CS5242 - Assigment03

Obtaining Filter Characteristics with Neural Networks

Deadline: 23.10.2018 @11:55pm

In this assignment, characteristics of smoothing filters employed in a DC motor control system will be obtained by using neural networks. Advantages of memory capabilities of recurrent neural networks over multi-layer perceptrons will be analyzed.

In Figure 1, block diagram of a typical closed-loop DC motor feedback control system is shown. In such a system, angular speed of the DC motor (orange box), which drives the load (purple box), is controlled by the speed controller (blue box) based on the error signal obtained as the difference between reference angular speed N_{ref} and smoothened angular speed N_{smooth}. Smoothing filter (red box) in the feeadback loop is responsible for removing the noise on the measurements of tachometer (brown box), which measures the instant angular speed of the DC motor. This filter gets noisy instant speed data $N_{noisy}(t)$, then filters out the noise and provides smoothened data $N_{smooth}(t)$ to error calculation. $N_{smooth}(t = t_0)$ is calculated by the filter over the set of last five noisy data points $\{N_{noisy}(t) \mid t = t_0, t_0 - 1, t_0 - 2, t_0 - 3, t_0 - 4\}$.

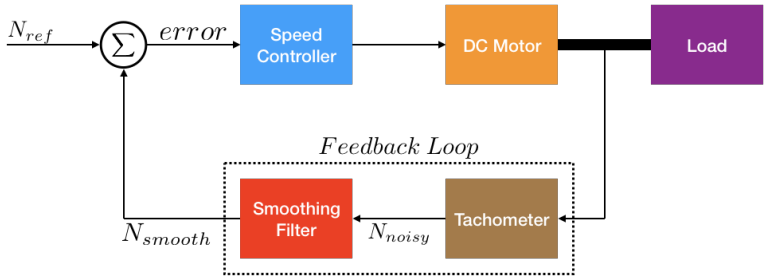

Figure 1: Block diagram of a typical closed-loop DC motor feedback control system.

Dataset: You are given 1000 noisy data points in input file 'noisy_data.txt' and corresponding smooth data points in output file 'smooth_data.txt' for 1000 time steps. Noisy data points within input file are normalized to interval of [-0.1,0.1]. These files are located under 'filter_data' folder. First 100 data points in noisy and smooth data files are plotted in Figure 2.

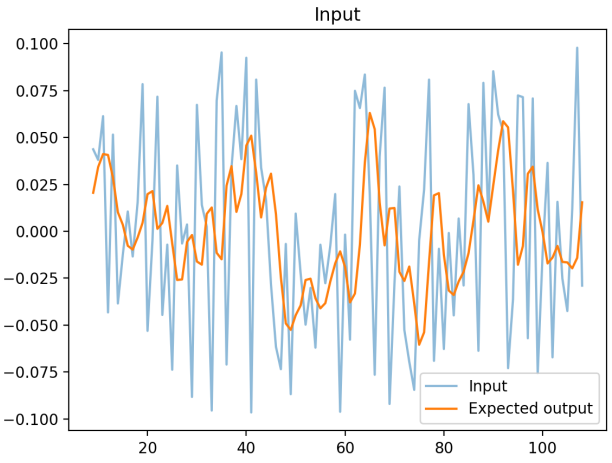

Figure 2: First 100 data points in noisy and smooth data files.

Aim: Our aim is trying to predict the smoothened data at filter output, $\hat{N}_{smooth}(t)$, for different length L input data sequences $\{N_{noisy}(t)|t = t, t-1, t-2, \cdots, t-(L-1)\}$ in different architectures.

In Figure 3, an example neural network model is shown for input length sequence of $L = 3$ at time $t = t_0$. Model gets $\{N_{noisy}(t_0), N_{noisy}(t_0 - 1), N_{noisy}(t_0 - 2)\}$ and predicts $\hat{N}_{smooth}(t_0)$ while the true value is $N_{smooth}(t_0)$.

!!! Please note that while preparing length L input sequences from the given original time series data, we are losing $L - 1$ data points in our dataset. In other words, our first input - true value pair is $\{N_{noisy}(2), N_{noisy}(1), N_{noisy}(0)\}$ and $N_{smooth}(2)$!!!

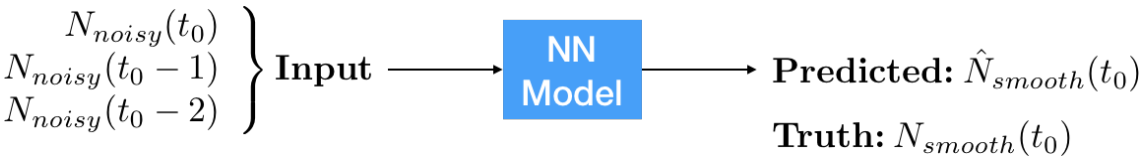

Figure 3: Example neural network model with input length sequence of three at time t_0.

1 Multi Layer Perceptron

In this part, we will try to predict filter output from different length input sequences by using multi layer perceptron architecture, i.e. fully-connected neural network. This arechitecture will have one input layer with input length L units, one hidden layer of 20 units with ReLU activation function and one output layer of 1 unit without activation, which is shown in Figure 4.

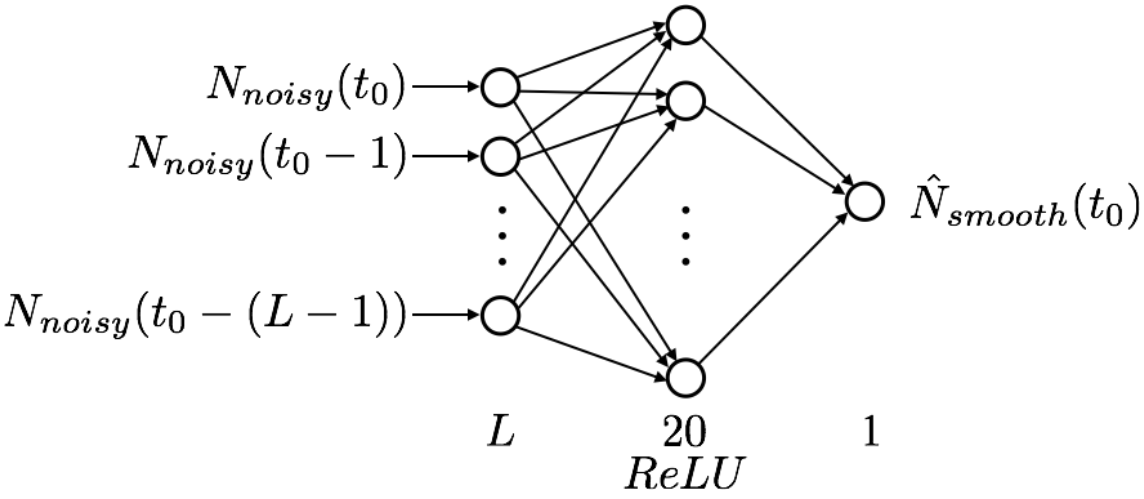

Figure 4: Fully connected neural network model with input length sequence of L at time t_0.

Jobs to be done:

- Train 10 model for $L = 1, 2, \cdots, 10$ length input sequences for 10 epochs with batch size 1. **!!! Do not shuffle data; otherwise you will lose time sequence info !!!**
- Save your model weights at the end of the training with filename 'fc_model_weights_length_L.h5' (e.g. 'fc_model_weights_length_1.h5' for L=1). You will use 'model.save_weights()' function of Keras, so please refer to Keras API.
- Plot and save your loss vs iteration curves for the training and test sets on the same graph and include them in your report in appendix (10 graphs in total in one page).

- Calculate root mean square error values for 10 models for $L = 1, 2, \cdots, 10$ length input sequences in test set at the end of training and save them with filename 'fc_model_rmse_values.txt'. There will be 10 values in this file with 10 rows and 1 column. You can use np.savetxt().

- Plot and save rmse vs length of input sequences graph and include it in your report in appendix. You will have one curve in this graph. Please include legend in your graph.

- Comment on the performance of different models in terms of memory properties of architectures. Especially, comment on the performances of $L = 1$, $L = 5$ and $L = 10$ models in your report. If you see anything interesting, please also comment on this.

2 Vanilla RNN

We will do the same thing by using Vanilla RNN architecture with 'stateful' and 'stateless' properties (please refer to SimpleRNN in Keras API). This arechitecture will have one SimpleRNN cell with 20 units with ReLU activation and different time steps of $L = 1, 2, \cdots, 10$, which is length of input sequences. Architecture is shown in Figure 5.

