Artificial Neural Network for RUL Prediction of Lithium Ion Batteries

Abstract

Accurate equipment remaining useful life prediction is critical to effective condition based maintenance for improving reliability and reducing overall maintenance cost by noticing the batteries' life periodically. An artificial neural network (ANN) based method is developed for achieving more accurate remaining useful life prediction of Lithium Ion batteries subject to condition monitoring. The ANN model takes the capacity attribute as a target against multiple measurement values as the inputs, and the life expectancy as the output. The proposed method is validated with the data provided by Prognostics Center of Excellence at Ames Research Center, NASA. A comparative study is performed between the proposed ANN method and other methods, and the results demonstrate the advantage of the proposed method in achieving more accurate remaining useful life prediction.

Introduction

In equipments running on Lithium Ion batteries, condition monitoring data, such as number of discharges and charges, time related data, and many other factors are collected and processed to determine the battery's health condition; Future health condition and thus the remaining useful life (RUL) of the battery is predicted; and optimal maintenance actions can thus be scheduled based on the predicted future battery health condition, so that preventive replacements can be performed to prevent unexpected failures and minimize total maintenance costs. Accurate remaining life prediction is the critical to effective implementation of condition based maintenance pertaining to the batteries' capability and chemical combination. Existing battery health condition and RUL prediction methods can be roughly classified into model-based (or chemistry/physics-based) methods and data-driven methods. The model-based methods predict the remaining useful life using damage propagation models based on damage mechanics. If properly used, such models can greatly improve the RUL prediction accuracy. However, discharge-charge processes are typically very complex, and authentic physics based models are difficult to build for many components and systems. Data-driven methods utilize collected condition monitoring data for RUL prediction. Artificial neural networks (ANNs) have been considered to be very promising tools for battery health condition and RUL prediction due to their adaptability, nonlinearity, and arbitrary function approximation ability. Neural network methods do not assume the analytical model of the degradation propagation, but aim at modeling the degradation process, based on the collected condition monitoring data using neural networks and perform battery's health condition prediction.

In this paper, we propose an ANN based method for achieving accurate RUL prediction of Lithium Ion batteries. The ANN model takes the capacity and multiple condition monitoring measurement values (Voltage, Current, Temperature, Cycle) at discrete timed inspection points as the inputs and the life percentage as the output. The prediction accuracy is improved mainly by careful implementation of necessary activation functions that are relevant to the batteries' degradation in the condition monitoring data, and by utilizing the validation mechanism in the ANN training process.

The proposed ANN method is presented in section "The proposed ANN remaining useful life prediction method". Section "Case study" discusses the case study, in which the proposed ANN method is validated using NASA dataset, and a comparative study is performed. The final section presents the conclusions.

The proposed ANN remaining useful life prediction method

In this section, we first present a modified ANN method based on various activation functions, which can be used to deal with condition monitoring data collected at discrete inspection time points.
Our main activation function used in the model is ReLU. Activation functions in general are used to convert linear outputs of a neuron into nonlinear outputs, ensuring that a neural network can learn nonlinear behavior. Rectified Linear Unit (ReLU) does so by outputting x for all x >= 0 and 0 for all x < 0. In other words, it equals max(x, 0). This simplicity makes it more difficult than the Sigmoid activation function and the Tangens hyperbolicus (Tanh) activation function, which use more difficult formulas and are computationally more expensive. In addition, ReLU is not sensitive to vanishing gradients, whereas the other two are, slowing down learning in your network.

