CHALMERS
UNIVERSITY OF TECHNOLOGY

Road Friction Estimation using Machine Learning

Master's thesis in Electrical Engineering

SHUANGSHUANG CHEN

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Master's thesis 2018
EX060/2018

Road Friction Estimation using Machine Learning

SHUANGSHUANG CHEN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2018

Road Friction Estimation using Machine Learning
SHUANGSHUANG CHEN

Supervisor: Mats Jonasson, Sohini Roy Chowdhury, Volvo Cars Group
Examiner: Tomas McKelvey, Department of Electrical Engineering

Master's Thesis 2018
Department of Electrical Engineering
Signal Processing
Chalmers University of Technology
SE-412 96 Gothenburg

Cover: Road slippery sign.

Typeset in LaTeX
Gothenburg, Sweden 2018

Road Friction Estimation using Machine Learning
Shuangshuang Chen
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Road friction coefficient, as the most critical variable that controls vehicle motion, has significant impact on the optimal motion control and the warning of slippery road. However, due to the complexity of tire characteristics and vehicle dynamics models, estimating road friction coefficient with acceptable accuracy in various scenarios is still an unsolved research question in the field of vehicle dynamics. Currently, most of road friction estimation algorithms are built based on the vehicle dynamics or acoustic effect from tire, so carefully-tuned model works only on the limited scenarios and is not generalized well for different conditions. Besides, the current state-of-the-art algorithm still experiences the low confidence of estimation when the tire is not excited to an adequate level. In this thesis, we build more generalized models using machine learning methods: applying Echo State Networks ESNs to build on-board road friction estimation algorithm; proposing Hidden Markov Model-based clustering framework to model the spatial-temporal pattern of the road friction over different geometrical locations. We obtain substantial accuracy improvement of estimation algorithm compared to the in-house physical-based estimation algorithm. And we are able to extract the underlying spatial-temporal patterns of road friction by the proposed method, which enables to model the statistics of reality for simulation as well.

Keywords: Road Friction Estimation, Vehicle Dynamics, Hidden Markov Model, Clustering, Spatial-temporal pattern recognition, Echo State Network, Feature Selection, Generative Model, Sequential modeling, Machine learning.

Acknowledgements

I would like to express my grateness to my supervisor and advisor at Volvo Car Corporate Mats Jonasson, Sohini Roy Chowdhury and Srikar Muppirisetty to give me guidance during the project. Thank you for the time on the insightful discussion and assistance. I would also like to thank Tomas McKelvey, my examiner at Chalmers for the great feedback and thoughtful idea.
I would also like to give my thanks to my manager Mikael Edvardsson at Volvo Car Corporate to provide me with this precious opportunity for the thesis and share me with all available tools and contacts. I would also like to thank my colleagues who gave me kind help, Niklas Ohlsson, Guanqun Cao, and Minming Zhao. Furthermore, thanks my family for your solid support and continuous encouragement during the difficult period.

Shuangshuang Chen, Gothenburg, June 2018

Contents

List of Figures

List of Tables

1
Introduction

This chapter first introduces the background of the thesis project and emphasizes on the current challenges to motivate the project. In addition, it will elaborate the research problem that we are to solve in this thesis, and its scope. Lastly, the outline of thesis will be given for readers to follow easily.

1.1 Background

As the most convenient way of transportation nowadays, automotive has become more and more available and affordable to everyone. The last few decades have witnessed the rapid development of new technologies on cars, especially in active safety, autonomous driving and electrification. More and more advanced driver assistance system such as autopilot system frees drivers from tedious driving tasks and prevents the occurrence of accidents. Along with some critical breakthroughs in artificial intelligence such as object detection, tracking, path planning and etc., automotive industry endeavors to offer more intelligent and failure-free autonomous driving system. In order to design self-driving system for all scenarios, it not only relies on the redundant sensors to reconstruct the surrounding environment and advanced algorithms to make driving decision, but also a robust motion control system to execute maneuvers safely, accurately and responsively. To achieve this goal, road friction coefficient, as one of the dominant parameters for maneuvers must be estimated accurately. The assistance system could adapt different maneuvers according to the surface condition and avoid collision under the critical situations. Meanwhile, the road friction information could be utilized in other applications. Volvo Cars has developed road slippery warning system that warns upcoming drivers of slippery road condition if there is reported as slippery by previous cars. The road authority could also utilize this information for better traffic plan and maintenance scheduling [1]. This information could be accessible by the navigation system such as Google Map, therefore travel plan could be easily adjusted according to current but also foreseen road conditions, and accidents will be avoided.
However, the road friction estimation is still one of unsolved research questions in vehicle dynamics although hundreds of methods have been proposed and developed in the last few decides. Most proposals reply on well studied tire characteristics that tire behavior is modelled by mathematical equations. Furthermore, most algorithms limit themselves on some specific scenarios such as pure longitudinal, lateral, self-alignment dynamics or friction limit. There is no single algorithm could estimate road friction accurately in all scenarios. Meanwhile, the current state-of-the-art

solution also suffers the inaccurate estimation and low confidence due to the complexity of vehicle dynamics model, tire model and noise measurement.
Nowadays, machine learning as a fiercely growing branch of computer science, is a statistical technique to learn the patterns from the data especially when it comes to enormous amount of samples. We have observed its success in computer vision, natural language processing and decision making in gaming. But it still lacks of studies to introduce this technique in the traditional vehicle dynamics field for modeling and estimation. The most applied algorithm is the Kalman filter and its extensions. Currently, more and more sensors are mounted on-board such as camera, LiDAR, IMU, and etc. And the performance of sensors gets improved substantially in both measurement frequency and accuracy. However, this enormous data has not been fully utilized and studied well so far. Machine learning method offers possible way to deal with high dimensional data and it requires less expert knowledge to give better prediction once the model is proved to generalize well. At the same time, powerful computation unit like CPU or GPU facilitates more complicated algorithms to be executed in real-time applications. Therefore, we introduce machine learning methods to build a better road friction estimation algorithm in this thesis.

1.2 Purpose

The biggest challenge faced by current state-of-the-art road friction estimation algorithm is the low confidence and poor accuracy of estimate when tire is not adequately excited. For instance, it is difficult to estimate the road friction value when car runs on the highway straightly since the corresponding longitudinal force is low and utilization of friction is low as well. However, this kind of driving scenario takes large proportion of daily driving. In addition, there is coupling effect between longitudinal and lateral dynamics, therefore the assumption of either pure longitudinal or lateral dynamics does not apply for the scenario of slight braking and steering at the same time. And other estimation algorithm based on the vision or acoustic effect does not generalize well in the wild even though some have been tested well in the experimental environment. There lacks of a general algorithm that could estimate accurately under all the scenarios. Therefore, the main goal of this thesis is to use data-driven methods to learn the model to estimate road friction with relevant signals. Except that some of raw measurement signals are pre-processed to have less noise-to-signal rate. And the intermediate estimates such as tire force of longitudinal, latitude and vertical directions, or slip rate are obtained by filtering algorithm. The first part of thesis mainly focuses on building on-board estimation algorithm by Echo State Networks ESNs using selected features. In addition, the second part of thesis studies on the temporal spatial pattern extraction of road friction over different locations which could serve as prior knowledge to enhance on-board estimation algorithm. Certainly, except from constructing suitable models for this application, more analysis of results will be discussed in the later chapters.

1.3 Scope

In this thesis, we apply machine learning methods to estimate road friction for all different conditions. It is the first data-driven project from our knowledge for road friction estimation algorithm. However, there is void of open data resource and most of research is based on their own simulation model or field test, which makes bench-marking difficult. The proposed method has been verified with comparison to the internally developed method by Volvo Cars and the internal expedition data is used. The result might not be consistently comparable to methods from other publications especially the input features used in different literature vary. More thorough comparison to other state-of-art algorithm will be carried in future work to validate the algorithm further.
Furthermore, even though current data covers different road condition variants such as dry, wet, snowy, ice, mixed, and road types like asphalt, gravel, lake, and different tire types. However, the effect of variance of data on the algorithm is impossible to measure and there is no established method to quantify the robustness of algorithm, since the performance possibly degrades under completely new conditions during test time. Or the characteristics of some particular tyre deviate substantially from the typical car tyres. Although we fairly divide data into separate subsets for training, validation and test, the performance of model on unseen test data might fail to be a fair measure and it is basically impossible to predict how good model performs in wild. It is still an unsolved problem in machine learning community, which we will not discuss in depth in this thesis. In addition, the interpretation of "black-box" network behavior is still an active research topic, therefore it is difficult to find components that are responsible for the failure of model and it is beyond the scope of this thesis as well.
For the spatial-temporal pattern recognition of road friction, we have carried some experiments to validate the proposed Hidden Markov Model-based clustering method. We observe that there is bias existed in data which makes pattern extracted sensible. For example, some expedition tests were carried on specific locations only in summer or some in summer only, which might be easy for algorithm to cluster. Moreover, unlike the estimation algorithm has access to the ground truth of road friction coefficient, the true geometric pattern is impossible to obtain to evaluate, and much more data is needed. We are looking forward to see that more further investigation in this field.

1.4 Thesis outline

The thesis includes the knowledge from multiple disciplines such as vehicle dynamics, statistics and machine learning. As stated previously, the thesis will mainly tackle two parts of the problem: one is to build road friction estimation algorithm using Echo State Network; another is to use Hidden Markov Model to study the temporal pattern of road friction over different locations. We start with the brief introduction of fundamentals to better understand the methods and results in the second chapter: fundamentals of vehicle dynamics and vehicle-dynamics based road friction estimation algorithm; fundamentals of ESNs and proposed modification of model are explained in details; fundamentals of Hidden Markov Model about model architecture and learning methods. In the method chapter, there is detailed description about data sets. It also covers the extended HMM based spatial-temporal pattern recognition method, and the feature selection technique for the input of ESNs. In the result chapter, the performance of proposed road friction estimation algorithm using ESNs is compared to the vehicle-dynamics baseline model. Hyper parameter optimization is also given with details in the same chapter. Furthermore, the detailed analysis of spatial-temporal pattern is explained with plots.

2
Theory

In this chapter, we include all the fundamental theories required to understand the rest of thesis. It starts with a brief introduction of a conventional road friction estimation algorithm based on vehicle dynamics. It describes how to estimate road friction coefficient from signals such as tire longitudinal and lateral forces, slip rates, slip angles using Kalman filter and analytic tire models. We also review literature and analyze the bottleneck of the traditional methods. In the second part, we introduce the sequence modeling methods in machine learning context mainly Recurrent neural networks. Detailed description of Echo State Network, a specific sequential model, is given regarding its structure, training method, hyper parameters. The last section mainly introduces the fundamentals of Hidden Markov Model HMM including its graphic model representation, parameters and training method for readers to understand the HMM based clustering method for spatial-temporal pattern in chapter 3.

2.1 Road friction estimation

Before our in-depth discussion, we first illustrate the definition of friction coefficient to be estimated in this thesis. Seen from figure 2.1, "friction utilized" indicates the dynamic friction that the tire experiences and is the ratio between the horizontal force and nominal force, while friction potential indicates the peak friction which is the maximum dynamic friction. We also call the friction potential as friction coefficient, which is the value we are to estimate in this thesis and is correlated to the property of road surface and tyre pairs.

Therefore, the road friction coefficient is defined as the ratio between the maximum horizontal force magnitude F_{xy} and nominal force magnitude F_z. If we assume the isotropic adhesion properties of tyre in the lateral and longitudinal directions, the horizontal force F_{xy} could be decomposed into longitudinal and lateral forces and road friction coefficient is assumed to be isotropic in both direction as well:

$$\mu_i \triangleq \frac{F_{xy,i}}{F_{z,i}} = \frac{\sqrt{F_{x,i}^2 + F_{y,i}^2}}{F_{z,i}} \tag{2.1}$$

where $F_{x,i}$ is the longitudinal force of tire i, $F_{y,i}$ is the lateral force of tire i, $F_{z,i}$ is nominal force of tire i. Due to the isotropic property, the maximum horizontal force forms a circle in the longitudinal and lateral force plane, called "friction limit circle". In practice, tire usually does not have perfectly isotropic properties in lateral and longitudinal directions, so the "friction limit circle" converts to "friction limit

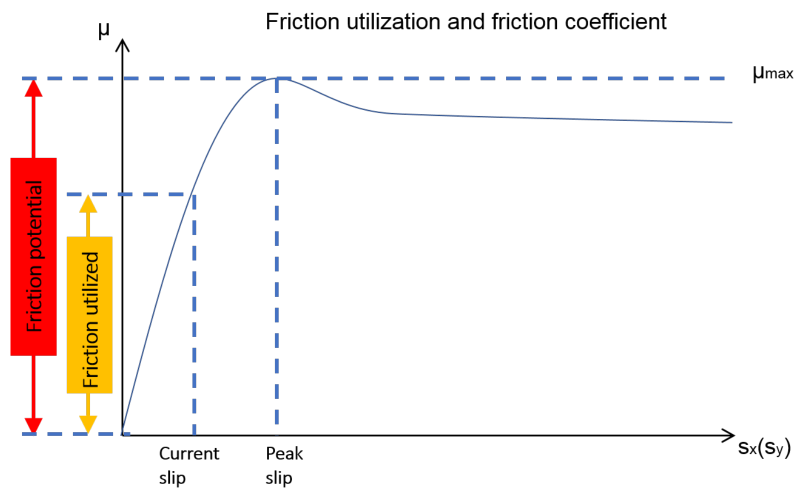


Figure 2.1: Friction utilization and friction potential given slip rate s_x or s_y

ellipse". However, due to the complexity of non-isotropic behavior of tire model and small deviation brought by non-isotropic behavior, we use the isotropic assumption in our discussion.
[2] and [3] give comprehensive reviews of different road friction estimation techniques including effect-based estimation methods that estimate it from tire response like tire slip, vibration and noise, and cause-based approaches that estimate friction directly from vision, laser scan or temperate to measure unevenness and lubricant presented on road surface. Under effect-based approaches, there are two main branches: vehicle dynamics-based methods and acoustics-based methods. Vehicle dynamics-based methods utilize longitudinal, lateral dynamics or self-aligned moment dynamics to estimate road friction using some tire models such as *Magic formula*, *Brush model*, *LuGre model* and etc. Those tire models are mathematical formulas to describe the tire characteristics with parameters. Except from tire models, different vehicle dynamics models describe the motion of vehicle using equations of motion to build up appropriate observers. The common vehicle dynamics models include single wheel model, bicycle model, two-track model and more complicated vehicle model with roll dynamics and etc. Estimation algorithms usually do not use unprocessed measurement of acceleration, yaw rate, force sensors, but utilize intermediate such as slip rate, slip angle or even tire forces. The state estimation algorithm include recursive least square, extended Kalman filter, sliding mode observer. Those intermediate estimate algorithm are beyond our discussion due to limit of space. [4], [5] and [6] describe details about longitudinal force estimation methods, and [7] introduces extended and unscented Kalman filter to estimate lateral individual tire forces. With intermediate estimates, the estimation algorithm of road friction is to calculate the road friction that maximizes the probability of estimate given forces and slip rate, slip angle under the specified tire model. However, there are several limitations of vehicle dynamics-based methods: it requires accurate tire model which is carefully calibrated and measured by a flat belt tire testing equipment; it requires tire to have adequate excitation about 60 % of maximum tire potential forces; the coupling effect between longitudinal and lateral dynamics is assumed negligible to reduce the

complexity of problem; tires on the customers' vehicle will be worn out along with the time of usage and the information of tire mounted is also impossible to acquire in advance. These shortcomings limit the wide applications of road friction on the optimal control, and motivates researchers to find more robust and general methods. Because of the complexity of tire model and vehicle dynamics, some research resorts other resource to sense road friction based on vision [1] or acoustics [8]. However, the disadvantages of those methods are evident: the acoustics-based method is sensitive to the background noise which is hard to remove completely; the vision-based method fails especially on the mixed surface such as sluggish snow on the surface of ice. Although those studies achieve certain level of accuracy in the experiments, the accuracy is tremendously reduced in practice and algorithms are extremely sensitive especially when testing conditions are deviated from the experiment conditions, therefore the robustness of algorithm is hard to guarantee. Besides, sensors such as camera or acoustic sensors that fulfill the performance requirement are usually too expensive for production car. Compared to vision- and acoustics-based method, vehicle dynamics-based method is still of great interest due to its cost-effectiveness and robustness. Therefore, we will focus on the discussion on vehicle dynamics-based methods mainly in this thesis, and the features used for machine learning methods are also dynamics related.

As mentioned before, there is bottleneck of vehicle dynamics-based method that different algorithms has limited applicability on the certain scenarios. Figure 2.2 illustrates the performing areas of different algorithms according to the different vehicle dynamics utilized. When tire reaches the friction limit or stability control is activated, it is not difficult to estimate road friction coefficient since longitudinal or lateral forces are confined in the friction limit ellipse. For the pure lateral dynamics approach, the longitudinal dynamics is assumed to have negligible effect i.e. when vehicle is steady-state cornering, so that friction coefficient is estimated from pure lateral forces and tire characteristics. Similarly, pure longitudinal dynamics approaches ignore the lateral effect i.e. when straight braking or accelerating. Both approaches work well if there is no large lateral motion for pure longitudinal dynamics or it is quasi steady-state (approximately constant longitudinal speed) for pure lateral dynamics. However, this assumption of pure longitudinal or lateral dynamics does not hold in practice when braking and steering are involved at the same time. And the coupling effect of dynamics for two directions degrades the estimation accuracy substantially when it deviates too far from pure lateral/longitudinal regions. Besides, the tire experiences low slip at normally driving scenarios without any braking and steering. The road friction utilization is so low that estimation algorithm fails as [3] addresses the difficulties of accurate estimation without adequate tire excitation. Since different estimation algorithms could function with certain accuracy in different regions in figure 2.2, it requires to integrate multiple algorithms to an integrated friction estimation algorithm like in [9]. However, the integrated algorithm is complicated but still cannot cover all regions and be generalized.

Therefore, it is significant to construct a more generalized approach to tackle most of scenarios with acceptable accuracy. Before we move into machine learning methods, we illustrate one simple estimation algorithm based on the nonlinear tire model with 8-degree-of-freedom dynamics first which is the baseline model we will compare

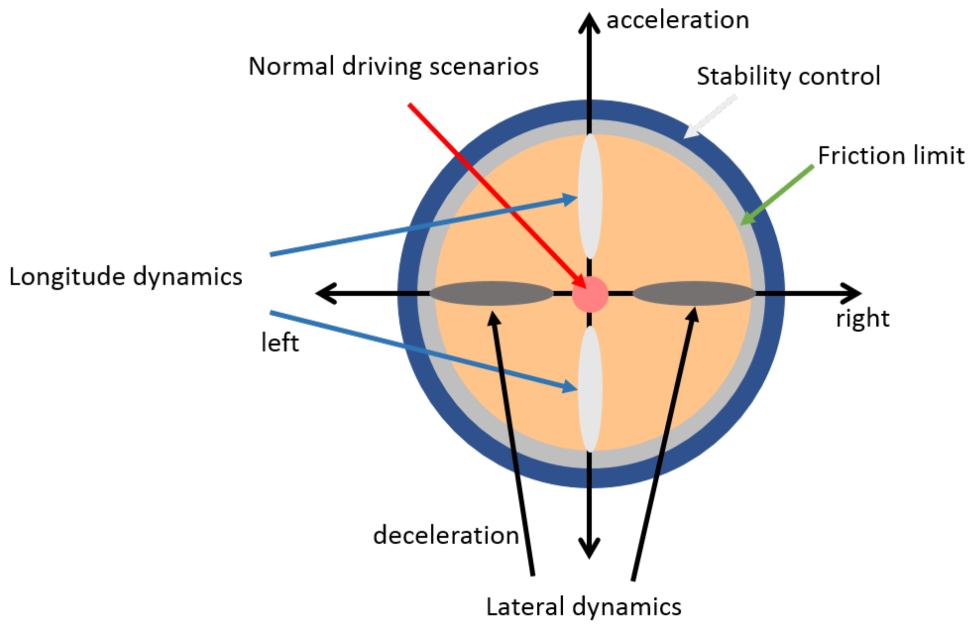


Figure 2.2: Performance area of different road friction estimation algorithm vehicle dynamics

to.

2.1.1 Vehicle dynamic based nonlinear tire force and friction estimation

The baseline algorithm of road friction estimation is proposed by Ray in [10]. It uses an eight-degree-of-freedom vehicle model which combined four-wheel model with vehicle roll dynamics.
Before deriving the estimation algorithm, we take a glance at tire model first. Figure 2.3 shows a series of tire characteristics curves under different road friction coefficients for longitudinal and lateral dynamics by *Brush model*. The x-axle is the slip rate for longitudinal dynamics or slip angle for lateral dynamics, while y-axle is the normalized longitudinal or lateral force respectively. For longitudinal dynamics, the curves of normalized longitudinal force F_x/F_z given longitudinal slip rate s_x vary with the road friction coefficients. Similar observation applies on the lateral dynamics as well: the curves of normalized lateral force F_y/F_z given lateral slip s_y vary with road friction coefficients, but the slope and the location of maximum are different to that of longitudinal characteristics. All curves can be simplified to linear and nonlinear regions. The linear region could be parameterized by the slope which we usually denote it as longitudinal stiffness for longitudinal dynamics or cornering stiffness for lateral dynamics. Within linear region, the longitudinal stiffness under different road friction coefficient is found different in experiment. Some research like [11] utilizes the estimated longitudinal stiffness to estimate road friction. However, this phenomenon varies from different types of tire and the estimate of stiffness largely relies on the accuracy of forces which requires extremely expensive

stress-strain force sensors. And analytic tire model in [12] or *Brush model* [13] do not integrate this phenomena, so that the stiffness is similar among different road friction, which makes the estimation in low slip rate region hard. In nonlinear region, we could clearly observe the discrepancy between curves. The saturation force is dominantly controlled by road friction value for both longitudinal and lateral dynamics. More complicated tire models such as *Magic Formula* or the analytic model [12] used in [10] have more complex shape than *Brush model* in figure 2.3. However, in general, we could consider tire force curves as the function of varying road friction coefficients μ. The discussion of tire models is beyond this thesis, but one thing to be emphasized here is that it does play a significant role on the validity of estimation algorithm.

In order to infer road friction from the tire model given forces and slip rate or

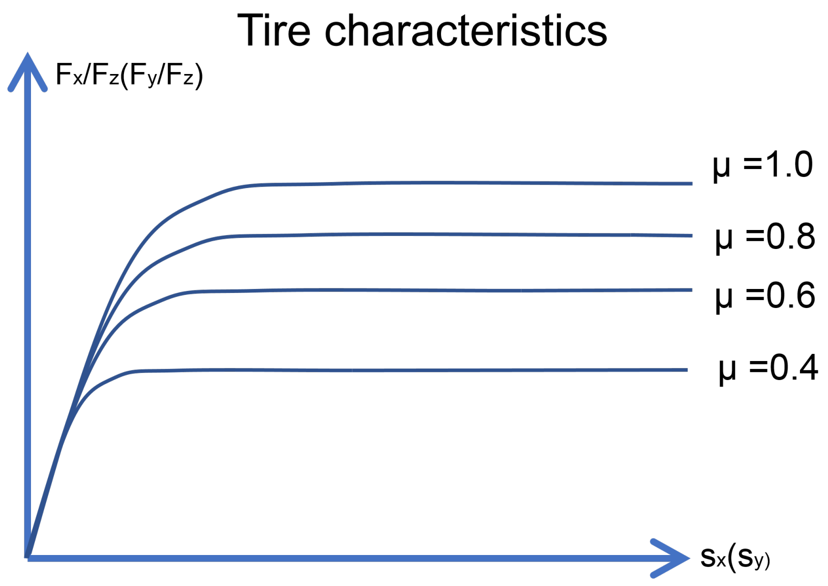


Figure 2.3: Tire characteristics curve of normalized longitudinal/lateral force $F_x(F_y)$ given slip rate/slip angle $s_x(s_y)$ by the *Brush model*

slip angle, it is necessary to model vehicle planar motion. [10] uses eight-degree-of-freedom vehicle model with roll dynamics whose diagram is shown in figure 2.4.

The equations of motion are:

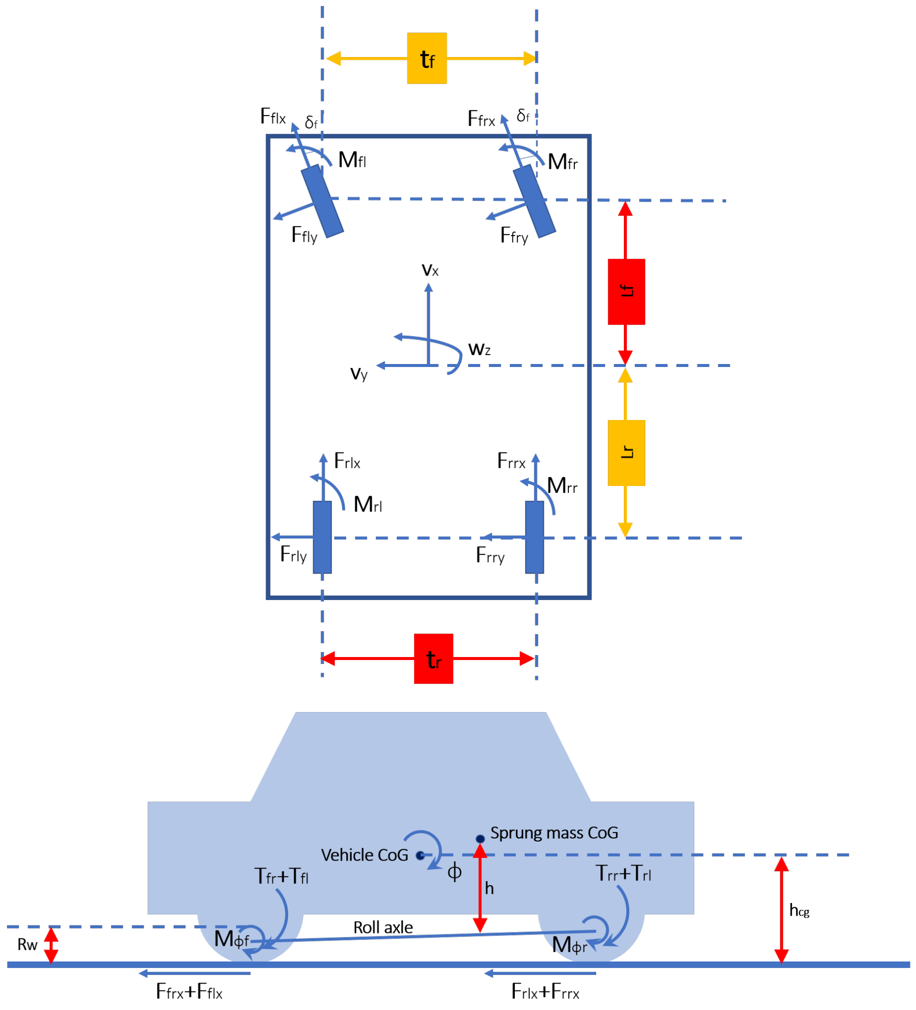


Figure 2.4: Eight-degree-of-freedom vehicle model

$$
\begin{aligned}
&m(\dot{v_x} - v_y r) = -m_s h r p + F_{xf} + F_{xr} \\
&m(\dot{v_y} + v_x r) = m_s h \dot{p} + F_{yf} + F_{yr} \\
&I_{zz}\dot{r} = I_{xz}\dot{p} + F_{yf}L_f - F_{yr}L_r + (F_{xfl}cos\delta_{fl} - F_{xfr}cos\delta_{fr} + F_{yfl}sin\delta_{fl} - F_{yfr}sin\delta_{fr})\frac{t_f}{2} \\
&\qquad + (F_{xrl}cos\delta_{rl} - F_{xrr}cos\delta_{rr} + F_{yrl}sin\delta_{rl} - F_{yrr}sin\delta_{rr})\frac{t_r}{2} + M_z \\
&I_{xxs}\dot{p} = m_s h(\dot{v_y} + v_x r) + I_{xzs}\dot{r} + m_s h g\varphi + M_{\varphi f} + M_{\varphi r} \\
&\dot{\varphi} = p \\
&\dot{\omega_{fl}} = (F_{xfl}R_w - T_{fl})\frac{1}{I_w} \\
&\dot{\omega_{fr}} = (F_{xfr}R_w - T_{fr})\frac{1}{I_w} \\
&\dot{\omega_{rl}} = (F_{xrl}R_w - T_{rl})\frac{1}{I_w} \\
&\dot{\omega_{rr}} = (F_{xrr}R_w - T_{rr})\frac{1}{I_w}
\end{aligned}
\tag{2.2}
$$

where

$$\begin{aligned}
F_{xf} &= (F_{xfl}cos\delta_{fl} + F_{xfr}cos\delta_{fr}) + (F_{yfl}sin\delta_{fl} + F_{yfr}sin\delta_{fr})\\
F_{yf} &= (F_{xfl}sin\delta_{fl} + F_{xfr}sin\delta_{fr}) + (F_{yfl}cos\delta_{fl} + F_{yfr}cos\delta_{fr})\\
F_{xr} &= (F_{xrl}cos\delta_{rl} + F_{xrr}cos\delta_{rr}) + (F_{yrl}sin\delta_{rl} + F_{yrr}sin\delta_{rr})\\
F_{yr} &= (F_{xrl}cos\delta_{rl} + F_{xrr}cos\delta_{rr}) + (F_{yrl}sin\delta_{rl} + F_{yrr}sin\delta_{rr})
\end{aligned} \tag{2.3}$$

Table 2.1 shows the list of all variables and parameters with their notations.
We could transform all of equations to state space model. So that the state vec-

symbols	variable name	
v_x	longitudinal velocity at CoG	state variable
v_y	lateral velocity at CoG	state variable
ω_{ij}	front/rear left/right wheel angular velocity	state variable
r	yaw rate at CoG	state variable
p	roll rate at CoG	state variable
φ	roll angle at CoG	state variable
F_{xij}	front/rear left/right longitudinal force	intermediate estimate variable
F_{yij}	front/rear left/right longitudinal force	intermediate estimate variable
M_z	total self-alignment torque	intermediate estimate variable
$M_{\varphi,i}$	roll moment	(ignored) input variable
δ_{ij}	front/rear left/right wheel steering angle	input variable
T_{ij}	front/rear left/right wheel torque	input variable
m	total mass of vehicle	parameter
m_s	sprung mass	parameter
h	distance of roll axis to sprung mass at CoG	parameter
L_f, L_r	distance of front/rear axle to CoG	parameter
R_w	wheel radius	parameter
I_{zz}	moment inertia around yaw axis	parameter
I_{xxs}	moment inertia around roll axis	parameter
I_{xzs}	sprung mass product of inertia about roll, yaw axis	parameter
I_{xz}	product of inertia about roll and yaw axes	parameter
I_w	wheel rotational moment inertia	parameter
t_f, t_r	front and rear track width	parameter

Table 2.1: Table of all variable and parameter symbols

tor $\mathbf{x}(t)$ components are longitudinal, lateral velocity, yaw and roll rate, 4 wheel angular velocities, and roll angle φ as $\mathbf{x}(t) = [v_x, v_y, r, p, \omega_{fl}, \omega_{fr}, \omega_{rl}, \omega_{rr}, \varphi]$. The input vector are the steering angle and braking torques of each wheel so $\mathbf{u}(t) = [\delta_{fl}, \delta_{fr}, \delta_{rl}, \delta_{rr}, T_{fl}, T_{fr}, T_{rl}, T_{rr}]$. The force vector includes longitudinal and lateral tire forces at each wheel, and total tire self-alignment moment so that it can be denoted as $\mathbf{F}(t) = [F_{xfl}, F_{xfr}, F_{xrl}, F_{xrr}, F_{yfl}, F_{yfr}, F_{yrl}, F_{yrr}, M_z]$. Meanwhile, the longitudinal slip rate and lateral slip angle estimates for each wheel could be derived from the state estimate. Lateral slip angles $\hat{\boldsymbol{\alpha}} = [\hat{\alpha}_{fl}, \hat{\alpha}_{fr}, \hat{\alpha}_{rl}, \hat{\alpha}_{rr}]$ and longitudinal slip rate estimates $\hat{\mathbf{s}} = [\hat{s}_{fl}, \hat{s}_{fr}, \hat{s}_{rl}, \hat{s}_{rr}]$:

$$\begin{bmatrix}\hat{\alpha}_{fl}\\ \hat{\alpha}_{fr}\\ \hat{\alpha}_{rl}\\ \hat{\alpha}_{rr}\end{bmatrix} = \begin{bmatrix}\delta_{fl}\\ \delta_{fr}\\ \delta_{rl}\\ \delta_{rr}\end{bmatrix} - \tan^{-1}\begin{bmatrix}\frac{\hat{v}_y+L_f\hat{r}}{\hat{v}_x}\\ \frac{\hat{v}_y+L_f\hat{r}}{\hat{v}_x}\\ \frac{\hat{v}_y-L_r\hat{r}}{\hat{v}_x}\\ \frac{\hat{v}_y-L_r\hat{r}}{\hat{v}_x}\end{bmatrix} \tag{2.4}$$

$$\begin{bmatrix}\hat{s}_{fl}\\ \hat{s}_{fr}\\ \hat{s}_{rl}\\ \hat{s}_{rr}\end{bmatrix} = 1 - R_w\begin{bmatrix}\frac{\hat{w}_{fl}}{\hat{v}_f\cos\hat{\alpha}_{fl}}\\ \frac{\hat{w}_{fr}}{\hat{v}_f\cos\hat{\alpha}_{fr}}\\ \frac{\hat{w}_{rl}}{\hat{v}_r\cos\hat{\alpha}_{rl}}\\ \frac{\hat{w}_{rr}}{\hat{v}_r\cos\hat{\alpha}_{rr}}\end{bmatrix} \tag{2.5}$$

where $\hat{v}_f$ and $\hat{v}_r$ are the estimated magnitudes of velocity of front and rear axle respectively:

$$\begin{aligned}\hat{v}_f &= \sqrt{(\hat{v}_y + L_f\hat{r})^2 + \hat{v}_x^2}\\ \hat{v}_r &= \sqrt{(\hat{v}_y - L_f\hat{r})^2 + \hat{v}_x^2}\end{aligned} \tag{2.6}$$

As mentioned before, the analytic tire force model of [12] is the functions of velocity, μ and normal force $\mathbf{F}_z = [F_{zfl}, F_{zfr}, F_{zrl}, F_{zrr}]$. Given the tire model $\mathbf{T}(\cdot)$, the normalized longitudinal force and lateral force is the function of slip:

$$\begin{aligned}\mathbf{F} &= [F_{xnfl}, F_{xnfr}, F_{xnrl}, F_{xnrr}, F_{ynf}, F_{ynr}]\\ &= \mathbf{T}(\mathbf{s}, \boldsymbol{\alpha}, \mathbf{F}_z, \mathbf{v}, \mu)\end{aligned} \tag{2.7}$$

where normalized force i.e.

$$\begin{aligned}F_{xnfl} &= \frac{F_{xfl}}{F_{zfl}}\\ F_{ynf} &= \frac{F_{ynfl}F_{zfl} + F_{ynfr}F_{zfr}}{F_{zfl} + F_{zfr}}\end{aligned} \tag{2.8}$$

One thing needs to emphasized is that the tire model here simply considers the decoupling effect between lateral and longitudinal dynamics. Therefore, the lateral forces are independent to the longitudinal forces.
The nonlinear tire forces are firstly estimated by Extended Kalman-Bucy filter EKBF [14]. The measurement equation is:

$$\begin{aligned}z(t) &= [r \quad \omega_{fl} \quad \omega_{fr} \quad \omega_{rl} \quad \omega_{rr} \quad a_x \quad a_y \quad p]^T\\ &= \mathbf{h}[\mathbf{x}(t), \mathbf{F}(t), \mathbf{u}(t)] + \mathbf{n}(t)\end{aligned} \tag{2.9}$$

which is the function of three parts – states, forces and inputs and a_x and a_y are the longitudinal and lateral acceleration respectively, and $\mathbf{n}(t)$ is Gaussian noise. The estimation model requires the pre-knowledge of either tire forces or road friction μ. However, the forces are the variable to be estimated in the force estimator. Therefore, in order to estimate both forces and road friction, the state vector is augmented to include the first order differential of forces as well. The augmented state and differential equation change to:

$$
\begin{aligned}
\hat{\mathbf{x}_A}(t) &= \begin{bmatrix} \hat{\mathbf{x}}(t) \\ \hat{\mathbf{F}}(t) \end{bmatrix} = \begin{bmatrix} \mathbf{f}(\hat{\mathbf{x}}, \hat{\mathbf{F}}, \hat{\mathbf{u}}) \\ \mathbf{A}\hat{\mathbf{F}} \end{bmatrix} = \mathbf{f}_A(\hat{\mathbf{x}}_{\mathbf{A}}, \mathbf{u}) \\
\hat{\mathbf{y}}(t) &= \mathbf{h}_{\mathbf{A}}(\hat{\mathbf{x}}_{\mathbf{A}}, \mathbf{u})
\end{aligned}
\tag{2.10}
$$

where $\boldsymbol{A}$ is a block-diagonal matrix, $\mathbf{x}_A$ is augmented state vector concatenated by original state vector $\hat{\mathbf{x}}(t)$ and $\hat{\mathbf{F}}(t)$ is six estimated forces $[\hat{F}_{xfl}, \hat{F}_{xfr}, \hat{F}_{xrl}, \hat{F}_{xrr}, (\hat{F}_{yfl}+ \hat{F}_{yfr}), (\hat{F}_{yrl} + \hat{F}_{yrr})]$ with their first order differentials, and $\hat{\mathbf{y}}(t)$ is reconstructed output. EKBF is a standard estimation algorithm by integrating equation 2.10. The details about force estimator could refer to [10] if readers are interested in implementation. We would assume that we have obtained force estimates to focus on our discussion about friction estimation algorithm.
Once we have all variables available, we could estimate road friction coefficient. Therefore the conditional probability of $\hat{\mathbf{F}}$ given μ is [10]:

$$
\begin{aligned}
Pr[\hat{\mathbf{F}}|\mu] =& Pr[\hat{\mathbf{F}}|\mathbf{T}(\mathbf{s}, \mathbf{a}, \mathbf{F}_z, \mathbf{v}, \mu)] \\
=& \frac{1}{(2\pi)^{n/2}\mathbf{S}^{1/2}} \exp\{-\frac{1}{2}(\hat{\mathbf{F}} - \mathbf{T})^T \mathbf{S}^{-1} (\hat{\mathbf{F}} - \mathbf{T})\}
\end{aligned}
\tag{2.11}
$$

where $\mathbf{S}$ is the covariance matrix. Then the estimated road friction coefficient takes the expectation:

$$
\hat{\mu} = \mathbb{E}_{Pr[\mu|\hat{\mathbf{F}}]}[\mu] \tag{2.12}
$$

However, the expectation is not computationally feasible for real-time estimation algorithm. Therefore, we pre-define finite number of hypothesis of μ. Given each hypothesis, the conditional probability of μ_j given $\hat{\mathbf{F}}(t)$ is according to Bayes theorem:

$$
Pr[\mu_j|\hat{\mathbf{F}}(t)] = \frac{Pr[\hat{\mathbf{F}}(t)|\mu_j] Pr[\mu_j|\hat{\mathbf{F}}(t-1)]}{\sum_{j=1}^{J} Pr[\hat{\mathbf{F}}(t)|\mu_j] Pr[\mu_j|\hat{\mathbf{F}}(t-1)]} \tag{2.13}
$$

The hypothesis μ_j is selected as 10 i.e. $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$ and the initial condition probability $Pr[\mu_j|\hat{\mathbf{F}}(\mathbf{0})]$ is $1/J$ that there is no prior knowledge of μ. The estimated $\hat{\mu}_t$ for each time instance becomes :

$$
\hat{\mu}(t) = \sum_{j=1}^{J} Pr[\mu_j|\hat{\mathbf{F}}(t)]\mu_j \tag{2.14}
$$

2.2 Sequence modeling for on-board estimation algorithm

Most data in practice comes in a sequential manner, so is the data we used in this thesis. It can be formulated in non-sequential model, however, the data will violate the independent identically distributed assumption and sensor reading has temporal pattern underlying in the measurement which non-sequential model can not capture

it. Besides, most sensor readings are not noise-free, so non-sequential model might suffer from large signal-to-noise ratio and make learning system extremely unstable and overfitting.
There are multiple alternative approaches to model sequential data such as Dynamic Bayesian network and Recurrent neural networks. The Dynamic Bayesian Network is a Bayesian network that relates variables to each other over the adjacent time steps. Two widely used Dynamic Bayesian Network are hidden Markov model [15] and Kalman filter [16]. Hidden Markov model [15] models the temporal pattern through the latent variables which are discrete states and state transits to each other according to transition probability, and emission probability determines the probability of observation given latent states. However, in this thesis, Hidden Markov model will be used for temporal pattern recognition and discussed in detail in next section. Kalman filter [16] is a linear Gaussian state space model. With specified linear dynamics of system (state-transition model and observation model) and assumed Gaussian noise, the state could be inferred from the observation. On the other hand, Recurrent neural network regains huge popularity recently in the sequence learning especially with deep neural networks due to its capability to learn the complex nonlinear mapping and improved computation power of embedded computation unit. Recurrent neural networks RNN have applied in many different domains such as natural language processing, sentiment analysis, neural machine translation, speech recognition, rhythm learning, hand-writing recognition, human action recognition and etc. Compared to the normal multi-layer perceptron method, RNNs could model variable-length data so that we do not have to fix the input length, because parameter of model is shared for all time instance. The basic idea of RNN is to learn the mapping between the hidden state, input and output for each time instance in sequence:

$$\begin{aligned} \mathbf{h}^{(t)} &= f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}; \boldsymbol{\theta}) \\ \mathbf{y}^{(t)} &= g(\mathbf{h}^{(t)}) \end{aligned} \tag{2.15}$$

where $\mathbf{h}^{(t)}$ is hidden states at time t that stores all necessary historical information until t, $\mathbf{x}^{(t)}$ is input vector, $\boldsymbol{\theta}$ is parameters of networks, $\mathbf{f}(...)$ is internal activation function and $\mathbf{g}(...)$ is output activation function, mapping from hidden space to output space, and $\mathbf{y}^{(t)}$ is output vector. The figure 2.5 shows the graphic representation of Recurrent neural networks. There are various architectures in RNN based on the different dependencies: fully recurrent network [17] (also called vanilla RNN), Elman networks [18], Jordan networks [19], and Long Short Term Memory networks LSTM [20], Echo state networks ESNs [21], Recursive networks [22]. Most of them have been proved successful in sequential modeling in different applications. However, for some networks such as LSTM [20], fully recurrent network [17], Elman network [18] and Jordan networks [19] suffer from the difficulties of training especially for large input dimension. The computational burden is a serious problem when using back-propagation through time which will be explained in detail later. Therefore, in this thesis, we focuses on Echo state network ESN which has a sparsely connected reservoir network and only the weight of output neurons needs to be learned.

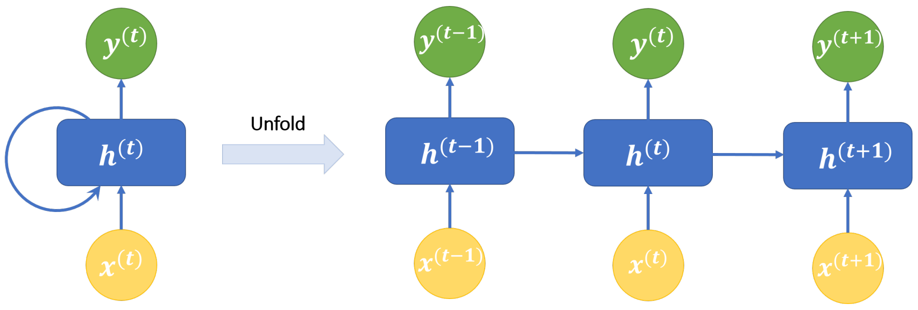


Figure 2.5: Graphic representation of Recurrent neural networks

2.2.1 Echo state networks

Echo state network provides an architecture for recurrent neural network which models the temporal mapping from input to output learned by supervised learning. ESNs follows the principle of RNN as stated in equation 2.15. The concept of Echo state networks is first introduced by Jaeger in [21] and [23], and later successfully applied on the adaptive nonlinear system identification in [24]. Figure 2.6 shows the architecture of ESNs. The reservoir network in the middle of figure 2.6 is a randomly initialized but fixed sparsely connected neural network. The connections between reservoirs are specified by the sparse matrix $\mathbf{W}$. The input unit is transformed into hidden space via weights $\mathbf{W}^{in}$. And $\mathbf{W}^{out}$ is learnable weights to map the hidden state to output [21]. There is an option path from previous outputs which is the autoregressive mechanism similar to Autoregressive model [25] by the weights $\mathbf{W}^{fb}$.

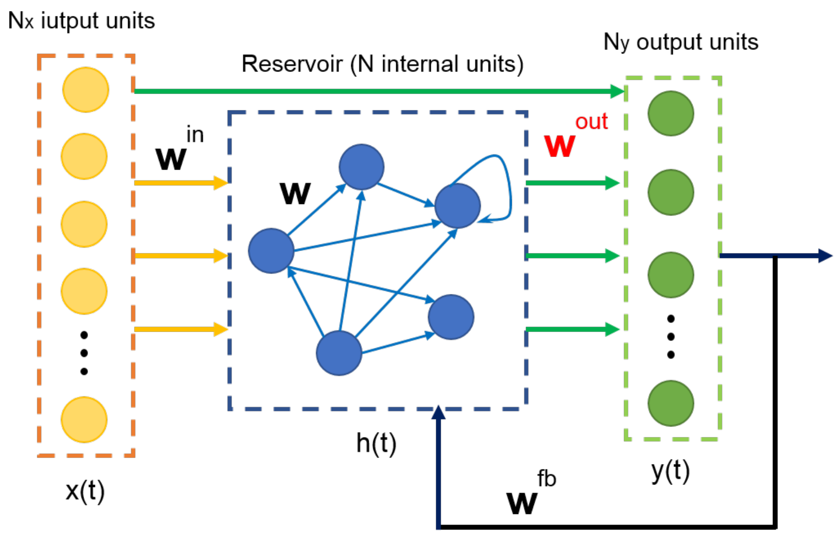


Figure 2.6: Echo state networks architecture

Therefore, the echo state $\mathbf{h}^{(t)}$ is updated by previous state $\mathbf{h}^{(t-1)}$, current input $\mathbf{x}^{(t)}$ and/or feedback from previous output $\mathbf{y}^{(t-1)}$:

$$\mathbf{h}^{(t)} = f(\mathbf{W}^{in}[1; \mathbf{x}^{(t)}] + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{W}^{fb}\mathbf{y}^{(t-1)}), t = 1, ..., T \tag{2.16}$$

where $\mathbf{x}^{(t)} \in \mathcal{R}^{N_x}$ is input signals, $\mathbf{h}^{(t)} \in \mathcal{R}^{N_h}$ is the vector of reservoir neurons, $\mathbf{y}^{(t-1)} \in \mathcal{R}^{N_y}$ is the optional previous output which could be targets or estimates and we will discuss it more in detail later. the input dimension is N^x, reservoir size is N and output size is N^y. $f(\dots)$ is an activation function, which can be chosen from linear, Tanh, sigmoid and etc. In this thesis, Tanh is used as default if not specified. $\mathbf{W}^{in} \in \mathcal{R}^{(N_h+1)\cdot N_x}$ is the input weight matrix. Bias term is also added in $\mathbf{W}^{in}$ to give the constant shift. $\mathbf{W} \in \mathcal{R}^{N_h \cdot N_h}$ is the internal reservoir connection weight matrix. And $\mathbf{W}^{fb} \in \mathcal{R}^{N_y \cdot N_h}$ is optional feedback weight matrix. $t = 1, 2, ...T$ is the discrete time instances and T is variable length of sequences. The state is usually initiated by a zero vector $\mathbf{x}^{(0)} = [0, 0, 0, \dots, 0]^T$. The notation used in this thesis follows the conventions in machine learning community which is slightly different to the original paper of ESNs [21].

If a linear readout layer is connected to hidden states, the output of network $\hat{\mathbf{y}}^{(t)}$ is harvested by the extended state vector including $\mathbf{x}^{(t)}$ and $\mathbf{h}^{(t)}$:

$$\hat{\mathbf{y}}^{(t)} = g(\mathbf{W}^{out}[1; \mathbf{x}^{(t)}; \mathbf{h}^{(t)}]) \tag{2.17}$$

where $\mathbf{W}^{out} \in \mathcal{R}^{N_y \times [1+N_x+N_h]}$ is linear readout weight matrix, $g(\dots)$ is the readout activation function. Similarly, it can be chosen from identity, tanh, sigmoid or other activation function. Here, the identity function is used as default if not specified. ESNs has very similar mathematical formulation as Elman network and Jordan network. But, the weight of reservoir connections $\mathbf{W}$ and input to reservoir $\mathbf{W}^{in}$ are initialized randomly. So only the weight matrix of output $\mathbf{W}^{out}$ is trainable. If the option path is not activated, there is no cyclic dependencies between the readout connection which gives the biggest advantage over other recurrent neural network because it eases the difficulty of training by the backpropagation through time [26]. However, in this thesis, we find that introducing the option feedback path could help reservoir to stabilize and handle the noise embedding in the input vectors if we train it appropriately although the option feedback path brings more difficulties for training. Compared to original ESNs, one-time feed-forward pass no longer has the closed form solution by linear regression or ridge regression for linear readout weight matrix $\mathbf{W}^{out}$. We will illustrate it in detail in the following sections.

Jaeger in [27] also introduces leaky rate which is similar to the concept of forget gate or time gate in other RNN methods [28]. But different to the time gate, leaky rate is a global hyper parameter. It is possible to extend it as a learnable parameter but it is out of scope of this thesis, which could be extended in the future work. The states update equation is modified from equation 2.16 to add leaky rate by:

$$\begin{aligned} \tilde{\mathbf{h}}^{(t)} = f(\mathbf{W}^{in}[1; \mathbf{x}(t)] + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{W}^{fb}\mathbf{y}^{(t-1)}) \\ \mathbf{h}^{(t)} = (1-\zeta)\mathbf{h}^{(t-1)} + \zeta\tilde{\mathbf{h}}^{(t)} \end{aligned} \tag{2.18}$$

Except from the same notation of variables as in 2.16, $\tilde{\mathbf{h}}^{(t)} \in R^{N_h}$ is the update and $\zeta \in (0, 1]$ is the leaky rate. When leaky rate is equal to 1, then the first equation above is equivalent to equation 2.16 so $\mathbf{h}^{(t)} \equiv \tilde{\mathbf{h}}^{(t)}$. Therefore, we could consider the formation without leaky rate is a special case of leaky rate as 1. If leaky rate approximates to 0, then $\mathbf{h}^{(t)} \equiv \mathbf{h}^{(t-1)} \equiv \dots \equiv \mathbf{h}^{(0)}$ so that all hidden states take the memory from previous state and ignore the current input completely. By these

two extreme examples, we could understand how leaky rate controls the trade-off between preserving previous information and taking new information from inputs and feedback outputs.
Figure 2.7 illustrates the difference of training process between ESNs (without feedback path) and other gradient-based RNN [29]. The bold line in figure 2.7 are the weights that need to be updated iteratively. For the gradient-based training method, the total number of gradients needs be calculated is $N_x \cdot N_h + N_h^2 + N_h \cdot N_y$. As for ESN, only the readout weight matrix $\mathbf{W}^{out}$ is trainable so that only $N_h \cdot N_y$ weights have to be updated. And as discussed before, without feedback optional path, ESNs could harvest all the reservoir state by one forward pass and use the linear regression to update the read out layer, which makes convergence much faster than that of other RNN architectures.

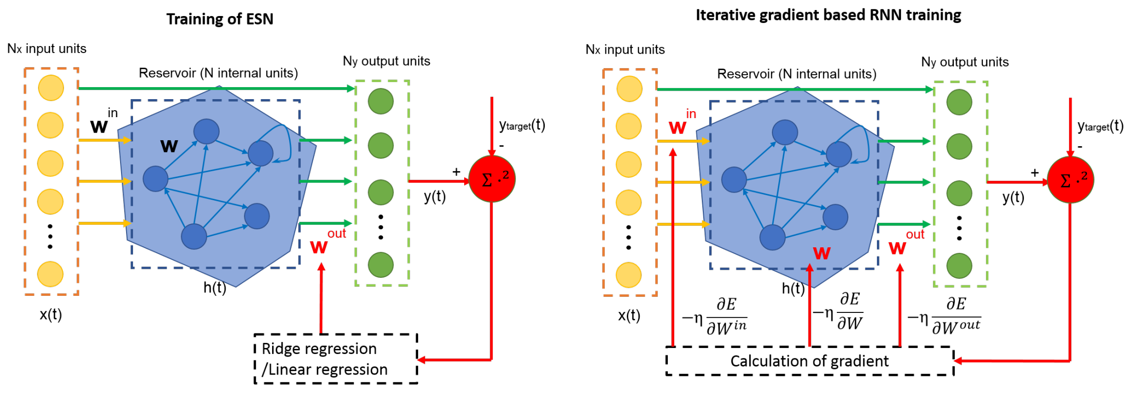

Training of ESN
Nx input units
Reservoir (N internal units)
Ny output units
W in
W
W out
h(t)
x(t)
y(t)
ytarget(t)
Ridge regression /Linear regression
Iterative gradient based RNN training
Calculation of gradient
$-\eta\frac{\partial E}{\partial W^{in}}$
$-\eta\frac{\partial E}{\partial W}$
$-\eta\frac{\partial E}{\partial W^{out}}$

Figure 2.7: The difference between gradient based and ESN training of RNN

In summary, the general training procedure of ESNs (without feedback path) is:

- Specify hyper parameters for ESNs including reservoir size, spectrum radius, sparsity of reservoir connectivity, spectrum radius, distribution of nonzero elements, leaky rate.
- Generate a randomized weight matrices $\mathbf{W}^{in}$, $\mathbf{W}$ according to the specified hyper parameters.
- Using inputs to $\mathbf{x}^{(t)}$ collect states $\mathbf{h}^{(t)}$ for all time instances using equation 2.18 (without feedback $\mathbf{y}$) by running forward pass of networks.
- Optimize readout weights $\mathbf{W}^{out}$.

During test time, states are calculated by equation 2.18 and estimation $\hat{\mathbf{y}}^{(t)}$ are calculated with the trained $\mathbf{W}^{out}$ using equation 2.17. In the subsequent sections, more deeper insight regarding training of ESNs with/without feedback path and model hyper-parameters will be discussed in details.

2.2.2 Optimizing readout layer without feedback path

As mentioned in previous session, the inputs $\mathbf{x}^{(t)}$ and states $\mathbf{h}^{(t)}$ could be collected through the forward pass of networks for all time instances. In order to make it more convenient to optimize weight over all samples, we rearrange all the vectors for each instance into matrix form:

$$\mathbf{X} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ \mathbf{x}^{(1)} & \mathbf{x}^{(2)} & \dots & \mathbf{x}^{(T)} \\ \mathbf{h}^{(1)} & \mathbf{h}^{(2)} & \dots & \mathbf{h}^{(T)} \end{bmatrix} \tag{2.19}$$

where $\mathbf{X} \in \mathcal{R}^{(1+N_x+N_h)\cdot T}$, T is the variable length of sequence. To be noted, the matrix $\mathbf{X}$ here is not only for reservoir states but also the constant and input vector $\mathbf{x}^{(t)}$.
We also collect all corresponding target and arrange them to matrix form:

$$\mathbf{Y}_{target} = [\mathbf{y}^{(1)}_{target}, \dots, \mathbf{y}^{(T)}_{target}] \tag{2.20}$$

where $\mathbf{Y}_{target} \in \mathcal{R}^{N_y \cdot T}$, and each element is column vector with length of N_y.
Following the previous definition of all matrices, the equation 2.17 could be reformulated as matrix term with $\mathbf{X}$:

$$\hat{\mathbf{Y}} = [\hat{\mathbf{y}}^{(1)}, \hat{\mathbf{y}}^{(2)}, \dots, \hat{\mathbf{y}}^{(T)}] = \mathbf{g}(\mathbf{W}^{out}\mathbf{X}) \tag{2.21}$$

where $\hat{\mathbf{Y}} \in R^{N_y \cdot T}$ and each element is a column vector with length of N_y.

2.2.2.1 Optimization metric

In order to find optimal weight matrix $\mathbf{W}^{out}$ in equation 2.21, we need to specify the objective function. For regression problem, there are several common metrics: mean squares error, mean absolute error, mean squares log error, median absolute error, coefficient of determination.
All metrics mentioned could be used to solve regression problem to find the optimal readout weight $\mathbf{W}^{out}$ to make estimation as close to target. However, due to the fact that sum-of-squares error is differentiable everywhere and convex, and easy to calculate the gradient (if we use the gradient based optimization method), sum-of-squares error function is nature to use to minimize the square error between $\hat{\mathbf{Y}}$ and $\mathbf{Y}_{target}$:

$$\begin{aligned} Err(\hat{\mathbf{Y}}, \mathbf{Y}_{target}) &= \sum_{t=1}^{T} (Err(\hat{\mathbf{y}}^{(\mathbf{t})}, \mathbf{y}^{(t)}_{target})) \\ &= \sum_{n=1}^{T} \sum_{i=1}^{N_y} (\hat{y}^{(t)}_i - y^{(t)}_{i,target})^2 \end{aligned} \tag{2.22}$$

The standard batch supervised training of ESN is to drive all the inputs and collect the reservoir states over the all sequences and optimize the linear output weights $\mathbf{W}^{out}$ by the target output $\mathbf{Y}_{target}$. So it is a regression problem to match $\mathbf{y}^{(t)}_{target}$ to $\hat{\mathbf{y}}^{(t)}$ as well as possible. There are several stable solutions for sum-of-square error: linear regression, general linear models, heteroscedastic models and so on. Here, we only discuss the linear regression model using least-squares estimation technique.

2.2.2.2 Linear regression and Ridge regression for linear readout layer

In the previous section, we mentioned that the readout layer can be chosen from linear function, multi-layer perceptron neutron network or other network architectures. If the settings of hyper parameters is appropriately chosen, the augmented state after reservoir computation should be linear to the target in the most cases [30]. Only if there is no optimal hyper parameters found for linear readout layer, other network architecture will be considered. In this thesis, we find linear readout layer is adequate. The nonlinear mapping using neural network and other architecture is out of the scope of this thesis, but reader could refer to [31], [32] and [33] for more complicate architecture.

If we could collect all reservoir states and inputs, and use equation 2.21, the optimization is formulated as linear regression problem with identity function as the activation function g:

$$\hat{\mathbf{Y}} = \mathbf{W}^{out}\mathbf{X} \tag{2.23}$$

The closed form solution is:

$$\begin{aligned}\mathbf{W}^{out} &= \mathbf{Y}^{target}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1} \\ &= \mathbf{Y}^{target}\mathbf{X}^{\dagger}\end{aligned} \tag{2.24}$$

where $\mathbf{X}^{\dagger} \equiv \mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}$ is Moore-Penrose pseudo-inverse of matrix $\mathbf{X}$. Sometimes, the term $\mathbf{X}\mathbf{X}^T$ might not be invertable which makes the solution extremely unstable. Moore-Penrose pseudo-inverse could help to give stable numerical solution. However, the weights tend be extremely large to make mapping less smooth and prediction more variant. Therefore, it is common to add a regularization term in error function:

$$Err(\hat{\mathbf{Y}}, \mathbf{Y}_{target}) + \lambda E_w \tag{2.25}$$

If we use L2 regularization term which is the sum-of-squares of weight vector, the optimization becomes to ridge regression:

$$Err(\hat{\mathbf{Y}}, \mathbf{Y}_{target}) + \lambda \|\mathbf{W}\|_2^2 \tag{2.26}$$

where λ is the regularization trade-off. And the closed form solution becomes:

$$\mathbf{W}^{out} = \mathbf{Y}^{target}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1} \tag{2.27}$$

This solution could mitigate the instability problem mentioned before since $\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I}$ is always invertable.

If the regularization term is L1, the optimization problem becomes Lasso. However, there is no closed form solution for Lasso and also difficult to be applied for gradient based optimization method. Besides, Lasso tends to shrink weights to exactly zero compared to ridge regression, which leads to sparse weight matrix.

Seen from equation (2.27), although the size of the product of matrix $\mathbf{Y}^{target}\mathbf{X}^T$ and $\mathbf{X}\mathbf{X}^T$ are independent to the size of training data, it requires huge memory to store all the collected vector in matrix $\mathbf{Y}^{target}$ and $\mathbf{X}^T$. Mantas in [27] mentioned one extension of ESN approach to train several small ESNs in parallel and average the output from each individual for the big data problem. However, it still does not solve the memory issue from root and also have scaling issue. Therefore, we have to resort to mini batch technique that we randomly pick up a small batch of data and perform optimization, and move to next batch and learn weights incrementally. Therefore, we have to use gradient-based method to optimize the weight matrix on mini-batches.

2.2.2.3 Stochastic gradient descent

As we mentioned in the last section, we have to resort to incremental learning method to deal with large scale of data. Stochastic gradient descent is an iterative method to optimize a differentiable objective function. It uses the stochastic approximation of gradient descent optimization, and samples in each iteration are randomly selected which stabilizes the training process.
If we use L2 regularization, we extend the terms of error function (the factor $\frac{1}{2}$ is for the convenience of deriving gradient):

$$\begin{aligned} E_{tot} =& \frac{1}{2}E(\hat{\mathbf{Y}}, \mathbf{Y}_{target}) + \frac{1}{2}\lambda\|\mathbf{W}\|_2^2 \\ =& \frac{1}{2}(\mathbf{Y}^{target} - \mathbf{W}^{out}\mathbf{X})^T(\mathbf{Y}^{target} - \mathbf{W}^{out}\mathbf{X}) + \frac{1}{2}\lambda\mathbf{W}^{out^T}\mathbf{W}^{out} \end{aligned} \tag{2.28}$$

The gradient of error on weight $\mathbf{W}^{out}$ is easy to calculated:

$$\frac{\partial \mathbf{E}}{\partial \mathbf{W}^{out}} = -(\mathbf{Y}^{target} - \mathbf{W}^{out}\mathbf{X})\mathbf{X}^T + \lambda\mathbf{W}^{outT} \tag{2.29}$$

Each iteration, the gradient is calculated and weight matrix is updated by:

$$\mathbf{W}^{out} := \mathbf{W}^{out} - \eta\frac{\partial \mathbf{E}}{\partial \mathbf{W}^{out}} \tag{2.30}$$

where η is learning rate. Decaying learning rate is used here for faster but stable convergence.
In [27], Jarger compares the stochastic gradient descent and Pseudo-batch output weight update mentioned in previous section. It is found that the first method is slower in convergence but more stable while Pseudo-batch method gives faster convergence but with an increasing risk of instability. Besides, once the feedback path is introduced, the optimization does not have closed-form solution anymore. Therefore, we select stochastic gradient descent to optimize weight matrix $\mathbf{W}^{out}$.

2.2.3 Output feedback path

It is nature to include the feedback connection in ESN since most of dynamics system has autoregressive mechanism in the signals. Therefore, it is necessary to include feedback from output to reservoirs by $\mathbf{W}^{fb}$ in equation 2.18. The same principle applies to initialize $\mathbf{W}^{fb}$ as to $\mathbf{W}^{in}$.
The feedback of output enhances the performance of reservoir computation since ESNs no longer only rely on input-driven dynamics but also the previous output, which brings some issues. Although $\mathbf{W}^{fb}$ is randomly initialized and not trainable, it affects the value of hidden state which indirectly affect optimal weight matrix

$\mathbf{W}^{out}$, as we see in equation 2.18. Therefore, we cannot collect all the hidden state unless we have access to the target as the feedback path. Mantas in [30] discusses two ways of training with feedback path: teacher forcing and adding state noise term $\mathbf{v}(n)$. The first strategy is to disengage the recurrent relation between reservoir and previous readout output by using target value $\mathbf{y}_{target}^{(t-1)}$ through the feedback connection. The target value $\mathbf{y}_{target}^{(t)}$ will boost learning if the model is prone to generate closely precise estimate that $\hat{\mathbf{y}}^{(t)} \approx \mathbf{y}_{target}^{(t)}$. And the training could be done by linear or ridge regression as discussed before. During test time, the real estimate $\hat{\mathbf{y}}^{(t-1)}$ has to be used for feedback path instead. However, we observe in the preliminary experiments that the distorted estimate error during test will propaganda through the feedback path if the estimate deviates from the target value and reservoir is not self-stable anymore. ESNs are trained with previous targets and incapable to tackle the estimation error itself so that the estimation error propagate through time and eventually estimate largely deviates to the target value after short period of time even though estimate is quite close to target value at beginning.
The second strategy in [30] is to add a noise term on the target value so that ESNs never get the exact target value but enforce network to account itself that there is always uncertainty on the previous estimate. In some extend, the uncertain previous estimate enhances the generality of model. However, the benefit of feedback then is substantially mitigated if the noise-to-signal ratio is too large and it makes difficult to evaluate estimates if the discrepancy of estimate to target is from the added noise or actual estimate error.
Although there are several proposed methods like BackPropagation-DeCorrelation BPDC in [34] and FORCE in [35] try to solve feedback reservoir computation for the iterative training problem, the current solutions still lack of capacity to deal with the arbitrary signal. Therefore, we would like to use the estimates during both training and test time so that the accuracy of estimate will not degrade.
It comes with the cost that the training of ESN no longer can be performed by linear or ridge regression if the estimates are used in feedback path. In order to train ESN with feedback path of the estimates, we propose the iterative training method shown in figure 2.8. For each mini-batch, there are few iterations of feedforward pass and backpropagation update. In each iteration, the process runs feedforward pass to collect all the hidden states $\mathbf{h}^{(t)}$ by the current network parameters $\boldsymbol{\theta}$ and rearrange them to the matrix form as equation 2.19. If it is the first iteration for the mini-batch, ESNs use the target value $\mathbf{Y}_{target}$ in feedback path as teacher forcing. Otherwise they use the estimates $\hat{\mathbf{Y}}$. With the rearranged matrix $\boldsymbol{X}$ and target $\mathbf{Y}_{target}$, the readout weights $\mathbf{W}^{out}$ are updated by stochastic gradient descent. For next iteration, ESNs run the feedforward pass using the new optimized $\mathbf{W}^{out}$ and estimates $\hat{\mathbf{Y}}$ to harvest all hidden states $\mathbf{h}^{(1:T)}$, and new optimization runs on the newly collected states and so on. After some iterations, the optimization converges and stabilizes and training moves to next mini-batch. The convergence criteria can be selected by specifying the number of iterations or evaluating the estimation error and stopping training when it decreases to a certain level. Here, we use the terminal criteria that when the root mean square error between target and estimate is lower than 0.05. By this way, the model learns to stabilize itself in practice and small deviation from target will not bring the catastrophic error propagation as teaching

force method.

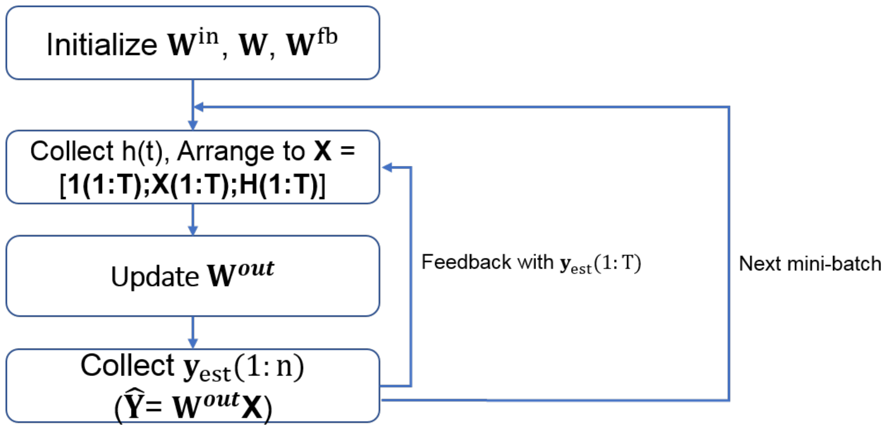

Initialize $\mathbf{W}^{\text{in}}$, $\mathbf{W}$, $\mathbf{W}^{\text{fb}}$
Collect h(t), Arrange to $\mathbf{X}$ = [1(1:T);X(1:T);H(1:T)]
Update $\mathbf{W}^{out}$
Collect $\mathbf{y}_{\text{est}}(1{:}\,\text{n})$ ($\hat{\mathbf{Y}} = \mathbf{W}^{out}\mathbf{X}$)
Feedback with $\mathbf{y}_{\text{est}}(1{:}\,\text{T})$
Next mini-batch

Figure 2.8: Iterative feedback training of ESN

Different to the classic ESNs, the optional feedback path could not only feedback one previous estimate, but also several previous steps. Therefore, we introduce a hyper parameter called memory size to specify the number of previous estimates that is used in the feedback path.
In summary, the general training procedure of ESNs with feedback path is:

- Specify hyper parameters for ESNs including reservoir size, spectrum radius, sparsity of reservoir connectivity, spectrum radius, distribution of nonzero elements, leaky rate, and memory size.
- Generate a randomized weight matrices $\mathbf{W}^{in}$, $\mathbf{W}$, and $\mathbf{W}^{fb}$ according to the specified hyper parameters.
- Select a mini-batch of data randomly
- Using inputs $\mathbf{x}^{(t)}$, and estimate $\hat{\mathbf{y}}^{(t-1)}$ collect states $\mathbf{h}^{(t)}$ for all time instances using equation 2.18 by forward pass of networks with current weights.
- Optimize readout weights $\mathbf{W}^{out}$ by the mini-batch. If the stop criteria is not fulfilled, return to step 4; otherwise, return to step 3 to start with a new mini-batch.

2.2.3.1 Evaluation metrics

As mentioned in optimization metrics section, there are multiple evaluation metrics. We use the mean squares error for optimization since it is differentiable and easy to calculate gradient. However, for the evaluation metrics, we do not have such constraints, but it should reflect the performance required for the application. Other metrics like mean absolute error helps us understand the performance of the model that how much is the estimate deviates to the target. Besides, small deviation to the target is allowed since it might not make huge difference for later decision making algorithm. For most scenarios, it is acceptable that estimate has error within the range of ± 0.1. Therefore, we also define the error rate under different absolute range: 0.05, 0.1, 0.15, 0.2. The result of evaluation under different metrics will be presented in later chapter. Here we give the definition for all metrics (now we assume the estimate is single dimension for each time instance to simplify expression, but it can easily be extended to multiple output dimensions):

- Mean squares error:

$$RMSE = \sqrt{\frac{\sum_{t=1}^{T} e_i^2}{T}} = \sqrt{\frac{\sum_{t=1}^{T}(\hat{y}^{(t)} - y_{target}^{(t)})^2}{T}} \tag{2.31}$$

- Mean absolute error:

$$MAE = \frac{\sum_{t=1}^{T}|e_i|}{T} = \frac{\sum_{t=1}^{T}|\hat{y}^{(t)} - y_{target}^{(t)}|}{T} \tag{2.32}$$

- Error rate within 0.05:

$$\begin{aligned} ErrorRate_{0.05} &= \frac{N_{error}}{T} = \frac{\sum_{t=1}^{T}\xi^{(t)}}{T} \\ \xi_t &= \begin{cases} 1 & |\hat{y}^{(t)} - y_{target}^{(t)}| > 0.05 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{2.33}$$

- Error rate within 0.1:

$$\begin{aligned} ErrorRate_{0.1} &= \frac{N_{error}}{T} = \frac{\sum_{t=1}^{T}\xi^{(t)}}{T} \\ \xi_t &= \begin{cases} 1 & |\hat{y}^{(t)} - y_{target}^{(t)}| > 0.1 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{2.34}$$

- Error rate within 0.15:

$$\begin{aligned} ErrorRate_{0.15} &= \frac{N_{error}}{T} = \frac{\sum_{t=1}^{T}\xi^{(t)}}{T} \\ \xi_t &= \begin{cases} 1 & |\hat{y}^{(t)} - y_{target}^{(t)}| > 0.15 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{2.35}$$

- Error rate within 0.2:

$$\begin{aligned} ErrorRate_{0.2} &= \frac{N_{error}}{T} = \frac{\sum_{t=1}^{T}\xi^{(t)}}{T} \\ \xi_t &= \begin{cases} 1 & |\hat{y}^{(t)} - y_{target}^{(t)}| > 0.2 \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{2.36}$$

These metrics give us an intuitive way to interpret the performance of model, but we also find that critical range is different at different level of friction. For instance, if the target road friction is 0.8, it will not be extremely critical if the estimate is 0.7, 0.9, or 1.0. However, for the low road friction i.e. as 0.2, it is very critical that if the estimate is 0.1 or 0.3. We have not found much literature to construct evaluation metric in such complication manner, not mention for optimization. But it is possible to define a weighted loss function on the different range. The construction of more complicated optimization and evaluation metric will be left to future work.

2.2.4 Hyper parameters of ESNs

As discussed in previous section, there are multiple hyper parameters of ESNs controlling its performance. We will discuss them in detail in the following sections.

2.2.4.1 Reservoir size

One critical hyper parameter of ESNs is the size of reservoir N. According to equation 2.18, it is evident that the larger the reservoir size is, the better performance of ESNs is. Because larger reservoir size empowers it with more expressiveness of reservoir networks in ESNs, so that the transformation could be learned better. But the improvement of performance does not increase linearly to to the size of reservoirs either. Theoretically, the reservoir size can not be over the number of data samples M since ridge regression will no longer have unique solution of weight matrix and it is over-parameterized. In [36], Fabian assesses the impact of reservoir size on the performance of reservoir computation. There is a limitation of reservoir size when the storage and inversion of matrix $\mathbf{XX^T}$ in equation 2.18 becomes problematic when applying ridge regression or linear regression. Herbert in [23] shows that the memory capacity in reservoirs cannot exceed N_h in theory. For input $\mathbf{x}^{(t)}$, the estimate of lower bound of reservoir size is N_h times how many time steps the inputs should be memorized. The conclusion drawn from theory, however, does not apply to practice since there are always inter- and temporal-correlations in the variables $\mathbf{x}^{(t)}$ [23]. Besides, the forgetting curve of reservoirs makes the analysis more complicated because the forgetting is not instantaneous but gradual. Therefore, the lower bound of reservoir size can be much smaller than the N_h in practice.
However, from the implementation perspective, the size of reservoir limited by the computational power and memory storage. The computation expense of training and test of ESNs increases quadratically with the reservoir size. Therefore, the trade-off between computational cost and performance is important to take into consideration. For the practical application, the real-time performance limits the reservoir size. Under that constraint, tuning for optimal hyper parameters can be obtained by the cross-validation. We select several proposals of reservoir size as 100, 200, 300, 400, 500, 600, 700, 800.

2.2.4.2 Sparsity of Reservoir Connectivity

The original ESNs in [21] recommend to have sparse connectivity between reservoirs to achieve better performance which is proved both in theory and practical experiments. However, in [30] Mantas mentions the sparsity of reservoirs does not affect the performance so much. So the parameter is less important to be optimized in practice.
With fixed reservoir size, the computational cost of reservoir updates only linearly relates to the number of reservoirs that are inter-connected if the programming environment supports efficient sparse matrices operation. However, the low level computational optimization is beyond the scope of this thesis. So the sparsity of reservoir connection will not affect so much and it will be optimized by the cross-validation.

2.2.4.3 Spectral Radius

Spectral radius is one of the most important hyper parameters of ESNs to control the reservoir connection matrix $\mathbf{W}$. The spectral radius $\rho(\mathbf{W})$ is defined as the maximal absolute eigenvalue of $\mathbf{W}$. It scales the nonzero elements distribution width in matrix $\mathbf{W}$. $\mathbf{W}$ is randomly initialized for each element within certain range first. Then the eigenvalue of initialized matrix is calculated. In order to make the spectral radius of $\mathbf{W}$ as we defined, we need to shift the matrix from the maximum eigenvalue of initialized matrix to the spectral radius by dividing $\mathbf{W}$ by its maximum eigenvalue then scaling to $\rho(\mathbf{W})$.
Theoretically, large spectral radius leads to the violation of echo state property [30]. However, in practice, spectral radius larger than 1 is not a necessary condition due to inter- and temporal-correlation in $\mathbf{x}^{(t)}$ as mentioned before. $\rho(\mathbf{W})$ can be selected to maximize performance by cross validation and 1 serves as the starting point.

2.2.4.4 Distribution of Nonzero Elements

The matrix $\mathbf{W}$ is initialized with nonzero elements from uniform distribution, or Gaussian distribution or discrete bi-valued distribution. Gaussian distribution is the most nature choice. The analysis shows the choice of different distribution does not matter to performance much [30]. Therefore, we choose unit Gaussian distribution $\mathcal{N}(0,1)$.
The input matrix $\mathbf{W}^{in}$ and feedback matrix $\mathbf{W}^{fb}$ are generated from the same distribution as matrix $\mathbf{W}$ but without sparsity.

2.2.4.5 Leaky Rate

Leaky rate serves as the speed of reservoir update dynamics as seen in equation 2.18. The reservoir update dynamics can be expressed [30]:

$$\dot{\mathbf{h}}^{(t)} = -\mathbf{h}^{(t)} + tanh(\mathbf{W}^{in}[1;\mathbf{x}^{(t)}] + \mathbf{W}\mathbf{h}^{(t-1)}) \tag{2.37}$$

If discretizing the equation above:

$$\frac{\Delta h}{\Delta t} = \frac{\mathbf{h}^{(t+1)} - \mathbf{h}^{(t)}}{\Delta t} = \dot{\mathbf{h}}^{(t)} \tag{2.38}$$

If we combine equations 2.37 and 2.38, we could find that Δt in the equation of dynamics equals to leaky rate. Therefore, leaky rate can be regarded as the time interval of two consecutive steps in the discrete representation. The optimal leaky rate ζ should match to the speed of dynamics of $\mathbf{h}^{(t)}$ and $\mathbf{y}^{(t)}_{target}$. In addition, the leaky rate integration is also considered as applying simple low-pass filter. In some special setting, a small leaky rate ζ enforces slow dynamics of $\mathbf{h}^{(t)}$ and contributes to longer duration of short-term memory. Since dynamics of hidden states is really difficult to analyze, the trail-and-error method is used to select the optimal leaky rate.

2.2.4.6 Memory size

In the most literature, the feedback path only bring back one previous estimate $\hat{\mathbf{y}}^{(t-1)}$ or ground truth $\mathbf{y}_{target}^{(t-1)}$. But if we feed back reservoirs with more previous estimation $\hat{y}^{(t-2)}$, $\hat{\mathbf{y}}^{(\mathbf{t-3})}$, $\hat{\mathbf{y}}^{(\mathbf{t-4})}$, …,$\hat{\mathbf{y}}^{(\mathbf{t-N_t})}$, ESNs could converge quicker than single estimate. We define N_t here as memory size.
However, the memory size cannot grow into infinite and it is limited by the length of sequences. As for optimal memory size, the feedback path is supposed to converge quicker but not propagate the estimation error into later estimation. In the result chapter, we will compare the result under different memory sizes to select the optimal value for the data.

2.3 Temporal pattern recognition for prior knowledge

Except from the on-board estimation algorithm based on ESN, we also study on the temporal pattern of road friction on different locations. If the underlying temporal pattern could be extracted, it could serve as a prior knowledge to help on-board estimation algorithm to estimate friction more accurately. If we are able to find such underlying pattern for each geometric location, then we could build up a pattern map to understand the spatial-temporal pattern. In this thesis we propose a hidden Markov model based method to build a pattern map and later using clustering to analyze the result of pattern map. The detailed description of the algorithm will be presented in the section 3.2. Here, we focus on the fundamentals of Hidden Markov Model.

2.3.1 Hidden Markov model

Hidden Markov Models HMM has been one of the most popular generative model of sequence. Rabiner proposes hidden Markov Model first systematically in [15]. It is a statistical first-order Markov model that is assumed that hidden states follows a Markov process. It can also be considered as a mixture model that hidden states control the mixture components for each observation and follow a Markov process. In hidden Markov model, the states are not directly visible, but the observations, dependent on the states, are visible. Let $\{o_t\}_{t=1}^{T}$ be observation sequence from some underlying hidden state sequence $\{i_t\}_{t=1}^{T}$ and the observation o_t is only dependent on the state i_t. Figure 2.9 shows the graphic representation of hidden Markov model of hidden states and observations where i_t stands for hidden state variables at time t and o_t is corresponding observation variables.

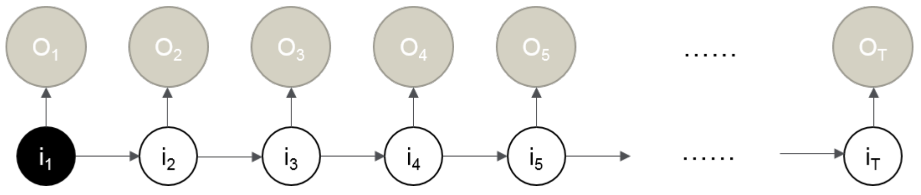


Figure 2.9: Graphic model representation of hidden Markov model

In summary, there are two main assumptions used in HMM:

Assumption 1 The Markov property: the conditional probability distribution of each hidden state i_{t+1} given all previous states is equal to its conditional probability distribution given only previous state i_t:

$$p(i_{t+1}|i_1, i_2, i_3, \ldots, i_t) = p(i_{t+1}|i_t) \tag{2.39}$$

Assumption 2 The observation o_{t+1} is independent to other previous observations and states, but only dependent on corresponding hidden state i_{t+1}:

$$p(o_{t+1}|o_t, o_{t-1}, \ldots, o_1, i_{t+1}, i_t, \ldots, i_1) = p(o_{t+1}|i_{t+1}) \tag{2.40}$$

If we assume the discrete symbols for the observations, figure 2.10 shows the hidden Markov model transition and emission with 3 states and 4 observation symbols, where q_n stands for different state symbols and v_k represents for the discrete observation symbols. The hidden states sample from N_q fixed states set $Q \in \{q_1, q_2, \ldots, q_{N_q}\}$. Similarly, observation symbols sample from N_v symbols set $V \in \{v_1, v_2, \ldots, v_{N_v}\}$. In this specific example, there are 3 states $Q = \{q_1, q_2, q_3\}$ and 4 observation symbols $V = \{v_1, v_2, v_3, v_4\}$. The edge between states gives the transition probability from one state to another which is denoted by a_{ij}, while the edge between state and observation gives the emission probability that state q_i generate the observation v_k which is denoted by $B_i(v_k)$.

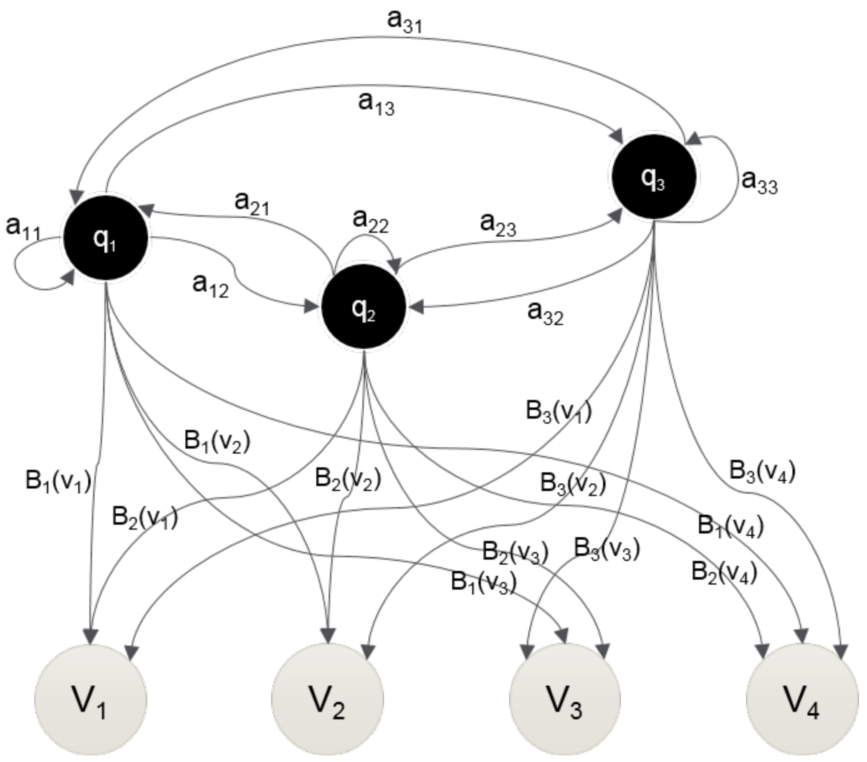


Figure 2.10: Hidden Markov model transition and emission

Transition matrix $\mathbf{A} \in \mathcal{R}^{N \times N}$ collects all the transition probability from state i to state j, $\mathbf{A}[i,j] = a_{ij} = p(i_t = q_j | i_{t-1} = q_i)$, $i = [1, 2, ..., N_q], j = [1, 2, ..., N_q]$. Emission matrix $\mathbf{B}$ gives the observation symbol distribution given state n, $\mathbf{B}[n,k] = b_n(k) = p(o_t = v_k | i_t = q_n) \quad n = [1, 2, ..., N_q]$, $k = [1, 2, ..., N_v]$. Except from transition probability and emission probability, $\mathbf{\Pi}$ gives the initial probability vector for the state initialization. $\mathbf{\Pi} = [\pi_1, \ldots, \pi_{N_q}]$ where $\mathbf{\Pi}[n] = \pi_n = p(i_1 = q_n)$, $n = [1, 2, ..., N_q]$. Therefore, total parameters for HMM are $\boldsymbol{\theta} = (\mathbf{A}, \mathbf{B}, \mathbf{\Pi})$. Given model $\boldsymbol{\theta}$, state sequence $\{i_t\}_{t=1}^{T} = [i_1, i_2, \ldots, i_T]$ and observation sequence $\{o_t\}_{t=1}^{T} = [o_1, o_2, ..., o_t]$ are generated by:

1. Choose initial state i_1 according to initial state distribution $\mathbf{\Pi}$;
2. Set t = 1;
3. Choose o_t according to the observation probability distribution $b_{o_t}(i_t)$, given i_t;
4. Choose i_{t+1} according to the transition probability $a_{i_t, i_{t+1}}$, $i_{t+1} \in Q$;
5. Set t = t+1; return to step 3 until t = T to terminate the process

In order to learn the transition and emission of HMM model, it requires to specify the number of states N_q and discrete symbols set V. Here, we specify the model with 3 hidden state and 10 observation symbols which are $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$. Before training of the model, we need to round the ground truth road friction to those symbols we defined. And for our application, the initial state probability distribution is not important since sequences are long enough that initial probability does not affect sequence too much.
To train HMM, there are three main problems to be solved:

Problem 1 To calculate the probability of observation sequence $Pr(\mathbf{O}|\boldsymbol{\theta})$ given the model $\boldsymbol{\theta}(\mathbf{A}, \mathbf{B}, \mathbf{\Pi})$

Problem 2 To predict optimal state sequence $\mathbf{I}$ given the observation sequence $\mathbf{O}$

Problem 3 To adjust model parameter $\boldsymbol{\theta}$ to maximize probability $Pr(\mathbf{O}|\boldsymbol{\theta})$

2.3.1.1 Solutions for three problems

For each problem, there are some established algorithms. Forward-backward algorithm is efficient to calculate the conditional probability $Pr(\mathbf{O}|\boldsymbol{\theta})$ of observation sequences without emulating all the possible state sequences. As for the second problem, there are two commonly used methods that are maximum probability algorithm and Viterbi algorithm. Maximum probability algorithm does not take global trellis structure into consideration but maximizes the state probability for time instance separately. Viterbi algorithm finds the single best state sequence with highest probability but it has more computational expense. Problem 3 is the most significant problem to find model parameters. The iterative procedure as Baum-Welch method or gradient techniques for optimization could be implemented to have decent trade-off between performance and computation speed. Our goal is to to understand the probability distribution of transition and emission given the observation sequence which involves all three problems.

Problem 1 First problem is to evaluate probability of observation sequence given the model parameters. The solution determines how good the model matches the observation later for problem 3. In order to calculate the conditional probability, we need to know states sequence $\mathbf{I}$ since the emission probability is conditioned on the states. The conditional probability of observation becomes if the state sequence assumes $\mathbf{I}$:

$$Pr(\mathbf{O}|\boldsymbol{\theta}, \mathbf{I}) = b_{o_1}(i_1) \cdot b_{o_2}(i_2) \dots b_{o_T}(i_T) \tag{2.41}$$

The probability of the state sequence I:

$$Pr(\mathbf{I}|\boldsymbol{\theta}) = \pi_{i_1} \cdot a_{i_1,i_2} \cdot a_{i_2,i_3} \dots a_{i_{T-1},i_T} \tag{2.42}$$

The joint probability $Pr(\mathbf{O}, \mathbf{I}|\boldsymbol{\theta})$ is the product of $Pr(\mathbf{O}|\boldsymbol{\theta}, \mathbf{I})$ and $Pr(\mathbf{I}|\boldsymbol{\theta})$. The marginal probability of sequence $Pr(\mathbf{O}|\boldsymbol{\theta})$ marginalizes joint probability summing over all possible state sequences:

$$Pr(\mathbf{O}|\boldsymbol{\theta}) = \sum_{allpossible\mathbf{I}} Pr(\mathbf{O}, \mathbf{I}|\boldsymbol{\theta}) = \sum_{allpossible\mathbf{I}} Pr(\mathbf{O}|\boldsymbol{\theta}, \mathbf{I}) \cdot Pr(\mathbf{I}|\boldsymbol{\theta}) \tag{2.43}$$

However, it is computationally unfeasible to emulate and sum over all possible state sequence instances since the space defined by all combination of possible state choice is too large. The complexity of computation is $\mathcal{O}(TN_q^T)$. Therefore, we have to resort other optimal way. In [15], the forward-backward algorithm is to simplify the formulation. The forward variable $\gamma_t(n)$ is defined as the joint probability of observation sequence until t and state i_t at t given model $\boldsymbol{\theta}$:

$$\gamma_t(n) \triangleq Pr(o_1, o_2, \ldots, o_t, i_t = q_n|\boldsymbol{\theta}) \tag{2.44}$$

Therefore, we could solve forward variable inductively:

$$\begin{aligned} &\alpha_1(n) = \pi_n b_n(o_1), \qquad 1 \le n \le N_q \\ &\alpha_{t+1}(m) = [\sum_{n=1}^{N_q} \alpha_t(n) a_{n,m}] b_m(o_{t+1}) \quad \text{for t = 1, 2,} \ldots, T-1 \quad 1 \le m \le N_q \end{aligned} \tag{2.45}$$

Therefore, the marginal distribution becomes $Pr(\mathbf{O}|\boldsymbol{\theta}) = \sum_{n=1}^{N_q} \alpha_T(n)$. The computation complexity decrease from $\mathcal{O}(TN_q^T)$ to $\mathcal{O}(N_q^2 T)$. The complexity drop substantially especially when the sequence is long.
In the same manner, the backward variable $\beta_t(n)$ is defined as conditional distribution of future observation given state i_t:

$$\beta_t(n) = Pr(o_{t+1}, o_{t+2}, \ldots, o_T|i_t = q_n, \boldsymbol{\theta}) \tag{2.46}$$

So that

$$\begin{aligned} &\beta_T(n) = 1, \qquad 1 \le n \le N_q \\ &\beta_t(n) = \sum_{m=1}^{N_q} a_{n,m} b_m(o_{t+1}) \quad \text{for t = T-1, T-2,} \ldots, 1 \quad 1 \le n \le N_q \end{aligned} \tag{2.47}$$

Similarly, the computation of backward variable has complexity of $\mathcal{O}(TN_2^T)$.

Problem 2 There are several possible way to solve problem 2. One of the best algorithm is called Viterbi algorithm [37]. There are 4 formal steps in Viterbi algorithm:

- Step 1 - Initialization

$$\begin{aligned} &\delta_1(n) = \pi_n b_n(o_1), \qquad 1 \le n \le N_q \\ &\Psi_1(n) = 0 \end{aligned} \tag{2.48}$$

- Step 2 - Recursion For $2 \le t \le T$, $1 \le m \le N_q$

$$\begin{aligned} \delta_t(m) &= \max_{1 \le n \le N_q} [\delta_{t-1}(n) a_{n,m}] b_m(o_t) \\ \Psi_t(m) &= \underset{1 \le n \le N_q}{\text{argmax}} [\delta_{t-1}(n) a_{n,m}] \end{aligned} \tag{2.49}$$

- Step 3 - Termination

$$\begin{aligned} P^* &= \max_{1 \le n \le N_q} [\delta_T(n)] \\ i_T^* &= \underset{1 \le n \le N_q}{\text{argmax}} [\delta_T(n)] \end{aligned} \tag{2.50}$$

- Step 4 - Sequence backtracking

$$i_t^* = \Psi_{t+1}(i_{t+1}^*) \quad For t = T-1, T-2, \ldots, 1 \tag{2.51}$$

The Viterbi algorithm is similar to forward-backward calculation in implementation. But the maximization in recursion process is to maximize on the previous state. The complexity of algorithm is $\mathcal{O}(N_q^2 T)$.

Problem 3 The third problem is to adjust the model parameters $\boldsymbol{\theta}$ to maximize the probability of observation sequence according to the evaluation method defined in problem 1. This is the most difficult problem among three and there is no known analytic way to solve the maximum likelihood model directly [15]. However, we could resort to some iterative procedure to approximately find the optimal model by such as Baum-Welch method or gradient techniques for optimization [15]. Here we only discuss the iterative procedure - Baum-Welch since it is easy to implement and understand.

So Baum-Welch re-estimation iterative process is quite simple: in each iteration, the most possible state sequence is estimated by problem 2, then the parameters is adjusted according to the optimal state sequence and observation sequence, the new model $\bar{\boldsymbol{\theta}}$ is evaluated by problem 1 to compare to previous model $\boldsymbol{\theta}$. If the marginal probability of observation sequence gets improved, then the parameters are updated by $\bar{\boldsymbol{\theta}}$. If we iteratively use the $\bar{\boldsymbol{\theta}}$ to replace $\boldsymbol{\theta}$ and repeat the reestimation procedure, we could improve the model until the optimization has reach the optimality. The reestimation formulas for $(\mathbf{A}, \mathbf{B}, \boldsymbol{\Pi})$ are:

$$
\begin{aligned}
\bar{\pi}_n &= \gamma_1(n), \qquad 1 \le i \le N_q \\
\bar{a}_{ij} &= \frac{\sum_{t=1}^{T-1} \nu_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(j)} \\
\bar{b}_j(k) &= \frac{\sum_{t=1}^{T} I(o_t = v_k)\gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}
\end{aligned}
\tag{2.52}
$$

where ν and γ is defined by:

$$
\begin{aligned}
\nu_t(i,j) &= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{Pr(\mathbf{O}|\boldsymbol{\theta})} \\
\gamma_t(i) &= \frac{\alpha_t(i)\beta_t(i)}{\sum_j \alpha_t(j)\beta_t(j)}
\end{aligned}
\tag{2.53}
$$

So the reestimation formulas for π_i is the probability of state q_i at $t = 1$. The reestimation for a_{ij} is the ratio of expected number of transition from state q_i to q_j divided by the total number of transition from state q_i. The estimation for $b_j(k)$ is the ratios of expected number of observing v_k given the state q_j.

3
Methods

In this chapter, we will first introduce the data used in this thesis. The road friction has underlying temporal pattern and it could give some prior information. We propose the HMM-based clustering method to analyze the underlying spatial-temporal pattern. The techniques involved in the pipeline will be illustrated in details: data aggregation, cross-validation, Kullback-Leibler divergence, Hierarchical clustering. Meanwhile, in order to improve real-time performance of on-board estimation algorithm, the feature selection technique is used to decrease dimensionality of data without compromising performance substantially.

3.1 Data

The data used in this thesis is 5808 expedition sequences from Sweden, Germany and other European countries between 2014 to 2016. There are over 7.5 million data samples. Figure 3.1 shows the geometric trajectory of all data. Except from filtered sensor measurements on the car like vehicle states from IMU, the data also includes the intermediate estimates such as tire forces, slip and rack force and so on. Since the amount of raw data is enormous, the processed data is shrink to 83 features that are relevant and also includes meta-data like tire type, surface inference and road type as additional features. Table A.1 in the Appendix lists all features in the processed data. The meta data annotated includes road type, surface type, tire type. The road type is encoded as "Asphalt", "Gravel", "Lake" and surface is encoded as "Dry", "Ice", "Mixed", "Snow" and "Wet", and tire type is encoded into 20 different categories as well.

The reference of road friction value is obtained by the test engineer's annotation and measured by a special equipment. The value ranges from 0 to 1. 1 means high road friction, while 0 means extremely low road friction.
For the study of spatial-temporal pattern of road friction, only the reference of road friction, longitude, latitude and timestamp are used. The sequences are split to small sub-sequences according to geometric locations. For the study of on-board estimation algorithm, however, it is too expensive to use all listed features in A.1. Therefore, feature space needs to be deduced to guarantee the real-time performance without compromising performance of learning algorithm. Feature selection is applied which will be discussed later. In addition, all sequences with selected features are further split to shorter sequences that each has 10000 samples corresponding to 100 seconds in order to make training and analysis more convenient.