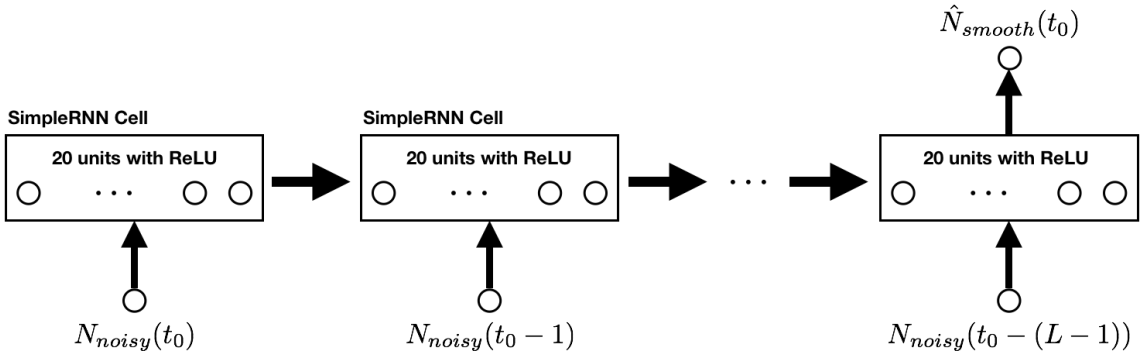

Figure 5: Vanilla recurrent neural network model with input length sequence of L at time t_0.

Jobs to be done:

- Train 10 models for $L = 1, 2, \cdots, 10$ length input sequences with 'stateful=True' for 10 epochs with batch size 1 and train 10 models for $L = 1, 2, \cdots, 10$ length input sequences with 'stateful=False' for 10 epochs with batch size 1. **!!! Do not shuffle data; otherwise you will lose time sequence info !!!**

- Save your model weights at the end of the training with filename 'rnn_stateful_model_weights_length_L.h5' (e.g. 'rnn_stateful_model_weights_length_1.h5' for L=1) and 'rnn_stateless_model_weights_length_L.h5' for stateful and stateless models, respectively. You will use 'model.save_weights()' function of Keras, so please refer to Keras API.

- Plot and save your loss vs iteration curves for the training and test sets on the same graph and include them in your report in appendix (10 graphs in total in one page for stateful and 10 graphs in total in one page for stateless).

- Calculate root mean square error values for 10 models for $L = 1, 2, \cdots, 10$ length input sequences in test set at the end of training and save them with filename 'rnn_stateful_model_rmse_values.txt' for stateful models and do the same thing for stateless models and save them with filename 'rnn_stateless_model_rmse_values.txt'. There will be 10 values in each file with 10 rows and 1 column. You can use np.savetxt().

- Plot and save rmse vs length of input sequences curves for stateful and stateless models on the same graph and include it in your report in appendix. You will have two curves in this graph. Please draw each curve with different color and include legend in your graph.
- Comment on the performance of different models in terms of memory properties of architectures. Especially, comment on the performances of $L = 1$, $L = 5$ and $L = 10$ models in your report. If you see anything interesting, please also comment on this.

3 LSTM

We will do the same thing by using LSTM architecture with 'stateful' and 'stateless' properties (please refer to LSTM in Keras API). This arechitecture will have one LSTM cell with 20 units and different time steps of $L = 1, 2, \cdots, 10$, which is length of input sequences. Architecture is shown in Figure 6.

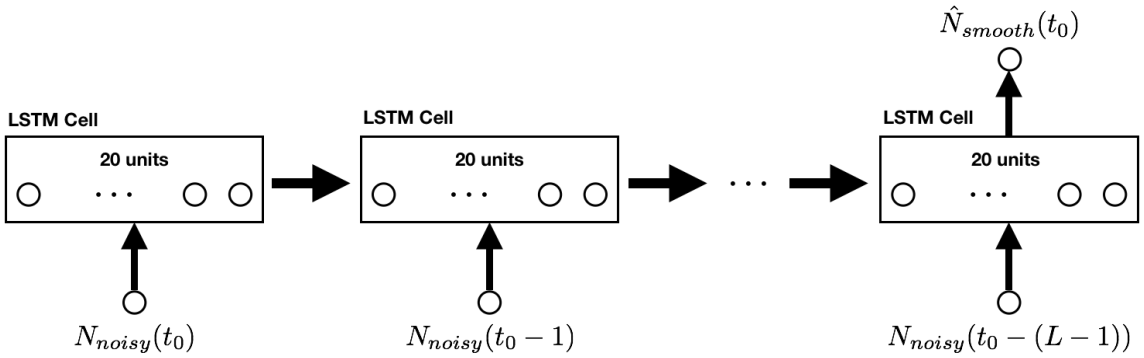

Figure 6: LSTM model with input length sequence of L at time t_0.

