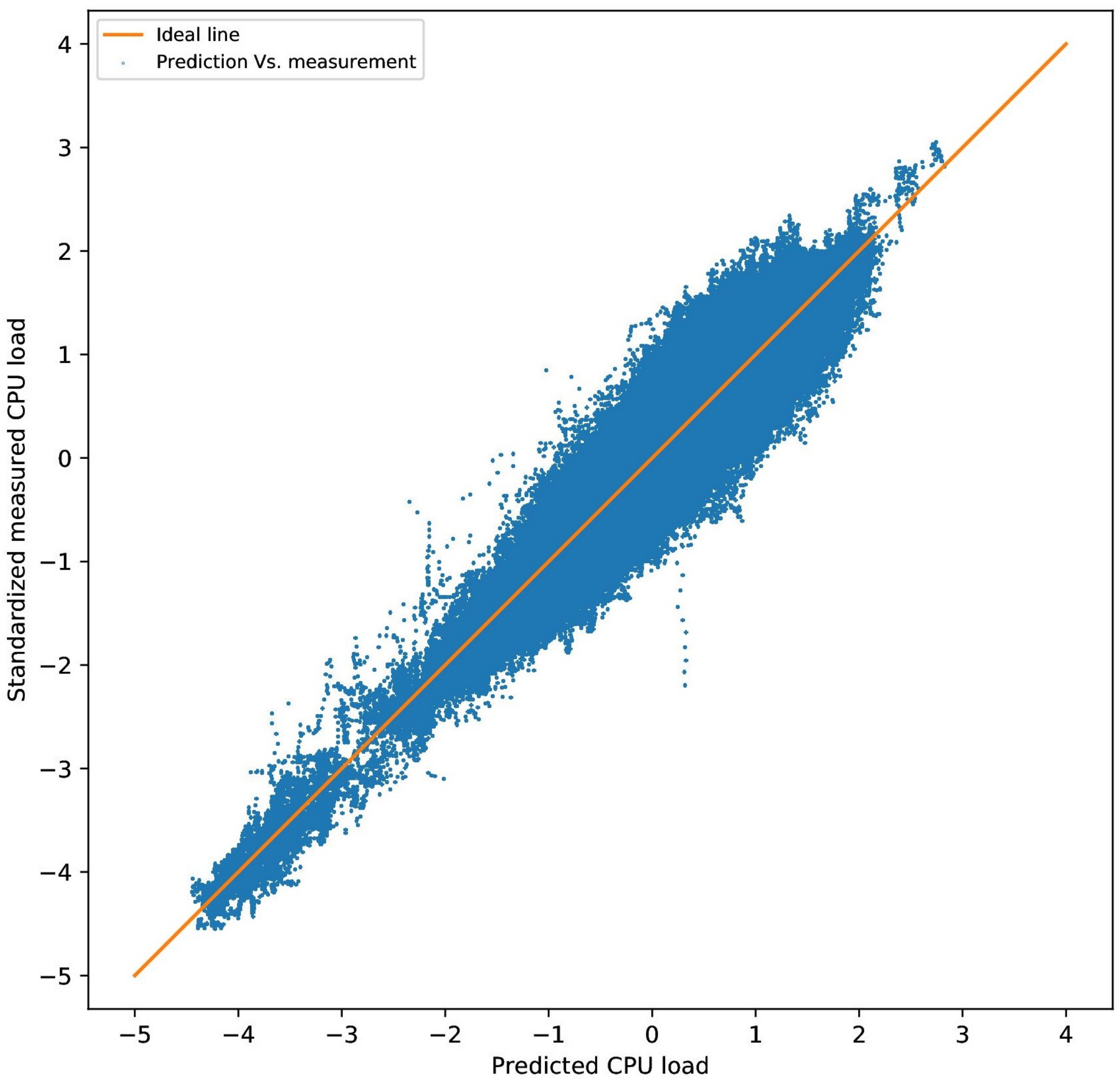

Workload Analysis of Active Safety Processor

Master of Science Thesis in the Embedded Electronic System Design

Haoge Liu

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2018

Master's thesis 2018

Workload Analysis of Active Safety Processor

Haoge Liu

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2018

Workload Analysis of Active Safety Processor
Haoge Liu

Supervisor: Lars Svensson, Computer Science and Engineering
Advisor: Muhammad Shahid, VOLVO CAR CORPORATION
Examiner: Per Larsson-edefors, Computer Science and Engineering

Master's Thesis 2018
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone + 46 (0)31-772 1000

Cover: Scatter plot of predicted CPU load and measured CPU load
Typeset in LaTeX
Department of Computer Science and Engineering
Gothenburg, Sweden 2018

Workload Analysis of Active Safety Processor
Haoge Liu
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Active Safety (AS) is an important feature to protect drivers and passengers in vehicles, however, the growing complexity of AS functions increases the CPU workload of the AS processor. This thesis attempts to 1) identify the signals that cause more CPU workload than others and 2) predict this CPU workload. In order to achieve the purpose, regression models of CPU workload in real-time contexts are proposed. Based on these models, different feature selection techniques are used to rank the contributions of signals. As a result, given the sampled input signals to all the software modules that are embedded running on the CPU, a model for predicting CPU workload in a real-time context is proposed. Signals and the associated contribution coefficients are listed using different feature selection methods and criteria. Volvo Car Corporation will be able to trace back to the functions that generate signals with high contribution coefficients according to the list of signals, and improve the corresponding AS functions to decrease the CPU workload.

Keywords: real-time system, feature selection, linear regression, autonomous driving.

Acknowledgements

This thesis project was carried out at Volvo Car Corporation. I would like to thank Driver Assistance and Active Safety group for giving me the thesis opportunity, and thank technical help, meetings and discussions from Muhammad Shahid– supervisor at Volvo and Lars Svensson– supervisor at Chalmers.

Haoge Liu, Gothenburg, July/2018

Contents

List of Figures

List of Tables

Glossary

AS	Active Safety
VCC	Volvo Car Corporation
AD	Autonomous Driving
ECU	Electronic Control Unit
HP	Host Processor
RTS	Real-time system
SWC	Software component
HIL	Hardware-in-the-loop
MSE	Mean squared error
PCA	Principal Component Analysis
PCR	Principal Component Regression
LASSO	Least absolute shrinkage and selection operator
VIF	Variance inflation factor
OLS	Oridinary least squares
IID	Independent and identically distributed
MLE	Maximum likelihood estimation
RSS	Regression sum of squares
TSS	Total sum of squares
ESS	Error sum of squares
LOOCV	Leave-one-out Cross-Validataion
AIC	Akaike information criterion
BIC	Bayesian information criterion

1
Introduction

1.1 Background

Active Safety (AS) is an important system in the field of vehicular safety. It is designed to activate protection measures in order to forestall accidents if possible or at least mitigate the effects. Autonomous driving (AD) has been stimulated by improvement of sensor technology and machine learning. National Highway Traffic Safety Administration (NHTSA) in US defined 6 levels of driver assistance technology advancements merging AS and AD [1].

