ex1

June 13, 2021

[1]:
```
'''Trains a simple deep NN on the MNIST dataset.

Gets to 98.40% test accuracy after 20 epochs
(there is *a lot* of margin for parameter tuning).
2 seconds per epoch on a K520 GPU.
'''
```

[1]:
```
'Trains a simple deep NN on the MNIST dataset.\n\nGets to 98.40% test accuracy
after 20 epochs\n(there is *a lot* of margin for parameter tuning).\n2 seconds
per epoch on a K520 GPU.\n'
```

[2]:
```
from tensorflow import keras
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import RMSprop
```

[3]:
```
batch_size = 128
num_classes = 10
epochs = 20
```

[4]:
```
# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

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')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-
datasets/mnist.npz
11493376/11490434 [==============================] - 1s 0us/step
60000 train samples
10000 test samples
```

[5]:
```
# 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)
```

[6]:
```
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()

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

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])
```

```
Model: "sequential"

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 512)               401920

_________________________________________________________________
dropout (Dropout)            (None, 512)               0

_________________________________________________________________
dense_1 (Dense)              (None, 512)               262656

_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0

_________________________________________________________________
dense_2 (Dense)              (None, 10)                5130
=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0

_________________________________________________________________
Epoch 1/20
469/469 [==============================] - 6s 11ms/step - loss: 0.4330 -
accuracy: 0.8617 - val_loss: 0.1243 - val_accuracy: 0.9604
```

```
Epoch 2/20
469/469 [==============================] - 4s 9ms/step - loss: 0.1082 -
accuracy: 0.9676 - val_loss: 0.0817 - val_accuracy: 0.9735
Epoch 3/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0711 -
accuracy: 0.9791 - val_loss: 0.0748 - val_accuracy: 0.9773
Epoch 4/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0614 -
accuracy: 0.9821 - val_loss: 0.0766 - val_accuracy: 0.9798
Epoch 5/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0476 -
accuracy: 0.9856 - val_loss: 0.0806 - val_accuracy: 0.9784
Epoch 6/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0406 -
accuracy: 0.9876 - val_loss: 0.0774 - val_accuracy: 0.9812
Epoch 7/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0363 -
accuracy: 0.9889 - val_loss: 0.0695 - val_accuracy: 0.9834
Epoch 8/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0333 -
accuracy: 0.9900 - val_loss: 0.0881 - val_accuracy: 0.9804
Epoch 9/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0301 -
accuracy: 0.9910 - val_loss: 0.0881 - val_accuracy: 0.9803
Epoch 10/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0239 -
accuracy: 0.9924 - val_loss: 0.0853 - val_accuracy: 0.9823
Epoch 11/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0260 -
accuracy: 0.9925 - val_loss: 0.0927 - val_accuracy: 0.9832
Epoch 12/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0229 -
accuracy: 0.9932 - val_loss: 0.0994 - val_accuracy: 0.9820
Epoch 13/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0230 -
accuracy: 0.9931 - val_loss: 0.1041 - val_accuracy: 0.9820
Epoch 14/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0208 -
accuracy: 0.9939 - val_loss: 0.0929 - val_accuracy: 0.9825
Epoch 15/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0188 -
accuracy: 0.9943 - val_loss: 0.0951 - val_accuracy: 0.9825
Epoch 16/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0166 -
accuracy: 0.9949 - val_loss: 0.1096 - val_accuracy: 0.9831
Epoch 17/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0185 -
accuracy: 0.9952 - val_loss: 0.0959 - val_accuracy: 0.9844
```

```
Epoch 18/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0168 -
accuracy: 0.9955 - val_loss: 0.1171 - val_accuracy: 0.9828
Epoch 19/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0162 -
accuracy: 0.9952 - val_loss: 0.1145 - val_accuracy: 0.9827
Epoch 20/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0155 -
accuracy: 0.9956 - val_loss: 0.1184 - val_accuracy: 0.9833
Test loss: 0.11838609725236893
Test accuracy: 0.983299970626831
```

[ ]: