Practical work 12 – 5/12/2019
Recurrent Neural Networks

Objectives

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

Submission

— **Deadline** : Monday 16 December, 10am

— **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 the jupyter notebook `name_classification_` and the data folder (`data`) 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 layer RNN with a *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) 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 notebook.

e) Explore whether there is a possible class-imbalance problem in the data. What would you do to compensate for that ? Apply it if applicable (i.e. if there is a class imbalance problem).

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).

You will 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 `activity_recognition_` and the data folder `UCI-HAR-Dataset`, 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 :
   - Model type : Based on *SimpleRNN*, *LSTM*, *GRU*, *Conv1d* + *Dense* (with softmax).
   - Number of layers : 1+
   - Different hyper parameters : batchsize, nepochs, number of hidden units
   - With/without regularization (dropout, recurrent dropout)

   Analyse at least 10 different settings (combinations of model/hyperparameters) - each model type should occur at least once. Describe quantitatively and qualitatively the resulting performances, the shape of the learning curves and the confusion statistics. Spot potential issues for specific settings / models. Report about your findings in the notebook.

d) Identify the best model suited for the given task. Report the results (test/train accuracy) for three independent runs.

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

Exercise 3 IMDb Sentiment Classification

Similarly to the previous exercise, you explore the hyperparameter/model space - here to identify the best model for classifying the sentiment in reviews from the IMDb database.

a) Download the zip-file `imdb_sentiment.zip` with the jupyter notebook `sentiment_imdb_stud.ipynb` and the data folder `imdb_data`, unzip the file, open the jupyter notebook.

b) The first cells contain some helper functions to load the training and test data. In this example, the preprocessing pipeline is very important and you will need to explore different settings in combination with the classification models. Study the different steps in detail and identify and describe the most important parameters that you can tune afterwards.

c) Now implement implement various different models (all are many-to-one) with different hyperparameter settings :
   - Model types : *SimpleRNN*, *LSTM*, *GRU*, *Conv1D* + *Dense* (with softmax) + *Embedding* (with/without finetuning)

— Number of layers : 1+

— Different hyper parameters : **Preprocessing parameters**.

— With/without regularization (dropout, recurrent dropout)

Analyse at least 10 different settings (combinations of model/hyperparameters) - each of the RNN model types should occur at least once. Describe quantitatively and qualitatively the resulting performances, the shape of the learning curves and the confusion statistics. Spot potential issues for specific settings / models. Report about your findings in the notebook.

d) Identify the best model suited for the given task. Report the results (test/train accuracy) for three independent runs.

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

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 rather in long than in short sentences ?

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

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

f) 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.

g) What is the difference between the word2vec embedding and the embedding learned on the IMDB corpus for positive/negative sentiment analysis.