Generalization of Electric Vehicles' Battery Cell Models with Machine Learning

YUE FANG
Generalization of Electric Vehicles’ Battery Cell Models with Machine Learning

Work conducted at Volvo Car Corporation

YUE FANG

School of Electrical Engineering and Computer Science
Information and Network Engineering
KTH Royal Institute of Technology
Stockholm, Sweden 2019
Nomenclature

Ah Ampere hour

ANN Artificial Neural Network

BMS Battery Management System

DST Dynamic Stress Test

ECM Equivalent Circuit Model

ELM Extreme Learning Machines

EV Electric Vehicles

ICE Internal Combustion Engine

Li – ion Lithium Ion

LSTMRNN Long Short-Term Memory Recurrent Neural Network

MC Monte Carlo

MLP Multilayer Perceptron
\textit{MSE} Mean Squared Error

\textit{NMC} Nickel Manganese Cobalt oxide

\textit{NN} Neural Network

\textit{OCV} Open Circuit Voltage

\textit{RMSprop} Resilient Mean Square Back-propagation

\textit{RPT} Reference Performance Test

\textit{RUL} Remaining-useful-life

\textit{RVM} Relevance Vector Machine

\textit{SOC} State-of-Charge

\textit{SOH} State-of-Health

\textit{SVM} Support Vector Machine

\textit{SVR} Support Vector Machine for Regression

\textit{TB} Traction Battery
Abstract

Battery cells are the core of any electric vehicles (EV). It is important to get the accurate value of different characteristics in a battery cell for safety and lifetime concern. And Two machine learning methods, support vector machine for regression (SVR) and multilayer perceptron (MLP) were introduced to improve the performance of predicting the battery cell voltage using certain circuit related features like current, capacity and temperature on two different kinds of datasets. Different configurations are validated using cross-validation to find the most suitable configuration of the parameters of these two machine learning models. Besides direct implementation, another pipeline is to build these machine learning models on one exist traditional model and improve its performance.
Batterikällor är kärnan i alla elektriska fordon (EV). Det är viktigt att få det exakta värdet av olika egenskaper i en batterilucka för säkerhet och livslängd. Och två maskininlärningsmetoder, stöd vektor maskin för regression (SVR) och flerlagsperceptron (MLP) introducerades för att förbättra prestanda för att förutsäga batterispännningen med användning av vissa kretsrelaterade funktioner som ström, kapacitet och temperatur på två olika typer av dataset. Olika konfigurationer valideras med hjälp av cross-validering för att hitta den lämpligaste konfigurationen av parametrarna för dessa två maskininlärningsmodeller. Förutom direkt implementering är en annan pipeline att bygga dessa maskininlärningsmodeller på en existerande traditionell modell och förbättra prestanda.
Acknowledgement

This thesis project is really interesting and practical. Big thanks to my supervisors at Volvo Cars, Kristian Frenander, Rajagopalan Badrinarayanan, Louise Lundberg and my supervisor at KTH Royal Institute of Technology, Ming Xiao, who were all very supportive and patient.

Thanks to many others who gave useful suggestions on the presentation and helped with the access of data within the group. Thanks to Vishnu Hari for conducting literature review, organizing data and helping tuning the parameters of one of the models together.
# Contents

Nomenclature v  
Abstract vii  
Sammanfattning viii  
Acknowledgement ix  
List of Figures xiii  
List of Tables xv  

1 Introduction 1  
1.1 Background ............................ 1  
1.2 Aim ...................................... 4  
1.3 Scope .................................. 4  
1.4 Methodology ............................ 4  
1.5 Thesis Outline .......................... 4  

2 Literature Review 6  
2.1 Extreme Learning Machines (ELM) ...................... 6  
2.2 Support Vector Machine (SVM) ........................ 7  
2.3 Long Short-Term Memory Recurrent Neural Network (LSTM RNN) . 8  

3 Modelling 9  
3.1 Multilayer Perceptron .......................... 9  
3.2 Support Vector Machine for Regression ............... 12
3.3 Data Used in the Modelling .............................................. 15
  3.3.1 Data Preparation ..................................................... 16

4 Results and Discussion ............................................. 18
  4.1 Pre-tests ................................................................. 18
  4.2 Direct Implementation ............................................... 20
    4.2.1 Multilayer Perceptron ......................................... 20
    4.2.2 Support Vector Machine for Regression ...................... 22
  4.3 On Top of ECM ......................................................... 24
    4.3.1 Multilayer Perceptron ......................................... 24
    4.3.2 Support Vector Machine for Regression ...................... 26
  4.4 Numerical Metric ...................................................... 27

5 Conclusion and Future Work ...................................... 28

References ................................................................. 31
List of Figures

1.1 Schematic of open-circuit voltage model. .......................... 2
1.2 Schematic of a basic equivalent circuit battery model. .......... 3

3.1 Schematic of multilayer perceptron model. ....................... 10
3.2 Schematic of SVM. ............................................. 13
3.3 Schematic of non-linear SVR function. ............................ 13
3.4 The voltage curve of original data. ............................... 16

4.1 Voltage prediction using MLP for RPT data. .................... 18
4.2 The schematic of building machine learning model on top of first order ECM. .................................................. 19
4.3 Voltage prediction using MLP on top of the first order ECM for RPT data. .......................................................... 20
4.4 The training result using MLP on dynamic data. ................. 22
4.5 The test result using MLP on dynamic data. ..................... 22
4.6 The training result using SVR on dynamic data. ................. 24
4.7 The test result using SVR on dynamic data. ..................... 24
4.8 The error voltage and prediction using MLP. .................... 25
4.9 The training result using MLP on top of first order ECM ........ 25
4.10 The test result using MLP on top of the first order ECM. ...... 26
4.11 The training result using SVR on top of the first order ECM. ... 26
4.12 The test result using SVR on top of the first order ECM. ...... 27
List of Tables

