Advanced Machine Learning - IMDB Final Project

December 2023

1 Instructions

- The entire project will be graded and will contribute 30% of the final grade.
- It is required to submit notebooks only, you should not write a final report. Nevertheless, the notebooks you submit should contain comments and explanations of your procedures.
- I will not grade you on the final quality of the classification tasks, but I will grade i) correctness of the Python implementation, ii) quality of the dataset visualization, iii) quality of the comments and findings, iv) scientific correctness of the performed steps.
- Please submit only one zip file containing all the scripts you wrote.
- Submit the zip file on iCorsi. The deadline is **January 6, 23:59**.
- You can talk about the task to your fellow students, but please do not share your implementation/code. Plagiarism in the source code or the report will be penalized with a grade of 0 and might be reported to the faculty.
- You are absolutely welcome to ask me anything about things that are unclear to you in this assignment. Feel free to contact me for feedback or suggestions.

2 Tasks

- If not specified, for evaluation of your results use the metric you think is more appropriate among those we saw in the lectures.
- The overall goal is to do sentiment classification on a movie review dataset

2.1 Task 1

Load the IMDB dataset in the following way:

```
from keras.datasets import imdb
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=10000).
```

Read the description of the dataset in https://www.tensorflow.org/api_docs/python/tf/keras/datasets/imdb/load_data and try to understand how it is structured (it is pretty simple). You can do plots and/or visualization if you like, but it is not mandatory.

Create a feedforward network in order to predict the sentiment score. Suggestion: pad the input sentences (with 0s should work) so that they all have the same length (suggested maximum length: 500).

Train few epochs (e.g., 10). Do not expect a good accuracy.

2.2 Task 2

Transform your input sentences into a lower dimensional space using embeddings (embeddings dimensionality could be 100). Use these embeddings as input for your previous network.

Train few epochs (5 should be enough), the accuracy should have a significant increase.

2.3 Task 3

Text can be thought of as 1-dimensional sequence (a single, long vector) and we can apply 1D Convolutions over a set of word embeddings.

Add a `Conv1D` layer before the dense part of your previous network and train the new model.

2.4 Task 4

Instead of Convolution+Dense, let us try to use Recurrent Networks. Keep the embedded inputs and test the performances of three simple networks: i) SimpleRNN; ii) LSTM; iii) Conv1D+LSTM (just one recurrent layer should be sufficient in all of these networks).

Bonus points
Improve your previous networks testing other ideas that may come to your mind: Batch Normalization, Dropout, Layer Normalization, MC Dropout, etc. Is there any improvement?