Level 0. **No Automation:** Driver handles all driving tasks.
Level 1. **Driver Assistance:** Driver takes control of vehicle under limited driving assist.
Level 2. **Partial Automation:** Some automated functions are included under some certain circumstances, such as auto braking and steering, but the driver needs to keep monitoring the environment.
Level 3. **Conditional Automation:** Based on the previous level, the driver do not need to monitor the environment, but the driver should be ready to control the vehicle all the time.
Level 4. **High Automation:** The vehicle is able to perform all the driving tasks under some certain circumstances. The driver has options to control the vehicle.
Level 5. **Full Automation:** The vehicle is able to perform all the driving tasks under all circumstances. The driver has options to control the vehicle.

There is no clear boundary between AS and AD. Roughly, low levels of automation are in the scope of AS and high levels of automation are associated with AD concept. Starting from level 3, vehicles are able to monitor and detect driving environment, which means more powerful sensors are going to be integrated into cars, which means an increasing computational capacity is required.

1.2 ECU CPU load

Electronic Control Unit (ECU) is an embedded system for controlling electrical systems and vehicle behavior. Different ECUs perform different functions in a car. Generally speaking, an ECU consists of CPU, RAM, ROM, I/O extender, A/D converter, sensors, etc.

In Volvo Car Corporation (VCC), AS is realized by several ECUs, which are responsible for different functions in a vehicle and drive actuators. AS functions like collision warning with full auto brake, lane keeping assist, animal detection, etc, are pre-programmed and running embedded on the ECU called Host Processor (HP), connected to sensor systems and actuators. During driving, CPU processes sensor data such as ultrasonic and radar data, and analyzes the data in real time to detect traffic scenario and decide whether to activate protection measures or not. When a dangerous situation is detected, AS will send command to electrical controller to take actions, like applying brake and auto steering.

In the trend of AD, VCC promised in 2016 a zero fatality rate in its vehicle by 2020 [2]. Among many ECUs, the CPU in HP ECU is responsible for executing AS functions in a real-time embedded system. As more and more functions associated with AD are introduced to AS system, the CPU in HP that is responsible for AS functions is facing growing pressures of computing load. The aim of this project is to find which signals in ECU have most impact on CPU load and use those features in predictive models.

The HP ECU is a real-time system, and in this section, the basics of real-time systems and CPU scheduling are introduced. From a hardware perspective, a task is finished before a time limit, which is called deadline. We assume that the effect of a sample on the CPU load is valid over the length of deadline corresponding to the signal related task. Thus, in our model, the current CPU load is regarded as a function of several past samples.

1.2.1 Tasks scheduling

The HP we are working on is a real-time system (RTS), which is based on Automotive Open System Architecture (AUTOSAR). AUTOSAR provides standardized software architecture for automotive ECU [3]. In a real-time system, generally speaking, tasks are scheduled by the CPU according to their priorities. Figure 1.1 is an example of scheduling tasks.

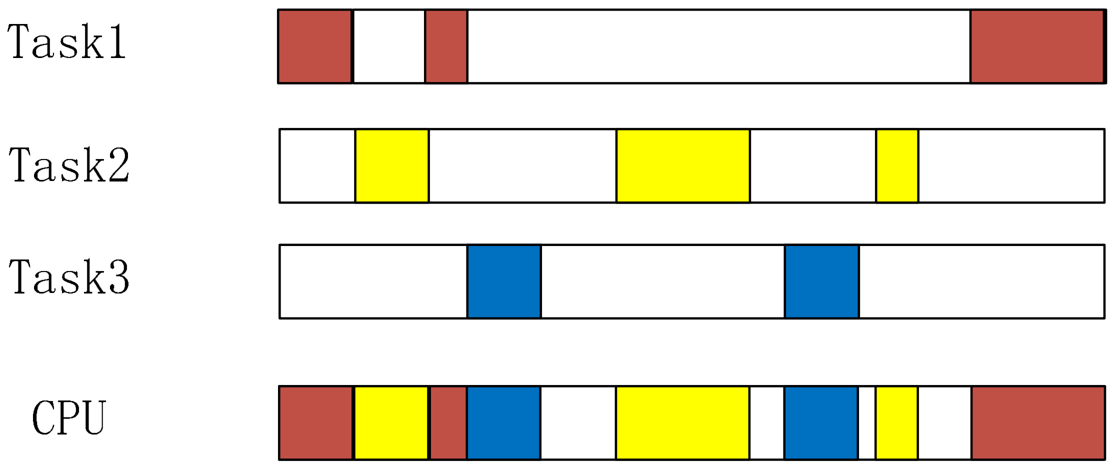

Figure 1.1: An example of scheduling tasks on single-core RTS: 3 tasks are scheduled by the CPU, and they consume different amount of CPU time represented by different color. For the CPU, the white block represents CPU is idle, otherwise the CPU is running.

1.2.2 CPU load calculation on AUTOSAR

According to AUTOSAR official documents [4], the CPU load is calculated by a averaging sliding window. The CPU usage is the average of CPU time over a period of time (T_w) :

$$L = \frac{\sum T_b}{T_w} = 1 - \frac{\sum T_i}{T_w}$$

where L is the CPU load measured over a sliding window, $\sum T_b$ is the total time when the CPU is busy, $\sum T_i$ is the total time when the CPU is idle, and T_w is the size of time window. Figure 1.2 is an example of calculating CPU usage.

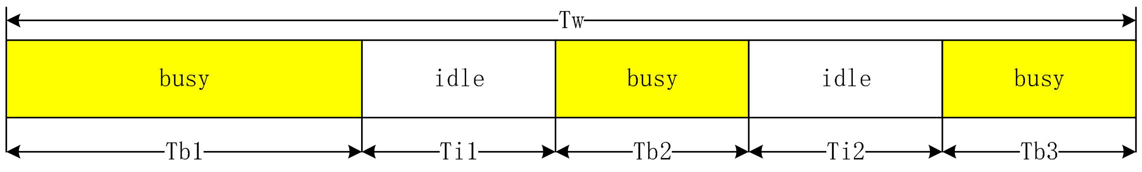

Figure 1.2: For the given case, the CPU load is $\frac{T_{b1}+T_{b2}+T_{b3}}{T_w}$

When the window is moving, the current CPU load L_t at time t is the ratio of the total busy time of CPU to the length of measuring window, so the CPU load is a continuous curve as a function of time t, which agrees with the sampled CPU load plotted in Figure 4.1. Figure 1.3 illustrates the way of measuring CPU load.

