Teachers: J. Hennebert, M. Melchior
Assistants: Y.-I. Beffa, S. Hackstein, O. Zayene

Practical Work 09 – 21/04/2022
Recurrent Nets

Objectives

The main objective of this PW is to build and train recurrent networks and experience some of the difficulties to do so. Again, you can carry through this practical work with your framework of choice (PyTorch or Tensorflow/Keras).

Submission

— **Deadline** : Wednesdays 4th May, noon

— **Format** : ipynb

Exercise 1 Names Classification

In this example, you will develop a character-based model for detecting the language (or country of origin) for given last names.

a) Download the zip-file `names_classification.zip` with a jupyter notebook and a data folder, unzip the file, open the jupyter notebook.

b) The first cells contain some helper functions to load the training and test data. Complete the generation of the alphabet (characters to be represented) and the functionality to create the vector representation for the characters. Use a one-hot-vector representation.

c) Implement a many-to-one model consisting of a single $SimpleRNN$ and a $Linear$ layer (with or without $softmax$) for the classification. Train the model - try different numbers of hidden units (dimension of hidden state vector) and play with different batch sizes. Analyse the accuracy and confusion matrix. Report about your findings in the notebook.

d) Can you improve the results by stacked more $SimpleRNN$? Do the comparison on the basis of the accuracy but also the confusion matrix.

e) Figure out a treatment of class imbalance, implement it and apply it to the given example. What improvement can you achieve ? Keywords : Minority over-sampling, class weights in loss. Make sure that you don't leak any information from the training set into the test set !

f) Explain why a treatment of class imbalance is important.

Exercise 2 Human Activity Recognition

In this example, you will study models for human activity recognition from accelerometer and gyroscope data recorded by smartphones (as presented in class).

The goal is to explore the hyperparameter/model space to find the best suited model for the given problem.

a) Download the zip-file `activity_recognition.zip` with a jupyter notebook and a data folder, unzip the file and open the jupyter notebook.

b) The first cells contain some helper functions to load the training and test data.

c) Now implement various different models (all are many-to-one) with different hyperparameter settings :

- Layer type : Consider `(Simple)RNN` | `LSTM` | `Conv1d` | `Dense|Linear`.
- Number of layers : 1+
- Different hyper parameters : `batchsize`, `nepochs`, `nhidden`, `clipnorm`.
- With/without regularisation

d) Compare at least 5 different architectures `SimpleRNN`, `LSTM`, stacked `LSTM`, `CNN`, `MLP` (each with a final `Dense | Linear` layer, quantitatively and qualitatively. Spot potential issues for specific settings / models. Make sure that the results are robust against different initialisation seeds.

e) What is the difference between a `Conv1d` with kernel size w and a `Conv2d` with kernel size $(w, 1)$?

Exercise 3 Optional : Review Questions

a) What are the different forms of sequence *mapping* allowed by recurrent neural networks ? Give for each form an example of application.

b) Compute the number of parameters to be trained for a two-layer $SimpleRNN$ and $softmax$ with hidden state dimensions 32 and 64, respectively, 10 classes to classify in the softmax and inputs given by sequences of length 100 and each element a vector of dimension 30.

c) Compute the number of parameters to be trained for a two-layer $LSTM$ and $softmax$ with hidden state dimensions 32 and 64, respectively, 10 classes to classify in the softmax and inputs given by sequences of length 100 and each element a vector of dimension 30.

d) Why is gradient clipping rather needed in long than in short sentences ?

e) Describe why SimpleRNNs have problems in learning long-term dependencies.

f) In what situations would you expect LSTMs (by its design) to perform better than SimpleRNNs ?

g) In some sequence problems, convolutional approaches may be better suited and more robust to learn. How could you apply conv layers when learning a time series forecasting task ? What do you need to pay attention to ?