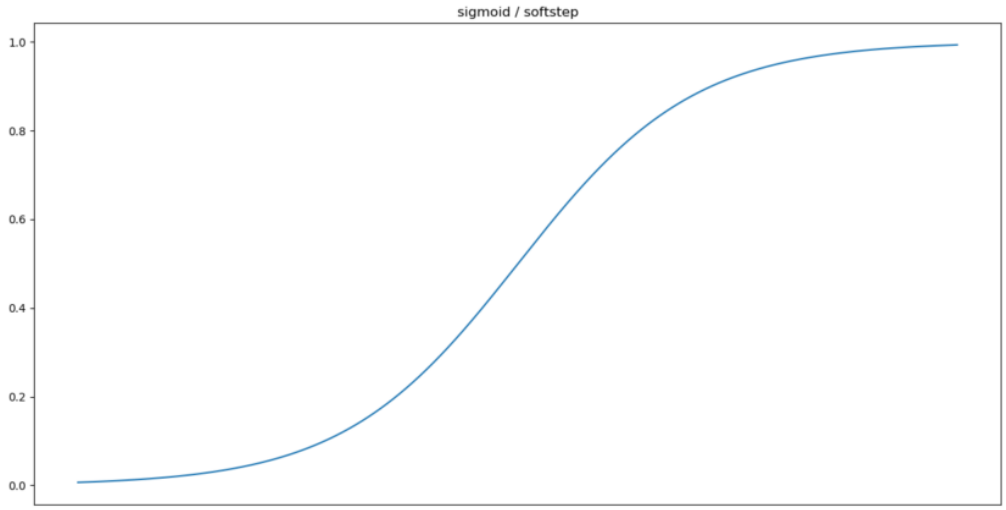
Activation Functions used

In any network, every neuron is composed of a weights vector and a bias value. When a new vector is input, it computes the dot product between the weights and the input vector, adds the bias value and outputs the scalar value. However in many cases, it does not. Because put very simply: both the dot product and the scalar additions are linear operations.
Hence, when we have this value as neuron output and do this for every neuron, we have a system that behaves linearly. Keeping in mind, most data that we possess is highly nonlinear. Since linear neural networks would not be capable of generating a decision boundary in our case, there would be no point in applying them when generating predictive models. The system as a whole must therefore be nonlinear.

The function, which is placed directly behind every neuron, takes as input the linear neuron output and generates a nonlinear output based on it, often deterministically. In such a way, with every neuron generating in effect a linear-but-nonlinear output, the system behaves nonlinearly as well and by consequence becomes capable of handling our data. Activation outputs increase with input

Sigmoid

In the following figure, a sigmoid function is presented, also known as the logistic curve:



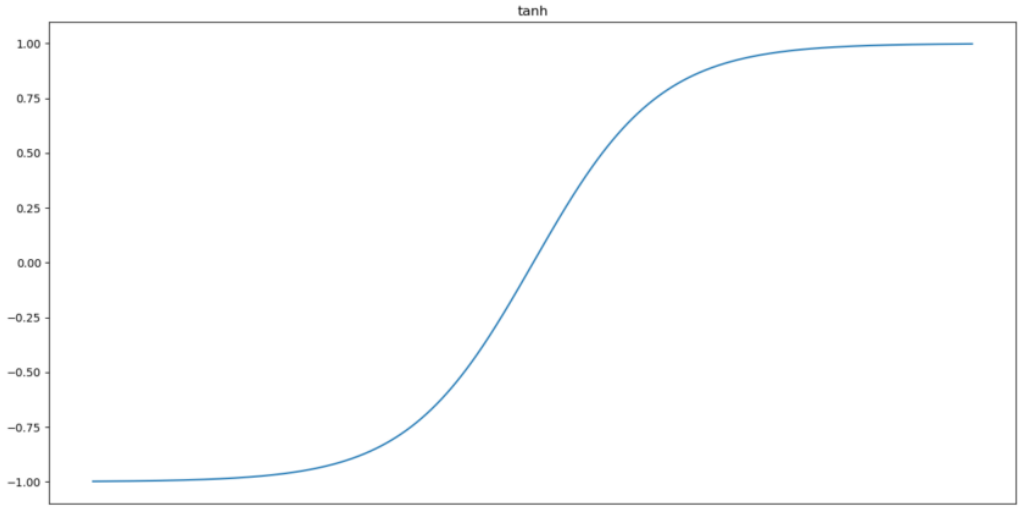
Mathematically, it can be represented as follows:

y: f(x) = 1 /(1 + e^{-x})

As seen in the plot, the function slowly increases over time, but the greatest increase can be found around x = 0. The range of the function is (0, 1); that is towards high values for x the function therefore approaches 1, but never equals it.
The Sigmoid function's step functions used in ancient neurons are not differentiable and hence gradient descent for optimization cannot be applied. Second, when we implemented the Rosenblatt perceptron, we noticed that in a binary classification problem, the decision boundary is optimized per neuron and will find one of the possible boundaries if they exist. This gets easier with the Sigmoid function, since it is more smooth.
Additionally, and perhaps primarily, we use the Sigmoid function because it outputs between (0, 1). When estimating a probability such as remaining life of Lithium Ion batteries, this is perfect, because probabilities have a very similar range of [0, 1]. Sigmoid functions allow us to give a very weighted estimate.

Tangens hyperbolicus: Tanh

Another applied activation function is the tangens hyperbolicus, or hyperbolic tangent / tanh function:



It works similar to the Sigmoid function, but is implemented regarding some differences.
First, the change in output accelerates close to x = 0, which is similar to the Sigmoid function.
It does also share its asymptotic properties with Sigmoid: although for very large values of x the function approaches 1, it never actually equals it.
The range of the activation function differs: (-1, 1).
Although this difference seems to be very small, it showcases a large effect on the model performance. This is related to the fact that they are symmetric around the origin. Hence, they produce outputs that are close to zero. Outputs close to zero are preferable as we initially deduct our X value prior to optimization, they produce the least weight swings, and hence let our model converge faster.

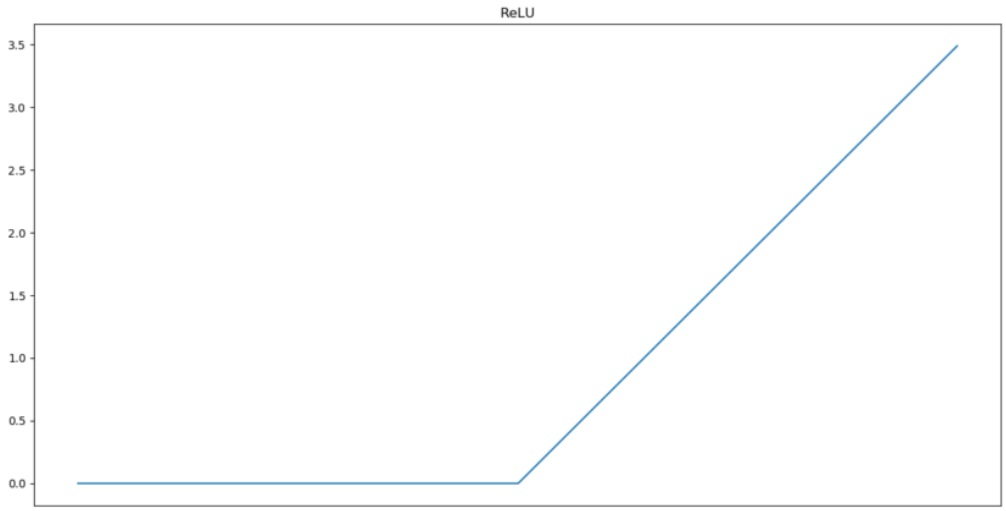
Challenges of Sigmoid and Tanh

Model's complexity can be viewed as the number of unimportant neurons that are still in our model. The fewer of them, the better - or sparser - the model.
Sigmoid and Tanh essentially produce non-sparse models because their neurons pretty much always produce an output value: when the ranges are (0, 1) and (-1, 1), respectively, the output either cannot be zero or is zero with very low probability.
Hence, if certain neurons are less important in terms of their weights, they cannot be 'removed', and the model is not sparse.
Another possible issue with the output ranges of those activation functions is the so-called vanishing gradients problem. During optimization, data is fed through the model, after which the outcomes are compared with the actual target values. This produces the loss. Since the loss