3.1 Different kernels used for SVR. .............................................. 14

4.1 Different configurations of MLP. ............................................. 21
4.2 Chosen configuration of MLP. ................................................ 21
4.3 Different configurations of SVR .............................................. 23
4.4 Chosen configuration of SVR ................................................ 23
4.5 Mean squared error comparison ............................................. 27
List of Tables

xvi
Introduction

The popularity of electric vehicles (EV) is the trend of the times. There are a few aspects make EV superior to traditional internal combustion engine (ICE) automobiles. EV use electric energy and cause less environmental pollution. Because of the use of single energy source, transmission, fuel tank, emission system and so on are omitted, so the structure is relatively simple [1]. EV have higher energy efficiency and lower noise than ICE automobiles [1].

Batteries are the core of any EV, and also called traction batteries (TB) [2]. The entire battery pack life is actually closely related to each battery cell. It is important for us to do battery management of EV under various complicated charging and discharging states and different ambient circumstances, make sure what is the actual output and input state [2].

In this report, data driven approaches using machine learning methods are investigated as the battery cell voltage measurement in the car. The approaches are chosen based on previous research and related characteristics of the battery cell.

1.1 Background

The study of battery cell models can be started by setting up analogs with common circuit elements. It is helpful to understand how battery cells act under different usage scenarios. In the simplest reasonable model, an ideal battery cell can be modeled as an ideal voltage source [3]. In an ideal battery cell, the voltage is neither a function of current nor a function of past usage, but a constant value [4]. This model is not adequate for practical application but provides a good start to under-
The function of battery cells is to supply a voltage to a load. The voltage is rather predictable while the battery cell is unloaded and in stability as shown in Fig. 1, known as an open-circuit voltage (OCV) [5]. In Figure 1.1, \( i(t) \) represents the current at time \( t \) and \( v(t) \) represents the voltage at time \( t \).

![Figure 1.1: Schematic of open-circuit voltage model.](image)

The OCV of a battery cell varies depending on how charged the battery cell is. The model can be improved by introducing a dependence on the charge status of a battery cell, the state of charge (SOC) \( z(t) \). It is defined such that \( z(t) = 100\% \) and \( z(t) = 0\% \) respectively when the battery cell is fully charged and fully discharged [6]. The total capacity \( Q \) is the amount of charge obtained when charging a battery cell from \( z(t) = 0\% \) to \( z(t) = 100\% \). \( Q \) is usually measured in Ah or mAh. OCV is a function of SOC and temperature and can be denoted as \( OCV(z(t), T(t)) \) [5].

A more realistic model is introduced by having a resistance in series with an ideal voltage source [5]. This is because a battery cell’s voltage drops when it is under load [4]. The model is given by Eq. 1.1.1. \( R_0 \) refers to the load resistance. This model is useful for many simple electronic circuits, but not the modelling of EV applications [4].

\[
v(t) = OCV(z(t), T(t)) - i(t)R_0
\]  

(1.1.1)

A battery cell’s voltage gradually approaches to OCV when it is allowed to rest. This phenomena is caused by slow diffusion processes in the battery cells and this slowly-changing voltage is known as diffusion voltage [6]. This can be approximated in a circuit using one or more parallel resistor-capacitor pair(s), which is called equivalent circuit models (ECM) [2]. ECMs are mostly used for the simulation of
the electrical and thermal behavior of lithium ion (Li-ion) battery cell [4]. While there’s only one resistor-capacitor pair it is called first order ECM, as presented in Fig. 1.2. $R_1$ and $C_1$ refer to the resistor and capacitor. The battery cell voltage can be modeled as Eq. 1.1.2. In Eq. 1.1.2, $i(t)$ is the sum of the current through $R_1$ ($i_{R_1}(t)$) and that through $C_1$.

$$v(t) = OCV(z(t), T(t)) - i(t)R_0 - R_1i_{R_1}(t) \quad (1.1.2)$$

**Figure 1.2:** Schematic of a basic equivalent circuit battery model.

The performance of TBs is of vital importance to EV, in terms of maintenance, efficiency and safety [4]. Therefore it is critical to model the transient behaviour of EV precisely and robustly. Among different features, battery cell voltage is strongly related and needs to be determined [3]. There are weaknesses of the first order ECM and specifically certain regions where the accuracy of the model is not sufficient, like certain relaxation phase during the transition from discharge process to charge process. Some studies such as that by Hu et al. [6], have proved that adding more RC pairs more than two RC pairs to increase the complexity does not increase accuracy.

This brings out an optimization task that can be solved in many different ways. Machine learning methods are more and more used in various fields to deal with difficult problems like prediction and classification. The application of machine learning methods are usually with high accuracy and no task related principles are needed. With all the practical data samples, the results obtained from machine learning methods are quite convincing. It is promising to manage machine learning algorithms for conquering such a challenge and no need to know the principle of inner structure of a battery cell.
1. Introduction

1.2 Aim

The aim of this thesis is to improve the performance of the first order ECM using suitable machine learning methods. Furthermore, the effectiveness and robustness of different approaches are compared. Utilizing the knowledge gained from this, models that can estimate the voltage performance of a battery cell under different conditions shall be built.

