Capstone Project Vikram Haravu

Udacity Machine Learning Engineer Nanodegree Feb 2018

# Definition

Project Overview

Rechargeable batteries are a very important component of several daily-use gadgets and EV automobiles too. Demand for Lithium batteries has grown tremendously in the last decade and is expected to continue growing in the foreseeable future. As a result, electronics suppliers and OEMs invest hugely on Battery Management Systems (BMS). Central to the BMS is the function of estimating the State of Charge (SOC) of a given battery. This battery parameter called SoC is directly related to the energy stored inside the battery at any given point of time.

In this project, I developed a Machine Learning model to predict the SoC of a rechargeable battery. It is a regression model since SoC lies between 0.0 (empty battery) and 1.0 (fully charged).

Dedicated and expensive hardware is designed into mobile device ICs to perform the SoC estimation function. I explore an ML method to solve this problem. This project was inspired by this IEEE publication (add link).

Problem Statement

The following strategy is followed:

1. Generate data using hardware (RTL) simulations. This is a large task by itself. Briefly, a SystemVerilog testbench is built around SoC estimation hardware. The testbench provides stimulus and observes the functioning of the hardware and writes out data files in CSV format.
2. Normalize and Scale the data – since the features/target are different scales of magnitude and units, it is good practice to normalize and standardize them for optimal algorithm performance.
3. Develop a simple benchmark model, to further assess the problem.
4. Develop a model that will better model non-linearities, as measured by metric scores.

For the project, I will address one phase of the BMS/SoC problem – namely, making accurate prediction of the battery’s “*first SoC*” – when a device is powered-on via a battery for the first time. The continuous estimation of SoC during normal operation of a device is critically dependent on accurate estimation of the first SoC. That is because, in the event the first SoC is inaccurate, continuous SoC estimation cannot converge to the correct SoC.

Metrics

Two metrics will be used in this project – R2 Coefficient of Determination, and MSE Mean Squared Error.

R2 is the square of the correlation (r) between predicted scores and actual scores. It ranges from 0.0 to 1.0. A model with an R2=0.0 (worst score) is a model that always predicts the mean of the target variable. On the other hand, a model with an R2=1.0 (best score) is one that perfectly predicts the target, in other words y\_pred = y\_test.

MSE gives an indication of how close the predicted values are to the real values. MSE is somewhat subjective (data range dependent) in the sense that it doesn’t lie in a fixed range like R2; a smaller MSE is obviously better. Given the mean of the target will be close to 0.0, and the range is approximately -1.0 to +1.0, an MSE of close to 0.0 (i.e., farthest away from 1.0) is a good value to achieve.

Keras models and optimizers will fundamentally attempt to minimize a loss function. MSE is chosen as that loss function in this project.

# Analysis

Data Exploration

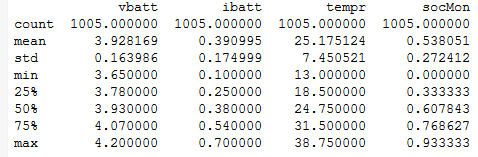
There are only a few proxies/features that are directly measurable and can be thought of as good indicators of the energy stored inside a rechargeable battery. These features are the voltage measured at the battery terminals (vbatt, volts), the current drawn from (or supplied to) a battery (ibatt, amps) and the operating temperature (tempr, Celsius).

**Fig. 1:** These three features and the target variable (socMon) are generated using hardware simulations.

|  |  |
| --- | --- |
|  |  |

The data collected from simulations is shown above. The raw data (CSV file format) is on the left. The raw data needs some minor post processing (such as hex to decimal conversion, divide by 255, etc) to result in actual v/i/tempr/SoC values shown on the right. I chose to not do this raw to actual conversions in the SystemVerilog simulation domain, because it is much simpler to do so with Python.

**Fig. 2:** Here are statistical characteristics of the features and target:



A note on outliers: The simulations to collect data were run under a controlled set of input stimulus. For instance, the hardware module under simulation is spec’d to operate to a max temperature of 39C. The specific battery being acted upon (for SoC est) is rated between 3.6V to 4.2V. These specifications were adhered to when developing simulation stimulus. So, I chose not to remove outliers.

Exploratory Visualization

**Fig. 3:** In the figure below, I have plotted each feature on individual x-axes with the target socMon on y-axis. One can see the strong or subtle influence of each feature on the target. It also shows the direct or inverse relation between each feature and target.

|  |
| --- |
|  |

**Fig. 4:** In the plots below, a subset of the data is used. All data-points which satisfy the following condition are removed: (ibatt < 0.3 & ibatt > 0.5). It is seen that while the relation with vbatt and tempr are similar to the full-data plots, the relation with ibatt has become even more subtle. This gives an indication of the nature of non-linearity inherent to the SoC estimation problem.

|  |
| --- |
|  |

**Note:** the subset used for this plot is only for initial exploratory purposes and not for model development.

Algorithms and Techniques

A Deep Learning model is proposed for this problem. It will be a Fully Connected MLP type. Given the size of the dataset and that MLPs are known to work well for regression problems, it is decided to go with this type.

The following hyperparameters can be tuned during the search for the best model:

* Epochs
* Batch size – the number of data-points to use for one feedforward-backprop step in each epoch. Example: Say X\_train=1000, and batch\_size=10, there will be 1000/10=100 ff-bp iterations (of 10 samples) per epoch to arrive at the weights/bias set. For a given problem, it is an experimental (hyperparam search) process to arrive at an optimal batch\_size.
* Optimizer – Stochastic Gradient Descent (sgd)
* Learning rate and Decay
* Momentum – to avoid local minima

Tuning of the Fully Connected network architecture:

* A wide vs. deep network
* Dropout – to avoid overfitting
* Weights initialization
* Activation: relu will be used because it is a continuous function; and given we are solving a regression problem, a continuous activation function is appropriate.

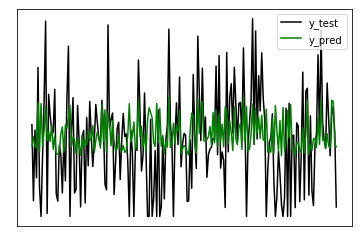
Benchmark

A linear model that uses the most correlated feature from Fig. 3 and Fig. 4 will be developed to model the target. The voltage at the terminals of a battery is the best indicator of the energy stored inside the battery. Scikit’s linear\_model.LinearRegression() was used for the benchmark.

This model resulted in the following metrics-

* + MSE = 0.961 and
  + R2 = 0.066

**Fig. 5** Benchmark model performance visualization



**Note 1:** Pre-processed data was used to develop the benchmark. Details about pre-processing are discussed next.

**Note 2:** Comparison of the benchmark and best model is discussed in the Results section.

# Methodology

Data Preprocessing

Implementation

Refinement

# Results

Model Evaluation and Validation

Justification

# Conclusion

Free-Form Visualization

Reflection

Improvement