can be considered to be an (optimizable) mathematical function in our case as the mean squared error, we can compute the gradient towards the zero derivative, in other words, the mathematical optimum.
When outputs are very small, the chains produced during optimization will get smaller and smaller towards the upstream layers. This will cause the model to learn very slowly, and make it questionable whether they will converge to their optimum at all, giving rise to the vanishing gradients problem.

Rectified Linear Unit: ReLU

In order to improve on our observations after applying tanh and sigmoid, another activation was used. Rectified Linear Unit or ReLU is much less sensitive to the problems faced in above activation functions and hence improved our training process.
It looks as follows:



And can be represented as follows:
f(x) = 0 for x < 0, x for {otherwise}
Or, in layman terms, it produces a zero output for all inputs smaller than zero; and x for all other inputs. Hence, for all inputs <= 0, it produces zero outputs.

- Sparsity: This benefits sparsity substantially: in almost half the cases, now, the neuron doesn't fire anymore.
- Fewer vanishing gradients: It also reduces the impact of vanishing gradients, because the gradient is always a constant: the derivative of f(x) = 0 is 0 while the derivative of f(x) = x is 1. Model hence learns faster and more evenly.
- Computational requirements: Additionally, ReLU does need much fewer computational resources than the Sigmoid and Tanh functions. The function that essentially needs to

be executed to arrive at ReLU is a max function: max(0, x) produces 0 when x < 0 and x when x >= 0.

With the data's Capacity as target, we take into account the cycle and measurement values as well as the changes of the measurement values at hidden layers' points, and this important information will be processed by ANN to estimate the remaining useful life. We only use particular data pertaining to 5 attributes from each mat file instead of incorporating data in ever mat file, because ANNs with less input nodes have better generalization capability, and the experiments results in this work show that adding more input nodes will not improve the RUL prediction performance.
It is worth justifying the use of the remaining life to represent the health condition of a specified battery. However, it is true that the health condition of a battery deteriorates with time as well as discharge cycles, and, without loss of generality, we can assume that the true inherent health condition index increases monotonically with time and cycles. Since it is hard to determine the true inherent health condition index based on the collected condition monitoring measurements, we can try to identify a measure that has a monotonic mapping relationship with the true inherent health condition index. We believe "remaining life percentage" is a good option for this purpose: (1) the mapping between the inherent health condition index and the life percentage is monotonically non-decreasing, and (2) remaining life is also able to indicate when the degradation occurs. The remaining life prediction problem is a complex nonlinear problem. The objective is to predict the battery's remaining useful life based on the available metrics and condition monitoring data. The relationship between the 5 inputs, and the output, the life percentage, is very complex and nonlinear. Because of the capability in modeling complex non-linear relationships, ANN is a powerful tool to be used to address the remaining useful life prediction problem in the proposed approach.

Training of proposed ANN Model

A critical issue of using ANN is to avoid overfitting the network. If an ANN is overfitted, noise factors will be modeled in the network, which affects the generalization capability of ANN, and thus affects the prediction accuracy. A widely used approach for avoiding overfitting is the use of a validation set. That is, during the ANN training process, the mean square error for the training set and that for the validation set are calculated. Both of the mean square errors drop early in the training process because the ANN is learning the relationship between the inputs and the outputs by modifying the trainable weights based on the training set. After a certain point, the mean square error for the validation set will start to increase, because the ANN starts to model the noise in the training set. Thus, the training process can be stopped at this point, and the trained ANN with good modeling and generalization capability can be achieved. Another question is how to construct the validation set. One method is dividing the available failure histories into two groups, one used as the training set and the other as the validation set. The percentage of histories used in the validation set is typically around 40%. This causes a problem if we do not have a lot of failure histories, which is the case in many practical applications.

