CS/ECE 528: Embedded Systems and Machine Learning
Fall 2021

Homework/Lab 5: Time Series and Anomaly Prediction

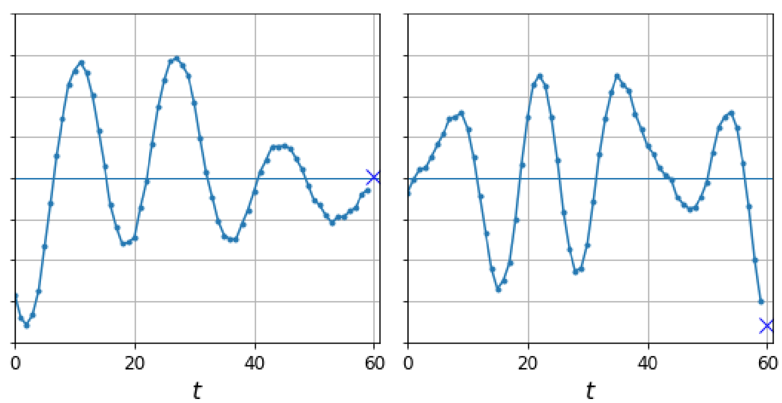
Assigned: 9 November 2021
Due: 16 November 2021

Instructions:

- Submit your solutions via Canvas.
- Submissions should include your jupyter notebooks in a zip file, with notebooks names q1.ipynb, q2.ipynb, etc. in a single folder. You can include comments in your notebooks to explain your design choices.
- **"Save and checkpoint" your notebook after running your notebook, so that cell outputs are preserved.** If you are using Colab, make sure to 'Save' your notebook after running it, before downloading it and submitting.

Q1. (150 points) Consider the problem of forecasting a time series (e.g., stock price, temperature readings). We will consider the case of a time series with a single value per time step, which is called a univariate time series. A typical task is to predict future values, which is called forecasting.

(a) [50 points] In this question you will forecast a single value for time series data. The ***notebook univariate-time-series-1-step.ipynb*** contains code that loads a time series dataset with 12,000 time series, each with 60 steps, and the goal here is to forecast the value at the next time step for each of them. The figure below shows two of the time series, with X representing the forecasted value at the 61st time step.



The 12,000 time series are split up into 9000 training, 2000 validation, and 1000 test time series. The notebook contains a simple linear regression model that achieves a mean square error (MSE) of 0.00142 on the test set. Design an RNN-based model with the lowest possible MSE for the test dataset that outperforms this model. Your score will depend on the MSE value achieved by your model.

(b) [50 points] In this question you will forecast multiple values for time series data. The notebook ***univariate-time-series-multi-step.ipynb*** contains code that is similar to that in (a) except that now the time series must be predicted for 10 steps instead of just 1 step in the future. The notebook contains a simple linear regression model that achieves a mean square error (MSE) of 0.0454 on the test set. Design an RNN-based model with the lowest possible MSE for the test dataset that outperforms this model. Your score will depend on the MSE value achieved by your model.

(c) [50 points] Instead of training the model to forecast the next 10 values only at the very last time step as we did in (b), we can train it to forecast the next 10 values at each and every time step. In other words, we can turn this sequence-to-vector RNN into a sequence-to-sequence RNN. The advantage of this technique is that the loss will contain a term for the output of the RNN at each and every time step, not just the output at the last time step. This means there will be many more error gradients flowing through the model, and they will also flow from the output of each time step. This will both stabilize and speed up training. To be clear, at time step 0 the model will output a vector containing the forecasts for time steps 1 to 10, then at time step 1 the model will forecast time steps 2 to 11, and so on. So each target must be a sequence of the same length as the input sequence, containing a 10-dimensional vector at each step.

To turn the model into a sequence-to-sequence model, we must set return_sequences=True in all recurrent layers (even the last one), and we must apply the output Dense layer at every time step. Keras offers a TimeDistributed layer for this very purpose: it wraps any layer (e.g., a Dense layer) and applies it at every time step of its input sequence. It does this efficiently, by reshaping the inputs so that each time step is treated as a separate instance (i.e., it reshapes the inputs from [batch size, time steps, input dimensions] to [batch size × time steps, input dimensions]; then it runs the Dense layer, and finally it reshapes the outputs back to sequences (i.e., it reshapes the outputs from [batch size × time steps, output dimensions] to [batch size, time steps, output dimensions]. Another change that needs to be made is related to the loss metric. While all outputs are needed during training, only the output at the last time step is useful for predictions and for evaluation. So although we will rely on the MSE over all the outputs for training, we will use a custom metric for evaluation, to only compute the MSE over the output at the last time step

The notebook ***univariate-time-series-multi-step-enhanced.ipynb*** contains code that is similar to that in (b) except for the variations discussed in this question. The notebook contains a simple linear regression model that achieves a last time step mean square error (MSE) of 0.096 on the test set. Note the use of the TimeDistributed layer in the model and the calculation of the last step MSE. Design a RNN-based model with the lowest possible last step MSE for the test dataset that outperforms this model. Your score will depend on the MSE value achieved by your model.

Q2. (50 points) Sometimes it is required to predict anomalies that occur in time series data. RNNs can be used to predict anomalies in such time series data. In this question, you will predict anomalies in a time series data of an IoT device utilization. When an anomalous observation occurs in such a time series, there will likely be a large deviation between the predicted and observed series, because an anomalous event is often difficult to model. An example of metric that can capture the extent of this deviation is a root mean square error (RMSE) plot between predictions and observations. A large enough RMSE can be used to signal when anomalies occur. The dataset for this question is in the file iot-util.csv. Starting from the notebook ***time-series.ipynb***, fill out the missing code to create an RNN model for predicting the time series, and detect anomalies in that series. Note that if your model is poor, you will end up with more false positives, as well as false negatives. Note also that the threshold specified for anomaly detection in the notebook has been empirically determined. Show your RMSE plot generated from the notebook as well as the times at which your model detects the anomalies.

Q3. (50 points) In this question, you will perform anomaly detection for IoT network intrusion detection. The dataset we will use can be found here: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html You will need to download kddcup.data.gz which has the full data set (18M; 743M Uncompressed). Starting from the notebook ***isolation-forest.ipynb***, fill out the missing code to create and use an Isolation Forest classifier on the dataset, for anomaly detection. Your score will depend on the highest AUC-ROC value achieved by your model on the test set (you can read up on AUC-ROC here: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc).