Tutorial IV: Sequence-to-Sequence Regression

Acknowledgment

1. This tutorial was prepared by Braden Stevens (bcsteven@uark.edu) at the University of Arkansas.
2. The dataset and algorithms used in this tutorial are from *C. Dunlap, H. Pandey, E. Weems, and H. Hu, "Nonintrusive Heat Flux Quantification Using Acoustic Emissions During Pool Boiling," Appl. Therm. Eng., 228, (2023) 120558. [link] and H. Hu and C. Heo, "Integration of Data Science into Thermal-Fluids Engineering Education", in Proceedings of the ASME 2022 International Mechanical Engineering Congress and Exposition, Oct – Nov 2022, Columbus, OH, IMECE2022-88193. [link]*
3. This tutorial was implemented in the MEEG tech elective "Machine Learning for Mechanical Engineers." [link]

1) Introduction

1a) Problem Definition

The purpose of this report is to demonstrate the use of a Long Short Term Memory (LSTM) regression model in forecasting future data given past data. The raw data given is comprised of 50,962 data points representing the vapor fraction of boiling images sequences sampled at a frequency of 3,000 Hz. The following two objectives are required for this assignment.

A. Develop a baseline model with an input sequence length of 16.33 ms (50 data points) and an output sequence length of 16.33 ms (50 data points). Plot the model-predicted signal vs. the true signal.
B. Vary the input and output sequence lengths to evaluate their effect on the error of the model predictions.

1b) Regression and Self-Regression

Consider the following scenario: a car company is trying to determine when to increase or decrease its manufacturing capabilities to more closely align with market trends. The benefit of this is that car companies will have extra stock to provide for customers when demand is high but won't need to have extra cars sitting on the lot during low demand times. This is a simple problem to solve using a concept called **regression**. In statistics, regression is the measure of the relation between the mean value of the dependent variable and the corresponding values of various independent variables. In the car analogy, the dependent variable is the volume of cars being manufactured. The independent variables would be external factors such as fuel price, seasonality, how late in the month it is, unemployment rates, or any other factors that one considers when buying a car. By using regression, an engineer can predict with reasonable accuracy how many cars should be produced depending on various factors.

In the above example, the assumption is that a wide range of variables are known and available; however, this luxury is not always available to engineers. It is not uncommon for data to be incomplete, imperfect, or even missing. This makes regression quite difficult to perform, but another model is available that drastically reduces the complexity and requirement of large amounts of data: **Self-Regression**. Similar to regression, self-regression is a tool used in statistics to predict dependent variables. However, unlike regression, self-regression can be used to predict future values based on only past values. Continuing with the car example, a manufacturer might not care why an uptick in vehicles sold is seen every year in December, but they would still want to know when that uptick occurs so that volume could be increased in November.

Self-regression is a useful tool for engineers by predicting future states on past data. Building HVAC controls can utilize self-regression to pre-condition spaces prior to an extremely cold weather event. Signals engineers can predict noise and interference patterns which allow for more accurate filtering. Electrical engineers can reduce electrical outages by forecasting energy demand increases based on past events. By utilizing a simple self-regression model, we can more effectively build systems that can withstand a wide range of external forces.

In this tutorial, we will be using self-regression to predict future vapor fractions found during boiling using only past values. This will be demonstrated through the prediction of a **sequence output** given a **sequence input**. Once the sequence output is predicted, we will then investigate the effect that different sequence lengths have on model accuracy.

2) Methodology

2a) Mathematical Theory

The architecture implemented in an LSTM model is what makes it unique and allows it to keep or forget data from earlier in a sequence. The key components of an LSTM model are:

A. Cell State (C_t): This can be considered the "memory" of the cell. This is the information that is remembered across time steps and allows the model to remember information from the past if necessary. In simple terms, this is what tracks the overall trend of the data. A real-world example of this is the stock market as shown in *Figure 1*. There might be an occasional downward trend,

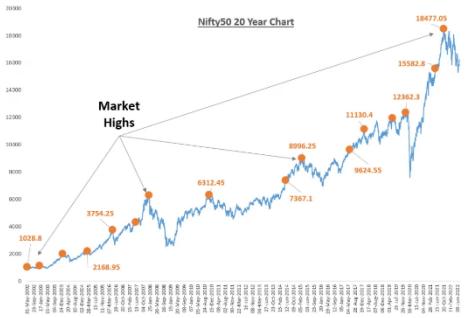


Figure 1: A graph of the stock market trend showing multiple valleys but an overall positive trend upward.[1]

but the cell state allows the model to remember the overall upward trend. This is considered the "long-term" in a long short term memory model.