1.3 Scope

The research conducted in this thesis is limited to lithium nickel manganese cobalt oxide (NMC) battery cells. Industrial standard ambient temperature is considered within a reasonable fluctuation range, not under extreme temperature. For confidential reason, the exact value of temperature is not revealed. Besides, only two kinds of machine learning methods, multilayer perceptron (MLP) and support vector machine for regression (SVR) are tested in this thesis.

1.4 Methodology

This work is divided into three parts:

- A literature review was carried out on previous research about characteristics estimation of battery cells using machine learning methods.
- Two machine learning algorithms were implemented to estimate the voltage of battery cells in the first order ECM and minimize the error.
- Analysis and validation were carried out to determine the performances of two algorithms.

1.5 Thesis Outline

This thesis is structured as follows:

- Chapter 2 presents previous research about features estimation of battery cell, such as SOC and state of health (SOH), using different machine learning methods.
- Chapter 3 explains the fundamental knowledge of the two machine learning algorithms used in this thesis and lays the modelling framework. Data preparation, feature engineering and experiments setting are all included.

- Chapter 4 demonstrates the results of the two implemented algorithms. Meanwhile it illustrates the analysis and comparison of the two investigated algorithms based on the obtained results regarding their performance and effectiveness.

- Chapter 5 summarizes the conclusions of this research and suggests feasible future work to do.
2

Literature Review

This chapter includes three widely used machine learning methods for the estimation of some battery cell characteristics like SOC, SOH and remaining useful life (RUL).

2.1 Extreme Learning Machines (ELM)

ELM is a single hidden layer feed forward neural network (NN) based on the traditional neural networks. It has simple structure and it does not easily fall into local minimum [7]. The hidden layer weights and biases are obtained randomly so only output layer weights are needed to be adjusted, which makes the network learning fast and more suitable for online training [8].

To track the SOC of a Li-ion battery cell, a SOC estimation algorithm based on ELM is proposed by Wang et al. in [9]. The battery cell voltage and current are inputs and SOC is output of the model. Experiments are conducted using an ambient temperature of 25°C, and the battery cell is discharged from the fully charged state until reaching 2.5V at different constant currents. To improve the training efficiency and eliminate the effect that battery cell voltage and current are different during charge and discharge process, data normalization is implemented on data set. Inputs and output are restrained in [0, 1]. The experiments eventually show that the fast training speed and high accuracy are verified.

Du et al. [10] also achieve a good performance of SOC estimation using ELM. They point out that for Li-ion battery SOC, the sampling time should be short enough to satisfy the accuracy requirement affected by quickly changing current during SOC estimation process.
2. Literature Review

2.2 Support Vector Machine (SVM)

SVM is a kernel supervised learning method with good generalization ability first introduced in [11]. It builds a hyperplane to perfectly partition different classes and especially has good performance on the classification of high dimensional data [12]. SVM is mostly used for offline training and has no local minimal problems like artificial neural network (ANN) [13]. There are two variations of SVM that are often used, SVR and relevance vector machine (RVM) [14].

Álvarez et al. [15] proposed the use of SVR to predict the SOC of a high-capacity lithium-ion manganese phosphate (LiFeMnPO4) battery cell. The input variables are the battery cell voltage, battery cell current, and battery cell temperature obtained from battery cell charging and discharging procedure cycles or dynamic stress test (DST) cycles. This model is tested to predict the battery cell voltage applying the same methodology and attains high accuracy under dynamic conditions.

A SVR model is implemented by Meru et al. in [16] to design an efficient RUL estimation model. Raw data features include capacity, voltage, voltage load, current, current load and temperature, for each discharge cycle of the battery cells. Crucial variables such as capacity, energy of voltage curve over time, fluctuation index of voltage curve are included as inputs of the model. This model is tested under different scenarios and with adequate accuracy.

RVM is the Bayesian form representation of SVM with the same function. The only difference is RVM provides probabilistic interpretation of outputs. In [14], SOH is the target output variable and inputs are calculated form discharge voltage data from tested battery cells. The results indicate that this method is feasible for the good performance of SOH prediction using SVM and RVM.
2.3 Long Short-Term Memory Recurrent Neural Network (LSTM RNN)

LSTM RNN is a deep learning NN that can learn long term dependencies. There are three gates in the network make the model able to remember information for certain periods. The forget gate is to drop out unnecessary information, the input gate is to choose useful information to store in the internal state and the output gate is to decide output information [17].

RUL estimation of Li-ion battery cell could assess battery reliability to identify failures and reduce risk since it can give the approximate failure time in advance [1]. It is challenging of RUL prediction to learn the long term dependencies over hundreds of cycles based on limited data. Zhang et al. [17] proposed an offline LSTM-RNN for battery cell RUL prediction. A technique designed for mini-batch training called the resilient mean square back-propagation (RMSprop) is used to train this NN model, combined with dropout technique to avoid over-fitting, increase the generalization [17]. Monte Carlo (MC) simulation is implemented to randomize the prediction. The study is based on 4 different cells under 25°C and 45°C, and with different current rates, including 1C, 2C, and 3.5C. The result indicates that LSTM RNN has high accuracy and precision on RUL prediction of Li-ion battery cells.
3

Modelling

In this chapter, two machine learning methods used in this thesis are demonstrated with their basic structure, mathematical background and parameters need to be tuned in application. Data description and data preparation are also included.

