ECE 4750/6750 + CS 4501/6501: Digital Signal Processing **Due Tu 4/29**

Homework 5: Time Series Prediction with LTI Systems and Neural Networks

Instructions: Submit a single Jupyter notebook (.ipynb) of your work to Canvas by 11:59pm on the due date. All code should be written in Python. **Be sure to show all the work involved in deriving your answers! If you just give a final answer without explanation, you may not receive credit for that question.**

You may discuss the concepts with your classmates, but write up the answers entirely on your own. Do not look at another student's answers, do not use answers from the internet or other sources, and do not show your answers to anyone. **Cite any sources you used outside of the class material (webpages, etc.), and list any fellow students with whom you discussed the homework concepts.**

1. Download the data `cho_weather.csv`, which contains hourly weather at Charlottesville Albemarle Airport (CHO) since 2019 [1]. The columns are:

`tmpf`	Temperature (F)
`dwpf`	Dew Point (F)
`relh`	Relative Humidity (%)
`drct`	Wind Direction (deg)
`sped`	Wind Speed (MPH)
`mslp`	Sea Level Pressure (mb)
`p01i`	Precipitation (in)

   Clean the data by filling in missing values with the previous valid value in that column. Missing values are marked with an 'M'. The letter 'T' indicates a trace amount of precipitation. Set these values to 0.

2. Let's use LTI systems to predict the weather. Take the exponential moving average (EMA) system that we discussed in class:

$$y[n] = (1-g)x[n] + gy[n-1].$$

   Given a time series $x[n]$ for $n = 0, \ldots, L-1$, we can use this as a prediction model for the next timepoint $x[n+1]$ (that our model has not seen yet) by taking the last output, $y[n]$, as our prediction for $x[n+1]$. Implement this exponential averaging system and test it on the provided data of hourly temperatures. Do the following:

   (a) Make predictions for $x[n]$ (using $y[n-1]$) for $n = 1, \ldots, L-1$. Plot these predictions over a plot of the original data. Do this four times, with gain parameters $g = 0.0, 0.25, 0.5, 0.75$. What difference do you see with the four different parameters?

   (b) Compute the mean absolute error (MAE) of the four different models ($g = 0.0, 0.25, 0.5, 0.75$). The MAE is

$$MAE = \frac{1}{L-1} \sum_{n=1}^{L-1} |x[n] - y[n-1]|.$$

   Which choice of g gave the best prediction (lowest MAE)?

[1] Weather data downloaded from `https://mesonet.agron.iastate.edu/request/download.phtml`.

(c) Because this is hourly data, our model is really only predicting one hour into the future, which isn't so hard! Let's predict further into the future. Repeat your exponential moving average model with a delay of 24 (instead of 1). Now predict $x[n+24]$ using $y[n]$. Plot the predictions over the original data again for $g = 0.0, 0.25, 0.5, 0.75$. Report the MAE for each model.

(d) Now try running your EMA system with a delay of 8 hours to predict 8 hours ahead. You don't need to plot, just report the MAE for the same four gain values. What do you notice about how this model performs compared to predicting a full 24 hours ahead? Give a conjecture for why you think the performance difference is they way it is.

Neural Networks

At the end of this exercise, you'll have several trained models: 3 linear and 1 RNN (possibly one more if you do extra credit problem #8). For each model, save it as a `*.pt` file using the provided code. **You will turn in these models with your code.** At the end of the assignment report the test accuracy (MAE) for all of the competing models. Write a paragraph about what you learned in terms of which models perform the best, what are the challenges in getting them to work, and what are the pros and cons of each model.

3. Now let's train neural networks to predict the weather. Throughout this part, we will reserve the last year of data to use as test data to evaluate the performance of our models. The last year is 365days $\times$ 24 hrs / day = 8760 time points. First, take your best-performing EMA models above and evaluate their performance (MAE) on the test data. (You may want to write a function that runs a model on the test data and outputs its MAE because you'll need to repeat this for several models.)

4. Train a single-layer, linear neural network prediction model (`LinearPrediction` in the provided code). Use at least one full year of timepoints for training data (make sure your training data is separate from the test data!). Experiment with the window size (how many time points to incorporate in the prediction), the learning rate, and the number of epochs to train. You should be able to choose parameters such that the training runs in a reasonably short amount of time and the final result performs better than EMA on the test data.

5. Train two more `LinearPrediction` models, now with the targets being 8 hours (`offset = 7`) and 24 hours (`offset = 23`) in the future. You might start with the same parameters you used in the previous problem and adjust them only if they don't work well.

6. Now train an recurrent neural network (RNN) model to predict the temperature one hour into the future. Now, you will need to experiment with the hidden variable size, the learning rate, the window size, and the number of epochs. Start small with the hidden variable size and window size, and increase them if the computation time is not too long. **Note:** It is okay if your final RNN doesn't perform as well as the previous models—this can be hard to tune well—but it should at least work reasonably well!

Grad Students Only (Extra Credit for Undergrads)

7. Write a function to use your RNN to predict farther into the future by recursively feeding the outputs y_n as the next inputs x_{n+1}. Use this to predict 8 hours into the future and compare

the performance with the EMA and linear neural network models. Repeat this for 24 hours into the future.

Extra Credit for Everybody!

8. Train an improved neural network such that you can beat the performance of EMA and the linear neural network in predicting 24 hours into the future. You might consider training the RNN to directly predict targets 24 hours ahead, modifying the RNN to be an LSTM, using the other weather variables (wind, pressure, etc.), or other ideas. This is open ended!