]

The labels are loaded in Y_train and Y_test.

The network architecture is

Input = 28 * 28 = 748 nodes

Hidden Layer: 10 nodes

Output layer: 1 node using the Softmax activation which outputs probabilities that sum to 1. It aggregates the 10 answers provided by the 10 neuros in the hidden layer.

Weights = (784 x 10) + (10 x 1) = 7840 + 10 = 7850

The cycle is epochs of training and validation. And when finished, testing.

```
import tensorflow as tf
import numpy as np
from tensorflow import keras

# for reproducibility
np.random.seed(1671)

# hyper-pramters
EPOCHS = 200
BATCH_SIZE = 128
VERBOSE = 1
NB_CLASSES = 10   # number of outputs = number of digits
N_HIDDEN = 128
VALIDATION_SPLIT=0.2 # how much TRAIN is reserved for VALIDATION

# loading MNIST dataset
# verify
# the split between train and test is 60,000, and 10,000 respectively
# one-hot coding is automatically applied
mnist = keras.datasets.mnist
(X_train, Y_train), (X_test, Y_test) = mnist.load_data()
print(X_train.shape[0], 'train samples')
print(X_test.shape[0], 'test samples')

#normalize in [0,1]
X_train, X_test = X_train / 255.0, X_test / 255.0
#X_train is 60000 rows of 28x28 values --> reshaped in 60000 x 784
RESHAPED = 784
#
```

```
X_train = X_train.reshape(60000, RESHAPED)
X_test = X_test.reshape(10000, RESHAPED)
Y_train = Y_train.astype('float32')
Y_test = Y_test.astype('float32')
```

```
#create the neural network
model = tf.keras.models.Sequential()
model.add(keras.layers.Dense(NB_CLASSES,
  input_shape=(RESHAPED,),
  kernel_initializer='zeros',
  name='dense_layer',
  activation='softmax'))

# summary of the model
model.summary()
```

You must specify the optimiser used to update the weights. See https://www.tensorflow.org/api_docs/python/tf/keras/optimizers

Options include SDG, Adam, etc.

You must specify the Loss (error / cost / objective) function, see:

https://www.tensorflow.org/api_docs/python/tf/keras/losses

Common choices are:

- MSE: Mean Squared Error
- categorical_crossentropy: which defines the multiclass logarithmic loss. If the true class is c, and the prediction is y, then

$$L(c,p) = -\sum_i C_i \ln(p_i)$$

You must then specify the metrics used to evaluate performance, see

https://www.tensorflow.org/api_docs/python/tf/keras/metrics

Common choices are

- Accuracy: the % of correct predictions
- Precision
- Recall

```
# compiling the model
model.compile(optimizer='SGD',
  loss='sparse_categorical_crossentropy',
```

```
            metrics=['accuracy'])

#train the moodel
model.fit(X_train, Y_train,
  batch_size=BATCH_SIZE,
  epochs=EPOCHS,
  verbose=VERBOSE,
  validation_split=VALIDATION_SPLIT)

#evalute the model
test_loss, test_acc = model.evaluate(X_test, Y_test)
print('Test accuracy:', test_acc)
```

Exercise 2: Add Hidden Layers

Add the following lines of code before the model declaration

```
#One hot representation of the samples
Y_train = tf.keras.utils.to_categorical(Y_train, NB_CLASSES)
Y_test = tf.keras.utils.to_categorical(Y_test, NB_CLASSES)
```

Change the model as follows

```
model = tf.keras.models.Sequential()
model.add(keras.layers.Dense(N_HIDDEN,
   input_shape=(RESHAPED,),
   name='dense_layer', activation='relu'))
model.add(keras.layers.Dense(N_HIDDEN,
   name='dense_layer_2', activation='relu'))
model.add(keras.layers.Dense(NB_CLASSES,
   name='dense_layer_3', activation='softmax'))
```

Exercise 3

At the end of this lab please work through Colab basics:

https://colab.research.google.com/notebooks/intro.ipynb?utm_source=scs-index#scrollTo=GJBs_flRovLc