B. Hidden State (H_t): Conversely, the hidden state can be thought of as the short-term memory of the model. While the stock market might be trending higher, a stockbroker might be interested in the next valley in order to buy more stock, or when the next peak would be in order to sell. The hidden state is what tracks the recent trends in data.

C. Gates: The gates are how the information flows through the cell state. This is where a neural network is implemented to regulate information flow. There are three types of gates.
   1. Forget Gate (f_t) – Decides which information from the previous cell state to remove.
   2. Input Gate (i_t)– Decides what new information should be added.
   3. Output Gate (o_t)– Decides which part of the cell state to output as a hidden state.

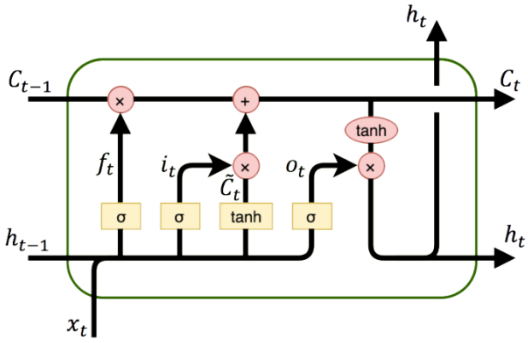


Figure 2: A generic LSTM cell depicting multiple gates. Each gate is implemented using a neuron or layer of neurons.[2]

Figure 2 shows a generic cell state found in a layer of an LSTM model. An LSTM layer can be composed of multiple cells that are calculated using neurons. Unlike a linear regression model, the LSTM model uses cells instead of individual neurons as the primary calculation unit.

An LSTM cell works using the following equations. First, the model decides what information to forget from previous cell states. *Equation 1* shows how the forget gate is calculated using a sigmoid function (σ), the previous hidden state (h_{t-1}), current input (x_t), and the weights and biases of the forget gate (W_f and b_f, respectively).

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \qquad Equation\ 1$$

The LSTM then decides which new information to store in the cell state by implementing new weights and biases (W_i and b_i) (*Equation 2)*.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \qquad Equation\ 2$$

Once the input gate and forget gate are calculated by the LSTM, it creates a new vector of potential new values for the cell state (*Equation 3*). This new vector, when combined with the previous cell state, produces an updated cell state cell state (*Equation 4*).

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \qquad Equation\ 3$$

$$C_t = f_t * C_{t-1} + i_t + \tilde{C}_t \qquad Equation\ 4$$

Finally, the LSTM decides the output for the current time step (*Equation 5*). The hidden state is also calculated from this and passed on (*Equation 6*) to the next cell state.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_C) \qquad Equation\ 5$$

$$h_t = o_t * \tanh(C_t) \qquad Equation\ 6$$

2b) Key Concepts

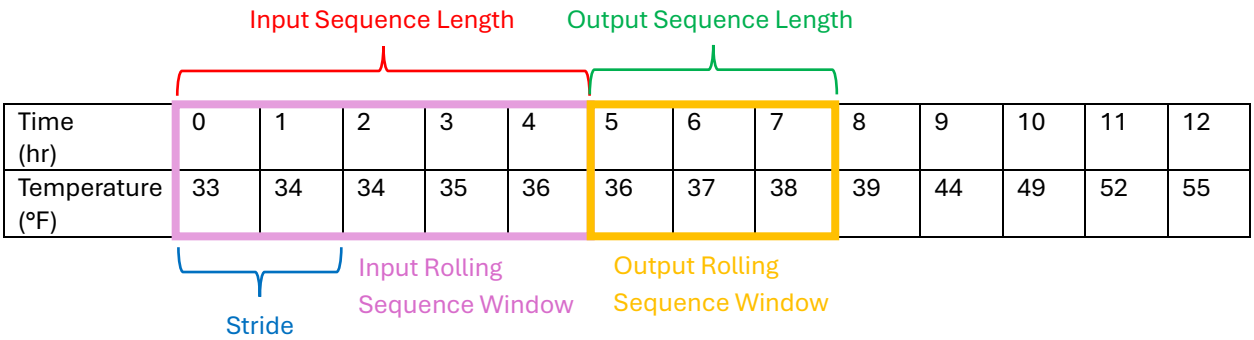
In order fully utilize self-regression, we must first understand the underlying key concepts that allow for prediction of target variables. The following section will discuss **rolling sequence windows**, **sequence length**, and **stride**. These concepts are important to understand, as they help to contextualize the forecast as a function of historical values. Without correctly utilizing these inputs, a predictive model might not capture essential patterns; for example, in a scenario where hourly temperature data is used to predict future temperatures, an improperly set sequence length could either include irrelevant noise leading to inaccurate forecasts, or on the opposite end of the spectrum omit important information such as seasonal trends.