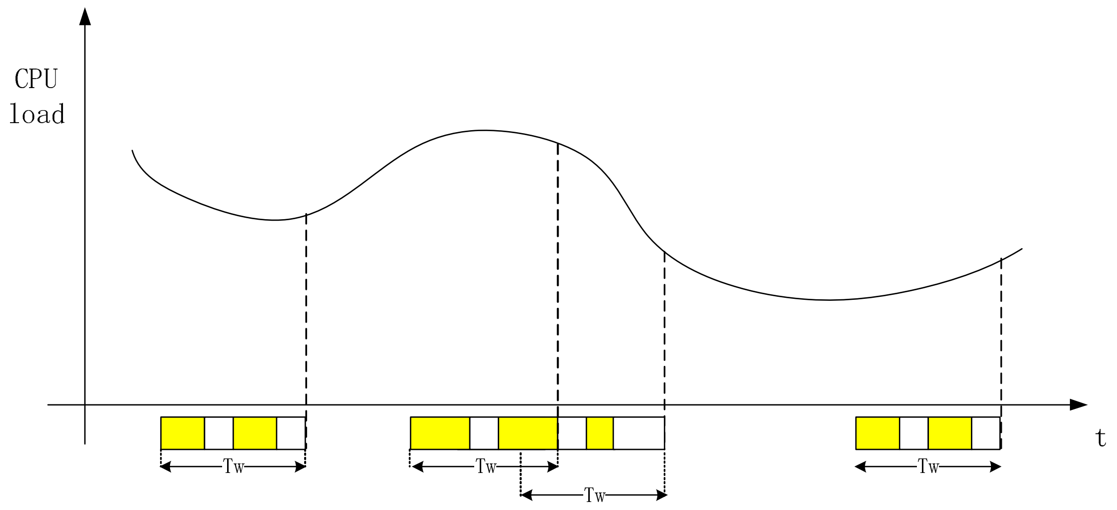

Figure 1.3: CPU usage curve calculated by the sliding window. White blocks indicate the length of CPU idle time, and yellow blocks indicate the CPU busy time.

The CPU usage is directly dependent on the size of busy time of the CPU. The models we are trying to build actually map the values of external and internal signals to the required CPU busy time in one T_w.

1.2.3 Dynamic model assumption

The sampled data are the CPU usage and the input signals of the functions that are executed by the CPU. Figure 1.4 illustrates how the data are sampled.

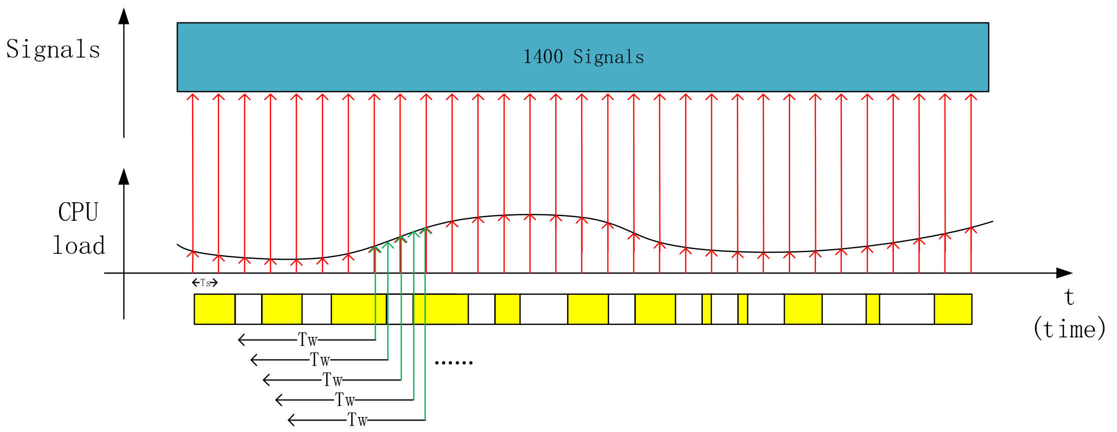

Figure 1.4: The blue bar represents the 1400 sampled signals. CPU load is updated by the sliding window with the length of T_w. Red arrows show the sampling time with sampling interval (Ts). Green arrows show that CPU load curve is plotted by calculating the average of CPU busy time over the past interval T_w.

Figure 1.5 illustrates the current CPU load only depends on the samples in the past effect window (250 ms). We assume the effects of the previous signals on CPU load are not earlier than 250 ms, which means some of the samples inside the effect window may have contributions to the current CPU load and some may not. Among those samples, some are excluded by calculating their correlation with the current CPU load.

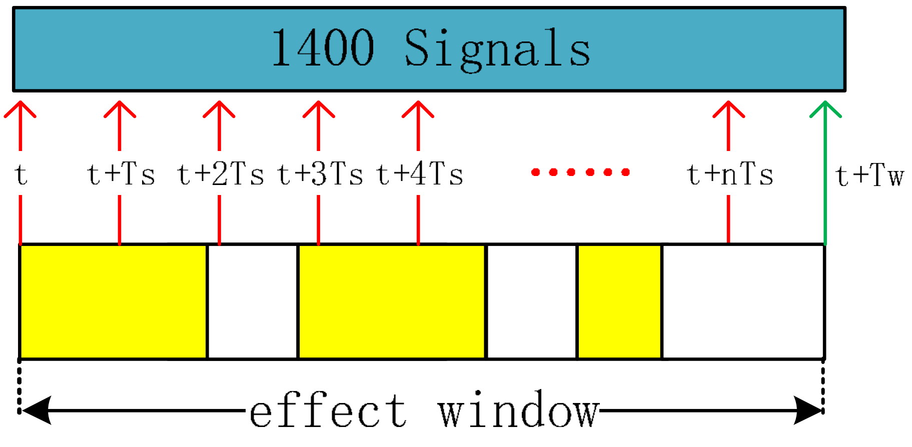

Figure 1.5: The CPU load calculated at time $t + T_w$ (green arrow) may depend on some of the sampled values of the 1400 signals at time *t*, *t+Ts*, *t+2Ts*, ..., *t+nTs*, where Ts is the sampling interval.

Denote the values sampled at time t as $\boldsymbol{X}^t$ and the CPU load sampled at time $t+T_w$ as Y^{t+T_w}, the hypothesis model is:

$$Y^{t+T_w} = \boldsymbol{X}^t\boldsymbol{\beta}^0 + \boldsymbol{X}^{t+Ts}\boldsymbol{\beta}^1 + \boldsymbol{X}^{t+2Ts}\boldsymbol{\beta}^2 + ... + \boldsymbol{X}^{t+nTs}\boldsymbol{\beta}^n$$

$$Y(t+T_w) = \boldsymbol{X}(t)\boldsymbol{\beta}^0 + \boldsymbol{X}(t+Ts)\boldsymbol{\beta}^1 + \boldsymbol{X}(t+2Ts)\boldsymbol{\beta}^2 + ... + \boldsymbol{X}(t+nTs)\boldsymbol{\beta}^n$$

$$Y(t+T_w) = \sum_{i=0}^{n} \boldsymbol{X}(t+iTs)\boldsymbol{\beta}^i$$

where Ts is the sampling interval, $\boldsymbol{X}^{t+nTs}$ is sampled signals right before the CPU load ($Y(t+T_w)$) is updated at time $t+T_w$, T_w is the CPU calculation interval, and $\boldsymbol{\beta}^0, \boldsymbol{\beta}^1, ..., \boldsymbol{\beta}^n$ are the coefficients vectors associated with $\boldsymbol{X}^t, \boldsymbol{X}^{t+Ts}, ..., \boldsymbol{X}^{nTs}$. Each sample, $\boldsymbol{X}(t+iTs)$, and the associated coefficient $\boldsymbol{\beta}^i$ has 1400 elements corresponding to each signal in the dataset.