3.1 Multilayer Perceptron

ANNs are part of the machine learning algorithms widely used in all kinds of fields recently. The initial thought was inspired by the biological NN. NN is a broad concept including many different machine learning algorithms to work on complicated data. The purpose of NN is to achieve tasks without task related principles, only by considering data samples and then find the key features and pattern [18].

The most crucial element of an ANN is a collection of interconnected nodes called neurons. The connections between neurons are like the synapses in a human brain. Signals are processed in the neurons and transmitted through the connection to other neurons. The output of each neuron is attained by applying some non-linear function on the sum of its inputs [19]. The connections typically have a weight would be replaced with new value during training process, and the weights decide how strong the connection between two neurons are [20]. In general, the weights decide how important one input is to the output. Usually, neurons are assembled as layers and different layers can perform different transformations on their inputs according to the non-linear function, which is called activation function [19].

A MLP is one kind of the feedforward ANNs. A feedforward configuration means all the information in the network moves only forward, from the input neurons to the output neurons, without any feedback loop [21]. A MLP model consists of at least
three layers: 1) the input layer; 2) one hidden layer and 3) the output layer. Hidden layers are the layers between the input layer and the output layer. Each neuron connects with a certain weight $w_{ij}$ to every neuron in the next layer. The indices $i$ and $j$ specify the neurons in the current layer and following layer respectively. Figure 3.1 shows the schematic of a MLP network.

![Figure 3.1: Schematic of multilayer perceptron model.](image)

The training technique utilized by MLP is called backpropagation. The whole training process consists of forward process and backward process. In the forward process, we calculate the output of each neuron as shown in Eq. 3.1.1. Here $N$ represents the number of neurons in the current layer, $b$ represents the bias term and $\varphi$ represents the activation function. The same operation passes through layers, and eventually get the final output. The function used to measure how close the prediction and the true value are to each other is called cost function [18]. Usually the mean squared error (MSE) is implemented as the cost function for regression problem [20]. Equation 3.1.2 shows the mathematical expression. $Y_i$ denotes the true value of target, $\hat{Y}_i$ denotes the prediction of target and $n$ denotes the number of data points.

$$y_j = \varphi\left(\sum_{i=1}^{N} w_{ij} x_i + b\right)$$
3. Modelling

\[
MSE = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 \tag{3.1.2}
\]

The point of MLP is to minimize the cost function using iterative training and which is dependent on the backpropagation. During the backpropagation, gradient descent is used to find the minimum of a function. Training a MLP with gradient descent requires the calculation of the gradient of the cost function \(E(X, \theta)\) with respect to the weights and biases [20]. Then the weights and biases are updated with new value during each iteration as shown in Eq. 3.1.3. \(\theta_t\) denotes the parameters (weights and biases in this case) of MLP at iteration \(t\) in gradient descent. \(\alpha\) denotes the learning rate, which has influence on the training speed and prediction accuracy.

\[
\theta_{t+1} = \theta_t - \alpha \frac{\partial E(X, \theta^t)}{\partial \theta} \tag{3.1.3}
\]

It is helpful to initialize the parameters of the network to achieve robustness in the MLP. If initialization is done correctly then optimization will be achieved in shorter time [22]. Initialization means to assign a statistical distribution to the parameters, in case it will not converge to the minimum using gradient descent [21]. Batch normalization is a technique designed to accelerate the training process and improve the performance and stability of the model via its regularization effect, which can be usefully applied behind each layer [23].

There are some mostly used activation functions for regression problems. Such as tanh, ReLU, ELU and SELU. The formulas are listed in order as shown in Eq. 3.1.4, 3.1.5, 3.1.6 and 3.1.7. The purpose is to transform the linear combination of weights, neurons and biases into a non-linear relationship, which is more suitable for the problems we meet in real life.

\[
f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.1.4}
\]

\[
f(x) = \begin{cases} 
0 & \text{for } x < 0 \\
x & \text{for } x \geq 0 
\end{cases} \tag{3.1.5}
\]

\[
f(x) = \begin{cases} 
\alpha(e^x - 1) & \text{for } x < 0 \\
x & \text{for } x \geq 0 
\end{cases} \tag{3.1.6}
\]

\[
f(x) = \lambda \begin{cases} 
\alpha(e^x - 1) & \text{for } x < 0 \\
x & \text{for } x \geq 0 
\end{cases} (\lambda = 1.0507, \alpha = 1.67326) \tag{3.1.7}
\]
During training, the model performance is evaluated based on the available data but in practice the model should be able to give good predictions of unseen observations. Overfitting occurs when a model fits well with training data but lacks a good generalization ability [19]. Early stopping is a regularization technique used to avoid overfitting while training with an iterative algorithm such as gradient descent [24]. This training algorithm improves its fitting performance within each iteration. However, after a certain point, the improvement of fitting performance of training data is at the cost of increasing the generalization error [24]. Early stopping rules make training process stop before it overfits.

There are two main parameters that control the performance of MLP: the number of hidden layers and the number of neurons in each hidden layer. It is not easy to analytically obtain the optimal configuration of layers and neurons. The choice of parameters' values should be made based on systematic experiments with the specific dataset. However in general, deep networks with more layers might be a heuristic approach for challenging problems [25].

### 3.2 Support Vector Machine for Regression

The purpose of SVM is to find a hyperplane in an N-dimensional space where N refers to the number of features, to separate and classify data points. A hyperplane is actually a decision boundary and the data points falling on different side of the hyperplane can be classified as different classes. The hyperplane is just a line when the number of input features is 2, and it is a two-dimensional plane when the number of input features is 3 [12]. Support vectors refer to the data points that are closer to the hyperplane and have an influence on the position and direction of the hyperplane [16]. With these support vectors, the margin of the SVM is maximized, as shown in Fig. 3.2.