To visualize these concepts, let's look at a simple hourly dataset of outdoor dry bulb temperature found in *Table 1*. We need to break this down into a **rolling sequence window**. This is what will build our sub-datasets. Think of a rolling sequence window as a filter that is passed over the data to split it into smaller groupings.

Time (hr)	0	1	2	3	4	5	6	7	8	9	10	11	12
Temperature (°F)	33	34	34	35	36	36	37	38	39	44	49	52	55

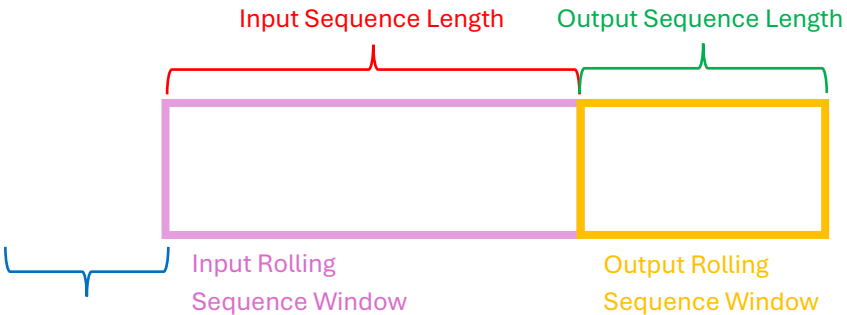
Table 1: The predicted temperature in Springdale, AR starting at 12:00 A.M. according to Weather.com.

However, to build the rolling sequence window, we need to know which parameters are important. This is where **sequence length** and **stride** come in. **Sequence length** is defined by the number of elements found within a sub-dataset. **Stride** is the movement of the rolling sequence window across the dataset; or, more simply, how many datapoints are skipped between each sub-dataset. Let's revisit *Table 1* again. A rolling window has been created using an **input sequence length** (red) of 5, **output sequence length** (green) of 3, and a **stride** (blue) of 2. The below image shows what the rolling window would look like for the first input sequence in the dataset.



$$x_1 = [33, 34, 34, 35, 36], \qquad y_1 = [36, 37, 38]$$

After the first array is built, the rolling window is moved by a number of datapoints (equal to the stride) to build the next datapoint.



Time (hr)	0	1	2	3	4	5	6	7	8	9	10	11	12
Temperature (°F)	33	34	34	35	36	36	37	38	39	44	49	52	55

$$x_2 = [34, 35, 36, 36, 37], \qquad y_2 = [38, 39, 44]$$

These windows are then passed over the entirety of the dataset to build the variables that will then be passed to the LSTM model for prediction. This builds the input array, *x*, and output array, *y*. The purpose of using the LSTM model is to predict the output sequence of any chosen input sequence. By changing these parameters, we can adjust the usefulness of our model in self-regression and prediction.

2c) Code Layout and Explanation

The code utilized in this assignment utilizes PyCharm with the Tensorflow Keras, MatPlotLib, Scikit Learn, and Numpy libraries. Section 1 of the code is a generic import section that imports all relevant libraries. Section 2 consists of a section to import the data and section building/training the LSTM model. Section 3 is comprised of 2 subroutines designed to analyze the data and plot as required per the assignment. Section 3 is discussed further in the results section.

Code Section 2a

The data was imported from a text file titled "vapor_fraction_Boiling-81_110W.txt". To reduce computational time, only the first 6000 values in the file were used.

Code Section 2b

To prepare the data, the vapor fractions found in the text file were placed in a numpy array. The data was looped through to create an array containing groups of 50 data points. Once the array was created and the data grouped, it then was passed to the Train_Test_Split function to get an Input Training set, Input Validation set, Input test set, Output Training set, Output Validation set, and Output Test set.

Code Section 2c

The LSTM model was built using 4 layers: two LSTM layers consisting of 32 neurons each, a dense layer with 64 neurons, and an output layer that output the data into groupings of 50. An early stop function was implemented to avoid overfitting.