1.3 Context

1.3.1 Problem description

In this thesis, the database is a 475835-by-1400 matrix taken from test vehicles driven in real traffic. There are 1400 types of signals in total and each signal has 475835 samples, corresponding to 198 minutes of driving. Based on this database, we want to build mathematical models which are able to identify which input signals cause more CPU load than others and predict CPU usage. The predictors in the models are selected subset of 1400 kinds of signal including sensor signals like ultrasonic [5], radar [6], lidar [7] signal, etc and output signals of internal SWCs. The response variable in the model is the CPU usage in percent of full load.

1.3.2 Proposed model

All the AS functions are running embedded on HP ECU, connected to sensor systems and actuators [8], so the CPU usage is directly affected by what is executed in HP and CPU usage in turn can be deduced from the internal signals in HP. In order to explain the CPU usage, we need a *mathematical model* that should be able to represent the relationship between CPU usage and internal signals. The *actual* relationship can be extremely complicated: different values of internal signal might trigger different functions that cause different CPU usage; CPU usage is very likely a non-linear function of input signals. We want to know which signals cause more CPU load than others in a interpretive way and predict CPU usage using signals, as input features, contained in the HP ECU, so interpretability of the model that we try to build to represent the relationship between CPU usage and internal signals is the first priority. Therefore our fundamental model is:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + ... + \beta_n x_n + \varepsilon \tag{1.1}$$

where y is the CPU usage we try to predict, $x_1, x_0, ..., x_n$ are input signals, $\beta_0, \beta_1, ..., \beta_n$ are model coefficients corresponding to input signals, and ε is stochastic term. Based on this linear regression model, polynomial regression model can be used to improve the quality of fit:

$$y = \beta_0 + \beta_1 x_1 + \beta_1' x_1^2 + \beta_2' x_2 + \beta_2 x_2^2 + ... + \beta_n x_n + \beta_n' x_n^2 + \varepsilon \tag{1.2}$$

As mentioned, the actual relationship between CPU usage and internal signals is unknown and can be extremely complicated, so it is necessary to evaluate the quality of the model before trusting its results.

1.3.3 Methodology context

In the field of statistics, statistical modeling, linear regression, can be used for ranking the inputs by the extent of correlation between each input and the output. Many statistical algorithms are available and well developed. For example, Hastie et al. have summarized comprehensive statistical techniques [9] that can be used in this thesis. The problem of selecting those of a set of inputs that are most significant for the output is known as the *feature selection problem* and different input signals are treated as different features in feature selection algorithms. One key aspect is to avoid over-fitting to the training data [10]. The project described here aims to investigate several alternative models generated by different algorithms and compare their performance.

Besides statistical methods, machine learning can also be used to build a regression model. Especially, it is good at mapping deep non-linearity between input and output, as discussed by Achim Zielesny [11]. Rogers et al introduced the ability of machine learning to handle regression problems in Chapter 1 of [12], but there are two drawbacks of machine learning methods:

1. Although feature selection is exploited well in [13], [14] and [15], the algorithm that directly selects features at the input layer is not well-studied. Inputs in this topic are samples of 1400 kinds of signals.
2. Compared with linear models, machine learning methods are good at modeling non-linearity of complex systems [11], but the model trained by machine learning algorithms can not quantitatively relate each input with output in the manner of a linear regression.

Because of these two drawbacks, general machine learning algorithms based on neural network are not suitable for this practical problem. But recent studies on sparse coding [16] and regularization [17] give us an inspiration that it is possible to overcome the drawbacks and select only the inputs that are relevant to the output right at the input layer: If it is a N-dimension input Neural network, we can weight the input by a N dimensional array (weight array) before the first hidden layer, then we add a Least Absolute Shrinkage and Selection Operator (LASSO) term [18] of the weight array to the loss function and train the neural network to minimize the loss to approximate to target value in the conventional way. As a result of training, ideally, not only the output approximate to the target value, but also the weight array becomes sparse because of the LASSO term. Each element of the weight array can be regarded as the importance of the corresponding input, therefore an important input is associated with an element that has a high value, and an element of zero value means the corresponding input is irrelevant to the output.

So the future research focus is to come up a machine learning algorithm that achieves building model and keeping interpretable. Because of the ability of selecting important input, such algorithm not only is able to handle the problem at hand, but also can be used in general feature selection problem, data compression, auto encoding [19], etc.

1.3.4 Current status

Given the subset of the 1400 kinds of signal selected by previous work, a linear prediction model exists [20], which ranks the most important signals that correlate strongly with CPU load and achieves 0.17 mean squared error (MSE, defined in Equation 2.24) predicting CPU load. This model employs two methods: elastic net and principal components regression (PCR) [13], and one evaluation factor: mean squared error (MSE) that is used to estimate the accuracy of prediction model.

For elastic net method, 250 selected features resulted in an MSE of around 0.18 after data re-scaling. Elastic net regularization [21] is one kind of penalty terms in regularization technique. Regularization can avoid over-fitting in linear and logistic regression by introducing penalty term to the loss function. Detailed explanation of regularization and elastic net can be found in [21]. For PCR method, 350 principal components resulted in around 0.17 MSE; input feature quantification was left as future work. PCR is a method for reducing data dimension by applying principal components analysis (PCA) [22] to regression problem. More introduction about PCR and partial least squares regression (PLSR) is presented in Section 2.

Although the main purpose of the existing model is to capture the main contributors to the CPU load in an interpretable way, there is room to increase complexity if accuracy could also be improved, and such statistical modeling methods are worth a comprehensive coverage so that we can draw more convincing conclusion of which inputs cause more CPU load than others.

1.4 Goal

For the sake of efficient usage of CPU in realistic driving conditions, we want to know which input signals are irrelevant to CPU usage and which signal cause more CPU load than others. Under the assumption of that 250 ms window can cover the effects of all the signals on the CPU load, the industry goals include:

Goal 1 Evaluate the effect of each signal on CPU load.

Goal 2 Based on the different feature selection algorithms and evaluation criteria, building regression models to quantitatively predict CPU usage given input signals.

Goal 3 Validate above results in hardware-in-the-loop (HIL [23]) virtual environment.

Goal 4 The best mean squared error (MSE) of the existing models achieved is 0.17. The aimed MSE is below 0.15.

Linear model is under the assumption that CPU usage is a linear summation of impact of signals that are executed in CPU. In practice, some signals are highly correlated with each other, so a high degree of multicollinearity (discussed in Section 2.1.4) exists in the dataset. As discussed in Section 2.1.4, we use variance inflation factor (VIF) to measure the degree of multicollinearity of a feature. We care about not only prediction accuracy, but also the impact of each feature on CPU usage, but high degree of multicollinearity prevent us from getting accurate estimation of coefficients of every signals, so we need to find a method to overcome the multicollinearity problem.

1.5 Chapter overview

After literature study, multiple linear regression, Ordinary least squares (OLS), multicollinearity problem, criteria for feature selection, Principal component regression (PCR), Partial least squares regression (PLSR) and Ridge regression (RR) are discussed in Chapter 2. Criteria that can be used for feature selection are derived in Section 2.1.5. Chapter 3 mainly proposes backward elimination and forward selection strategies for feature selection on principal components. By using different feature selection methods and criteria, MSEs and contribution coefficients are shown in Chapter 4. To sum up, Chapter 5 mainly summarizes 3 weaknesses of the model, and forecasts future work.