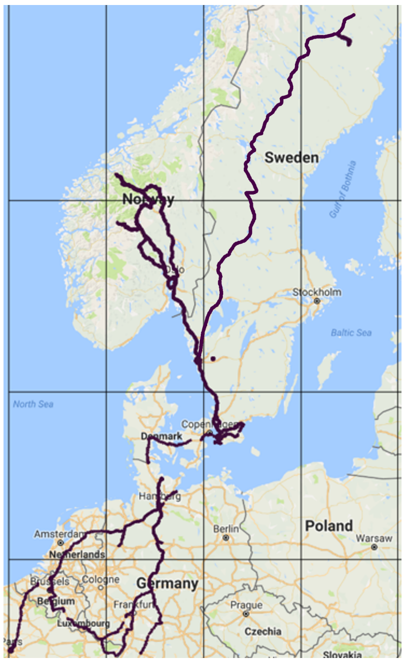


Figure 3.1: Geometric location of all available road friction

3.2 HMM-based spatial temporal pattern map and clustering

The underlying temporal pattern of road friction coefficient in a specific location could be used as a prior information for the on-board estimation algorithm. For instance, the road friction keeps constantly high during summer in California while it varies more during winter. Therefore, the spatial temporal pattern could give a rough guess of road friction coefficient given its location. Besides, it makes possible to generate the simulated scenarios under the statistics of reality. Previous study [38] shows that weather information could help to prediction road friction, which serves the prior information to the road friction.
There are various methods to learn temporal patterns such as Hebbian Learning, Associative Memory or Time Delay Multi-layer Perceptrons [39]. Since we would like to use the model later to sample road friction value based on the realistic statistics for simulation purpose, the generative model is the best option. And the model should be preferably simple to build hundreds of thousands of them over all different locations. Hidden Markov model is a simple dynamic Bayesian model which is able to capture the temporal pattern but is simply modeled with initial probability matrix, transition probability matrix and observation probability matrix, compared to other delicate but complex models.
In this thesis, we collect all the ground truth road friction coefficient and their geometric location. And we treat the ground truth as the observations in hidden Markov Model. For each location, one HMM is built to represent the temporal pattern. In

order to build up the relation between different locations, we use the upper bound of Kullback-Leibler Divergence between two HMMs as a distance metric. And finally Hierarchical clustering is used to find the similar geometric locations based on the distance metric. Therefore, not only the temporal pattern in one specific location, but the spatial-temporal pattern is learned over multiple locations.
There are several spatial-temporal models applied for prediction and analysis for infectious disease spread and biological system modeling. In [40], Menenberg proposes the network-based approach for topic modeling. Lv in [41] uses HMM to build up a pattern map to extract user mobility pattern. However, there is no research literature so far to study the spatial-temporal model for road friction. In this thesis, the Hidden Markov Model-based clustering framework is proposed to learn the spatial temporal pattern of road friction, which is inspired by the papers mentioned previously. The pipeline for the framework is shown in figure 3.2. First, the original sequential data is aggregated to sub-sequences based on their longitude and latitude according to the grid of selected resolution. K-fold cross-validation is used to train HMM model to prevent overfitting. Once we train all HMMs, upper bound of Kullback-Leibler Divergence between HMMs is used as similarity metric to build up dissimilarity matrix for all geometric grid. Hierarchical clustering then clusters the similar temporal patterns and differentiate the dissimilar ones.

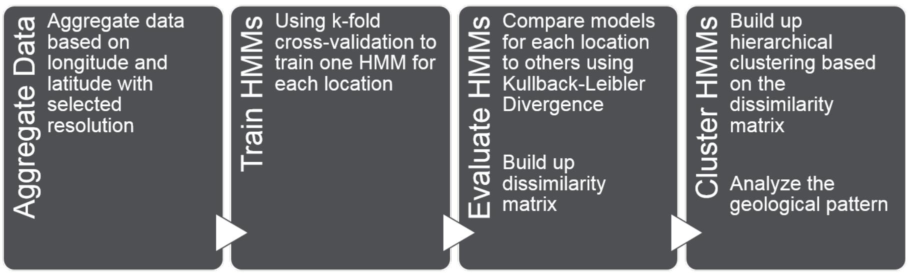

Aggregate Data
Aggregate data based on longitude and latitude with selected resolution
Train HMMs
Using k-fold cross-validation to train one HMM for each location
Evaluate HMMs
Compare models for each location to others using Kullback-Leibler Divergence
Build up dissimilarity matrix
Cluster HMMs
Build up hierarchical clustering based on the dissimilarity matrix
Analyze the geological pattern

Figure 3.2: Hidden Markov Models-based spatial temporal pattern clustering

The fundamentals of hidden Markov Model have already been covered in theory section which will not be repeated again. In the following sections, more detailed description about the pipeline and the related techniques will be illustrated.

3.2.1 Optimal geometric aggregation resolution

Before training the model, data needs to be aggregated to sub-sequences according to their geometrical locations on the grid. One parameter needs to be specified here is the resolution of aggregation. Larger the grid resolution is, more data will be in one grid, however, less representative the location is. Therefore, it is significant to trade off between the representativeness and the number of individual HMMs.
Figure 3.3 shows the amount of samples over different locations when the resolution of longitude and latitude is chosen as 1.5[degree]. From figure 3.3, there are only 44 areas occupied with data, and most of them are quite small. It is tricky to select

optimal resolution to ease the interpretation of result. More comparison on different resolutions will be presented in the result session.

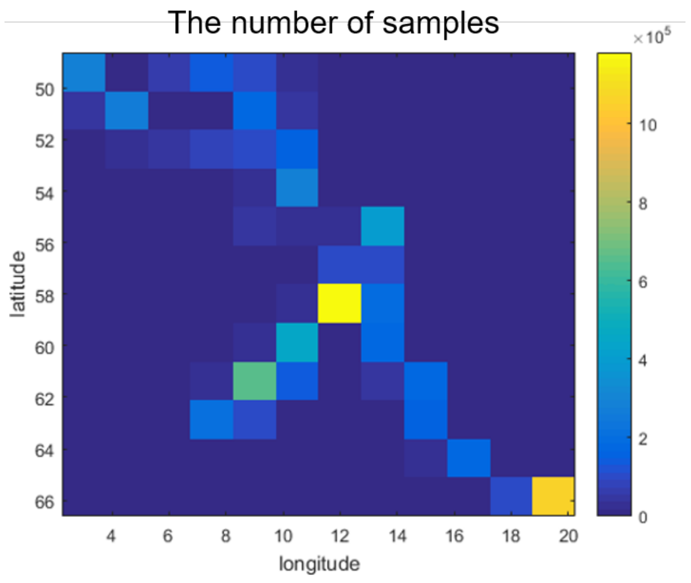


Figure 3.3: The amount of samples of different geometric locations on aggregated data

3.2.2 Cross-validation

The goal of learning HMMs is to extract the underlying patterns that how road friction changes so that we could generate road friction sequence in the simulation from realistic statistics. In order to prevent overfitting of training, K-fold cross-validation is used here. The cross-validation gives the rough evaluation of trained model on validation data. The model that has higher accuracy on validation data indicates it is more general. There are two main types of cross-validation: exhaustive and non-exhaustive. Exhaustive cross-validation trains and validates on all possible ways to divide original samples into training and validation set. However, it is computationally unfeasible especially when data size is large. Non-exhaustive cross-validation does not compute all different combinations of splitting. For example, K-fold cross-validation randomly partitions original samples into k equal sized subsets, and use one of subsets as validation data, and rest of them as training data. Figure 3.4 shows how 10-fold splits the data into training and validation sets. The metric of validation is the logarithmic probability of validation sequences given the trained model.

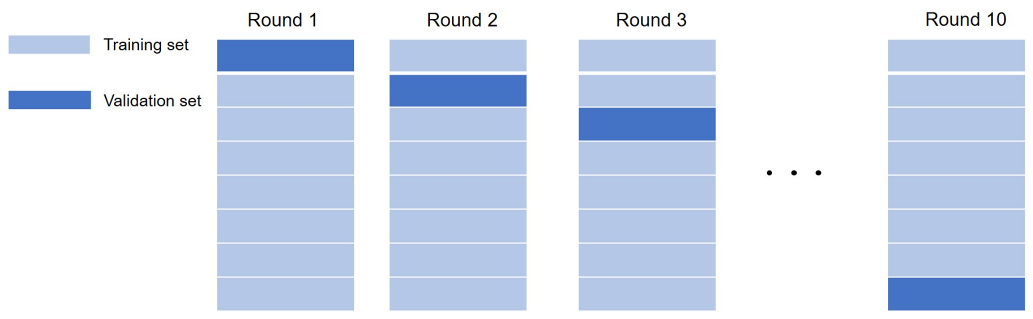


Figure 3.4: K-fold cross-validation

3.2.3 Dissimilarity metric

All HMMs trained need to be compared to measure how similar two patterns are. There are several ways to measure the similarity i.e. through distributions [42]. Here we use the upper bound Kullback-Leibler divergence as a discriminative measure for hidden Markov model.
The Kullback-Leibler Divergence provides an objective statistical indication to measure two distributions [43]. The original formation of KL Divergence between two probability distributions pair T_i and T_j is defined by:

$$KL(T_i||T_j) = \int T_i(x)\log(\frac{T_i(x)}{T_j(x)})dx \tag{3.1}$$

KL Divergence also can be used to compare probabilistic models from a discrimination point of view [43]. Singer in [44] derives the upper bound for KL Divergence for discrete observation HMMs. We use this upper bound as an approximation to compare between models. As theory section, the parameters of a hidden Markov model HMM is denoted by ($\mathbf{A}$, $\mathbf{B}$, $\mathbf{\Pi}$). The upper bound of $UBKL(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$ is used as discriminative measure to compare two HMMs:

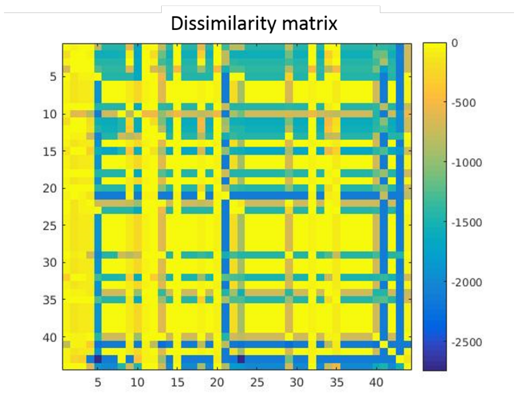
$$KL(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) \geq UBKL(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = KL(\mathbf{\Pi}_1||\mathbf{\Pi}_2)+\sum_{i}^{N_q}\mathbf{\Pi}_{1,i}(KL(\mathbf{A}_{1,i}||\mathbf{A}_{2,i})+KL(\mathbf{B}_{1,i}||\mathbf{B}_{2,i})) \tag{3.2}$$

where $KL(\mathbf{\Pi}_1||\mathbf{\Pi}_2)$, $KL(\mathbf{A}_{1,i}||\mathbf{A}_{2,i})$ and $KL(\mathbf{B}_{1,i}||\mathbf{B}_{2,i})$ can be easily calculated by the discrete definition of KL Divergence. One important thing to be noted is that the upper bound of Kullback-Leibler divergence is not a symmetric metric that $UBKL(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$ is not equal to $UBKL(\boldsymbol{\theta}_2, \boldsymbol{\theta}_1)$. Therefore, the symmetric extension is used as the distance measure instead:

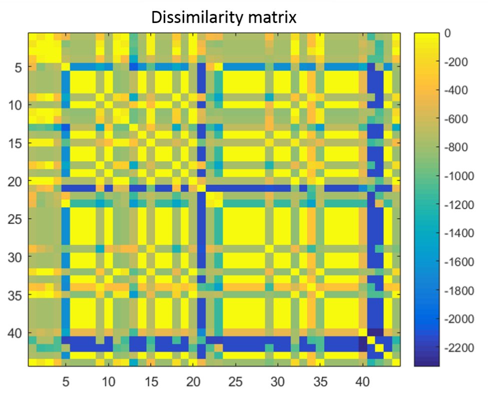
$$UBKL_{sum}(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) = \frac{1}{2}UBKL(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2) + \frac{1}{2}UBKL(\boldsymbol{\theta}_2, \boldsymbol{\theta}_1) \tag{3.3}$$

Besides, the upper bound has its deficiencies of infinite value when any element in matrix A_2, B_2, π_2 is zero. Therefore, we use a real small number replacing zero to avoid the problem.
Figure 3.5a shows one example of dissimilarity matrix calculated based on the upper bound of KL Divergence and figure 3.5b on the extended symmetric upper bound

of KL Divergence. In both figure, x-axis and y-axis indicate 44 different location indexes. The color of each grid stands for the value of the metric. It is clear that closer the value is to 0, more similar two locations is. The symmetric extension of KL Divergence upper bound which gives more easily interpretable result. The distinctive locations could be easily observed from the matrix. However, the geometric location information cannot be presented in this matrix which makes it difficult to directly see the geometric pattern. Besides, although the matrix helps us to distinguish the most distinctive locations, it is difficult to interpret the relations between multiple locations, therefore the dissimilarity matrix will be further used to cluster models to easily understand spatial-temporal patterns.



(a) Dissimilarity matrix by the upper bound of KL divergence



(b) Dissimilarity matrix by the extended symmetric upper bound of KL divergence

Figure 3.5: Dissimilarity matrix comparison between two definitions

3.2.4 Hierarchical clustering

In statistics, hierarchical clustering is one of clustering method to create a cluster tree. Not like other clustering method, tree is not a single set of clusters but has multiple hierarchies. There are two different categories of hierarchical clustering: agglomerative and divisive. Agglomerative is a bottom-up approach that each element starts with its own cluster and then pairs of clusters are merged as one moves up in the hierarchy. On the other hand, divisive is a top-down approach that all elements start with a single hierarchy and then split as one move down the hierarchy. Once the dissimilarity matrix is built, hierarchical clustering could determine how different regions are grouped together. In this master thesis, the implementation uses the function *linkage* in MATLAB to find out the linkage of two elements in dissimilarity matrix [45].
Before creating clusters, there is one more parameter required to specify which is the number of clusters or the inconsistency coefficient threshold. Here, we specify the number of clusters as 5 for convenience. One thing needs to be noted that hierarchical clustering is not the only way. Other alternatives like k-means clustering, Gaussian Mixture Model clustering can be applied. The reason to choose hierarchical clustering is due to its advantage of hierarchical interpretation. Figure 3.6 shows the hierarchy relation between different geometric locations based on the similarity

matrix shown in figure 3.5b. The x-axis indicates the location indexes, and y-axis gives the measure of linkage distance. We could clearly observe the linkage distance of each cluster to others and the hierarchical structures.

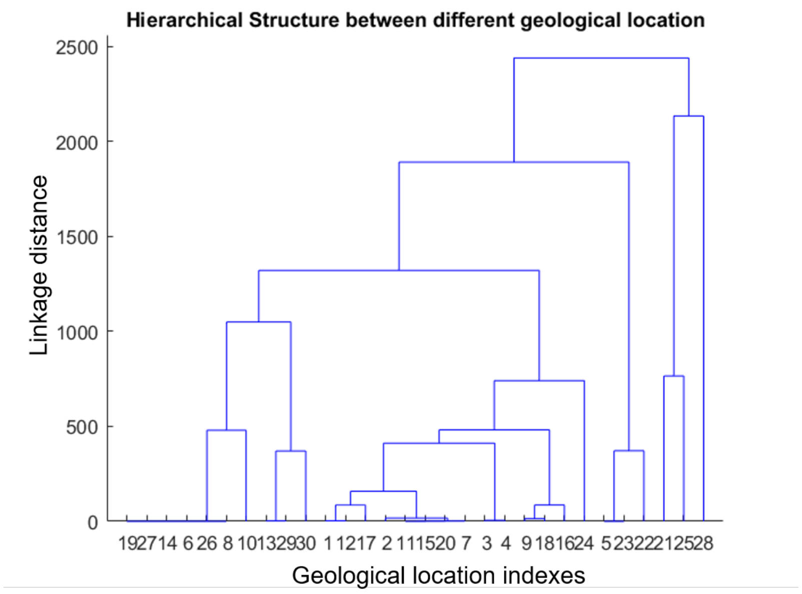


Figure 3.6: Hierarchy relation between different geometric locations based on KLD in figure 3.5b

3.3 Feature selection technique

Payam in [46], and Lei and Huan in [47] evaluate more than 20 common feature selection methods. There are two main broad categories: filter method and wrapper method [47]. The filter method selects features without involving any learning algorithm. On the contrast, wrapper method requires the predetermined learning algorithm during the feature selection. Wrapper model needs to learn a hypothesis for each new subset of features, which makes selection procedure computationally expensive and it is difficult to decide single learning algorithm to select features beforehand. Therefore, we resort simpler filter methods. Filter method is more suitable especially to large number of features and large number of data. Among multiple filter methods, we select the Fast Correlation-Based Filter FCBF [48] for dimension reduction due to its efficiency for large size of data. Meanwhile, the domain knowledge from expert in the field of vehicle dynamics gives a guidance to pinpoint useful features as well.
Figure 3.7 shows the feature selection procedure of Fast Correlation-Based Filter. During relevance analysis, the relevance of original feature set is evaluated to the class, and irrelevant features is discarded first. Redundancy analysis scrutinizes all the relevant features whether they are redundant to each other or complimentary. Only the complimentary features are kept, and all redundant features are discarded since other features have already carry similar information.

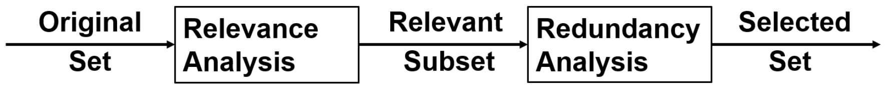


Figure 3.7: The feature selection procedure of Fast Correlation-Based Filter FCBF

The correlation-based measure used here is the symmetrical uncertainty. In section 3.3.2, the algorithm and pseudo code will be illustrated with details, and features selected on the data used in this thesis will be presented as well.

3.3.1 Correlation Measures

There exist two types of correlation measures between two variables: linear correlation and non-linear correlation. For linear correlation, Pearson's correlation coefficient measures the linear correlation between two variables. It ranges from -1 to 1 inclusive. If the value equals to -1 or 1, two variables are completely linear correlated; if the value is 0, there is no linear correlation between them; if the value is between -1 to 0 or 0 to 1, they are partially linearly correlated. However, it is not safe to assume the linear correlation among features or between feature to class in practice. Linear correlation measure cannot capture the nonlinear correlations which is common in practice. In addition, the Pearson's correlation coefficient does not fit to analyze features that are categorical.
To overcome these disadvantages, the correlation measure is used to capture nonlinear relations. In [48], they adopt the correlation measures based on the information theoretical concept of *entropy*, a measure of the uncertainty of a random variable. The entropy of variable X is defined as:

$$H(X) \triangleq -\int P(X) \log P(X) dX \tag{3.4}$$

and the entropy of X conditioned after observing Y is defined as:

$$H(X|Y) \triangleq -\int P(Y) \int P(X|Y) \log P(X|Y) dX dY \tag{3.5}$$

where $P(X)$ is the probability of X, and $P(X|Y)$ is conditional probability of X given Y. The amount that additional information gained from observing Y is called *information gain* [49]. The *information gain* reflects the benefit of knowing Y to predict X:

$$IG(X|Y) \triangleq H(X) - H(X|Y) \tag{3.6}$$

According to the definition of *information gain*, feature Y is more correlated to feature X than to feature Z if $IG(X|Y) > IG(Z|Y)$ [48]. Besides, *information gain* is a symmetrical measure. The detailed derivation could be found in [48]. However, *information gain* is biased to favor features with larger values. Therefore, the measure should be normalized to ensure it is comparable between different variable scaling. Therefore, *symmetrical uncertainty* is uesed instead [50]:

$$SU(X,Y) \triangleq 2[\frac{IG(X|Y)}{H(X)+H(Y)}] \tag{3.7}$$

Symmetrical uncertainty resolves the bias of *information gain* and constraints value in the range of [0,1]. Value as 1 stands for the complete knowledge to correctly predict X given Y and 0 indicates the independence between X and Y. And it is easy to prove that the *symmetrical uncertainty* is also symmetric measure since *information gain* $IG(X|Y)$ is symmetric measure. *Symmetrical uncertainty* applies on both categorical features and continuous ones.

3.3.2 Fast Correlation-Based Filter FCBF

FCBF uses *symmetrical uncertainty* SU as the metric for both relevance and redundancy analysis:

- Relevance analysis:
  To decide which features are relevant to the class. First, the relevance threshold δ_{rel} is pre-defined. The *symmetrical uncertainty* of each feature to class is calculated and store in matrix $SU_{i,c}$ which denotes the correlation between feature F_i and target value C. All the features that have $SU_{i,c}$ larger than δ_{rel} will be kept in relevant feature list S'_{list}, otherwise will be discarded.
- Redundancy analysis:
  To approximately determine if two predominant features F_i and F_j are redundant to each other. After relevance analysis, features are ranked according to their relevance to target value indicated by $SU_{i,c}$. Staring from the highest ranked features, two predominant features F_j (with higher rank) and F_i (with lower rank) will be determined whether they are redundant to each other: the feature F_i will be removed from list if the symmetric uncertainty $SU_{i,j}$ between two features is smaller than the symmetric uncertainty $SU_{i,c}$ between feature F_i to target C. Therefore, the redundancy analysis uses more predominant features to remove less predominant ones. By iterating over the whole relevant feature list, the features left are the ones selected through the process.

The pesudo code of original FCBF is presented in algorithm 1. First part (line 1-6) is the relevance analysis. δ_{rel} is pre-defined *symmetrical uncertainty* relevance threshold to filter out irrelevant features and only relevant features are stored in feature list S_{list}'. The feature list is sorted as descending order according to their relevance to target. The second part (line 10-20) further processes the ordered list and only keeps the predominant features among all relevant features. The algorithm stops until there is no more feature can be removed from S_{list}', and then selected feature list will be kept in the S_{best}. In FCBF, the relevance analysis has a linear time complexity in terms of the number of features $\mathcal{O}(\mathcal{N})$. The complexity of redundant analysis depends on the number of features removed. The best case is that all remaining features are removed in one iteration, and the worst case is no feature removed at all. On average, if we assume half of remaining features are removed in

Algorithm 1
Original FCBF

```
[!h]
 0: input: symmetrical uncertainty $SU_{i,c}$; relevance threshold $\delta_{rel}$
    output: $S_{best}$
 1: for i = 1 to N do
 2:    calculate $SU_{i,c}$ for $F_i$
 3:    if $SU_{i,c} > \delta_{rel}$ then
 4:       append $F_i$ to $S_{list}'$
 5:    end if
 6: end for
 7:
 8: order $S_{list}^{'}$ in descending $SU_{i,c}$ value
 9: $F_j$ = getFirstElement($S_{list}'$)
10: repeat
11:    $F_i$=getNextElement($S_{list}^{'}$,$F_j$)
12:    repeat
13:       if $SU_{i,j} \leq SU_{i,c}$ then
14:          remove $F_i$ from $S_{list}'$
15:          $F_i$=getNextElement($S_{list}'$,$F_i$)
16:       end if
17:    until $F_i$==NULL
18:    $F_j$ = getNextElement($S_{list}'$,$F_j$)
19: until $F_j$==NULL
20: $S_{best}$ = $S_{list}'$ =0
```

each iteration, the time complexity for redundancy analysis is $\mathcal{O}(\mathcal{N}log\mathcal{N})$[48]. Since the calculation of $SU_{i,j}$ or $SU_{i,c}$ is linear to the number of instances M in dataset, the overall complexity of FCBF is $\mathcal{O}(\mathcal{M}\mathcal{N}log\mathcal{N})$. Therefore, the biggest advantage of FCBF compared to other filtering methods is to save time to not compute all *symmetrical uncertainty* $SU_{i,j}$ between all different combinations of features F_i and F_j when they have already been removed by the pre-domainant features. Since the number of features and the number of samples are so large, FCBF assists to select features in a faster manner. However, when analyzing our data, we find that the *symmetrical uncertainty* between features is usually higher than that between feature and class (there are intra-feature correlations). If we use the classic FCBF proposed in [48], a lot of informative features will be left out thanks to the restrict criteria in redundancy analysis. Therefore, additional redundancy threshold is introduced to slack the restrict constraint. Instead of judging based on $SU_{i,j} < SU_{i,c}$ in line 13 in 1, lose condition $SU_{i,j} - SU_{i,c} \leq \delta_{red}$ is used. Here, the redundancy threshold is set to 0.3. The pesudo code after modification is list in Algorithm 2:

Algorithm 2
Improved FCBF algorithm

```
 0: input: symmetrical uncertainty $SU_{i,c}$
        relevance threshold $\delta_{rel}$
        redundancy threshold $\delta_{red}$
 1: for i = 1 to N do
 2:     calculate $SU_{i,c}$ for $F_i$
 3:     if $SU_{i,c} > \delta_{rel}$ then
 4:         append $F_i$ to S_list′
 5:     end if
 6: end for
 7:
 8: order $S_{list}'$ in descending $SU_{i,c}$ value
 9: $F_j$ = getFirstElement($S_{list}'$)
10: repeat
11:     $F_i$=getNextElement($S_{list}'$,$F_j$)
12:     repeat
13:         if $SU_{i,j}$ - $SU_{i,c} \leq \delta_{red}$ then
14:             remove $F_i$ from $S_{list}'$
15:             $F_i$=getNextElement($S_{list}'$,$F_i$)
16:         end if
17:     until $F_i$==NULL
18:     $F_j$ = getNextElement($S_{list}'$,$F_j$)
19: until $F_j$==NULL
20: $S_{best} = S_{list}' =0$
```

3.3.3 Selected Features

As discussed in previous section, the complexity of FCBF is linear to the number of samples. Using whole dataset of few million samples is not computationally feasible to select features. Therefore, data for feature selection is downsampled by 10, which will not change data distribution substantially.