SVR implements the same techniques as the SVM for classification, a non-linear function is leaned by linear learning machine mapping into higher dimensional kernel induced feature space, with only a few slight differences [15]. In the case of SVR, the main idea is still the same: to minimize the error, finding a hyperplane which maximizes the margin, so that the model has more generalization ability [14].
To find and optimize the generalization bounds for regression, the cost function is defined by setting up a tolerance margin, ignoring the errors, which are located within the certain distance of the true value [12]. This function is called $\varepsilon$-intensive cost function. Figure 3.3 illustrates an example of the regression function with $\varepsilon$-intensive band. For data points situated inside the margin, the prediction errors are considered as zero. As shown in Eq. 3.2.1, $\hat{y}$ refers to the prediction of target, $\hat{y}$ is the true value of target. The accuracy of prediction is measured by the cost function $L(y, \hat{y})$. Using the $\varepsilon$-intensive cost function guarantees the existence of the global minimum with reliable generalization ability at the same time [15].

$$L(y, \hat{y}) = \begin{cases} 0 & \text{if } |y - \hat{y}| \leq \varepsilon \\ |y - \hat{y}| - \varepsilon & \text{otherwise} \end{cases} \quad (3.2.1)$$

In SVR, the input $x$ is mapped into a $N$-dimensional feature space by nonlinear transformation, and then a linear function is constructed based on this feature space, as shown in Eq. 3.2.2, $K(x_i, x)$ is the non-linear transformation, $b$ is the bias term. $\alpha_i$ and $\alpha_i^*$ are the non-negative multipliers for each observation $x_i$. Besides minimizing $||\alpha_i - \alpha_i^*||^2$, non-negative slack variables $\xi_i, \xi_i^*$ are introduced to measure the deviation of training data points outside the margin [11]. Eventually the SVR is formulated to minimize Eq. 3.2.3 under the constraints shown in Eq. 3.2.4 [13]. $C$ is a positive constant that controls the penalty applied on samples that lie outside.
3. Modelling

the margin and helps to avoid overfitting.

$$\hat{y} = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) K(x_i, x) + b$$  \hspace{1cm} (3.2.2)

$$\frac{1}{2} \| \alpha_i - \alpha_i^* \|^2 + C \sum_{i=1}^{N} (\xi_i + \xi_i^*)$$  \hspace{1cm} (3.2.3)

$$\begin{cases}
y_i - \hat{y}_i \leq \varepsilon + \xi_i^* \\
\hat{y}_i - y_i \leq \varepsilon + \xi_i^* \\
0 < \alpha_i < C, 0 < \alpha_i^* < C
\end{cases}$$  \hspace{1cm} (3.2.4)

There are four common kernel functions used for SVR to convert the data into a higher dimensional feature space to make it possible be linearly separated, see Table 3.1. The first is the linear kernel, mainly used for the data that are linearly separable. The dimension of the feature space and the input space is the same, parameters are less and training process is fast. For linearly separable data, the performance is very good, so linear kernel is usually applied first to check the characteristics of data [11]. The second kernel function is the polynomial kernel, which can map the low-dimensional input space to the high-dimensional feature space, but it has many parameters to tune. When the order of the polynomial kernel is rather high, the element value of the kernel matrix will tend to infinity [16]. It can be too complex to calculate. The third kernel function (which is the most popular one) is the Gaussian radial basis function (RBF) which maps samples into a higher-dimensional space. It has better performance with both large and small number of samples, and fewer parameters than the polynomial kernel [12]. The fourth kernel function is the sigmoid kernel. Using this kernel, the SVR actually becomes a kind of multilayer NN.

Table 3.1: Different kernels used for SVR.

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Mathematical Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear Kernel</td>
<td>$K(x_i, x_j) = x_i' x_j + c$</td>
</tr>
<tr>
<td>Polynomial Kernel</td>
<td>$K(x_i, x_j) = (x_i' x_j + c)^d$</td>
</tr>
<tr>
<td>Gaussian RBF Kernel</td>
<td>$K(x_i, x_j) = e^{-\frac{| x_i - x_j |^2}{2\sigma^2}}$</td>
</tr>
<tr>
<td>Sigmoid Kernel</td>
<td>$K(x_i, x_j) = tanh(x_i' x_j + c)$</td>
</tr>
</tbody>
</table>
Therefore, choosing a kernel function can be tricky. If there are certain prior knowledge of the available data, those information can be used to select a kernel function that conforms to the data distribution. If the prior information is insufficient, it is common to use cross-validation to try different kernel functions, the one with lowest error is the best performing kernel function, or it is also feasible to combine multiple kernel functions to form a mixed kernel function [13].

The generalization performance of SVR highly depends on a good setting of parameters $C$, $\varepsilon$ and the kernel parameters. Parameter $C$ controls the trade off between the model complexity and the degree to which deviations larger than $\varepsilon$ are tolerated in the optimization formulation [26]. If $C$ is too large, then the objective is only to minimize the error, without considering the overfitting problem caused by the model complexity [11]. Parameter $\varepsilon$ controls the width of the margin, and the value of $\varepsilon$ can affect the number of support vectors used to build the regression function. The bigger $\varepsilon$, there are fewer support vectors. Hence, both $C$ and $\varepsilon$ affect model complexity but in a different way [16].