2
Theory

In this chapter, theories behind feature selection methods and regression algorithms will be discussed. Ordinary least squares (OLS) is the most popular method for building regression models and the underlying method for feature selection algorithms. OLS will be discussed first. Shrinkage methods[9], PCR[24] and PLSR[25] will also be explained.

2.1 Ordinary least squares

Maximum likelihood estimation (MLE) is a method for estimating coefficients in statistical models. Between 1912 and 1922, MLE was popularized by Ronald Fisher and became fundamental basis of OLS [26]. Below, the connection between MLE and OLS is discussed first. Before explaining multicollinearity, some properties of OLS will be derived. Some criteria to evaluate regression model is mentioned in Section 2.1.5.

2.1.1 Model assumptions and MLE

The proposed model of Equation 1.1 can be rewritten in matrix format:

$$\begin{bmatrix} Y_1 \\ Y_2 \\ ... \\ Y_n \end{bmatrix} = \begin{bmatrix} 1 & X_{11} & X_{21} & ... & X_{p1} \\ 1 & X_{12} & X_{22} & ... & X_{p2} \\ 1 & X_{13} & X_{23} & ... & X_{p3} \\ ... & ... & ... & ... & ... \\ 1 & X_{1n} & X_{2n} & ... & X_{pn} \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ ... \\ \beta_p \end{bmatrix} + \begin{bmatrix} \varepsilon_0 \\ \varepsilon_1 \\ ... \\ \varepsilon_p \end{bmatrix}$$

$$\begin{aligned} \boldsymbol{Y} &= \beta_0 + \beta_1 \boldsymbol{X^1} + \beta_2 \boldsymbol{X^2} + \beta_3 \boldsymbol{X^3} + ... + \beta_n \boldsymbol{X^n} + \boldsymbol{\varepsilon} \\ &= \boldsymbol{X\beta} + \boldsymbol{\varepsilon} \qquad (2.1) \\ &= \begin{bmatrix} 1 & \boldsymbol{D} \end{bmatrix} \boldsymbol{\beta} + \boldsymbol{\varepsilon} \qquad (2.2) \\ \text{s.t. } & \boldsymbol{X} \in \mathbb{R}^{n \times (p+1)},\ \boldsymbol{\beta} \in \mathbb{R}^{p+1},\ \boldsymbol{\varepsilon} \in \mathbb{R}^{p+1} \end{aligned}$$

where $\boldsymbol{D}$ and $\boldsymbol{Y}$ are data samples matrices, $\boldsymbol{\beta}$ is the model coefficients column vector that we are going to estimate and $\boldsymbol{\varepsilon}$ is a stochastic term. We assume $\boldsymbol{\varepsilon}$ is independent and identically distributed (IID) and has Gaussian distribution with zero mean. Therefore $\boldsymbol{Y}$ follows Gaussian distribution with mean of $\boldsymbol{X\beta}$:

$$\varepsilon \sim \mathcal{N}(0, \sigma^2)$$

$$\Rightarrow P(\varepsilon) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(\varepsilon-0)^2/2\sigma^2}$$

$$Y \sim \mathcal{N}(\boldsymbol{X\beta}, \sigma^2).$$

$$\Rightarrow P(Y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(Y-\boldsymbol{X\beta})^2/2\sigma^2}$$

where $P(\varepsilon)$ and $P(Y)$ are probability distribution function (PDF) of ε and Y respectively, and σ is the standard deviation of distribution.

Based on the model and assumption, we can deduce the $\boldsymbol{\beta}$ that maximizes the probability that the model reproduces sampled data that we have. Let $P(Y_n|\boldsymbol{\beta})$ represents the probability that the model with $\boldsymbol{\beta}$ as coefficients gives rise to Y_n and joint probability density function $P(\boldsymbol{Y}|\boldsymbol{\beta})$ represents the probability that the model with $\boldsymbol{\beta}$ as coefficients reproduces our sample matrix $\boldsymbol{Y}$, where $\boldsymbol{Y} = \begin{bmatrix} Y_1 & Y_2 & ... & Y_n \end{bmatrix}$. Because $\boldsymbol{Y}$ is IID, $P(\boldsymbol{Y}|\boldsymbol{\beta})$ is the product of $P(Y_n|\boldsymbol{\beta})$:

$$l(\boldsymbol{\beta}) = P(\boldsymbol{Y}|\boldsymbol{\beta}) = \prod_{i=1}^{n} P(Y_i|\boldsymbol{\beta}) \tag{2.3}$$

$l(\beta)$ is the likelihood function [27] and we are going to find the $\hat{\beta}$ that maximizes $l(\beta)$:

$$\begin{aligned}\hat{\boldsymbol{\beta}} &= \underset{\beta}{\operatorname{argmax}}\ l(\boldsymbol{\beta}) = \underset{\beta}{\operatorname{argmax}}\ \prod_{i=1}^{n} P(Y_i|\boldsymbol{\beta}) \\ &= \underset{\beta}{\operatorname{argmax}}\ (\frac{1}{\sigma\sqrt{2\pi}})^2 e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(Y_i-\boldsymbol{X}_i\boldsymbol{\beta})^2} \end{aligned} \tag{2.4}$$

For equation 2.4, it is obvious that $l(\boldsymbol{\beta})$ reach its maximum value, $(\frac{1}{\sigma\sqrt{2\pi}})^2$, if there is such a $\hat{\boldsymbol{\beta}} = \begin{bmatrix}\hat{\beta}_0, \hat{\beta}_1, ..., \hat{\beta}_p\end{bmatrix}^T$ that $\sum_{i=1}^{n}(Y_i - \boldsymbol{X}_i\hat{\boldsymbol{\beta}})^2$ is zero, so $\hat{\boldsymbol{\beta}}$ can be deduced by solving the following optimization problem,

$$\hat{\boldsymbol{\beta}} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{n}(Y_i - X_i\boldsymbol{\beta})^2, \tag{2.5}$$

which is exactly the key idea of ordinary least squares (OLS).

2.1.2 Formula for estimating model coefficients

Ordinary least squares (OLS) is a method for solving linear regression problem based on maximum likelihood estimation and the assumption of stochastic term is Gaussian distribution. In this section, formula for estimating model coefficients,$\boldsymbol{\beta}$ will be derived. First we define the estimated response value $\hat{\boldsymbol{Y}}$ as

$$\hat{\boldsymbol{Y}} = \begin{bmatrix}\hat{Y_1}, \hat{Y_2}, ..., \hat{Y_n}\end{bmatrix}^T \tag{2.6}$$

where $\hat{Y_n}$ is the predicted value based on the n-th observed data $\boldsymbol{X_n}$

$$\begin{aligned}\hat{Y_n} &= \hat{\beta_0} + X_n^1\hat{\beta_1} + X_n^2\hat{\beta_2} + X_n^3\hat{\beta_3} + ... + X_n^p\hat{\beta_p} \\ &= \boldsymbol{X_n}\hat{\boldsymbol{\beta}} \end{aligned} \tag{2.7}$$

where $\hat{\boldsymbol{\beta}}$ is estimated model coefficients vector, $\boldsymbol{X}^j$ is a column vector representing the j-th predictor, and $\boldsymbol{X} = \begin{bmatrix}\mathbf{1} & \boldsymbol{X}^1 & \boldsymbol{X}^2 & ... & \boldsymbol{X}^p\end{bmatrix}$. The error of the trained model is defined as:

$$\boldsymbol{\epsilon} = \boldsymbol{Y} - \hat{\boldsymbol{Y}} \tag{2.8}$$

OLS calculates the coefficients by minimizing the sum of the squares of the error terms. The sum of squares of error term is called loss function (L) and it is written as:

$$\begin{aligned} L &= \sum_{i=1}^{n}(Y_i - \hat{Y_i})^2 = \sum_{i=1}^{n}\epsilon_i^2 \qquad (2.9) \\ &= (\boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}^T)(\boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}) \\ &= \boldsymbol{Y}^T\boldsymbol{Y} + \hat{\boldsymbol{\beta}}^T\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\beta}} - 2\hat{\boldsymbol{\beta}}^T\boldsymbol{X}^T\boldsymbol{Y} \end{aligned}$$

where n is the number of data samples, y_i is the i-th actual value of response variable and $\hat{y}_i$ is the i-th estimated value of response variable. So OLS is actually a process of solving a minimization problem:

$$\hat{\beta_0}, \hat{\beta_1}, ...\hat{\beta_n} = \underset{\beta_0,\beta_1,...\beta_n}{\operatorname{argmin}} L = \underset{\beta_0,\beta_1,...\beta_n}{\operatorname{argmin}} \sum_{i=1}^{n}(Y_i - \hat{Y_i})^2 = \underset{\beta_0,\beta_1,...\beta_n}{\operatorname{argmin}} \sum_{i=1}^{n}\epsilon_i^2 \tag{2.10}$$

This minimization problem can be solved by setting the partial derivative of L with respect to $\hat{\boldsymbol{\beta}}$ to zero:

$$\begin{aligned} &\frac{\partial L}{\partial\hat{\boldsymbol{\beta}}} = 2\boldsymbol{X^T}\boldsymbol{X}\hat{\boldsymbol{\beta}} - 2\boldsymbol{X^T}\boldsymbol{Y} = 0 \\ &\Rightarrow \boldsymbol{X^T}\boldsymbol{X}\hat{\boldsymbol{\beta}} = \boldsymbol{X^T}\boldsymbol{Y} \qquad (2.11) \\ &\Rightarrow (\boldsymbol{X^T}\boldsymbol{X})^{-\mathbf{1}}\boldsymbol{X^T}\boldsymbol{X}\hat{\boldsymbol{\beta}} = (\boldsymbol{X^T}\boldsymbol{X})^{-\mathbf{1}}\boldsymbol{X^T}\boldsymbol{Y} \\ &\Rightarrow \hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y} \qquad (2.12) \end{aligned}$$

Equation 2.12 can be used to estimate model coefficients.

Table 2.1 summarizes the notations and its corresponding meanings

2.1.3 Properties of OLS

In order to explain the effect of multicollinearity of dataset on coefficients, in this section, some properties of OLS will be discussed and corresponding proof will be presented.

Table 2.1: Symbols meanings

$\boldsymbol{i}$	index of observed data points, $i \in \mathbb{Z}^+$
$\boldsymbol{j}$	index of observed predictor variables, $j \in \mathbb{N}$
p	number of observed predictor variables. p is 1400 for the problem at hand
n	number of observed data points. n is 475878 for the problem at hand
$\boldsymbol{Y}$	observed response variable in matrix; sampled CPU load for this problem
$\boldsymbol{X}$	observed predictor matrix
$\boldsymbol{X}^j$	the j-th predictor variable values in a column vector with length of n
$\boldsymbol{X}_i$	the i-th observed data point in a row vector with length of *p+1*
X_i^j	the the value of j-th predictor in i-th observed data point $\boldsymbol{X}_i$
$\hat{Y}_i$	the i-th predicted value based on the i-th observed data point $\boldsymbol{X}_i$
$\hat{\boldsymbol{Y}}$	predicted value in matrix, $\hat{\boldsymbol{Y}} = \left[\hat{Y}_1, \hat{Y}_2, ..., \hat{Y}_n\right]^T$
β_j	the coefficient corresponding to the j-th predictor $\boldsymbol{X}^j$
β_0	intercept of regression model
$\hat{\beta}_j$	estimator of β_j
$\hat{\beta}_0$	estimator of β_0
$\boldsymbol{\beta}$	$\boldsymbol{\beta} = \begin{bmatrix}\beta_0 & \beta_1 & ... & \beta_j\end{bmatrix}^T$; coefficients column vector with lengtn of $p+1$
$\hat{\boldsymbol{\beta}}$	estimator of $\boldsymbol{\beta}$
ε	stochastic term of regression model
$\boldsymbol{\epsilon}$	$\boldsymbol{\epsilon} = \boldsymbol{Y} - \hat{\boldsymbol{Y}}$ error of trained model

Equation 2.12 calculates model coefficients by setting partial derivative of *Loss* with respect to β_n to zero. Particularly,

$$\frac{\partial Loss}{\partial \beta_0} = \sum_{i=1}^{n} 2(Y_i - \hat{Y}_i)(-1) = 0 \Rightarrow \sum_{i=1}^{n} \epsilon_i = 0 \tag{2.13}$$

shows that the sum of error term ϵ_i is zero and ϵ_i is also called residual representing the information that can not be explained by our model.

Equation 2.13 gives rise to **property 1**: The sum and mean of residuals is zero. Rewrite Equation 2.11 and replace $\boldsymbol{Y}$ by $\boldsymbol{X\beta} + \boldsymbol{\epsilon}$:

$$\begin{aligned}
&\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\beta}} = \boldsymbol{X}^T\boldsymbol{Y} \\
\Rightarrow &\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\beta}} = \boldsymbol{X}^T(\boldsymbol{X}\hat{\boldsymbol{\beta}} + \boldsymbol{\epsilon}) \\
\Rightarrow &\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\beta}} = \boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\beta}} + \boldsymbol{X}^T\boldsymbol{\epsilon} \\
\Rightarrow &\boldsymbol{X}^T\boldsymbol{\epsilon} = 0 \\
\Rightarrow &\begin{bmatrix}\boldsymbol{1} & \boldsymbol{X}^1 & \boldsymbol{X}^2 & ... & \boldsymbol{X}^p\end{bmatrix}^T \boldsymbol{\epsilon} = 0
\end{aligned} \tag{2.14}$$

Equation 2.14 gives us **property 2**: Each predictor $\boldsymbol{X}^j$ is purely uncorrelated with residual ($cov(\boldsymbol{X}^j, \boldsymbol{\epsilon}) = 0, j \in \mathbb{Z}^+$).

$$\begin{aligned} cov(X_i^p, \boldsymbol{\epsilon}) &= \frac{1}{n}\sum_{i=1}^{n}(X_i^p - \overline{X^p})(\epsilon_i - \overline{\epsilon}) \\ &= \frac{1}{n}(\sum_{i=1}^{n} X_i^p \epsilon_i - \overline{X^p}\sum_{i=1}^{n}\epsilon_i - \overline{\epsilon}\sum_{i=1}^{n} X_i^p + \overline{X^p}\overline{\epsilon}) \end{aligned}$$