In this paper, we propose to construct the validation set using the actual measurement data from the collective dataset, and use the remaining fitted measurement data to construct the training set. In this way, data based on all attributes can be taken advantage of for modifying the trainable weights in the ANN training process, and the validation process can help to avoid overfitting the network. Experiments based on practical condition monitoring data have shown that the use of such a validation set in the training process can lead to more accurate and robust predictions.
Because of the uncertainty in the ANN training algorithms such as the LM algorithm, with the same training set and validation set, typically we will not obtain the same neural network after training. In this work, we train the ANN to obtain the lowest training mean square error (MSE). This is actually another advantage of using the validation mechanism in the training process. Without the validation process, a lower training MSE does not necessarily mean a better network, and in many cases it is not. Thus, we cannot select the best trained ANN based on the training MSE if we do not have the validation mechanism. However, if we do have the validation mechanism, as in the proposed ANN method, a lower training MSE will indicate a trained ANN with better modeling and generalization capabilities, and thus an ANN with better prediction capability.

Procedure of the proposed ANN method

The explanation of the procedure is given as follows. Step 0: We start from the available failure history data, which includes the age values and actual condition monitoring measurement values at inspection points for each failure history.

Step 1: Each measurement data for a battery is fitted in a collective dataset. The capacity values and the fitted measurement values of various mat datasets are used to construct the ANN training set.
Step 2: The ANN validation set is constructed using the collective dataset and the actual measurement values as inspection points are the Capacity values.
Step 3: Train the ANN model based on the training set and the validation set using the sequential model. This Step 3 gives the trained ANN model desired obtained for RUL prediction.
Step 4: At every iteration, we first fit each measurement series based on the measurement values up to the current range. The fitted measurement values as well as the target values in these two sets, are used as the inputs to the trained ANN model.
Step 5: The predicted remaining useful life from the current set of values is calculated using the trained ANN model.
Step 6: The RUL is calculated based on the current capacity of the battery and the measured attributes.
Step 7: For every epoch, repeat Step 4 to Step 6 based on the available data, and determine the RUL prediction.

RUL prediction using the proposed ANN method

The collective data from NASA mentioned above are used to test the RUL prediction performance of the proposed batteries subjected to condition monitoring regarding various aspects in the ANN method. The following approach is used to test the RUL prediction performance of the proposed ANN method. When we apply the ANN prediction method in practical situations, we first use the available data, to train the ANN, and then use the trained ANN model to predict the RUL of a specific observation in the test dataset which then compares the capacity against the prediction. Thus, we use X_test and y_test sets to construct the ANN validation set to test the prediction performance, and use the remaining data to construct the ANN training set and the validation set to train the ANN model. After that, we can use a different slice approach to construct the ANN test set to test the prediction performance and use the remaining data to train the ANN. We can repeat this process until we go through all the well performed approaches and select a prime range. Using the approach described above, the test data is not involved in the ANN training process, and we can make full use of the data available to test the prediction performance of the ANN method as many times as possible to achieve accurate prediction performance evaluation.

The ANN model we use has four hidden layers with two hidden neurons in the first layer and one neuron in the second output layer. From our experiments, such ANN configuration is found to be able to produce better prediction results compared to other ANN configurations with different numbers of hidden neurons.

The prediction performance of the trained ANN is evaluated using the test set constructed based on the remaining data. We test the prediction performance starting every epoch, since measurement values at every point are needed to fit the measurement series and generate the fitted measurement values used as the inputs to the trained ANN model. The predicted remaining life values are obtained, and are compared with the actual life values, which are calculated based on the attribute values in the given dataset.

Conclusion

The proposed ANN method is validated using the condition monitoring data collected by Prognostics Center of Excellence at Ames Research Center, NASA. Experiment results show that the proposed ANN method can produce satisfactory RUL prediction results, which will assist the condition based maintenance optimization.