Practical Work 11 – 06/05/2021
Recurrent Neural Networks

Objectives

The objective of this PW is to practice applications of Recurrent Neural Networks (RNN).

Submission

— **Deadline** : Wednesday 19th of May, 12 :00 (noon)

— **Format** : Zip with the jupyter notebooks.

Exercise 1 Language Classification by Last Names

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 `name_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 $softmax$ for the classification (by using tensorflow/keras). Train the model - try different numbers of hidden units (dimension of hidden state vector) and play with different batch sizes. Report about your findings in the notebook.

d) Figure out a treatment of class imbalance, implement it and apply it to the given example. What improvement can you achieve ?

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

f) Implement a model with more than one $SimpleRNN$ layer and a final $softmax$ for the classification. Train the model - try different numbers of hidden units (dimension of hidden state vector) and play with different batch sizes. Report about your findings in the notebook.

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 : Only consider $SimpleRNN$, $Conv1D/Conv2D$, $Dense$ (with softmax) and other Helper structures such as $InputLayer$, $Flatten$, $Reshape$.
   - Number of layers : 1+
   - Different hyper parameters : batchsize, nepochs, number of hidden units
   - With/without regularisation

d) Compare at least 5 different architectures (pure RNN, pure CNN, pure MLP, mixtures) quantitatively and qualitatively. Spot potential issues for specific settings / models. Make sure that the results are robust against different initialisation seeds.

Try to beat the test accuracy reported in class (and on the slides) !

Exercise 3 Sequence generation - startup names

The objective of this exercise is to build a generator of startup names using a RNN. We will use to train the system a corpus of existing startup names composed of 172K entries. Figure 1 illustrates the 20 first entries of the corpus.

Many-to-one approach

a) Read again the examples on the Shakespeare text generation in the section *Generative RNN* from the class support. Make sure you understand the different steps for the data preparation. You can also start from the ipython notebooks provided on Moodle for the Shakespeare text generation.

b) Get from Moodle the file `companies.csv` that will be used to train the system.

c) Read the data from the file. Treat the whole set of names as a unique string of text, like in the Shakespeare example.

d) Follow similar steps for the data preparation as in the Shakespeare example. First extract the set of chars and define the encoding and decoding dictionaries. Then chop the data into X and y, you may define here a sequence length of 10 and a skip of 3. Finally transform your data and labels into one-hot vectors.

```
Hashplay Inc.
New Incentives
GrabJobs
MediBookr
MelissaWithLove.co
Starting 11
The CarShare Guy
Allahabad Bank
Anlaiye
Any Time Loan
Asieris Pharmaceuticals
Birner Dental Management Services (PERFECT TEETH)
Blockchain Foundry
Breethe
Buffalo Wild Wings
Canadian Solar
Convert Group
DeepCam
Doctaly
Gaze Coin
```

FIGURE 1 – 20 first entries of the training set of startup names

e) Define your model using a SimpleRNN (64 units) and a Dense with softmax activation layers.

f) Train your model.

g) Generate startup names using different values of the hyperparameters : number of epochs, number of cells in the RNN, etc.

Report about your findings in the pdf report. Select 5-10 generated startup names that you find interesting.

OPTIONAL - Many-to-many approach

Redo the previous exercise using this time a many-to-many approach. You will need to modify slightly the way the target tensors are prepared in order to do so. Pay also attention to the parameters of the layer definition that are a bit different.

Exercise 4 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) Why is gradient clipping rather needed in long than in short sentences ?

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

e) How can you define a generative system ? Describe the two approaches seen in the class to build generative systems with RNNs.