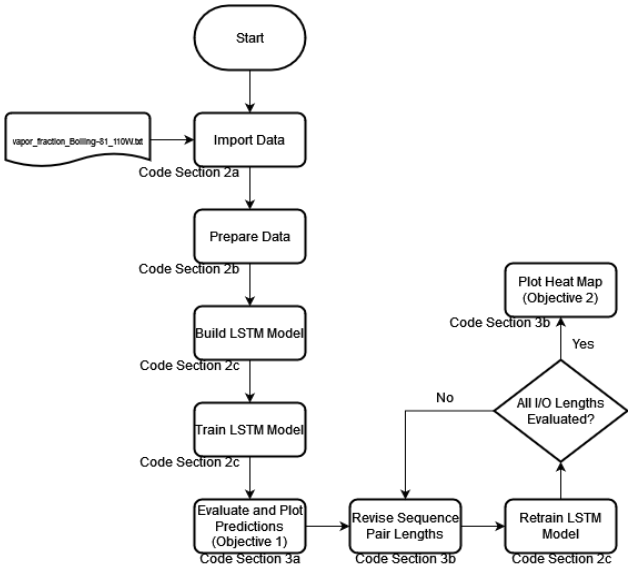


Figure 3: A flow chart of the program used to analyze the data.

The model was trained on the Input_Train and Output_Train arrays and validated on the Input_Validation and Output_Validation arrays.

3) Results

3a) Objective 1

An arbitrary sequence pair from the Input_Test set, as well as the corresponding Output_Test were chosen to compare predicted values. As seen in *Figure 4*, both the predicted and true output values trend down. The large gap in predicted vs. true values implies a poorly trained model; however, this cannot be considered a bad result in the context of this assignment. Sequence2Sequence regression requires a much more detailed model to have a reasonable error. A model needs to be optimized for the dataset used, whereas this assignment only requires a prediction. Future work could focus on optimizing the model for lower error in predicted values.

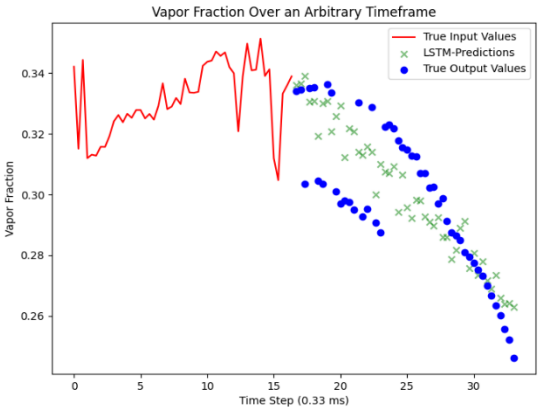


Figure 4: A plot of the vapor fraction over an arbitrarily chosen timeframe. Each data group shows 50 datapoints, corresponding to a total sampling time of 32.66 seconds.

3b) Objective 2

Once all other objectives were met, the LSTM model was re-trained using varying input sequence lengths and output sequence lengths. *Figure 5* shows a heat map of the mean squared error of the different combinations of input/output sequence lengths. As shown in the figure, the error increases drastically as the output length increases. However, the input sequence length seems to have little effect on the error. Future iterations of this homework would see a larger input/output length to determine at what point increasing these lengths provides diminishing returns. The major challenge in this section was running the original code a stride of 1 and with 64 neurons per layer. This led to a total training time of about 15 hours.

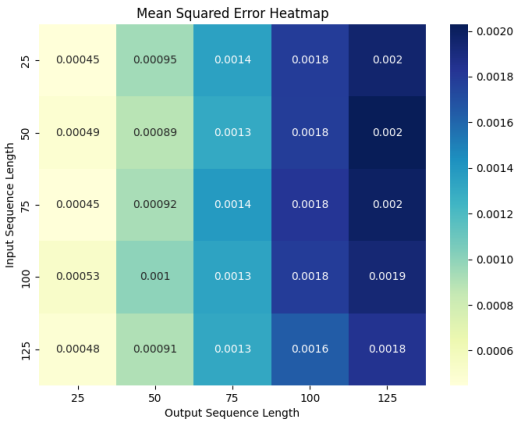


Figure 5: Heat map of mean squared error at varying input/output sequence lengths.

References

[1] Mani. *Nifty50 20 Year Stock Price Chart*. 9 Aug. 2022. *Getmoneyrich*, https://getmoneyrich.com/timing-the-market-vs-time-in-the-market/.

[2] Ruan, Xiaofei, et al. “Predicting the life of varistors via a nonlinear coefficient based on a small-scale data model.” *Applied Sciences*, vol. 14, no. 1, 24 Dec. 2023, p. 171, https://doi.org/10.3390/app14010171.