Project #3 Modeling Time Series Data with RNN

Due Apr. 13 at 6pm

Electronic Health Records (EHRs) contain a wealth of patient medical information that can improve the overall quality of care a patient receives when seeking medical assistance. For example, the record data can be mined to yield early warning signs of patients that may require extra care or an indication of the severity of a patient's illness. In this project, we will use deep learning techniques to predict patient mortality at any time of interest. Such prediction can be used to provide essential feedback to medical professionals when trying to assess the impact of treatment decisions.

The EHR dataset used here consists of patient data in a Pediatric Intensive Care Unit (PICU) at Children's Hospital Los Angeles. This dataset contains a collection of patient encounters with a set of variables measured or observed at different times during each encounter. It is important to note that time between measurements can vary from minutes to hours. In addition to non-uniform sampling, not all measurements were taken for all patients at each encounter. Measurements include:

- **Statics** *(e.g. gender, age, weight)*
- **Vitals** *(e.g. heart rate, respiratory rate)*
- **Labs** *(e.g. glucose, creatinine)*
- **Interventions** *(e.g. intubation, O2)*
- **Drugs** *(e.g. dopamine, epinephrine)*

The label (y) for each patient encounter is the ultimate result of alive (1) or not alive (0).

The provided data set includes two files of training data, X_train.hdf and Y_train.hdf, and two testing data files, X_test.hdf and Y_test.hdf. Both training and testing data sets are stored in HDF5 format, which stands for hierarchical data format version number 5. The HDF format is designed specifically to store and organize large amounts of scientific data because of its flexibility, efficiency, and compatibility with different operating systems. However, it is important to note that HDF is a binary format and hence lacks the human readable transparency of text based CSV files. Therefore, we use pandas, an open-source library of data management tools for the Python programming language, to access data in HDF format. Check out the pandas documentation for more info.

In this project, we will use the python library pandas to manage the dataset provided in HDF5 format and deep learning library Keras to build and train different types of recurrent neural networks (RNN). Finally, we will compare the performances.

Part 1. Data preparation

1.1 Load HDF5 data using Pandas API. The data imported by pandas is a multi-index dataframe where index level 0 is the unique patient encounter identifier and index level 1 is the time of each measurement in units of hours since first measurement.

1.2 Find how many patient encounters are there in the training set; how many timestamps are measured in total; and how many types of measurements are documented.

1.3 For each patient encounter, find how many times measurements have taken, and plot a histogram showing the number of times each patient encounter is measured.

1.4 Perform data normalization for all numerical measurements.

1.5 For each patient encounter, fill in missing data point using forward filling, i.e., use the last measurement for all missing measurement till the next time stamp when another measurement is made. If there was not a previous measurement, fill in missing point(s) by zeros.

1.6 Set the maximum length of sequence for each patient encounter to the 90 percentile of length. Truncate or zero-pad to the same maximum length for all patient encounter sequences.

Part 2 RNN Model Selection

2.1) A base-line LSTM model

a Build a single hidden layer LSTM model with 128 neurons that will take a sequence of data as input, and make a binary prediction on weather a patient can survive.

b. Train the model using the training set with a 80/20 training/validation split. Plot the training v.s. validation loss, as well as training v.s. validation accuracy curves.

c. Evaluate your model using the testing set. Find the sensitivity and precision for your model.

d. Plot the ROC curve of your model and compute the AUC for your model using sklearn.metrics functions.

2.2) Change the architecture of the LSTM model, such as using different nnmber of neurons, and adding more hidden layers, repeat part b-d in 2.1), and find the best performing model.

2.3) Change the LSTM models to GRU models, and repeat part 2.2), compare the results.