### 3.3 Data Used in the Modelling

Two different categories of data were used in this investigation. The first dataset is reference performance test (RPT) of one single battery cell. There are relaxation phases when it reaches certain constant voltage. The test was conducted under one industrial standard temperature, with small changes due to charge and discharge processes. There are 85507 data points which are sampled at 1 Hz. The dataset consists of a few fully discharged and charged cycles followed by a series of artificially manipulated charge and discharge curves under certain current rate. Figure 3.4 (a) illustrates the voltage curve of RPT. For confidential reason, the voltage has been normalized between 0 and 1 while plotting. The RPT dataset was only used to get a rough idea of the machine learning models’ performance.

The second dataset is composed of 62 different subdatasets, all from a battery pack with 96 battery cells in series. The subdatasets were obtained under a wide range of temperature, but still not extreme high or low temperature. There are 87582 data points which are sampled at 1 Hz. The subdatasets are all dynamic tests with dif-
ferent behaviours to improve the generalization ability of the models. Figure 3.4 (b) illustrates the voltage curve of dynamic data. For confidential reason, the voltage has been normalized between 0 and 1 while plotting as well. This dataset is the main dataset used to train the models and evaluate their performance.

![Image](image.png)

**Figure 3.4:** The voltage curve of original data.

### 3.3.1 Data Preparation

The choice of input features is of vital importance for the performance of the model. In this case, our output target is the battery cell voltage, and the input features are the current, ambient temperature and capacity. The capacity at each time step was calculated by multiplying the maximum value of capacity and SOC at that point. 80% of the dataset was assigned as training data and the rest 20% was test data.

To accelerate the training speed, make the value of different features into a similar scale and improve the accuracy and robustness of algorithms, normalization was implemented on the input features [7]. For features like capacity and temperature, a common way is to normalize them between 0 and 1 separately [7]. Equation 3.3.1 shows how the normalization was achieved. $x_{\text{norm}}$ denotes the values after normalization. The minimum and maximum value of variable $x$ are denoted by $x_{\text{min}}$ and $x_{\text{max}}$.

$$x_{\text{norm}} = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}$$

(3.3.1)
3. Modelling

For features with orientation, like current, it is assumed that the current value is negative during discharge and positive during charge. In this case the current should be normalized between -1 and 1. Equation 3.3.2 shows how the normalization between -1 and 1 was achieved. \( x_{\text{aver}} \) represents the mean value of the variable \( x \) for the whole dataset.

\[
x_{\text{norm}} = \frac{x - x_{\text{aver}}}{x_{\text{max}} - x_{\text{min}}}
\] (3.3.2)

Besides that, standardization can be useful as well. It means rescaling the distribution of values so that the mean of them is 0 and standard deviation is 1. \( x_{\text{std}} \) denotes the values after standardization, \( \mu \) stands for the mean value and \( \sigma \) is the standard deviation. Standardization demands that the mean value and standard deviation of observed values are easy to know or estimate. Equation 3.3.3 shows how the standardization was achieved.

\[
x_{\text{std}} = \frac{x - \mu}{\sigma}
\] (3.3.3)

The prediction of battery cell voltage is not only about the correspondence between voltage and inputs value at the same moment; the trend of how the variables vary is also important. That is the reason previous inputs values are need to be considered as well. A parameter called "look back steps" was introduced to decide how many previous values should be used.

A remaining problem is related to how to flatten previous observations of all inputs into feature vectors for machine learning models. First, three input features (current, capacity and temperature) were combined with the output target voltage together as a single dataset, where each row is a time step and each column is a different time series. If the "look back steps" is set as 4, the first sample should be the first four time steps of each parallel variables as the inputs to the model and the output of model would be the output series at the fourth time step.

For this purpose, a few values at the beginning from the output series have to be discarded since there are no four previous time steps for the first step. Hence the choice of how many steps to look back determines how many values to discard. The model requires that the shape of the input for each sample is a vector, thus, it is necessary to flatten the multiple vectors into one. The length of each input vector is the number of time steps to look back multiplied by the number of input features.
4

Results and Discussion

In this chapter, the two machine learning models introduced in Chapter 3 were implemented on two main datasets for training and evaluation. Visual results and numerical metric comparisons were made to evaluate the performance of SVR and MLP on the prediction of battery cell voltage.

4.1 Pre-tests

In this section RPT data was used to see if machine learning models perform well on the battery cell voltage prediction and then make the decision on how to conduct the experiments exactly.

(a) Discharge process

(b) Charge process

Figure 4.1: Voltage prediction using MLP for RPT data.

MLP was implemented on RPT data, using current, capacity and temperature as the input features and the measured voltage as the target. In general it can cap-
4. Results and Discussion

ture the trend of how voltage varies, but there are certain phases like the relaxation phase, which should be a fairly constant voltage period, but the prediction can not really follow the trend for both charge and discharge processes, as seen in Fig. 4.1 (a) and (b). However, in real life application, the aim is not only decreasing the prediction error but also making the prediction follow the trend of the target, instead of fluctuating around the target value even though with a rather small error. Furthermore, the performance of SVR is similar to MLP: neither model is able to reproduce the pattern of the relaxation phase.

To solve this issue with the relaxation phase, another pipeline was proposed to fix this problem. Another idea was to build machine learning models on top of the first order ECM. There are current, capacity, temperature, resistance and capacitance for each data point, it is easy to simulate the battery cell voltage based on the first order ECM. And the error between the voltage simulated by the first order ECM and the measured voltage was considered as the target of machine learning model to estimate. It gave the error prediction, adding up to the simulation voltage we got the final prediction of battery cell voltage, as shown in Fig. 4.2.

