DLM Labor 4. Activation Functions and Weight Initialization
WS 17-18

1 Solving MNIST

file: Labor.py

1.1 SETUP AND LOAD DATA

Set up the python environment.
Import the required libraries. In this practice we'll download the data using the datasets library from scikit-learn.

```
 7 import numpy as np
 8
 9 SEED = 76238
10 np.random.seed(SEED)
11
12 import keras
13
14 from keras.models import Sequential
15 from keras.layers import Dense, Activation
16 from keras.initializers import RandomNormal
17 from keras.optimizers import SGD
18 from keras.utils import to_categorical
19
20 import matplotlib
21 import matplotlib.pyplot as plt
22
23 def load_mnist():
24     ''' Loads MNIST dataset using scikit-learn datasets library
25         70000 examples of handwritten digits of size 28x28 pixels,
26         labeled from 0 to 9.
27         original data: yann.lecun.com/exdb/mnist
28     '''
29     from sklearn.datasets import fetch_mldata
30
31     mnist = fetch_mldata('MNIST original', data_home='./mnist')
32
33     # Rescale the data
34     X, y = mnist.data / 255., mnist.target
35
36     # one hot encoding
37     y = to_categorical(y, 10)
38
39     # use traditional train/test split
40     X_train, X_test = X[:60000], X[60000:]
41     y_train, y_test = y[:60000], y[60000:]
42
43     from sklearn.model_selection import train_test_split
44     X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=
45
46     return X_train, y_train, X_val, y_val, X_test, y_test
47
```

```
66 def main():
67
68     # Load data
69     X_train, y_train, X_val, y_val, X_test, y_test = load_mnist()
70
```

1.2 VISUALIZING THE LOSS AND ACCURACY

Use the Callback Tensorboard included in Keras to visualize the accuracy and loss for the training and validation sets. This will help you monitor the training process.

1.3 CREATING THE NETWORK

Create a multilayer neural network to classify the handwritten images. Use the Keras Documentation to build and train the model. https://keras.io/

```
# create model
model = Sequential()
## layer 1
model.add(Dense(units= ,
                input_shape= ,
                kernel_initializer=weigth_initializer))
model.add(Activation(activation))
## layer 2
...
## layer 3
...
.
.
.
## layer n
## predictions layer
...

optimizer = SGD(lr=0.01)

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

# Train model
model.fit(...
          validation_data= ,
          batch_size=10,
          epochs=10,
          shuffle=True)
```

1.4 ADJUSTING THE NETWORK

Vary the architecture, the weight initialization methods and the activation functions so that your network can achieve an accuracy greater than 96%

1.5 SAVING THE NET

Save the model with the best results using the help functions from keras.