System Identification Problem

Learning parameters of an unknown system using LMS algorithm

Project Report

**Author**: Mihir Juneja

**Mentor**: Hailan Shanbhag (ECE UIUC Undergraduate)

**Project Duration:** June 1 – July 6, 2020

# System Identification Problem

# Motivation

Having recently completed a course *Python for Research* offered by *Harvard University*, I was looking for a real life application of Python in engineering. I felt that this was a great opportunity for me, as it would test my Python skills and help me get an idea of applying Python in the field of Electrical & Electronics Engineering.

# Problem setting

The problem statement was to create an FIR filter which can be used to identify the parameters of an unknown system. The input **u** and expected output **y0**were given. Objective was to find the order and coefficients of the filter **H0** which gives the expected output **y0** for the input **u**. The adaptive filter *learns* the unknown coefficients by iterating over time index **n** using LMS learning algorithm. After each time index, error between expected and actual output is used to update the coefficients as shown in Figure 1 below. Difference equations for coefficients give the final converged weights.

|  |
| --- |
|  |

*Figure 1. Adaptive filter - Block diagram.*

# Approach followed

The Adaptive filter was implemented using Python code.

First step was to read given input data file in CSV format for filter input ‘***u’*** and expected output ‘**y0’**. This data had 800 samples.

Difference equations for the FIR filter were implemented in Python. The initial coefficients for the first time index were taken as zeros. The error signal was calculated by subtracting the desired output **y0** from the computed output **y1**. This error signal was used to generate coefficients for the next time index. Various values of MU were tried as well and 0.001 was finally chosen because higher values were giving exponentially high error values. This procedure was repeated for 800 time series samples.

## Ensemble Averages for Error (e) and Coefficients (h1)

The code was executed several times for different values of filter order **M** to arrive at a value which gave minimum mean square error (MSE). The code was also run with initial coefficients as random uniform distribution values. The plot for error obtained using initial random coefficients looked better as it approached zero faster but the plot for the initial coefficients as zeros was better due to the smaller absolute values of error.

The next step was to run the code for 50 \* 800 = 40,000 time series samples. The ensemble averages for **e** and **h1** were computed using the results of the 50 iterations. The ensemble average for MSE was further averaged for last 100 time samples and plotted with respect to order M. These are plotted in Figure 2 for orders 5 through 15. As can be seen, **MSE** decreases, reaches a minimum and rises again. An optimal value of **M** = 10 was chosen as the filter order which gave the least **MSE** = 8.93.

|  |
| --- |
|  |

*Figure 2. Ensemble Average for Error, time averaged for last 100 samples vs Order.*

In *both cases of initial coefficients*, Random or Zeros, *the best value found for the order was same* ie **M**=10. Plots for **M** = 10 is shown in Figure 3 for different initial coefficients. These plots show the peak values of error and how it approached zero for time series samples.

|  |
| --- |
|  |

*Figure 3. Ensemble Average for Error over 50 iterations.*

Ensemble averages for coefficients h1 were also computed for 50 iterations, similar to error.

## 3.2 Converged coefficients (h1)

To obtain the converged values for **h1**, these were ensemble averaged first and then time averaged over the last 100 samples. The converged values of **h1** for filter of order 10 were found to be

* *[-0.15  0.09  0.2  -0.6  -0.3   0.35 -0.4  -0.25  0.35  0.1 ]*

These values matched the **h0**values that were provided which were

* *[-0.15  0.1  0.2  -0.6  -0.3   0.35 -0.4  -0.25  0.35  0.1 ]*

|  |
| --- |
|  |

*Figure 4. Converged h1 values.*

Figure 4 above shows the plot for converged **h1** values.

These converged values were also the same as those from the last time sample i.e. **h1[k, 799]**

## 3.3 Verifying converged coefficients (h1)

To check, how good the converged **h1** weights were, graphs for **y0, y1** and error were plotted along with the error **e**. In this code written to verify results, h1 was fixed and not changing for time series samples.

Following three sets of data samples were run using the above h1 values.

* Data Samples - 0 to 799
* Data Samples - 19200 to 20000
* Data Samples - 39200 to 40000

# Final Results

## Order M = 10

* Converged h1 = *[-0.15  0.09  0.2  -0.6  -0.3   0.35 -0.4  -0.25  0.35  0.1 ]*
* Variance for h1 = 0. 10010756254472222
* MU = 0.001

The converged coefficients worked well for all three sets of data samples, with the **y1** and **y0**curves almost overlapping each other. The plots turned out to be pretty good.

Figure 5 shows these results for one of the above sets of data samples.

|  |
| --- |
|  |

*Figure 5. Verifying Filter outputs and error using Converged h1 values.*

# Conclusion

System Identification has been implemented in Python using LMS algorithm to arrive at filter order **M** and coefficients **h1**. In Machine learning terminology, these correspond to parameters (coefficients h1) and hyper-parameter M (filter order) of the model (FIR filter). Arrived values for **h1** work well for any of the provided data samples with **y1** and **y0** almost overlapping.

I would like to express my thanks to Hailan Shanbhag for her valuable and constructive suggestions during the project. She has been a great mentor. I also wish to sincerely thank Prof. Naresh Shanbhag for providing an opportunity to work on this project and also for reviewing my results and providing feedback.