NN Programming Tasks

1 Playground

1. Open the website *playground.tensorflow.org*.
2. This interactive website gives you the opportunity to play with some basic network parameters and to see their corresponding effects.
3. Take some time to go through the different datasets and try to find a working model by making a basic model more advanced step by step.

2 KERAS (Tensorflow) API

Classification

1. Open the notebook *keras_classification_tutorial*.
2. This notebook will guide you through the creation of a basic classification neural network in TensorFlow using the KERAS API.
3. For comparison with the previous exercises, the model is created for the known blob dataset.
4. Refer to the website *keras.io* for detailed information on the API.

Regression

1. Open the notebook *keras_regression_tutorial*.
2. This notebook will guide you through the creation of a basic regression neural network in TensorFlow using the KERAS API.
3. The dataset is manually created at the beginning of the notebook.
4. Refer to the website *keras.io* for detailed information on the API.

3 Classification

1. Open the notebook *classification_ex_diy*.
2. In this notebook, you will be guided through implementing a classifier for the MNIST fashion database. As a first step, download the database by executing the corresponding code block.

3. Work yourself through the next section to become familiar with the data. The database consists of 70,000 28x28 grayscale images with corresponding class labels.

4. Normalize the images in the preprocessing block.

5. Next, define the model architecture with a KERAS sequential model. Use core layers only, refer to *keras.io* for the layer documentation. Once you have a working model (verified by the next steps), you can play around with the parameters and layers.

6. Compile the model. Make sure to use *sparse_categorical_crossentropy*, otherwise you need to change the labels to one-hot encoded vectors.

7. Train and test your model. With a basic network, you should be able to achieve a test accuracy of at least 88 %.

8. Optional: Add and test an abortion condition by adding a callback function (*EarlyStopping*) for the training process. What purpose does adding such a callback function have?

4 Regression

1. Open the notebook *regression_ex_diy*.

2. In this notebook, you will be guided through implementing a regression model which should estimate a car's miles per gallon value from other car parameters. As a first step, read in the data by executing the corresponding code block.

3. Execute the train / test split.

4. Spend some time on the data statistics section to familiarize yourself with the data. Take a closer look at the correlation matrix. What information is provided through this matrix? What statements can be made about the car parameter relationships?

5. Create the target data (labels) by executing the next block.

6. The next block defines all the model and optimization parameters. Fill in the model structure. Your model should have three dense layers, where the first two layers have 64 neurons each and both apply a rectified linear unit as activation function. What should the last of the three layers look like?

7. Execute the training and the evaluation block. Is the performance satisfactory?

8. The next section will repeat the last two steps but with normalized data. Fill in the same network structure and run the code. Compare the performance with the previous model. Where does the difference come from?

9. Execute the next code block. The plot is a zoomed-in version of the training / evaluation history. What phenomenon do you observe? Is there a name for it?

10. To counteract the observed phenomenon, the next code block repeats the training process but with an abortion condition. Fill in the same network structure as above and then define the abortion condition yourself. Play around with the criterion parameters to obtain a satisfactory result.

11. As a last step, a correlation analysis is performed for the predictions on the separated test data. Discuss the results.