The figure 3.8 shows the $SU_{i,c}$ calculated for each feature to the target, and y axis indicates the corresponding relevance. The green points are the GPS related features which are not considered when applying FCBF, since we have observed bias introduced by those signals from preliminary experiment. Red features are recommended features by the expert who has domain knowledge. On the contrary, blue features are not recommended by expert. The yellow star features in 3.8 are the features selected by FCBF. We could see from the figure 3.8, most of selected features have high *symmetrical uncertainty* $SU_{i,c}$ and are recommended by expert. Except from some meta features such as road type and tyre type, ambient temperature, the lateral forces, longitudinal slip rate, steering wheel angle and etc. are usually used for physical based estimation algorithm.

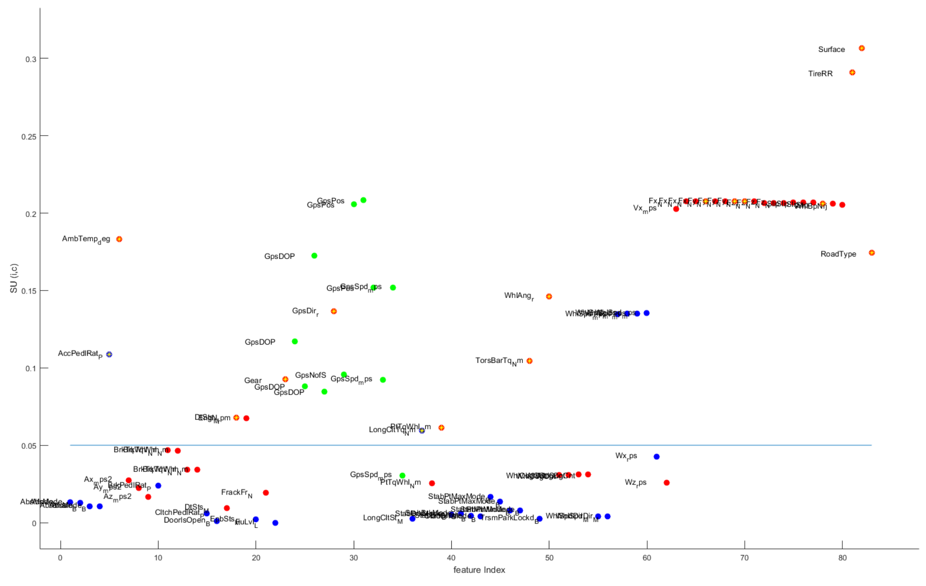


Figure 3.8: *symmetrical uncertainty* between features and class by FCBF feature selection

In order to show the validity of FCBF on our datasets, complete symmetrical uncertainty among all features are calculated as well. Figure 3.9 shows the *symmetrical uncertainty* $SU_{i,j}$ among 83 features, which could help us closely understand more about how different features correlate from information perspective. One thing needs to be emphasized that it is not necessary to calculate it since FCBF is used to decrease computation expense. In figure 3.9, the lighter the color is, the higher the *symmetrical uncertainty* $SU_{i,j}$ between feature F_i and F_j is. For instance, the first four features have high values as almost 1 since they are FLAG signal indicating ABS trigger for each wheel. The similar observation is found for feature 11 to 14 which are breaking torque for each wheel respectively.

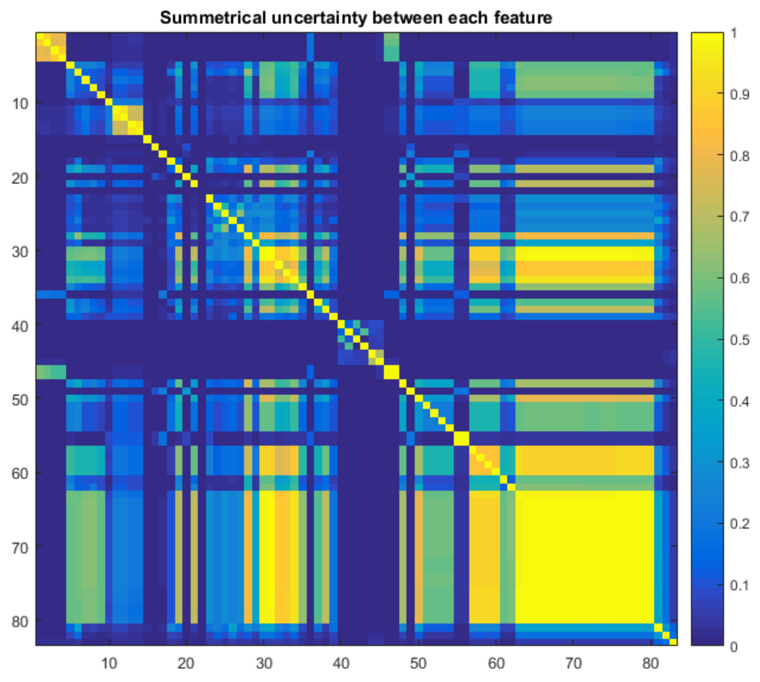


Figure 3.9: symmetrical uncertainty between features by FCBF feature selection

Table 3.1 lists all 29 features selected by FCBF in the descending order. 24 of 29 features have physical bounding to the road friction. All selected features will be used for training and test for on-board estimation algorithm by ESNs which will be presented later.

Table 3.1: Selected feature list by FCBF

Index	Name	Description
83	RoadType	Road type
81	Tire	Tire type
6	AmbTemp_deg	Ambinent temperature in Celsius
76	Slip.1	Slip rate of front left wheel
78	Slip.3	Slip rate of rear left wheel
68	Fy_N.1	Lateral force of front left wheel
82	Surface	Surface type
50	WhlAng_r	Steering wheel angle
48	TorsBarTq_Nm	Steering wheel torque
23	Gear	Current active gear
5	AccPedlRat_P	Acceleration pedal ratio
18	DtSts_M	Rear axle is engaged in drive train
28	GpsDir_r	GPS heading [rad]
39	PtTqWhl_Nm	Powertrain torque for rear axle
37	LongCltTq_Nm	Longitudinal clutch status

19	EngN_rpm	Engine rotational speed [rpm]
61	Wx_rps	Roll rate
53	WhlCogCnt	Wheel cog counters for rear left wheel
54	WhlCogCnt	Wheel cog counters for rear right wheel
52	WhlCogCnt	Wheel cog counters for front right wheel
51	WhlCogCnt	Wheel cog counters for front left wheel
10	BrkPedlRat_P	Break pedal ratio
11	BrkTqWhl_Nm	Break torque for front left wheel
7	Ax_mps2	Longitudinal acceleration $[m/s^2]$
14	BrkTqWhl_Nm	Break torque for rear right wheel
44	StabPtMaxMode_B	Engine traction control for front left wheel is at maximum
8	Ay_mps2	Lateral acceleration $[m/s^2]$
62	Wz_rps	Yaw rate
45	StabPtMaxMode_B	Engine traction control for front right wheel is at maximum

Feature selection is significant to affect the performance of learning algorithm. As mentioned previously, we utilize some expert knowledge from vehicle dynamics domain. However, there still lacks of method to merge this useful knowledge in the feature selection process, which will be left for the future work.

4 Results and analysis

In this chapter, we will present the results in two parts: spatial-temporal pattern of road friction using HMM and clustering methods and on-board road friction estimation algorithm using ESNs. The temporal pattern will be presented over different geometric locations and clustering result will further motivate the validity of method. On the other hand, the road friction estimation algorithm with selected feature from previous chapter, will be compared to the in-house physical model based method to evaluate its performance.

4.1 Hidden Markov model based spatial temporal pattern recognition of road friction

In section 3.2, we have proposed a method for temporal modeling of road friction over different locations to understand the spatial temporal pattern, therefore it could serve as a potential prior information source for the on-board estimation algorithm in the future.

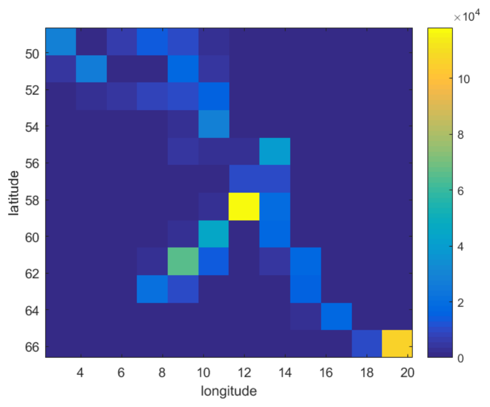
Once we specify the resolution of longitude and latitude for data aggregation, multiple HMMs could be learned for each location. We take an example that resolution of longitude and latitude are chosen as 1.5 degree, and the number of cluster as 5 to illustrate how the proposed method performs.

With the given resolution, we aggregate data sequences to sub-sequences according to their locations. Figure 4.2c shows the sample size for each region and there are only 44 locations that have adequate data to be analyzed. The x-axis and y-axis stand for the longitude and latitude respectively, while the color indicates the data size specified by the color map on the right. Once aggregation is completed, the data sequences in each location will be used to train HMMs using k-fold cross-validation. Therefore, we will get 44 HMM models in this case which is denoted by $\Gamma = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_{44}\}$. Then the extended symmetric upper bound of KL divergence between HMM is used as dissimilarity metric to calculate the distance between all HMM pairs. So we will get a matrix with size of 44×44 and we show it in figure 4.3c. The x-axis and y-axis are the index of location from 1 to 44, and the color stands for dissimilarity metric between location i and location j. The lighter the color is, the higher the correlations are between two locations. We could distinguish 4 distinctive locations immediately from 4.3c: location 5, 21, 41 and 42 since they have quite low correlation to most of other locations. And location from 1 to 4 are close to each other, same applies to location from 23 to 27. Although location 5 is distinctive to other locations, it has quite high correlation to 42. We could continue to observe more connections from the plot, but it is quite time-consuming and exhaustive to understand the relations between any pair of the locations. Besides, the geometric information is missing in the figure that neighbor index does not necessarily mean that they are geometrically close, so it makes difficult to understand the spatial pattern. As introduced in method chapter, hierarchical clustering is a powerful tool to build hierarchical represent of relations between locations based on the dissimilarity matrix, and we could tell how strong is the relations from the linkage distance. We show the hierarchical relations of all locations in figure 4.1c. The x-axis is the geometric location index, and y-axis is the linkage measure which tells how strong is the relation. From the plot, it is easy to interpret how each location is relates to others and how firm is the bound. In addition, we could set up a threshold to get different clusters based on the linkage measure. For instance, if we set the threshold of 1000, there are 5 distinguished clusters. If we change the threshold to 1500, there are only 4 clusters instead.

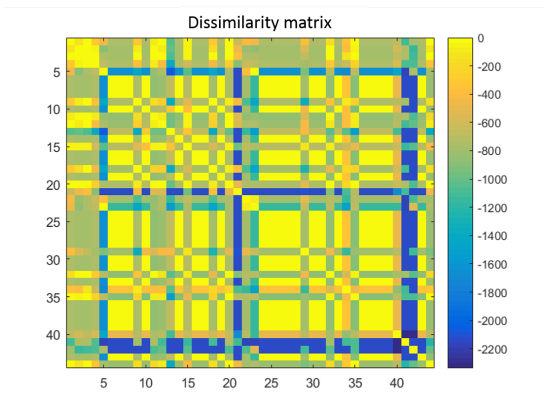
Although hierarchy structure shows the relations that can be directly interpreted, location information is still missing i.e we do not know their geometric relations although we know i.e. location 12 is closely related to 25. Therefore we resort to Google Map API to plot clustering result on the map. Figure 4.1d shows the hierarchy clustering result on the corresponding locations. The color of circle indicates the cluster index and index in circle is the same location index as in hierarchical clustering plot 4.1c or dissimilarity matrix in figure 4.3a. The location of the circle is the center of the region. In figure 4.1d, we could clearly spot spatial-temporal pattern i.e. location 5 is close to 42 and they are also geometrically close. In the northern of Norway and Sweden, the patterns are clustered to the same group except from the Northeast Norway. If we manually check that data, we find that the road friction keeps constantly low in northeast Norway while the road friction varies from low to high from time to time in other northern regions. In Denmark and Germany, the data is collected during summer so that the road friction keeps constantly high so it is not strange to see those regions are clustered into the same group.

Moreover, as we discussed in the section 3.2, the geometric segment resolution is a global parameter that requires to be specified in order to aggregate data. The finer the resolution is, the more regions are. However, the amount of data in each area will be less and make more difficult to train and validate the model since the observation might keep constant always. Besides, if the resolution is fine enough, we have to build large amount of HMMs . On the other hand, the finer the resolution is, the more representative the model is for each location. So that the different pattern will be easily captured by the model and differentiated from others.

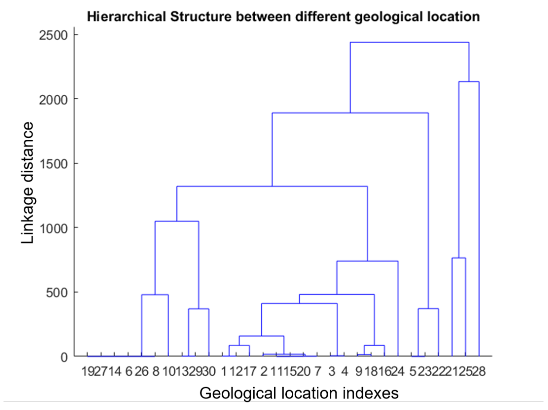
Here, we experiment four different resolutions for longitude and latitude: 0.5, 1, 1.5 and 2 degree. Figure 4.2 shows the comparison of sample sizes under different resolutions. And table 4.1 lists the total number of grid and non-occupied locations and the rate of occupancy under different resolutions. We could easily conclude that the sample size for each location decreases with increasing resolution. Samples are more sparsely distributed over all the locations with smaller resolution since the occupancy ratio is lower. Besides, it is observed that model is slower to converge during training if the value keeps constant. So it takes longer time to build up all



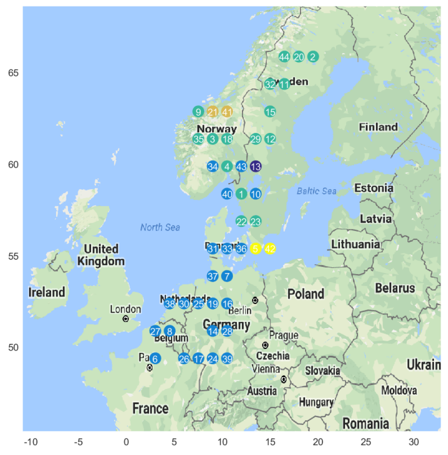
(a) Size of aggregated data under 1.5 degree resolution



(b) Dissimilarity matrix under 1.5 degree resolution



(c) Hierarchy relation between different geometric location



(d) Hierarchical clustering of HMMs

Figure 4.1: Analysis result of spatial-temporal pattern of road friction using hidden Markov model based method

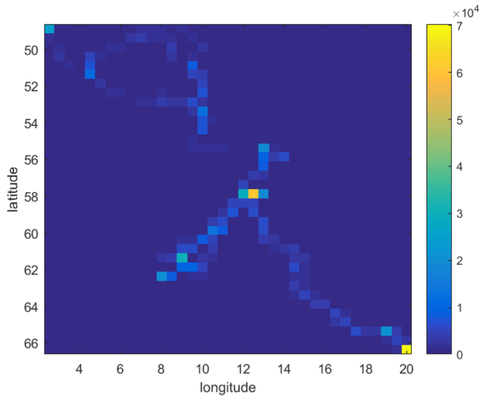
HMMs for all locations. And from 4.2, the data collected from the expedition is quite sparse, which we will leave it to future work to analyze more data stream possibly.

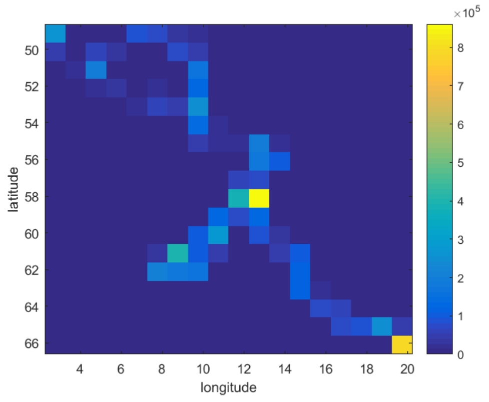
Figure 4.3 shows the comparison of dissimilarity matrix under different aggregation

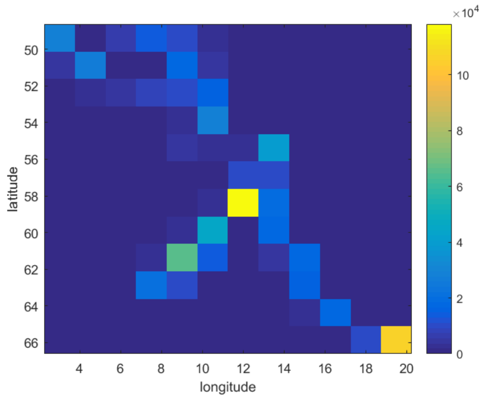
Table 4.1: The number of grid and non-empty location under different resolution

Resolution	Number of grid	Number of occupied grid	Ratio of occupancy
0.5	1296	157	12.1%
1	324	68	21.0%
1.5	144	44	30.6%
2	81	30	37.1%

resolutions. The x-axis and y-axis stands for the index of locations, and the color


(a) Sample size of aggregated data under 0.5 degree resolution


(b) Sample size of aggregated data under 1 degree resolution


(c) Sample size of aggregated data under 1.5 degree resolution

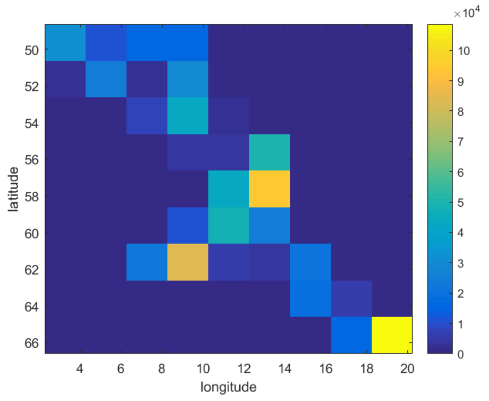
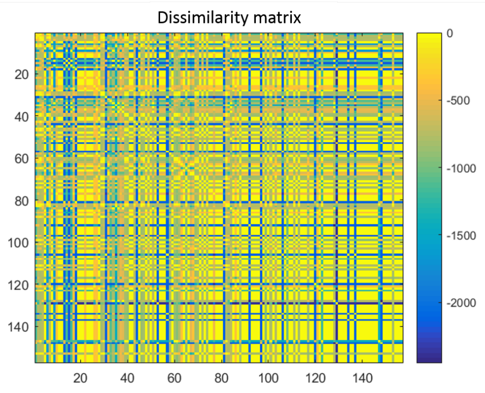

(d) Sample size of aggregated data under 2 degree resolution

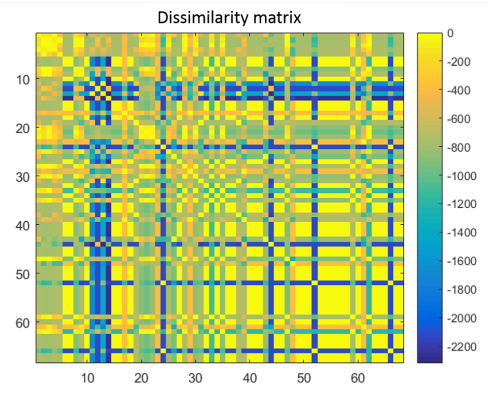
Figure 4.2: Sample size of aggregated data under different resolutions

is the extended KL Divergence upper bound approximation between location i and location j. One thing to be declared is that the indexes of locations among these 4 conditions in Figure 4.3 are different for the same location since the index is actually sorted according to the sample size. Therefore, the location of the distinguished "blue" columns in the matrix varies from one to other. However, regardless of location index, we could still observe the distinguished "blue" columns in plots, and finer the aggregation resolution is, more the "blue" columns appear. One explanation is that the distinctive large regions are split into small regions but those small areas are still distinguished to others. Except from this reason, we also find that there are some areas becomes more distinctive in finer resolution which is not distinctive using larger resolution. Therefore, finer the resolution is, the more distinguished and representative location is.

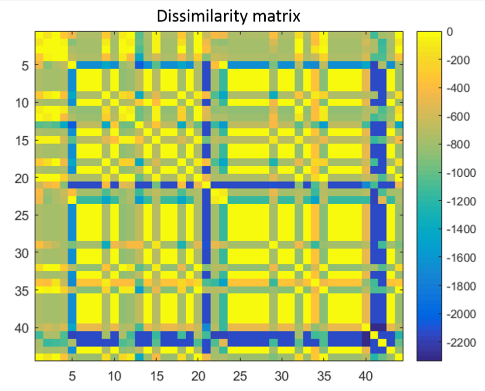
Figure 4.4 shows the comparison of hierarchical clustering of temporal pattern under different resolutions. As denoted previously, the color of circles gives the indication the cluster index and the number in circles corresponds to the geometric index used in dissimilarity matrix plots. The location of circle is the center of the grid



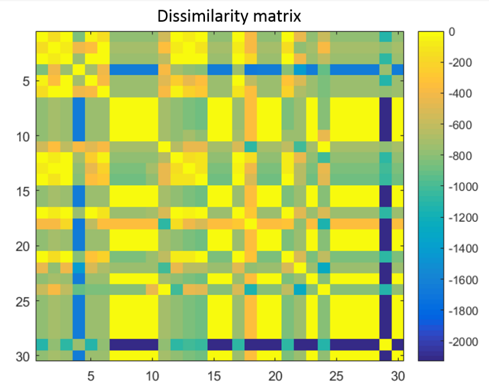
(a) KLD matrix under 0.5 degree resolution



(b) KLD matrix under 1 degree resolution



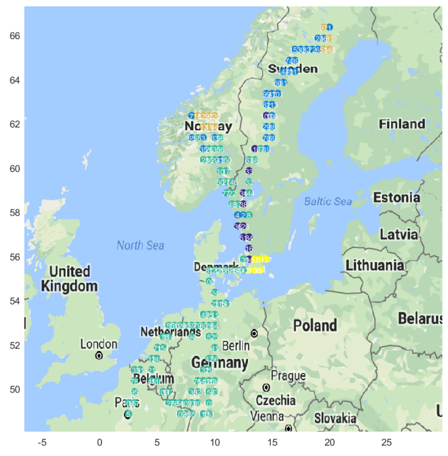
(c) KLD matrix under 1.5 degree resolution



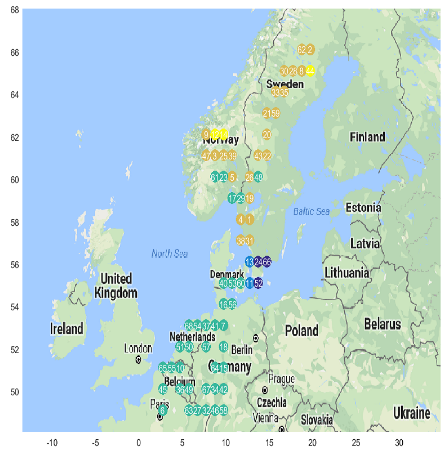
(d) KLD matrix under 2 degree resolution

Figure 4.3: KLD matrix under different resolution

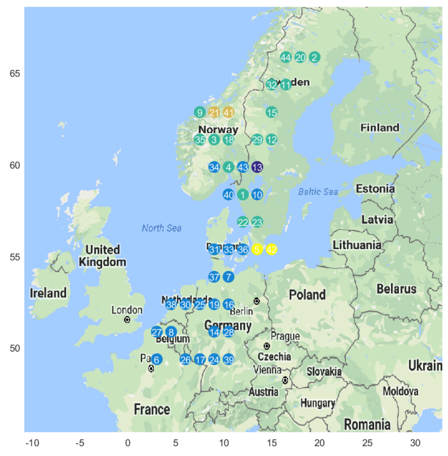
area. However, the color among different figures do not indicate the same cluster, so the assigned color for one specific location on different plot is unnecessarily the same. As mentioned in previous discussion, the finer resolution is, more representative the model is to each location, and the number of distinctive area increases not only because the big regions split into small regions but also some small areas are distinguished from their neighbors. If we take a close look at figure 4.4a and figure 4.4b, the cluster boundaries are more complicated in figure 4.4a and more cluster mixing is observed such as in Norway and Sweden. For instance, the deep blue area in middle of Sweden is distinguished among its neighbors in figure 4.4a. However, in figure 4.4b it is clustered in the same group. Therefore, with finer resolution, those locations, having more representative pattern are no longer mixed with their neighbors. However, the general geometric pattern is still similar no matter what resolution is selected: northern Sweden and Norway share quite similar pattern; the northeast part of Norway and Sweden is in the same group; southern Sweden has distinguished pattern; all areas in European continent share the same pattern.