![Figure 4.2: The schematic of building machine learning model on top of first order ECM.](image)

This new pipeline was implemented to verify and compare the performance with using only machine learning model. For both charge and discharge processes, the prediction strictly captured the trend also on details, and followed the variation if the voltage changed abruptly (the turning point) or was constant for a longer period, as shown in Fig. 4.3 (a) and (b). The results are similar for SVR. The addition of the first order ECM to the machine learning models resulted in a significant improvement on the voltage prediction for all parts of the RPT dataset.
4. Results and Discussion

(a) Discharge process  
(b) Charge process

Figure 4.3: Voltage prediction using MLP on top of the first order ECM for RPT data.

4.2 Direct Implementation

In this section, MLP and SVR were implemented on the dynamic data directly without any help of the first order ECM. Since the performance with dynamic data might be different to that with the RPT data, direct implementation of machine learning models without including first order ECM was also considered.

4.2.1 Multilayer Perceptron

As stated in Section 3.1, for MLP there are many parameters that need to be settled and there is no one optimal setting of these parameters, since it depends on the data are dealt with. Fine tuning these parameters is of vital importance to the performance of model on different data. The combination of different parameter settings that were used in this project are listed in Table 4.1.

Cross-validation is an important technique introduced here. It is commonly used to select a suitable model for given tasks and estimate the performance under different parameter settings. The result obtained mostly have a lower bias than other methods. A 5-fold cross-validation was used to improve the generalization ability in this thesis. K-fold means splitting the training data into K parts with the same size, one of the K subsets is retained as the validation data, the rest K -1 subsets
4. Results and Discussion

Table 4.1: Different configurations of MLP.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Look back steps</td>
<td>1, 5, 10, 15, 20</td>
</tr>
<tr>
<td>Number of layers</td>
<td>3, 4, 5</td>
</tr>
<tr>
<td>Number of neurons</td>
<td>30, 40, 50, 60, 80, 100</td>
</tr>
<tr>
<td>Activation function</td>
<td>tanh, ReLU, ELU, SELU</td>
</tr>
<tr>
<td>Batch size</td>
<td>90, 900</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adam, SGD, Adadelta, Adagrad</td>
</tr>
<tr>
<td>Initializer</td>
<td>Zeros, Ones, RandomNormal, RandomUniform, None</td>
</tr>
<tr>
<td>Data preparation</td>
<td>Normalization, Standardization</td>
</tr>
</tbody>
</table>

are used as the new training data [20]. Then the cross-validation process is repeated K times, making sure that each subset is used as validation data and only once. During training process, once the validation error stops decreasing or increasing in a certain period, training stops, and then we can get the appropriate training epochs. This is also how early stopping worked in this case. Then the most suitable settings for this dynamic data is listed in Table 4.2.

Table 4.2: Chosen configuration of MLP.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Look back steps</td>
<td>5</td>
</tr>
<tr>
<td>Number of layers</td>
<td>5</td>
</tr>
<tr>
<td>Number of neurons</td>
<td>40-50-60-80-100</td>
</tr>
<tr>
<td>Activation function</td>
<td>ReLU</td>
</tr>
<tr>
<td>Batch size</td>
<td>900</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adam</td>
</tr>
<tr>
<td>Initializer</td>
<td>RandomUniform</td>
</tr>
<tr>
<td>Data preparation</td>
<td>Normalization</td>
</tr>
</tbody>
</table>

Figures 4.4 and 4.5 illustrate the comparison between measured voltage and simulation voltage from the first order ECM, as well as the prediction result obtained using MLP, for training data and test data respectively. Overall, the deviation of
the result attained from MLP is less than the simulation result from the first order ECM. Especially for test data, there are certain periods that the prediction form MLP fits really good with the measured voltage. But for some other part, it seems like it can not really follow the trend of measured voltage going up as the simulation from the first order ECM.

Figure 4.4: The training result using MLP on dynamic data.

Figure 4.5: The test result using MLP on dynamic data.

4.2.2 Support Vector Machine for Regression

As stated in Section 3.2, for SVR there are a few parameters that need to be determined and they are quite sensitive, especially the choice of kernel function. The
4. Results and Discussion

A combination of different parameter settings that were validated in this project are listed in Table 4.3.

**Table 4.3: Different configurations of SVR**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Look back steps</td>
<td>1, 5, 10, 15, 20</td>
</tr>
<tr>
<td>(\varepsilon)</td>
<td>0.001, 0.01, 0.1</td>
</tr>
<tr>
<td>(C)</td>
<td>0.1, 1, 10</td>
</tr>
<tr>
<td>Kernel function</td>
<td>RBF, Linear, Sigmoid, Polynomial</td>
</tr>
<tr>
<td>(\gamma)</td>
<td>0.01, 0.1, 1, 10</td>
</tr>
<tr>
<td>Stopping criteria</td>
<td>0.0001, 0.001</td>
</tr>
<tr>
<td>Data preparation</td>
<td>Normalization, Standardization</td>
</tr>
</tbody>
</table>

A 5-fold cross-validation was also used to validate the performance using the different parameter settings given in Table 4.3. The most suitable settings for this dynamic data is listed in Table 4.4. Figures 4.6 and 4.7 illustrate the comparison between actual voltage, simulation voltage from the first order ECM and prediction result using SVR. In general the deviation of the result from SVR is smaller than the result from the first order ECM. However, it can not always capture the trend of how the voltage varies as better as the first order ECM.