Equation 2.13 shows that the $\sum_{i=1}^{n}\epsilon_i = 0$ and therefore $\overline{\epsilon} = 0$,

$$cov(X_i^p, \boldsymbol{\epsilon}) = \frac{1}{n}\sum_{i=1}^{n} X_i^p \epsilon_i = (\boldsymbol{X^p})^{\boldsymbol{T}}\boldsymbol{\epsilon} = 0 \tag{2.15}$$

which mean each predictor $\boldsymbol{X}^p$ is purely uncorrelated with the residuals. Predicted values of response variable $\hat{Y}$ is also purely uncorrelated with residuals:

$$\begin{aligned} cov(\hat{\boldsymbol{Y}}, \boldsymbol{\epsilon}) &= \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - \overline{\hat{Y}_i})(\epsilon_i - \overline{\epsilon}) \\ &= \frac{1}{n}(\sum_{i=1}^{n}\hat{Y}_i\epsilon_i - \overline{\hat{Y}_i}\sum_{i=1}^{n}\epsilon_i - \overline{\epsilon}\sum_{i=1}^{n}\hat{Y}_i + \overline{\hat{Y}_i}\overline{\epsilon}) \\ &= \frac{1}{n}\sum_{i=1}^{n}\hat{Y}_i\epsilon_i = \hat{\boldsymbol{Y}}^T\boldsymbol{\epsilon} \\ &= (\boldsymbol{X}\hat{\boldsymbol{\beta}})^T\boldsymbol{\epsilon} = \hat{\boldsymbol{\beta}}^T\boldsymbol{X}^T\boldsymbol{\epsilon} \end{aligned} \tag{2.16}$$

Equation 2.14 shows that $\boldsymbol{X}^T\boldsymbol{\epsilon} = 0$, so $cov(\hat{\boldsymbol{Y}}, \boldsymbol{\epsilon}) = 0$, which gives us **Property 3**: Predicted value $\hat{\boldsymbol{Y}}$ is purely uncorrelated with residuals.

Recall that Equation 2.13 shows that the $\sum_{i=1}^{n}\epsilon_i = 0$ and definition (Equation 2.8) shows $\boldsymbol{\epsilon} = \boldsymbol{Y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}$, calculate expectation value of two sides:

$$\begin{aligned} \boldsymbol{\epsilon} &= \overline{\boldsymbol{Y}} - \begin{bmatrix}\overline{\boldsymbol{X}}^0 & \overline{\boldsymbol{X}}^1 & \overline{\boldsymbol{X}}^2 & ... & \overline{\boldsymbol{X}}^p\end{bmatrix}\hat{\boldsymbol{\beta}} = 0 \\ \Rightarrow \overline{\boldsymbol{Y}} &= \begin{bmatrix}\mathbf{1} & \overline{\boldsymbol{X}}^1 & \overline{\boldsymbol{X}}^2 & ... & \overline{\boldsymbol{X}}^p\end{bmatrix}\hat{\boldsymbol{\beta}} \end{aligned} \tag{2.17}$$

which gives us **Property 4**: The trained model goes through the means of observed data, $(\overline{\boldsymbol{X}}, \overline{\boldsymbol{Y}})$, without error: $\overline{\boldsymbol{Y}} = \overline{\boldsymbol{X}}\hat{\boldsymbol{\beta}}$.

Most of linear regression tools are using Equation 2.12 to estimate model coefficients. Combining Equation 2.12 and Equation 2.1, we have

$$\begin{aligned} \hat{\boldsymbol{\beta}} &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y} \\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}) \\ &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} + (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{\varepsilon} \\ &= \boldsymbol{\beta} + (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{\varepsilon} \end{aligned}$$

which gives us **Property 5**: Estimated model coefficients $\hat{\boldsymbol{\beta}}$ is a linear function of assumption model coefficients $\boldsymbol{\beta}$:

$$\begin{aligned} \hat{\boldsymbol{\beta}} &= \boldsymbol{\beta} + B\boldsymbol{\varepsilon} \\ B &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T \end{aligned} \tag{2.18}$$

Since dataset $\boldsymbol{X}$ is determined, $E[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\varepsilon] = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T E[\boldsymbol{\varepsilon}]$, where $E[]$ is averaging operation, so

$$E[\hat{\boldsymbol{\beta}}] = E[\boldsymbol{\beta}] + (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T E[\boldsymbol{\varepsilon}] \tag{2.19}$$

By the assumption that stochastic term has Gaussian distribution with zero mean, we have

$$E[\boldsymbol{\varepsilon}] = 0 \Rightarrow E[\hat{\boldsymbol{\beta}}] = E[\boldsymbol{\beta}] \tag{2.20}$$

Property 6: Estimated model coefficients $\hat{\boldsymbol{\beta}}$ is an unbiased estimator of assumption model coefficients $\boldsymbol{\beta}$

2.1.4 Multicollinearity

Based on the proposed model (2.1) and assuming ε is independent and identically distributed (IID) and has Gaussian distribution with zero mean, we have shown the relationship between MLE and OLS in 2.1.1 and proved 6 useful properties of OLS in 2.1.3. In this section, a common problem in regression model will be discussed.

Under Gauss-Markov assumptions and the mentioned properties, chapter 3 of [28] showed that

$$Var[\hat{\beta}_j] = \frac{\sigma^2}{SST_j(1 - R_j^2)} \tag{2.21}$$

where $SST_j = \sum_n^{i=1}(X_i^j - \overline{X^j})^2 (j \in \mathbb{N})$ is the summation of squares of deviation of the j-th predictors, R_j^2 is the R-squared value of predictor X^j regressing on the other predictors, and $Var[]$ is variance operator. R-squared value will be discussed in Section 2.1.5.

In estimation, the large variance of estimated coefficient is unwanted. Equation 2.21 shows that $Var[\hat{\beta}_j]$ depends on σ^2, SST_j and $1 - R_j^2$, so we take these 3 factors into account when reducing $Var[\hat{\beta}_j]$.

σ^2 is the variance of distribution of stochastic term ε. Intuitively, large uncertainty (large σ^2) in our database means it is difficult to estimate the effect ($\hat{\beta}_j$) of predictors on the CPU load. σ^2 is a characteristic of our sampled data and a quality of our proposed model, so there is no way to decrease σ^2 under the linear model assumption.

Obviously, we can increase SST_j to decrease $Var[\hat{\beta}_j]$. Under the same population, larger number of samples will give rise to larger SST_j and therefore smaller $Var[\hat{\beta}_j]$. Technically, the quality of our estimation benefits from the large sample size. However, large database will cause more computing effort, especially when feature selection algorithms are applied where matrix operations are heavily used. Table 2.2 shows computational complexity of matrix operation [29] and

$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}$(Equation 2.12) is used to estimate coefficients. So the computational complexity of Equation 2.12 is $O(p^2(n+1)+p^3+pn)$, where p is the number of predictors and n is the number of samples.