Jobs to be done:

- Train 10 models for $L = 1, 2, \cdots, 10$ length input sequences with 'stateful=True' for 10 epochs with batch size 1 and train 10 models for $L = 1, 2, \cdots, 10$ length input sequences with 'stateful=False' for 10 epochs with batch size 1. **!!! Do not shuffle data; otherwise you will lose time sequence info !!!**
- Save your model weights at the end of the training with filename 'lstm_stateful_model_weights_length_L.h5' (e.g. 'lstm_stateful_model_weights_length_1.h5' for L=1) and 'lstm_stateless_model_weights_length_L.h5' for stateful and stateless models, respectively. You will use 'model.save_weights()' function of Keras, so please refer to Keras API.
- Plot and save your loss vs iteration curves for the training and test sets on the same graph and include them in your report in appendix (10 graphs in total in one page for stateful and 10 graphs in total in one page for stateless).
- Calculate root mean square error values for 10 models for $L = 1, 2, \cdots, 10$ length input sequences in test set at the end of training and save them with filename 'lstm_stateful_model_rmse_values.txt' for stateful models and do the same thing for stateless models and save them with filename 'lstm_stateless_model_rmse_values.txt'. There will be 10 values in each file with 10 rows and 1 column. You can use np.savetxt().
- Plot and save rmse vs length of input sequences curves for stateful and stateless models on the same graph and include it in your report in appendix. You will have two curves in this graph. Please draw each curve with different color and include legend in your graph.

- Comment on the performance of different models in terms of memory properties of architectures. Especially, comment on the performances of $L = 1$, $L = 5$ and $L = 10$ models in your report. If you see anything interesting, please also comment on this.

4 Comparison of Different Architectures

In this section, you are given trained weights for the above models in 'trained_models' folder and required to obtain the test set root mean square error values in different models by using the given weights.

Jobs to be done:

- Create 10 models for $L = 1, 2, \cdots, 10$ length input sequences for each architecture (fc_model, rnn_stateful_model, rnn_stateless_model, lstm_stateful_model and lstm_stateless_model) and load the given weights. You will use 'model.load_weights()' function of Keras, so please refer to Keras API. Filename convention is the same with above convention, we have just added '_trained' at the end of file name (e.g. 'fc_model_weights_length_1_trained.h5' for L=1).

- Calculate root mean square error values for each architecture type for $L = 1, 2, \cdots, 10$ length input sequences in test set and save them with filename 'all_models_rmse_values.txt'. There will be 50 values in the file with 5 rows and 10 columns, each row corresponding to an architecture with the order of fc_model, rnn_stateful_model, rnn_stateless_model, lstm_stateful_model and lstm_stateless_model and each column corresponding to the length of input sequences $L = 1, 2, \cdots, 10$. You can use np.savetxt().

- Plot and save rmse vs length of input sequences curves for all architectures on the same graph and include it in your report in appendix. You will have five curves in this graph. Please draw each curve with different color and include legend in your graph.

- Comment on the performance of different architectures in terms of memory properties of architectures. Especially, comment on the performances of $L = 1$, $L = 5$ and $L = 10$ models of architectures in your report. If you see anything interesting, please also comment on this.

5 Given Data

You are given filter data, trained model weights and code templates in the main assigment folder. The directory structure of the main folder is shown in Figure 7.

6 Submission

You are required to upload a compressed data file into IVLE, whose name is your id (e.g. e0210516.zip). Directory structure of the main folder to be uploaded to IVLE is shown in Figure 8.

Your report must be inside the main folder and named with your id (e.g e0210516.pdf). Main part of the report can be at most one page and should consist of four parts (one for each section). You will give your graphs in the appendix of the report.

Your code, weight files and error file(s) for each section must be inside the corresponding folder.

!!! Your submission will be graded by an auto-grading program. All file and variable names are case-sensitive. Please strictly follow the file name format and directory structure. There will be 0 mark for corrupt submissions. !!!