**Table 4.4: Chosen configuration of SVR**

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Look back steps</td>
<td>5</td>
</tr>
<tr>
<td>(\varepsilon)</td>
<td>0.01</td>
</tr>
<tr>
<td>(C)</td>
<td>0.1</td>
</tr>
<tr>
<td>Kernel function</td>
<td>RBF</td>
</tr>
<tr>
<td>(\gamma)</td>
<td>0.1</td>
</tr>
<tr>
<td>Stopping criteria</td>
<td>0.0001</td>
</tr>
<tr>
<td>Data preparation</td>
<td>Standardization</td>
</tr>
</tbody>
</table>
4. Results and Discussion

Figure 4.6: The training result using SVR on dynamic data.

Figure 4.7: The test result using SVR on dynamic data.

4.3 On Top of ECM

In this section, MLP and SVR are implemented on the dynamic data on top of the first order ECM, as described in Section 4.3.

4.3.1 Multilayer Perceptron

Figure 4.8 shows the error voltage curve. The range of the error voltage value is fairly small, especially compare to the range of the measured voltage value.
4. Results and Discussion

Figure 4.8: The error voltage and prediction using MLP.

Figures 4.9 and 4.10 illustrate the comparison between measured voltage, simulation voltage from the first order ECM and prediction result using MLP on top of first order ECM, for the training results and test results respectively. For both training data and test data, the deviation of the result from the proposed model is apparently smaller than the result from the first order ECM. Besides, compared to using MLP directly, the curve seems really understand the trend of how the voltage varies.

Figure 4.9: The training result using MLP on top of first order ECM
4. Results and Discussion

4.3.2 Support Vector Machine for Regression

For error prediction, SVR has similar result as MLP. Figures 4.11 and 4.12 illustrate the comparison between measured voltage, simulation voltage from the first order ECM and prediction result using SVR on top of the first order ECM. For both training data and test data, the deviation of the result from the proposed model is really close to the measured voltage. Besides, compared to using SVR directly, the curve seems really stable to follow the trend of how the voltage varies.

Figure 4.10: The test result using MLP on top of the first order ECM.

Figure 4.11: The training result using SVR on top of the first order ECM.
4. Results and Discussion

Figure 4.12: The test result using SVR on top of the first order ECM.

4.4 Numerical Metric

Besides evaluating the performance visually, numerical results are also necessary to evaluate the performance of different usage. MSE was chosen as the metric to compare different results as stated in Section 3.1. Table 4.5 includes 5 different occasions to predict the voltage, training result and test result are listed separately. The results are only from the dynamic data since the result of RPT data was only based on one kind of data without behaviour variance, it is not meaningful to evaluate the performance. All the four occasions with machine learning methods involved have smaller MSE than using only the first order ECM. The occasions with MLP involved have smaller MSE than those with SVR involved. And using only machine learning methods have smaller MSE than using them on top of the first order ECM.

Table 4.5: Mean squared error comparison

<table>
<thead>
<tr>
<th>Model</th>
<th>Training Data</th>
<th>Test Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECM</td>
<td>0.00357</td>
<td>0.00546</td>
</tr>
<tr>
<td>MLP</td>
<td>0.00036</td>
<td>0.00226</td>
</tr>
<tr>
<td>SVR</td>
<td>0.00040</td>
<td>0.00338</td>
</tr>
<tr>
<td>ECM + MLP</td>
<td>0.00071</td>
<td>0.00236</td>
</tr>
<tr>
<td>ECM + SVR</td>
<td>0.00043</td>
<td>0.00356</td>
</tr>
</tbody>
</table>
Conclusion and Future Work

As seen in Chapter 4, the two data driven machine learning approaches used in this thesis (SVR and MLP) are able to predict the voltage curves of battery cells with higher accuracy compared to the first order ECM. All the four experiments, including prediction using only MLP, using only SVR, using MLP on top of the first order ECM and using SVR on top of the first order ECM, they all have obvious smaller MSE than the prediction from first order ECM. In general, models with MLP involved shows smaller MSE than models with SVR involved. And only implementing MLP or SVR gives smaller MSE than using them only for performance optimization for the first order ECM.

However, the direct implementation of machine learning models can not really follow the trend of how battery cell voltage varies all the time as good as the performance built on top of the first order ECM. For some phases it can not follow the trend of the actual voltage but eventually can be compensated by great performance of other phases. While in real world application, considering of the safety and lifetime of the battery cell, the purpose is to decrease the prediction error and make sure the voltage prediction follows the actual voltage all the time. In this way, machine learning models built on top of the first order ECM performs better, since they can achieve if it is in the relaxation phase with a fairly constant voltage value, or if the voltage varies abruptly. Among all these four new application scenarios, MLP on top of the first order ECM gave the optimal result that has smaller MSE but also follows the trend of voltage really well.

Machine learning models have great performance in many fields like computer vision, handwriting recognition, natural language processing and recommendation systems. Compared to those, applications like battery cell voltage prediction has solid physi-
5. Conclusion and Future Work

cal formulas as support, which means that the improvement of introducing machine learning methods might not be as impressive as in those fields. Nevertheless machine learning methods can still be used to optimize the performance of existing first order ECM, so that the burden for machine learning methods to find the pattern accurately is absolutely mitigated.

In the future, with more and more data samples fed in the model proposed, MLP on top of the first order ECM can be more generalized and accurate, the performance of only using machine learning models might also be improved. Using different models like bayesian learning and LSTM is also a reasonable attempt. This model can be used in battery management system (BMS) in real-time with larger data scale.
5. Conclusion and Future Work
Bibliography