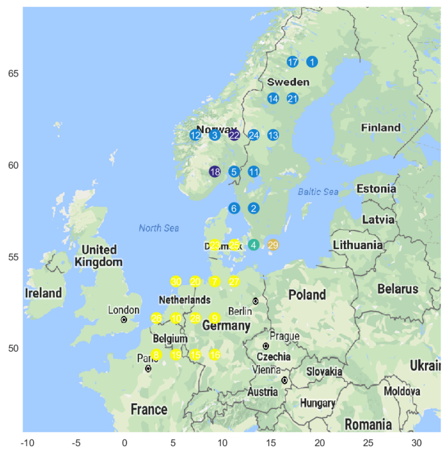
(a) Hierarchical clustering over different locations under 0.5 degree resolution



(b) Hierarchical clustering over different locations under 1 degree resolution



(c) Hierarchical clustering over different locations under 1.5 degree resolution



(d) Hierarchical clustering over different locations under 2 degree resolution

Figure 4.4: Hierarchical clustering over different locations under different resolution

In summary, the resolution of aggregation affects extracted pattern of road friction over different locations. And with finer resolution, the extracted pattern is more representative and easier to be distinguished from neighbor areas. The attendant disadvantages is that there is less data in each location and it takes more time to train due to slower convergence.
As mentioned before, since the data is distributed sparsely over space and pattern learned is difficult to be validated so far due to the lack of similar expedition multiple times. Therefore, it is hard and unsafe to draw firm conclusion without validation. We will leave the further investigation and merge with estimation algorithm in the future work.

4.2 On-board estimation algorithm using Echo State Network

4.2.1 Training, validation and test datasets split

Before presenting result, we need to define the way to split training, validation and test data. Training data are only used during training, while validation data is to evaluate the model performance to optimize hyper parameters. The test data is only used to test model performance after training and validation is completed and best model is selected to make an unbiased evaluation of model performance on unseen data.

As mentioned in data section, the long sequences are partitioned into small sequences with length of 10000 that correspond to 100 seconds for each. We could estimate road friction quite accurately by physical based model when tire is excited to certain level. Therefore, we assume that we could obtain some approximate "ground truth" road friction from it, which could be used to estimate road friction in the future. Therefore, we would like to train the ESN model to estimate the subsequent data stream when physical model does not work so well so that we have redundant alternative to compliment each other. Different from the usual definition of train/validation/test data split for the i.i.d data, we take the first part of short sequences as training data, and the second part as validation or test data. Figure 4.5 shows how long sequence is split into different dataset. The red part indicates the training data, while the green indicates the validation or test data. The ratio between the length of training and whole short sequences is called train-test split to partition sequences, which ranges from 0 to 1 and 1 indicates whole sequence is training data while 0 means that whole sequence is validation/test data. Besides, the ratio between validation and test data is set as 1 to 1. The purpose of setting up different train-test split is to evaluate the model's limit that how long the following sequence can be accurately estimated, since it is nature that model gives worse and worse prediction when less relevant and accurate information is given at beginning. The train-test split is selected first as 0.3 for the following discussion. The different split ratio will be compared in the end of this chapter. The ESNs used in the following discussion is the one with feedback path using the iterative training proposed in section 2.2.2.4, and estimated $\hat{y}$ is used for feedback path.

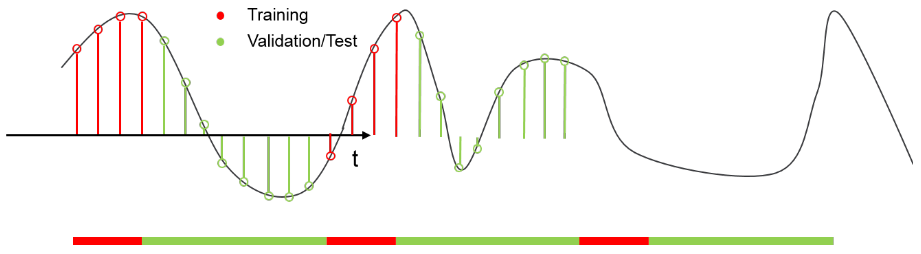


Figure 4.5: Training, validation and test data split

4.2.2 Hyper parameters optimization

As discussed in the theory section that there are several hyper parameters of ESNs, including reservoir size, leaky rate, sparsity of connectivity of reservoir, memory size and spectral radius. The hyper parameters need to be tuned for model to perform better. There are several ways for hyper parameter optimization methods such as grid search, random search, Bayesian optimization, Gradient-based optimization and evolutionary optimization and etc. Since hyper parameter optimization is not the main focus of thesis, we will not compare different methods but select grid search here due to its simplicity.
Grid search is a simply exhaustive search on the manually specified subset of hyper parameter space. Table 3.1 lists all hyper parameter proposals for reservoir size, spectral radius, leaky rate, sparsity of connectivity, memory size.

Table 4.2: Hyper parameter proposals for hyper parameter optimization

Hyper parameters	Range	Interval
Reservoir size	100:800	100
Spectral radius	{0.5,1,2}	
Leaky rate	0:1	0.2
Sparsity of connectivity	0.1:0.3	0.05
Memory size	{0,1,2,3,4,5,7,10,15,20,30}	

Except from the specified parameter candidates, the grid search algorithm also needs the evaluation metric. We have discussed different evaluation metrics in section 2.2.3.1 that we choose MAE, RMSE, average error rate within different ranges to evaluate performance. In most cases, all evaluation metrics follows the similar trend, but not necessarily always the same. For example, if model gives estimation that always deviates to reference within certain range, the error rate within that range could be low, but MAE and RMSE might be high. Therefore, we need to utilize all the evaluation metrics to draw safer conclusions.
However, grid search suffers from large dimensionality of hyper parameter space and is extremely demanding to run training over all different hyper parameter combinations. There is more than 3200 different settings in total to be trained and validated which is infeasible for this thesis due to limited time. Therefore, author uses the greedy grid search to decrease the number of combinations. The greedy grid search is to test all candidate of one hyper parameter while fixes all other hyper parameters, therefore it selects optimal hyper parameter one by one. By doing so, the search space is shrunk to around 30 which is only 1 % of full grid search. Certainly, we compromise the time cost with the optimality of hyper parameter selection. In order to better apply greedy grid search to mitigate the performance degrading, we decide to optimize more importance hyper parameters first which have larger effect on the performance than others, then optimize other less important parameters later. As analyzed in the theory of ESNs, the significant parameters include reservoir size and memory size.
Therefore, experiments are carried first on the different reservoir sizes with fixed

other hyper parameters: spectral radius as 1, leaky rate as 0.5, connectivity as 0.1, memory size as 1. The reservoir size varies from 100 to 800 with interval of 100. Table 4.3 shows the result of MAE, RMSE, error rate within 0.1 on the validation data under different reservoir sizes. All the training and validation are run on 8 cores i7-6700HQ CPU @ 2.6GHz.

Table 4.3: ESN performance under different reservoir size on validation data

Reservoir size	MAE	RMSE	Average Error Rate within 0.1	Average test time per sample [ms]
100	0.0902	0.1053	31.567 %	**0.1322**
200	0.0809	0.0967	27.987 %	0.1394
300	0.0627	0.0764	16.987 %	0.1419
400	0.0649	0.0793	18.593 %	0.1512
500	0.0606	0.0754	17.505 %	0.1553
600	**0.0500**	0.0640	13.234 %	0.1685
700	0.0503	**0.0637**	**12.697 %**	0.1777
800	0.0531	0.0666	14.195 %	0.1794

As we discussed before, there is trade-off between performance and computation cost on large reservoir size. The results in Table 4.3 shows that larger the reservoir size is, the smaller the MAE and RMSE are as we expected. The MAE has best performance at reservoir size as 600, while RMSE and average error rate within 0.1 have best performance on reservoir size as 700. Larger reservoir size introduces more non-linearity in the mapping so that the activation output after ESNs is easier to learn weights of readout layer. There is limitation of performance that it does not grow with reservoir size so that the performance saturated at reservoir size of 700. We observe the performance drop at 800 which is explained by overfitting. As reservoir size increases, the complexity of model increases and model fit training data to more detail noise but less generalized to the unseen validation data. Therefore larger reservoir size leads to the overfitting. Certainly, we could use some technique like regularization as we applied, or early stopping to prevent overfitting. We realize larger reservoir size is not necessary to boost performance.
Besides, the computation cost increases with the size of reservoirs both for training and test. Theoretically the cost goes linearly with reservoirs size if sparse matrix computation could be implemented for reservoirs. However, in practice, we do not optimize in low level programming but use the normal matrix computation. And test time does not grow linearly with reservoir size in Table 4.3 since additional cost i.e. form vector for input variables is static. The average inference time is around 0.2 ms even when reservoir size is large which still fulfills the real time performance requirement, therefore, the computation cost is not the bottleneck. Certainly, more tests need to be done to deploy the algorithm on the ECU or other computation resource on-board in order to evaluate the real time performance when other functions are activated at the same time. Besides, if sparse matrix computation could be implemented in low level programming, the run time performance will be improved

further.
Based on the analysis before, we choose reservoir size 700 as the optimal parameter to carry on the subsequent experiments on memory size. The feedback path helps faster convergence especially when more previous estimation values are feedback into reservoirs if estimates do not deviate to target too much, otherwise it brings the risk that the error in the estimate will be propagated to later estimate if the model cannot handle the noise in estimate. Meanwhile, it is more likely that larger memory size slows down the ESNs dynamics so that it is harder to follow the quick dynamics and give accurate prediction if the ground truth changes rapidly all the time. Therefore, it is significant to select a proper memory size. Table 4.4 shows MAE, RMSE and average error rate within 0.1 under different memory sizes. Before memory size increases to 10, larger memory size gains the benefit to give more accurate estimation. However, after 20, the benefit is mitigated and error increases along with the memory size increasing. For certain, the model is expected to perform even worse if we continue to increase the memory size to i.e. 100, 200, ..., 1000. All of the evaluation metrics have the best performance when memory size is 15. Therefore, we select 15 as the optimal memory size.

Table 4.4: ESNs performance under different memory sizes on validation data

Memory size	MAE	RMSE	Average Error Rate within 0.1
0	0.052983	0.064527	12.787 %
1	0.050464	0.063936	12.627 %
2	0.049412	0.062554	12.313 %
3	0.048915	0.061879	11.892 %
4	0.048323	0.061135	11.628 %
5	0.048075	0.060781	11.555 %
7	0.047557	0.060116	11.294 %
10	0.046953	0.059343	11.021 %
15	**0.046566**	**0.058812**	**10.823** %
20	0.046707	0.058948	10.838 %
30	0.046809	0.059154	10.97 %
40	0.046294	0.058569	10.795 %
50	0.046744	0.05913	11.109 %

Among the rest of hyper parameters, the sparsity of connectivity has more dominant effects because it affect the number of active reservoirs and nonlinear mapping. Theoretically, the larger the connectivity is, the more reservoirs are linked to others and the recurrency of model is kept more. But as discussed in the theory section, most studies recommend to keep the sparsity of reservoirs to maintain the echo property of model both from theoretic analysis and practical experiments. Although the computational cost of reservoir is linearly to sparsity, the time consumption is not the bottleneck as we tested before. Table 4.5 shows MAE, RMSE and average error rate within 0.1 under different sparsity of connectivity value. Seen from Table 4.5, the ESNs give the lowest MAE and RMSE when the connectivity is 0.25. Below 0.25, the error decreases with the increase of connectivity which can be explained by increasing reccurency of model. After 0.25, the error increases again with increasing connectivity, which can be explained by deviation of echo state property. Therefore, 0.25 is selected as the optimal connectivity. And as discussed before, connectivity does not affect the performance of ESNs as much as previous two predominant hyper parameters, which is shown by the insensitivity of performance between different connectivity.

Table 4.5: ESNs performance under different connectivity on validation data

Connectivity	Average MAE	Average RMSE	Average Error Rate within 0.1
0.1	0.046566	0.058812	10.823 %
0.15	0.04754	0.058958	11.991 %
0.2	0.049114	0.061085	12.027 %
0.25	**0.044709**	**0.057082**	**10.67** %
0.3	0.04637	0.057834	10.986 %

After optimizing the sparsity of connectivity, we continue to run experiments over the different spectral radius $\rho(\mathbf{W})$ with frozen optimal reservoir size, memory size and connectivity rate. Table 4.6 shows MAE, RMS and average error rate within 0.1 under different spectral radius as 0.5, 1 and 2. The smaller spectral radius is, the echo state property could be guaranteed more. However, less randomization of W matrix is reserved which might jeopardize the performance of ESNs. In several literature, it is recommended to set spectral radius as 1. But in practice, the spectral radius less than 1 will not bring catastrophic effect to model because of the intra-correlations between input variables. Table 4.6 shows MAE, RMSE and average error rate within 0.1 under different spectral radius. Seen from the Table 4.6, the spectral radius as 2 gives the best average MAE and RMSE, while spectral radius as 1 gives the least error rate within 0.1. But the difference of performance between different spectral radius is slight. The reason could be that the effect of spectral radius on $\mathbf{W}$ is mitigated during training and the intra-correlation between variables makes large spectral radius not applied anymore. Therefore, we choose 1 as the optimal spectral radius.

Table 4.6: ESNs performance under different spectral radius on validation data

Spectral radius	MAE	RMSE	Average Error Rate within 0.1
0.5	0.044719	0.057116	10.664 %
1	0.044764	0.057156	**10.617** %
2	**0.044487**	**0.056833**	10.629 %

From previous discussion, leaky rate serves as the speed of reservoir update dynamics. Therefore, the larger the leaky rate is, more prone model is to follow the fast dynamics. However, the estimate could be more unstable as well. If the leaky rate is set to zero, the dynamic effect is completely ignored and estimate is prone to stay constant as previous estimate. Table 4.7 shows MAE, RMSE and error rate within

0.1 under different leaky rate as 0, 0.2, 0.4, 0.6, 0.8, 1. It is clear that the validation error is large when leaky rate is 0 since dynamics is completely deactivated. With increasing leaky rate, the error also increases since the estimate tends to be less stable and sensitive to current estimation. Small leaky rate as 0.2 gives the best performance on all evaluation metrics. Therefore, we choose 0.2 as the optimal leaky rate.

Table 4.7: ESNs performance under different leaky rate on validation data

Leaky Rate	MAE	RMSE	Average Error Rate within 0.1
0	0.10927	0.10174	39.256 %
0.2	**0.041787**	**0.053441**	**9.089 %**
0.4	0.043824	0.056065	10.265 %
0.6	0.045638	0.05815	11.091 %
0.8	0.048217	0.061007	11.641 %
1	0.049711	0.062642	12.066 %

In conclusion, Table 4.8 lists all the optimal hyper parameters from the experiments above. And we use the best model to evaluate model performance on the test data later.

Table 4.8: Optimal hyperparameters for ESN

Hyper parameters	Optimal value
Reservoir size	700
Memory size	15
Connectivity	0.25
Spectral radius	1
Leaky Rate	0.2

4.2.3 Evaluation of ESNs

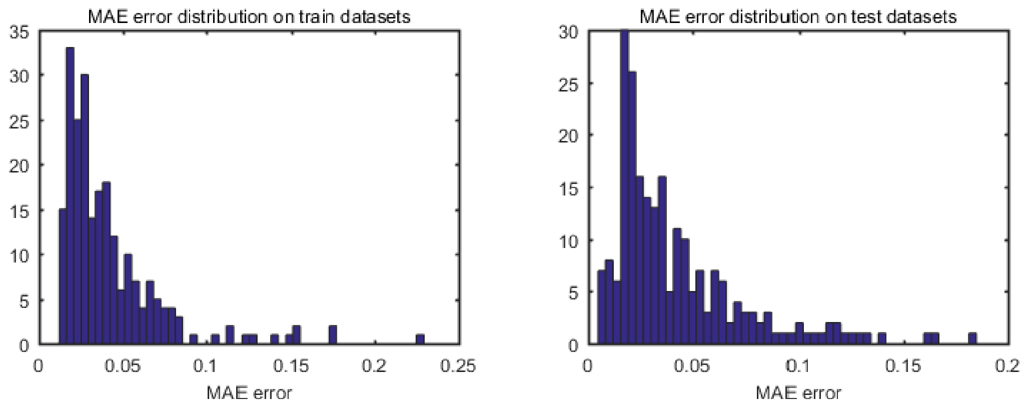
After hyper parameter optimization, the optimal model is used to test the performance on the test data sets. Table 4.9 lists the comparison between ESNs with the optimal hyper parameters and the physical-based model on both training and test data sets. Seen from Table 4.9, ESNs outperform under all the evaluation metrics tremendously compared to that of physical-based model. For instance, MAE is decreased from 0.4723 to 0.0437 by almost 1/10. The average error rate within 0.1 is improved to 9.310% from 97.143%. As we mentioned in our motivation, the physical-based model has extremely bad performance when tire is not excited adequately. However, ESNs are able to give accurate estimate on those circumstances. Besides, ESNs have quite good generality that the performance degrades from training and validation data to test data is small. The average error rate within 0.1 is small enough to apply for the decision making algorithm without huge mistake.

Table 4.9: Comparison between ESN and physical model on training and test data

	Model	MAE	RMSE	Average error rate within			
				0.2	0.15	0.1	0.05
Train	ESN	**0.0423**	**0.0525**	**1.278%**	**3.101%**	**8.995%**	**28.372%**
	Physical	0.301	0.332	71.333%	81%	89.333%	97.667%
Test	ESN	**0.0437**	**0.0564**	**2.478%**	**4.528%**	**9.310%**	**25.794%**
	Physical	0.472	0.492	94.714%	96.000%	97.143%	98.571%

In order to evaluate the performance of model thoroughly, we also plot the error distribution of MAE, RMSE, and average error rate within four different range on training and test sequences. Figure 4.6 shows the MAE error distribution on training and test data where the y-axis shows the number of sequences within range and x-axis is the MAE error. The median and average error of training is a bit smaller than that of test but performance drop is slight. And from 4.6, most of test sequences have MAE error lower than 0.05, and only very few sequences is larger than 0.1. The similar trend we could observe from figure 4.7 as well: most of sequences has small RMSE less than 0.1, while only very few has larger error than 0.15.

Figure 4.8 to figure 4.9, 4.10 and 4.11 show the error rate within different ranges on

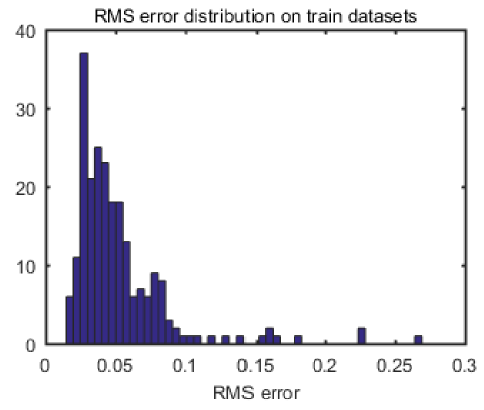


(a) MAE distribution on training data **(b)** MAE distribution on test data

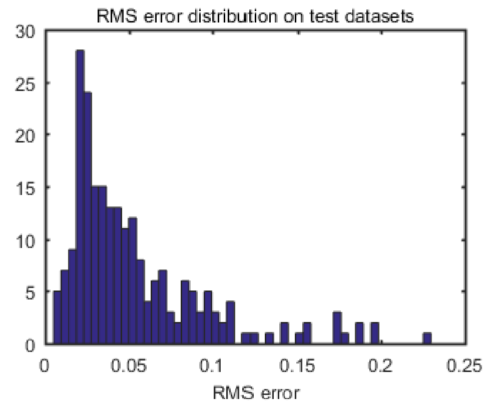
Figure 4.6: MAE distribution comparison of ESN between training and test data

train and test data sets. The x-axis indicates the error rate from 0% to 100 % while y-axis indicates the number of sequences at the particular error rate range specified by the bins. At most strict range of error, most of sequences still have error rate less than 20 % only few sequences have high error rate from 4.8. If the range of error is less strict, more proportion of sequences haves lower error rate from figure 4.8 to figure 4.9, 4.10 and 4.11. Only less than 10% of sequences has error rate larger than 10 % which indicates that the algorithm is generalized well to different scenarios.

In addition, we shows some typical examples that how model performs. Figure 4.12 and 4.13 show examples of sequence that have small and large deviation of estimate respectively. In all examples, the ground truth road friction varies with time, which is indicated by the blue line for reference. The red line indicates the

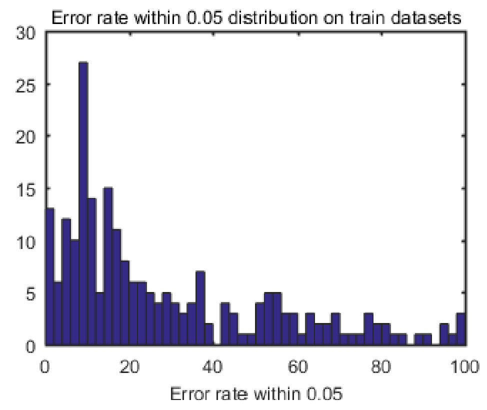


(a) RMSE distribution on training data

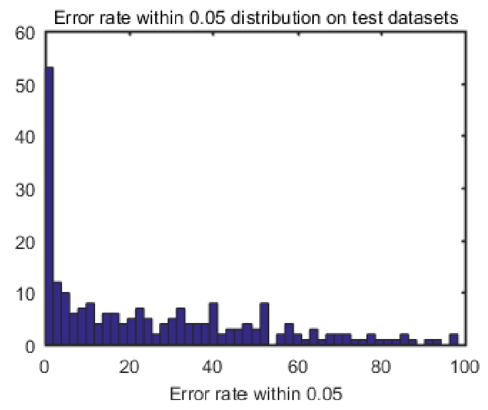


(b) RMSE distribution on test data

Figure 4.7: RMSE distribution comparison of ESN between training and test data

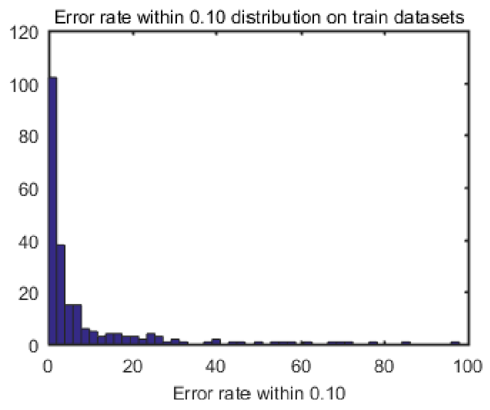


(a) The error rate within 0.05 distribution on training data

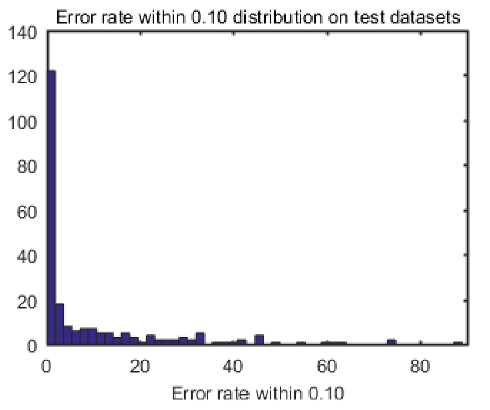


(b) The error rate within 0.05 distribution on test data

Figure 4.8: The error rate within 0.05 distribution on training and test data

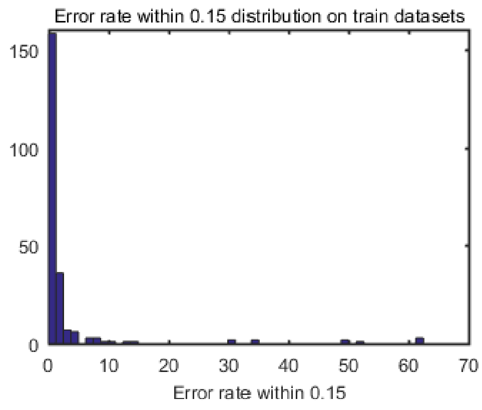


(a) The error rate within 0.1 distribution on training data

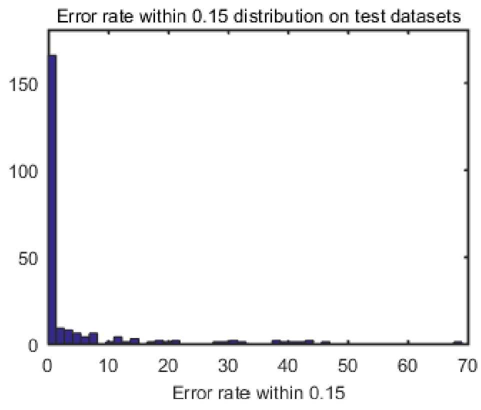


(b) The error rate within 0.1 distribution on test data

Figure 4.9: The error rate within 0.1 distribution of ESN on training and test data

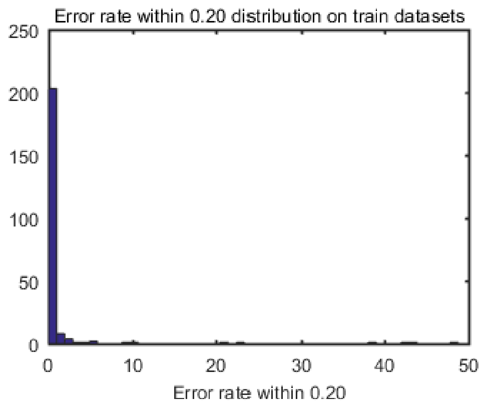


(a) The error rate within 0.15 distribution on training data

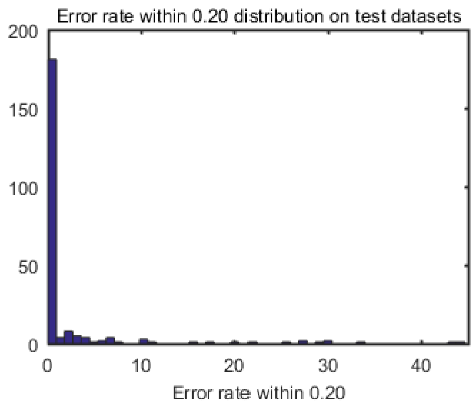


(b) The error rate within 0.15 distribution on test data

Figure 4.10: The error rate within 0.15 distribution of ESN on training and test data



(a) The error rate within 0.2 distribution on training data

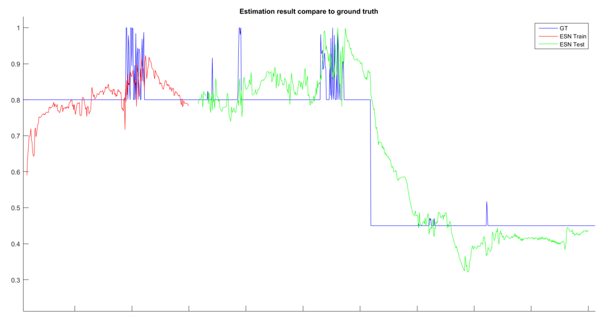


(b) The error rate within 0.2 distribution on test data

Figure 4.11: The error rate within 0.2 distribution of ESN on training and test data

estimate for training data, while green one plots the estimate for test data. In figure 4.12, the estimation algorithm could estimate road friction with acceptable accuracy on both high and low level. When the ground truth changes rapidly, the estimator is still able to converge the trend quickly and stabilize, even though the first part of sequence is almost constant. Certainly, the estimate is not as smooth as the reference which could be filtered by some low pass filters. In general, estimate is only deviated within acceptable range to the reference at most of time.