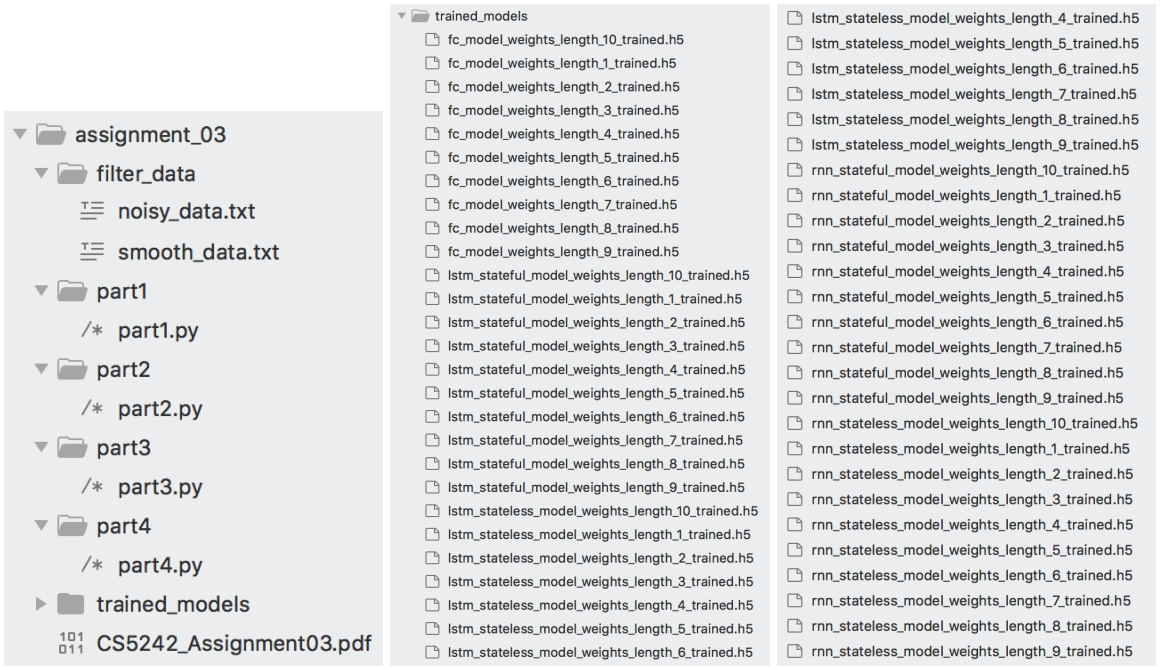

Figure 7: Given data folder structure.

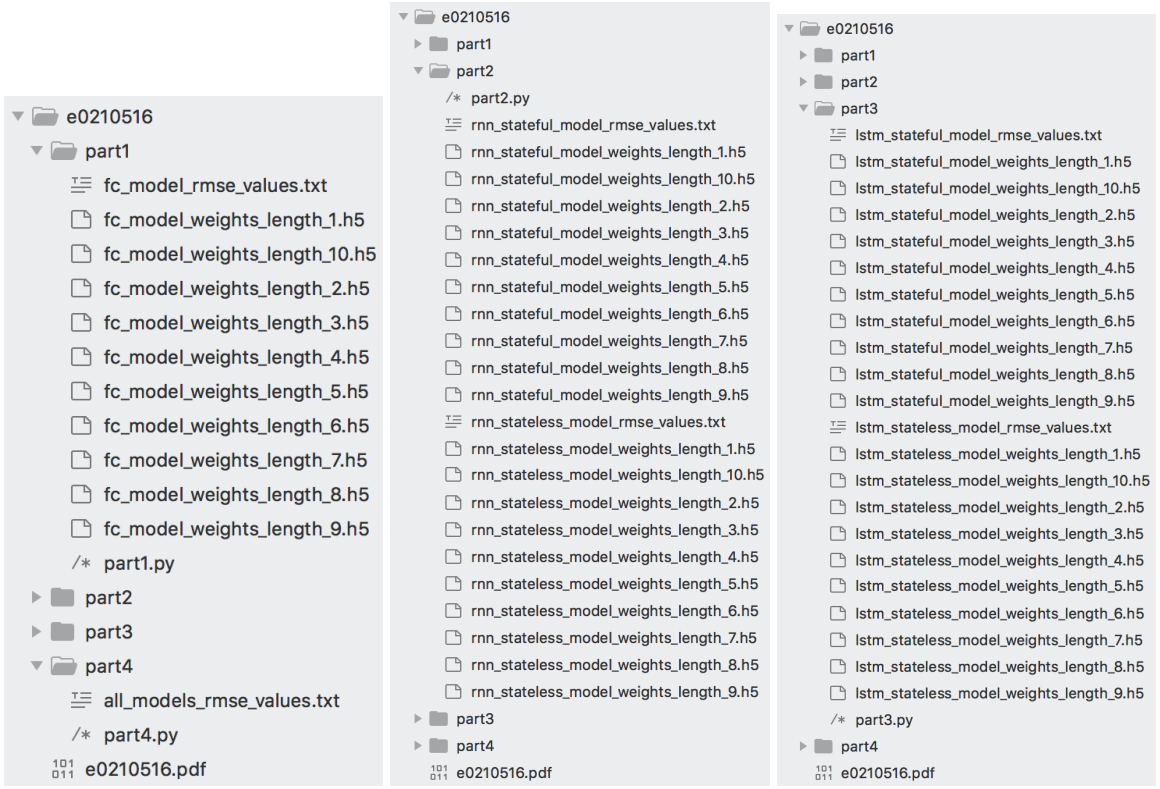

Figure 8: To be submitted folder structure.