Prediction of Remaining Useful Life (RUL) of Batteries using ODE-RNNs

Shubham Mate and Nandanavanam VS Sai Saketh

Department of Mathematical Science, IIT (BHU), Varanasi

mate.shubhamvijay.mat23@itbhu.ac.in,
nvssai.saketh.mat23@itbhu.ac.in

November 2024

1 Objective

To predict the Remaining Useful Life (RUL) of a battery by giving 20 observations which consists of the various features of the batteries using a neural ODE without using the capacity (mAH) feature which is used in traditional methods. The model was to be trained on the given dataset.

1.1 Dataset

The dataset to be used is the Battery Remaining Useful Life Prediction dataset on Kaggle. The Hawaii Natural Energy Institute examined 14 NMC-LCO 18650 batteries with a nominal capacity of 2.8 Ah, which were cycled over 1000 times at 25°C with a CC-CV charge rate of C/2 rate and discharge rate of 1.5C. This data released by Hawaii Natural Energy Institute was used to make this dataset. It contains the following features:

Cycle Index The Cycle number of current cycle

Discharge Time (s) time that takes the voltage to reach its minimum value in one discharge cycle.

Decrement 3.6-3.4V (s) It represents the time taken for voltage to drop from 3.6V to 3.4V during a discharge cycle.

Max. Voltage Dischar. (V) The initial and maximum voltage in the discharging phase.

Min. Voltage Charg. (V) The initial value of voltage when charging.

Time at 4.15V (s) The time to reach 4.15V in charging phase.

Time constant current (s) The time in which the current stays constant at its max. value.

Charging time (s) The total time for charging.

RUL The Remaining Useful Life of battery after every cycle.

2 Mathematical Formulation and Explaination

In this section, we'll look at and recap the mathematical formulation of the various techniques and components used in the model

2.1 Neural ODE

Numerous Models build complicated transformations by composing a sequence of transformations to the input or a maintained hidden state. These transformations are mathematically expressed as:

$$\mathbf{z}_{t+1} = \mathbf{z}_t + f(\mathbf{z}_t, \theta_t)$$

where $t \in \{0, \ldots, T\}$ and $h_t \in \mathbb{R}$. These iterative updates can be seen as an Euler discretization of a continuous transformation. Neural ODE aims to parameterize the continuous dynamics of z_t given by the equation:

$$\frac{d\mathbf{z}(t)}{dt} = f(\mathbf{z}(t), t, \theta)$$

The output of Neural ODE is generated with a black box differential equation solver. Therefore, the $\mathbf{z}(T)$ can be obtained by:

$$\mathbf{z}(T) = \mathbf{z}(t_0) + \int_{t_0}^{T} \mathbf{f}(\mathbf{z}(t), t; \theta)\, dt = \text{ODESolve}(\mathbf{z}(t_0), f, t_0, T, \theta)$$

In order to avoid backpropogating through the black box differential solver, Neural ODE uses adjoint sensitivity method to calculate required gradients. At every

point, it maintains an adjoint state $\mathbf{a}(t)$ whose dynamics are given by another ODE:

$$\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t)^T \frac{\partial \mathbf{f}(\mathbf{z}(t), t; \theta)}{\partial \mathbf{z}(t)}$$

Another ODE solver is used to solve the dynamics of $\mathbf{a}(t)$. The Gradient $\frac{d\mathbf{L}}{d\theta}$ is finally computed by the equation:

$$\frac{d\mathcal{L}}{d\theta} = \int_0^T \mathbf{a}(t)^T \frac{\partial \mathbf{f}(\mathbf{z}(t), t; \theta)}{\partial \theta} dt$$

We use Neural ODEs to model the dynamics of the hidden state of a RNN.

2.2 Neural ODE based GRU

RNN (Recurrent Neural Network) is a type of neural network which maintains a hidden state $\mathbf{h}_t$ which gets updated each time a new input is fed into it. This allows it to remember information from previous inputs. This makes them suitable to model sequential/time-series data in which current input may be dependant on previous inputs.
GRU (Gated Recurrent Unit) is a variation of RNNs where each unit has an update gate (z_t) and reset gate (r_t) which allows them to select what information should be maintained by hidden states and what information should be forgotten. The equations for the GRU are:

$$z_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1} + \mathbf{b}_z) \tag{1}$$

$$r_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1} + \mathbf{b}_r) \tag{2}$$

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_h \mathbf{x}_t + \mathbf{U}_h (r_t \odot \mathbf{h}_{t-1}) + \mathbf{b}_h) \tag{3}$$

$$\mathbf{h}_t = (1 - z_t) \odot \mathbf{h}_{t-1} + z_t \odot \tilde{\mathbf{h}}_t \tag{4}$$

To incorporate a Neural ODE to update the hidden state, we give the following method: Suppose we have two inputs $\mathbf{x}_{t_1}$ and $\mathbf{x}_{t_2}$. Let us define a function UpdateGRU($\mathbf{x}$, $\mathbf{h}_t$) which takes an input and current hidden state and updates the states according to the equations given above. We also define a function f (a neural network) with parameters θ We use the following method to get the output:

$$\mathbf{h}_{t_1} = \mathbf{UpdateGRU}(\mathbf{x}_{t_1}, \mathbf{h}_{t_0})$$

$$\mathbf{h}_{t_1} = \mathbf{ODESolve}(\mathbf{h}_{t_1}, f, t_1, t_2, \theta)$$

$$\mathbf{h}_{t_2} = \mathbf{UpdateGRU}(\mathbf{x}_{t_2}, \mathbf{h}_{t_1})$$

This helps us model the continuous dynamics of the hidden state of GRU in between the updates, which can help us even model time series data with irregular time interval.