Homework 04

IANNWTF 21/22

Submission until 21th Nov 23:59 via https://forms.gle/n6ERdhYx3uBPzuGn9

Welcome back to the fourth homework for IANNWTF. Last week you coded your first MLP using tensorflow. Well done! In this week's lecture, you should have read about different optimization and regularization techniques. Now as you know the theory, we want you to use them in practice!
You will work on a simple dataset solvable with an MLP. We want you to code a very small basic model and train it on the task. Use this model's performance as a benchmark. Now comes the fun part: Apply at least 3 optimization and regularization techniques of your choice featured in the lecture to increase your model's performance and training convergence! Feel free to toy around a little and see how different techniques affect what's happening. Have fun! *As always, there are hints (denoted by the superscript numbers) at the end of the document to help you get started. Do not read them immediately, but think about it first, and only go to them if you're stuck. Sometimes instructions might be a bit vague and that usually is intentional. We leave most things up to you specifically to enable the respective learning experiences which we deem necessary. But sometimes there are also mistakes from our side, so don't be shy to ask if something is unclear for you or doesn't make sense.*
Also, feel free to copy code snippets from your last week's homework or the courseware!

1 Data set

In this homework we would like you to be sophisticated: The dataset we want you to work on is the `wine_quality` dataset, containing a set of collected features for each wine and a rank of that wine as found by 'wine experts' on the scale of 1 to 10.

You can also find this dataset in the Tensorflow Datasets Collection , however, as `tfds` has been proven to be a bit difficult at times, this week we would like you to create your own Tensoflow Dataset from a .csv using `pandas` [1]. Do not worry, this is very easy!

1.1 Load Data into a Dataframe

You can download the .csv and store it on your local computer or directly read in the url when working with pandas. Read the data into a pandas dataframe. [2]

Wine Quality Dataset: https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv

Make yourself familiar with the dataset.

- What keys are there?
- What should be the input and what should be the target for our NN? [3]

1.2 Create a Tensorflow Dataset and a Dataset Pipeline

Split the dataset into a train, test and validation split. [4]

Separate the **labels** from the **input** and store them accordingly. [5]

Out of the resulting dataframes, build a **Tensorflow Dataset**. [6]

Now we want to make this a **Binary Classification Task**: As we are not as sophisticated as our wine experts, we only care about *good* wine vs. *bad* wine. Write a function `make_binary(target)` that receives a target and returns a target fit for a binary classification task. [7] You can hardcode (i.e. fix) a threshold or use statistics obtained from your data, e.g. the median ranking of all wines.[8]

Create a **Data Pipeline** with all the necessary steps. You should at least map the `make_binary()` function to the dataset and apply batching.

2 Model

You can pretty much recycle the model you defined for last week's task with some small modifications.

- As the task is maybe a bit simpler and you are aiming to explore the effect of optimization techniques, also try to make your model a bit smaller. The baseline model should barely be able to perform sufficiently good at the task (better than random).
- As we now are dealing with a binary classification task, you have to change the configuration of the output layer. [9]

3 Training

Then start by training your network for 10 epochs using a learning rate of 0.1. You can again copy most of the training procedure from last week. However, you have to change the loss function to fit our binary task! [10]. For the beginning try using SGD as an optimizer.

Also, when computing the accuracy, slight changes are required to transfer to the binary task. [11]

4 Fine-Tuning

Now comes the fun part. Your task is to apply *at least 3* optimization and regularization techniques featured in the lecture with the goal to significantly increase your model's performance. [12]

You can also think about another approach and normalize the input data. [13]

In the end, everything is up to you! You should witness an increase in performance, stability, and generalization.

You may notice that some optimization techniques such as dropout seem to rather decrease performance. However, please bear in mind that these techniques often address ensuring better *generalization*. So before you start going all out on the test set, don't forget what the validation set is for.

5 Visualization

Visualize accuracy and loss for training and test data using matplotlib.

6 How-To Outstanding

The general rule for achieving the mark outstanding is the following: Your submission is outstanding, if it could be used as a sample solution for the other students. That means, the code must be instructive for other students in that it is clean, nicely structured, well commented, and efficiently using the methods introduced in Courseware.

For this week, we also want to see an increase in performance after applying your optimization/regularization techniques of at least **5%** and it should generalize to the validation set.

Additionally, we would like you to **visualize** how the different hyperparameter, optimization techniques and regularizers you made use of altered your training behaviour in comparison to your plain baseline model. Provide your own **explanation/interpretation** of what you see.

Have fun!

Notes

[1]Don't forget to import pandas! Usually, '`import pandas as pd` is used, thus in the future we will refer to `pandas` with `pd`

[2]Check out $\texttt{pd.read}_{c}sv()and think about how to set the \texttt{delimiter} argument$[14]

[3]The column 'quality' contains the respective targets to the feature vector specified by all previous columns i.e. each point of the dataset should have 11 values (fixed acidity, volatile acidity,...) and one target (quality).

[4]check out df.sample(frac=, $\text{random}_{s}tate =)and df.drop()$

[5]check out pandas `df.drop( ,axis=)`

[6]again check out $\texttt{tf.data.Dataset.from}_{t}ensor_{s}lices()$

[7]this function should compare the rank to some threshold and return a 0 for bad and a 1 for good wine

[8]if you use the mean, you may notice that your dataset becomes extremely imbalanced

[9]you only need 1 output neuron and you should use something like a `sigmoid` as an activation function

[10]Have a look at `tf.keras.losses.BinaryCrossentropy()`

[11]instead of `np.argmax()` think about using `np.round(,0)`

[12]If you are using Dropout, think about the flag training= that you can pass onto the call of Dropout.
[15]

[13]For that the stats returned by pandas `df.describe()` could prove useful to you