Computational Neuroscience

AY 2022-23

Additional Bonus Tracks
Laboratory Assignment 3 (LAB3)
Gated Recurrent Models & Char RNN

General Information

The assignments indicated in the present file are bonus track assignments, meant to be for those who finish early the mandatory assignments, and are not formally required for completing the Lab Assignments.

The subfolder for this lab should be called "LAB3_1" and should include the scripts and the other files as requested in the assignment below. You can organize the code as you wish, implementing all the helper functions that you need, provided that these are included in the subfolder and are appropriately called in the scripts.

Detailed instructions for submissions are given in the lecture note "Intro to the course".

The recommended programming language is MATLAB, but you are allowed to use Python (e.g., with NumPy, PyTorch, or TensorFlow).
If you use MATLAB, you will provide:

- the source files as a collection of the scripts (.m files) used to solve the assignment, or as a live script (.mlx file) with all the necessary code.
- the output data as a collection of binary .mat files, or in a single .mat file.

If you use Python, you will provide:

- the source files in one of these forms:
    - a collection of the scripts (.py files) used to solve the assignment, with a main script that calls all the necessary other scripts;
    - a Jupyter nootebook (.ipynb file) with all the necessary code.
- the output data as a collection of pickle or json files, or in a single pickle/json file
    - in this case it is required to include a script to load the saved data in the chosen format.

In both cases, the required figures can be provided in .pdf files or in a graphics format (e.g., png, jpg). When multiple images are required, you are allowed to generate a single file including multiple subplots (or pages, if appropriate).

Supporting Material

For the Character RNN part, you can find a supporting reference in the file "Lab3_1-AdditionalMaterial-CharRNN.pdf" (on the Moodle website), and at the link: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Matlab

Documentation on:

- LSTM https://it.mathworks.com/help/deeplearning/ug/long-short-term-memory-networks.html
- GRU https://it.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.grulayer.html
- An tutorial example on time-series classification with LSTM https://www.mathworks.com/help/deeplearning/examples/sequence-to-sequence-classification-using-deep-learning.html

Matlab User's Guide https://www.mathworks.com/help/index.html
Matlab documentation using the *help* command

Python (TF/Keras)

An online guide to create, customize and use RNN layers:
https://keras.io/guides/working_with_rnns/
Basic API for recurrent models in Keras: https://keras.io/api/layers/recurrent_layers/
GRU: https://keras.io/api/layers/recurrent_layers/gru/
LSTM: https://keras.io/api/layers/recurrent_layers/lstm/

Python (PyTorch)

Documentation on recurrent layers in PyTorch:
https://pytorch.org/docs/stable/nn.html#recurrent-layers
LSTM:
https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html#torch.nn.LSTM
GRU:https://pytorch.org/docs/stable/generated/torch.nn.GRU.html#torch.nn.GRU
Vanilla RNN:
https://pytorch.org/docs/stable/generated/torch.nn.RNN.html
Linear layer module in PyTorch:
https://pytorch.org/docs/stable/generated/torch.nn.Linear.html

Bonus Track Assignment 4: benchmarking RNN models on the sequential MNIST task

The sequential MNIST can be downloaded as described in the Lab 3 - bonus track assignment #2.

The goal of this assignment is to compare the performance achievable by various RNN models on this task. As such:

(1) Write your custom code that solves the sequential MNIST time-series classification task with several recurrent neural networks. You must consider at least: a vanilla RNN model, a gated recurrent neural network (e.g., GRU and/or LSTM). Optionally, you can implement / use a bidirectional RNN, a deep RNN, and an Antisymmetric RNN (as described in lecture slides “Part3_Lecture2.pdf”).
(2) For every model, identify a suitable set of hyper-parameters and their possible values to explore by model selection
(3) Perform model selection, choosing, for each model individually, the values of the hyper-parameters on the validation set.
(4) For each model individually, train the network with the chosen configuration on the whole training set, and assess it on the test set. For each model, run a number of Ng identical experiments (same hyper-parameters’ values, different random seeds), where Ng >= 5, and record the training and test accuracies achieved. Compute averages and std over the Ng repetitions for each model individually.
(5) Provide a Table in which you report the training and test statistics of the achieved results (computed as indicated in the previous point).

The output of the assignment should then consist in the following

- The source code.
    - include a specification of the set of hyper-parameters and the range of values considered for the model selection
- The table generated at point (5)

Bonus-Track Assignment 5: Char RNN, or “The Unreasonable Effectiveness of Recurrent Neural Networks”

Implement a Character RNN, train it to generate sequential data from your favorite author (you can also consider generating lyrics for songs). Experiment using different choices for the RNN-based language model (e.g., LSTM, GRU, RNN, etc.) and temperature for the sampling function.

A tutorial on how to start organizing your code is available at https://colab.research.google.com/drive/1WsETcyfV7IGibKG2OojHN5AfzJmNBHT6?usp=sharing (here I chose to use as source text Dante's Divina Commedia)

The output of the assignment should then consist in the following

- The source code.
    - Please indicate the text you have used to train your model
- The values of the hyper-parameters of the model (including the temperature) used in your favorite runs
- The generated text in your favorite runs