Practical work 11 – 29/11/2018
Recurrent Neural Networks with Keras - part 1

Objectives

The objective of this PW is to practice with some applications of Recurrent Neural Networks (RNN) for sequence classification and sequence generation. Another objective is to experiments with the implementations of RNN in Keras.

As for the last practical work, we ask you to submit the solution for next week.

Submission

— **Deadline** : Wednesday 12 December, 12am

— **Format** : Zip with report and iPython notebook.

Exercice 1 Human Activity Recognition

In this example, you will develop a model for human activity recognition from accelerometer and gyroscope data recorded by smartphones (as described in the class).

a) Download the jupyter notebook `activity_recognition_stud.ipynb` and the zip-file with the data `UCI-HAR-Dataset.zip`, place them in the same folder, unzip the file. Open jupyter notebook.

b) The first cells contain some helper functions to load the training and test data. Complete the functionality to load the labels (y-data) : Two functions need to be implemented for transforming the label values to one-hot vectors and back.

c) Implement a first many-to-one model with a $SimpleRNN$ and a $softmax$ for the classification (by using keras).

— Train the model - try different numbers of hidden units (dimension of hidden state vector) and different batch sizes.

— Inspect the learning curve and spot potential issues.

Report about your findings in the pdf report.

d) Implement L2 regularisation for the $SimpleRNN$ and $softmax$ layer (see week 6 for the details ; learn how you can do that in Keras).
   — Train the model with regularisation (tune the hyper parameter λ, i.e. the weight given to the L2 regularisation term).
   — Describe whether you can observe qualitative changes in the learning curves.

   Report about your findings in the pdf report.

e) Implement gradient clipping for the model.
   — Again train the model. Choose a suitable form for how to clip and threshold values.
   — Inspect now the learning curves and describe qualitative changes.
   — Compute the confusion matrix and discuss which classes are more or less easy to identify.

f) Implement a model with two stacked $SimpleRNN$s and a $softmax$. Choose all the regularisation that you consider as important to achieve a good accuracy. Again report about it.

Exercice 2 Language Classification by Last Names

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

a) Download the jupyter notebook `name_classification_stud.ipynb`, `stud_names.csv`, the zip-file with the data (`names_by_country.zip`), place them in the same folder, unzip the file. Open 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 layer RNN with a $SimpleRNN$ and a $softmax$ for the classification (by using 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 pdf report.

d) Implement a two layer model with a $SimpleRNN$ and a $softmax$ for the classification (by using 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 pdf report.

Exercice 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.

```
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

Many-to-one approach

a) Read again the examples on the Shakespeare sonnets 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 sonnets.

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.

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.

Exercice 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 rather in long than in short sentences ?

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