Figure 4.13 shows some extreme but rare examples on the test data that the estimates have large deviation to the reference. The first example in 4.13 shows the estimation has constant deviation to the ground truth while the second example shows that estimator has difficulty to follow the extremely rapid changes of road friction. One potential explanation for the first phenomenon is that the reservoir states in estimator fails to capture the change of features under different road friction and makes it diverges from the reference. As for the second example, the reference bounces between 0.8 to 0.3 rapidly while the estimate only bounces between 0.8



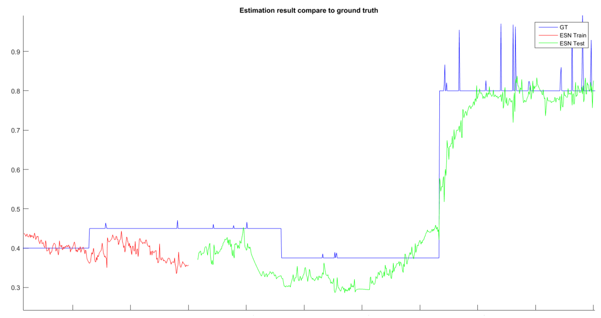
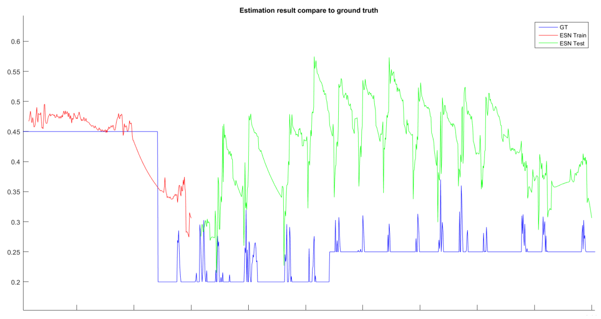


Figure 4.12: Example of estimate having small deviation from ground truth



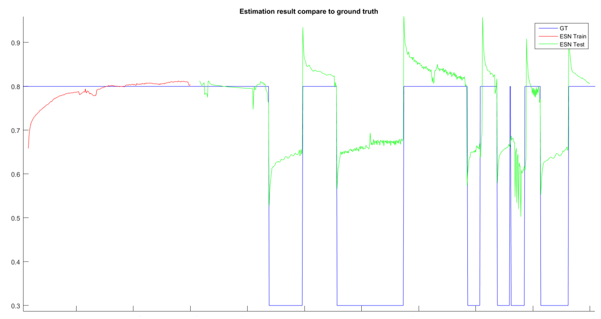


Figure 4.13: Example of estimate having large deviation from ground truth

to 0.6. The estimator tends to give more discreet estimate without rapid change. Although the estimator does not manage to give the completely accurate value, it is able to monitor the exact time when the underlying dynamics changes.
In addition, in order to evaluate the consistency of model performance when less data stream is fed, we experiment on even smaller train/test split ratio to estimate the maximum reasonable estimate length. Table 4.10 reports the comparisons of evaluation metrics between different train-test split ratio with the same hyper parameter settings. It is reasonable that with larger ratio, the model is easier to estimate accurately. The average MAE for test data increases from 0.0437 to 0.0598 if the train-test split decreases from 0.3 to 0.1. And the average error rate within 0.1 for test data increase from 9.310% to 16.471%. The performance degrades with decrease of train-test split since there is less information from sequence could be utilized to estimate for future and the total number of training sample is decreased substantially as well. However, even though the train-test split comes to 0.1, ESNs still give much better performance than physical-based model and approximately 16% error rate within 0.1 is acceptable for current application. So the model still functions well even if small amount of instances is available for accurate estimation from other source.

Table 4.10: The performance of ESN on training/test data under different train-test split percentage

Train-test split		MAE	RMSE	Average error rate within			
				0.2	0.15	0.1	0.05
0.3	Train	0.0423	0.0525	1.278%	3.1007%	8.995%	28.372%
	Test	**0.0437**	**0.0564**	**2.478%**	**4.528%**	**9.310%**	**25.794%**
0.1	Train	0.0406	0.0487	1.161%	2.840%	8.510%	27.162%
	Test	0.0598	0.0752	4.690%	8.395%	16.471%	39.640%

5
Conclusion and Future work

In this chapter, we summarize the conclusion drawn from the result and discussion chapter and future development is suggested in the end.

5.1 Conclusion

To conclude, both proposed method for on-board estimation algorithm using ESNs and framework to analyze spatial-temporal pattern using HMM based clustering achieve promising performance. As for on-board estimation algorithm, the proposed Echo State Network with feedback path has promising performance over the physical-based model for road friction estimation. The average mean absolute error between estimation to ground truth is improved from approximately 0.472 for physical model to 0.0437 for ESNs, as shown in Table 4.10. The avergae error rate within 0.1 is improved from 97.143 % to 9.310 %. The estimation algorithm works well both under high and low road friction and rapid change scenarios. From experiment, the performance of ESNs is no longer sensitive to the tire excitation as physical-based model to give accurate estimation. Besides, the proposed method could be updated with new coming data which gives advantage for continuous learning to improve performance further. The average test time for one sample is under 0.2 ms on the personal computer which fulfills real time performance requirement even though detailed low level programming optimization could improve it further.
As for the study of spatial-temporal pattern of road friction, the results shown in previous chapter indicates that the spatial-temporal pattern could be extracted by the proposed HMM based clustering framework. And clustering result could be affected by the choice of aggregation resolution. The finer the resolution is, the better the representative of the pattern at each location is, but more difficult to train HMMs due to smaller data available in small regions. But the general pattern is quite similar regardless of resolution choice.

5.2 Future work

For future work, the proposed on-board road friction estimation needs to be transferred to the embedded programming language and tested on ECU or computation unit on car to evaluate its real time performance as we discussed in previous chapter. Although we have achieved 0.2 ms test time on the dedicated CPU, the performance drop could be expected when deploying algorithm on the ECU on car since it runs hundreds of applications at the same time. And the data has been synchronized offline already which might be another influential effect on the run time performance. Besides, the sensor measurements, collected by different frequencies need to be synchronized, which cost more time and could impair the performance. Furthermore, although we motivate the reason to choose ESNs over other recurrent neural network method due to its simplicity of training, benchmark those methods on the data will be of great interest to further analyze the strengths and weakness of ESN with respect to other methods.

As for feature selection, more different techniques could be explored and compared, and ablation test on each selected feature could also help us to understand how each feature affects performance so that the network's behaviors are more interpretable and more improvement could be done to make model insensitive to data, and we could avoid disorder of algorithm in case of some abnormal readings in system.

As for the spatial-temporal pattern of road friction, there is no undergoing study yet in this field to formulate the problem in such way, therefore it is difficult to compare to state-of-the-art algorithm and evaluate the performance of proposed method. And the temporal pattern we discussed here is the short term pattern instead of long term in the scale of different months, or even years due to lack of data. Based on the analysis before, we have already known that season plays a significant role on road friction distribution. Therefore, we need not only aggregate data based on the geometric locations, but also according to date in year as well. Due to the scarcity of current data, it impossible to extract long-term pattern yet. Furthermore, from figure 3.3, a large amount of data are concentrated close to Göteborg and northern Sweden. Except from these two locations, the data is quite sparse which makes analysis more difficult. Besides, most of data are collected on highway or country road that do not cover all the variance of road. Therefore, the pattern we discovered is bias to those road.

Last but not the least, because of lack of data to validate the spatial-pattern algorithm, it is not possible yet to combine both algorithms although we do find spatial-temporal pattern could be a good prior information resource for on-board algorithm. Therefore, it could be very valuable to use spatial-temporal pattern to give rough estimate prior to merge with on-board estimation algorithm to further correct it with sensors even when there is no other information resource available.

Bibliography

[1] Mats Andersson et al. “Road friction estimation”. In: *Saab Automobile AB, Trollhättan, Sweden* (2007).

[2] Seyedmeysam Khaleghian, Anahita Emami, and Saied Taheri. “A technical survey on tire-road friction estimation”. In: *Friction* 5.2 (2017), pp. 123–146.

[3] Manuel Acosta, Stratis Kanarachos, and Mike Blundell. “Road Friction Virtual Sensing: A Review of Estimation Techniques with Emphasis on Low Excitation Approaches”. In: *Applied Sciences* 7.12 (2017), p. 1230.

[4] Laura Ryan Ray. “Nonlinear state and tire force estimation for advanced vehicle control”. In: *IEEE Transactions on Control Systems Technology* 3.1 (1995), pp. 117–124.

[5] Junmin Wang, Lee Alexander, and Rajesh Rajamani. “Friction estimation on highway vehicles using longitudinal measurements”. In: *Journal of dynamic systems, measurement, and control* 126.2 (2004), pp. 265–275.

[6] Jamil Dakhlallah et al. “Tire-road forces estimation using extended Kalman filter and sideslip angle evaluation”. In: *2008 American control conference*. IEEE. 2008, pp. 4597–4602.

[7] Moustapha Doumiati et al. “Onboard real-time estimation of vehicle lateral tire–road forces and sideslip angle”. In: *IEEE/ASME Transactions on Mechatronics* 16.4 (2011), pp. 601–614.

[8] J Alonso et al. “On-board wet road surface identification using tyre/road noise and support vector machines”. In: *Applied Acoustics* 76 (2014), pp. 407–415.

[9] Chang Sun Ahn. “Robust Estimation of Road Friction Coefficient for Vehicle Active Safety Systems.” In: (2011).

[10] Laura R Ray. “Nonlinear tire force estimation and road friction identification: Simulation and experiments1, 2”. In: *Automatica* 33.10 (1997), pp. 1819–1833.

[11] Fredrik Gustafsson. “Slip-based tire-road friction estimation”. In: *Automatica* 33.6 (1997), pp. 1087–1099.

[12] R Wade Allen, TJ Rosenthal, and HT Szostak. *ANALYTICAL MODELING OF DRIVER RESPONSE IN CRASH AVOIDANCE MANEUVERING. VOLUME 1: TECHNICAL BACKGROUND. FINAL REPORT*. Tech. rep. 1988.

[13] H.B. Pacejka and Technische Universiteit Sectie Voertuigtechniek. *Modelling of the Pneumatic Tyre and Its Impact on Vehicle Dynamic Behaviour: Hauptbd.* TU Delft, 1988.

[14] Simon J Julier and Jeffrey K Uhlmann. “New extension of the Kalman filter to nonlinear systems”. In: *Signal processing, sensor fusion, and target recognition*

VI. Vol. 3068. International Society for Optics and Photonics. 1997, pp. 182–194.

[15] LR Rabiner. “An Introduction to Hidden Markov Models”. In: *iEEE ASSP MAGAZINE* (1986).

[16] Greg Welch, Gary Bishop, et al. “An introduction to the Kalman filter”. In: (1995).

[17] Ronald J Williams and David Zipser. “A learning algorithm for continually running fully recurrent neural networks”. In: *Neural computation* 1.2 (1989), pp. 270–280.

[18] Paul Rodriguez, Janet Wiles, and Jeffrey L Elman. “A recurrent neural network that learns to count”. In: *Connection Science* 11.1 (1999), pp. 5–40.

[19] Peter J Angeline, Gregory M Saunders, and Jordan B Pollack. “An evolutionary algorithm that constructs recurrent neural networks”. In: *IEEE transactions on Neural Networks* 5.1 (1994), pp. 54–65.

[20] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. “Learning to forget: Continual prediction with LSTM”. In: (1999).

[21] Herbert Jaeger. “The “echo state” approach to analysing and training recurrent neural networks-with an erratum note”. In: *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report* 148.34 (2001), p. 13.

[22] Joe Touch, Yu-Shun Wang, and Venkata Pingali. “A recursive network architecture”. In: *ISI, Tech. Rep* 2006-626 (2006).

[23] Herbert Jaeger. *Short term memory in echo state networks*. Vol. 5. GMD-Forschungszentrum Informationstechnik, 2001.

[24] Herbert Jaeger. “Adaptive nonlinear system identification with echo state networks”. In: *Advances in neural information processing systems*. 2003, pp. 609–616.

[25] Hirotugu Akaike. “Fitting autoregressive models for prediction”. In: *Annals of the institute of Statistical Mathematics* 21.1 (1969), pp. 243–247.

[26] Paul J Werbos et al. “Backpropagation through time: what it does and how to do it”. In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560.

[27] Herbert Jaeger et al. “Optimization and applications of echo state networks with leaky-integrator neurons”. In: *Neural networks* 20.3 (2007), pp. 335–352.

[28] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. “Phased lstm: Accelerating recurrent network training for long or event-based sequences”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 3882–3890.

[29] Mantas Lukoševicius. *Echo state networks with trained feedbacks*. Tech. rep. Technical report, 2007.

[30] Mantas Lukoševičius. “A practical guide to applying echo state networks”. In: *Neural networks: Tricks of the trade*. Springer, 2012, pp. 659–686.

[31] Boudjelal Meftah, Olivier Lézoray, and Abdelkader Benyettou. “Novel approach using echo state networks for microscopic cellular image segmentation”. In: *Cognitive Computation* 8.2 (2016), pp. 237–245.

[32] Abdelkerim Souahlia et al. “An experimental evaluation of echo state network for colour image segmentation”. In: *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE. 2016, pp. 1143–1150.

[33] Petia Koprinkova-Hristova et al. “Clustering of spectral images using Echo state networks”. In: *Innovations in Intelligent Systems and Applications (INISTA), 2013 IEEE International Symposium on*. IEEE. 2013, pp. 1–5.

[34] Jochen J Steil. “Backpropagation-decorrelation: online recurrent learning with O (N) complexity”. In: *Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on*. Vol. 2. IEEE. 2004, pp. 843–848.

[35] David Sussillo and Larry F Abbott. “Generating coherent patterns of activity from chaotic neural networks”. In: *Neuron* 63.4 (2009), pp. 544–557.

[36] Fabian Triefenbach et al. “Phoneme recognition with large hierarchical reservoirs”. In: *Advances in neural information processing systems*. 2010, pp. 2307–2315.

[37] G David Forney. “The viterbi algorithm”. In: *Proceedings of the IEEE* 61.3 (1973), pp. 268–278.

[38] Ilkka Juga, Pertti Nurmi, and Marjo Hippi. “Statistical modelling of wintertime road surface friction”. In: *Meteorological Applications* 20.3 (2013), pp. 318–329.

[39] DeLiang Wang. “Temporal pattern processing”. In: *The handbook of brain theory and neural networks* 2 (2003), pp. 1163–1167.

[40] Max Menenberg et al. “Topic modeling for management sciences: A network-based approach”. In: *Big Data (Big Data), 2016 IEEE International Conference on*. IEEE. 2016, pp. 3509–3518.

[41] Qiujian Lv et al. “Big data driven hidden Markov model based individual mobility prediction at points of interest”. In: *IEEE Transactions on Vehicular Technology* 66.6 (2017), pp. 5204–5216.

[42] Sung-Hyuk Cha. “Comprehensive survey on distance/similarity measures between probability density functions”. In: *City* 1.2 (2007), p. 1.

[43] Jorge Silva and Shrikanth Narayanan. “Upper bound Kullback-Leibler divergence for hidden Markov models with application as discrimination measure for speech recognition”. In: *Information Theory, 2006 IEEE International Symposium on*. IEEE. 2006, pp. 2299–2303.

[44] Yoram Singer and Manfred K Warmuth. “Training algorithms for hidden Markov models using entropy based distance functions”. In: *Advances in Neural Information Processing Systems*. 1997, pp. 641–647.

[45] MathWorks. *Hierarchical Clustering Algorithm Description for MATLAB Statistics and Machine Learning Toolbox*. 2018. URL: `https://www.mathworks.com/help/stats/hierarchical-clustering.html`.

[46] Payam Refaeilzadeh, Lei Tang, and Huan Liu. “On comparison of feature selection algorithms”. In: *Proceedings of AAAI workshop on evaluation methods for machine learning II*. Vol. 3. 4. 2007, p. 5.

[47] Lei Yu and Huan Liu. “Feature selection for high-dimensional data: A fast correlation-based filter solution”. In: *Proceedings of the 20th international conference on machine learning (ICML-03)*. 2003, pp. 856–863.

[48] Lei Yu and Huan Liu. “Efficient feature selection via analysis of relevance and redundancy”. In: *Journal of machine learning research* 5.Oct (2004), pp. 1205–1224.

[49] J. Ross Quinlan. “Induction of decision trees”. In: *Machine learning* 1.1 (1986), pp. 81–106.

[50] William H Press et al. “Numerical recipes in C”. In: *Cambridge University Press* 1 (1988), p. 3.

A

Appendix 1

Table A.1: Feature list of datasets

Index	Name	Description
1	AbsMode_B	Anti-Brake system control is active or not for front left wheel
2	AbsMode_B	Anti-Brake system control is active or not for front right wheel
3	AbsMode_B	Anti-Brake system control is active or not for rear left wheel
4	AbsMode_B	Anti-Brake system control is active or not for rear right wheel
5	AccPedlRat_P	Acceleration pedal ratio
6	AmbTemp_deg	Ambinent temperature in Celsius
7	Ax_mps2	Longitudinal acceleration $[m/s^2]$
8	Ay_mps2	Lateral acceleration $[m/s^2]$
9	Az_mps2	Vertical acceleration $[m/s^2]$
10	BrkPedlRat_P	Break pedal ratio
11	BrkTqWhl_Nm	Break torque for front left wheel
12	BrkTqWhl_Nm	Break torque for front right wheel
13	BrkTqWhl_Nm	Break torque for rear left wheel
14	BrkTqWhl_Nm	Break torque for rear right wheel
15	CltchPedlRat_P	Clutch pedal ratio
16	DoorIsOpen_B	Door or trunk is open
17	DtSts_M	Front axle is engaged in drive train
18	DtSts_M	Rear axle is engaged in drive train
19	EngN_rpm	Engine rotational speed [rpm]
20	EpbSts_M	Electric Parking barke status
21	FrackFr_N	Rack force in steering column
22	FuLvl_L	Fuel level [liters]
23	Gear	Current active gear
24	GpsDOP	GPS horizontal dilution of precision
25	GpsDOP	GPS vertical dilution of precision
26	GpsDOP	GPS position dilution of precision
27	GpsDOP	GPS time dilution of precision
28	GpsDir_r	GPS heading [rad]

29	GpsNofS	Number of GPS satellites
30	GpsPos	Longitude position
31	GpsPos	Latitude position
32	GpsPos	Altitude position
33	GpsSpd_mps	GPS longitude change rate
34	GpsSpd_mps	GPS longitude change rate
35	GpsSpd_mps	GPS longitude change rate
36	LongCltSt_M	Longitudinal clutch status
37	LongCltTq_Nm	Longitudinal clutch status
38	PtTqWhl_Nm	Powertrain torque for front axle
39	PtTqWhl_Nm	Powertrain torque for rear axle
40	StabBrkMode_B	ABS/ESC/RSCintervention for front left wheel
41	StabBrkMode_B	ABS/ESC/RSCintervention for front right wheel
42	StabBrkMode_B	ABS/ESC/RSCintervention for rear left wheel
43	StabBrkMode_B	ABS/ESC/RSCintervention for rear right wheel
44	StabPtMaxMode_B	Engine traction control for front left wheel is at maximum
45	StabPtMaxMode_B	Engine traction control for front right wheel is at maximum
46	StabPtMinMode_B	Engine traction control for rear left wheel is at maximum
47	StabPtMinMode_B	Engine traction control for rear right wheel is at maximum
48	TorsBarTq_Nm	Steering wheel torque
49	TrsmParkLockd_B	Transmission park is active
50	WhlAng_r	Steering wheel angle
51	WhlCogCnt	Wheel cog counters for front left wheel
52	WhlCogCnt	Wheel cog counters for front right wheel
53	WhlCogCnt	Wheel cog counters for rear left wheel
54	WhlCogCnt	Wheel cog counters for rear right wheel
55	WhlSpdDir_M	Wheel speed direction of rear left wheel
56	WhlSpdDir_M	Wheel speed direction of rear right wheel
57	WhlSpd_mps	Wheel speed for front left wheel
58	WhlSpd_mps	Wheel speed for front right wheel
59	WhlSpd_mps	Wheel speed for rear left wheel
60	WhlSpd_mps	Wheel speed for rear right wheel
61	Wx_rps	Roll rate
62	Wz_rps	Yaw rate
63	Vx_mps	Longitudinal velocity at center of gravity
64	Fx_N.1	Longitudinal force of front left wheel
65	Fx_N.2	Longitudinal force of front right wheel
66	Fx_N.3	Longitudinal force of rear left wheel
67	Fx_N.4	Longitudinal force of rear right wheel
68	Fy_N.1	Lateral force of front left wheel

69	Fy_N.2	Lateral force of front right wheel
70	Fy_N.3	Lateral force of rear left wheel
71	Fy_N	Lateral force of rear right wheel
72	Fz_N.1	Vertical force of front left wheel
73	Fz_N.2	Vertical force of front right wheel
74	Fz_N.3	Vertical force of rear left wheel
75	Fz_N.4	Vertical force of rear right wheel
76	Slip.1	Slip rate of front left wheel
77	Slip.2	Slip rate of front right wheel
78	Slip.3	Slip rate of rear left wheel
79	Slip.4	Slip rate of rear right wheel
80	WhlBpNrj	Bandpass-filtered energy in suspension mode
81	Tire	Tire type
82	Surface	Surface type
83	RoadType	Road type