Foundations of Data Analysis - WS21
Lab assignment
Supervised learning

Due date: 9:45 am on 11.11.2021

Description and instructions

The maximum number of points achievable in this assignment is 100. Kindly follow the submission instructions carefully as failing to do so will result in a penalty.

- You should work on this assignment individually, however you are allowed and encouraged to discuss your approaches to the problems, as well as questions you may have, with your colleagues.
- You are however not allowed to share your code! (Except for very small parts, in order to discuss problems with your peers.)
- Remember to cite every external source that you use!
- Any act of plagiarism will be taken very seriously and handled according to university guidelines.
- Clearly label all figures and comment your code!
- Upload your submission to Moodle either as a python file with your code and a pdf containing your answers to the open questions, or as a jupyter notebook including both code and other answers. All files should be named `<last_name>.<pdf/py/ipynb>`, replacing `<last_name>` with your last name(s).

Do not hesitate to email me (Anja Meunier) at `anja.meunier@univie.ac.at` or post on the discussion forum on Moodle with any questions you may have.

Introduction

The purpose of this assignment is for you to put the theory about supervised learning algorithms we have discussed in the lectures into practice. You will get to know two different popular machine learning libraries for python, `scikit-learn` and `keras`. Please consult the respective documentations (found online) for questions on how to use the objects mentioned in this assignment.
We will use the CIFAR-10 dataset (`https://www.cs.toronto.edu/~kriz/cifar.html`), which contains labeled images. We will first load and inspect the data and then train a random forest, as well as different neural network models.

1 Dataset [20 points]

(a) [10 points] Load the CIFAR-10 dataset with `(x_train, y_train), (x_test, y_test) = keras.datasets.cifar10.load_data()`. Determine and print the following information:

- Number of training and test samples
- Size of the images
- Number of color channels
- Number of classes
- Number of samples for each class
- Class label corresponding to each class

(b) [2 points] The images are represented in RGB format, where each pixel can have a value between 0 and 255 for each color channel. Rescale `x_train, x_test`, so that all values lie between 0 and 1.

(c) [6 points] For each class, show five images. Use `matplotlib.pyplot.subplots` to display them all in one figure. In a second figure, display the class means (as images).

(d) [2 points] The chance level is the expected percentage of correct predictions which would be made by randomly choosing a label according to the class probabilities. What is the chance level of this dataset?

2 Random forest [20 points]

(a) [5 points] Train a random forest with 1000 trees on the training dataset, using `sklearn.ensemble.RandomForestClassifier`. Before training, set the random seed using `numpy.random.seed(1111)`.

Hint: This model requires flat (1-dimensional) input samples and scalar outputs. Use `numpy.reshape` to change the dimensions of the training and test data.

(b) [3 points] With the fitted classifier, predict the labels of the test set.

(c) [2 points] Compute and print the accuracy of the classifier (percentage of correct predictions).

(d) [5 points] Compute and display the confusion matrix of the classifier, using `sklearn.metrics.confusion_matrix`.

(e) [5 points] Which classes are most often confused with each other by this model? Give an explanation, why this may occur.

3 Neural networks [60 points]

Hints:

- All of the following networks need one-hot-encoded output, which you obtain by using `tensorflow.keras.utils.to_categorical` on `y_train`.
- Fully connected models take flattened input, convolutional networks take the input in the original shape of the images. Specify the input shape to the first layer with the `input_shape` argument.
- All models need an output layer with as many neurons as there are classes, with softmax activation function.
- The history object `h` is returned by the fit method of each model. Then access the learning curves with `h.history['accuracy']` and `h.history['val_accuracy']`.

(a) [7 points] Using `keras.models.Sequential` as model and `keras.layers.Dense` as layers, define a fully connected neural network with one hidden layer with 20 neurons with ReLu activation function. Before you start, set the random seeds using `numpy.random.seed(2222)` and `tensorflow.random.set_seed(2222)`.

(b) [5 points] Compile the model using binary crossentropy as loss, stochastic gradient descent with learning rate 0.01 as optimizer, and accuracy as additional metric to be measured.

(c) [3 points] Print the summary of the model. How many parameters does the model have in total?

(d) [7 points] Fit the model for 30 epochs, using 20% of the training data as validation set. Then plot the learning curves. Does the model overfit? If yes, what do you think is the reason? Would it make sense to train the model for more epochs?

(e) [8 points] Perform steps (a) - (d) for a fully connected neural network with one hidden layer with 100 neurons. Use the random seed `3333`.

(f) [12 points] Perform steps (a) - (d) for a convolutional neural network with one convolutional layer (`keras.layers.Conv2D`) with 64 filters and a kernel size of 3×3 and ReLu activation function, followed by a max-pooling layer (`keras.layers.MaxPooling2D`) with a pool size of 3×3. Briefly explain in your own words, what both of these layer types do. Use the random seed `4444`.

Hint: You need to add a flattening layer `keras.layers.Flatten` between the last `Conv2D` or `MaxPooling2D` layer and the `Dense` output layer.

(g) [10 points] Try to create a model that outperforms all of the previous three models by playing around with the model architecture (i.e. number and type of layers, number of neurons, etc.), as well as other parameters. (You may also use layers and parameters other than those discussed here, as well as use existing architectures, so long as you are transparent about your sources.) Use the random seed `5555`.
Briefly explain, which different approaches you tried, but only keep one model in your submission.

(h) [8 points] Using the best of all previous neural network models, predict the labels of the test set and compute and display accuracy and confusion matrix. Explain why it is important to compare the models with a validation set first, and only then evaluate the chosen model on the test set.