Table 2.2: Computational complexity of matrix operations

operation	**complexity**
a $p \times n$ matrix multiply by a $n \times m$ matrix	*O(pnm)*
a $p \times p$ matrix inversion	*O(p^3)*

R_j^2 in Equation 2.21 measures quality of fit of predictor X^j regressing on the other predictors. By definition, R_j^2 is between 0 to 1, and the larger R_j^2 is, the more of the variance of X^j is explained by the other predictors. $R_j^2 = 1$ means X^j is a linear function of the rest of predictors, and leads to $Var[\hat{\beta}_j]$ being infinite, which is called multicollinearity problem and should be avoided. Equation 2.21 can be rewriten as

$$Var[\hat{\beta}_j] = \frac{\sigma^2}{SST_j} \cdot VIF_j \tag{2.22}$$

$$VIF_j = \frac{1}{1-R_j^2}$$

The degree of multicollinearity linearly increases the $Var[\hat{\beta}_j]$, and therefore we measure multicollinearity by variance inflation factor (VIF) in Equation 2.22. Because multicollinearity always exists, it is hard to define multicollinearity problem, but [28] considers multicollinearity as a severe problem when VIF is larger than 10. For our problem at hand, because we are interested in the effect of each predictor on the CPU load, and if VIF_j is high (>10), it will inflate the variance of $\hat{\beta}_j$ and other estimated coefficients to some extent, which means we cannot precisely estimate $\hat{\beta}_j$: $\hat{\beta}_j$ is sensitive to new data and changes a lot. In Section 4.2, we show high degree of multicollinearity exists in dataset. Two ways to deal with this problem: a) recursively excluding the predictors associated with the VIF that is larger than a critical value; b) using alternative methods: ridge regression estimators[9], PCR[24] or PLSR[25].

2.1.5 Model evaluation

In this section, some factors for evaluating trained model will be discussed. Feature selection methods are based on these factors as criteria. For the problem at hand, after performing PCA, there are 1400 potential predictors, and including all of them may introduce overfitting [10] to our model, so feature selection is for finding the "best model". In this section, we will discuss what "best" means for a model.

Pearson value

Pearson value (Pearson correlation coefficient) is developed by Carl Pearson [30] in 1880s to measure the direction and degree of linearity of two variables. It is also known as *linear correlation coefficient* or *quantity* r and the fundamental of the most criteria in this thesis. Pearson value of two variables, x and y, is defined as

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{2.23}$$

where $\bar{x}$ and $\bar{y}$ are means of variable x and y respectively. The higher $|r|$ is, the stronger linear relationship x and y have. r is always between -1 and 1, and $r < 0$ means x and y are negative related; $r > 0$ means x and y are positive related; $r = 0$ means x and y do not have linear relationship at all.

R-squared and adjusted R-squared value

R-squared (R^2) value is one of the most fundamental factor to evaluate the quality of fit of trained model. It is defined as the ratio of explained variance, regression sum of squares (RSS), to the total variance, total sum of squares (TSS).

$$R^2 = \frac{RSS}{TSS}$$
$$RSS = \sum_{i=1}^{n}(\hat{\mathbf{Y}}_i - \bar{\mathbf{Y}}_i)^2$$
$$TSS = \sum_{i=1}^{n}(\mathbf{Y}_i - \bar{\mathbf{Y}}_{\boldsymbol{i}})^2$$

Alternatively, since error sum of squares (ESS) is the difference between TSS and RSS,

$$ESS = \sum_{i=1}^{n}(\mathbf{Y}_i - \hat{\mathbf{Y}}_i)^2$$
$$R^2 = \frac{RSS}{TSS} = \frac{TSS - ESS}{TSS} = 1 - \frac{ESS}{TSS}$$

Mean squared error (MSE) is defined as the mean of ESS and for measuring the quality of fit

$$MSE = n^{-1}\sum_{i=1}^{n}(\mathbf{Y}_i - \hat{\mathbf{Y}}_i)^2 \tag{2.24}$$

The larger R^2 is, the better the model matches the observed data, but R^2 is misleading, because more the predictors are included, the smaller ESS is, so R^2 cannot be used as a criterion for evaluating the quality of subset predictors.

Adjusted R-squared value (R^2_{adj}) introduces penalty term to R^2.

$$R^2_{adj} = 1 - \frac{(1 - R^2)(n - 1)}{n - p - 1} \tag{2.25}$$

where n is the number of data points and p is the number of predictors. R^2_{adj} can be a criterion for feature selection: if the increment of quality of fit by introducing a new predictor cannot cover the penalty, the new predictor should be eliminated.

Mallows's Cp

Because ESS always decreases when adding more predictors, even when the new predictor is just noise, it is possible to perfectly match the observed data, R-squared value = 1 (RSS=0), when the number of predictors are large enough. Some predictors in model are "unuseful" and our model matches the noise of the training data, which is call overfitting [10]. it is necessary to introduce criteria to eliminate dummy predictors.

Recall our model defined in Equation 2.1, $\boldsymbol{Y} = \boldsymbol{X\beta} + \boldsymbol{\varepsilon}$. Ideally, our trained model should be able to predict $\boldsymbol{Y}$ without memorizing noise, which means the MSE when predicting $\boldsymbol{Y}$ should be as same as or close to the MSE when predicting $\boldsymbol{Y'}$, where $\boldsymbol{Y'} = \boldsymbol{X\beta} + \boldsymbol{\varepsilon}'$ and $\boldsymbol{\varepsilon}$ and $\boldsymbol{\varepsilon}'$ are IDD. $\boldsymbol{Y}$ is our in-sample data, $\boldsymbol{Y'}$ is imaginary data (out-of-sample data), and the only difference is stochastic term ($\boldsymbol{\varepsilon}$ and $\boldsymbol{\varepsilon'}$):

$$\boldsymbol{\varepsilon}_{n\times 1}, \boldsymbol{\varepsilon'}_{n\times 1} \sim \mathcal{N}(0,\, \sigma^2 \boldsymbol{I}_{n\times n})$$
$$\boldsymbol{Y}_{n\times 1}, \boldsymbol{Y'}_{n\times 1} \sim \mathcal{N}(\boldsymbol{X\beta},\, \sigma^2 \boldsymbol{I}_{n\times n})$$

so the adequate sub-set predictors can minimize the expected in-sample MSE and the difference between expected in-sample MSE ($E[MSE_{in}]$) and expected out-of-sample MSE ($E[MSE_{out}]$).

$$E[MSE_{in}] = E[n^{-1}\sum_{i=1}^{n}(Y_i - \boldsymbol{X_i}\hat{\boldsymbol{\beta}})^2]$$
$$E[MSE_{out}] = E[n^{-1}\sum_{i=1}^{n}(Y'_i - \boldsymbol{X_i}\hat{\boldsymbol{\beta}})^2]$$

where $(Y_i - \boldsymbol{X_i}\hat{\boldsymbol{\beta}})^2$ is squared error of one single point and its expected value is

$$\begin{aligned} E[(Y_i - \boldsymbol{X_i}\hat{\boldsymbol{\beta}})^2] &= Var[Y_i - \boldsymbol{X_i}\hat{\boldsymbol{\beta}}] + E[Y_i - \boldsymbol{X_i}\hat{\boldsymbol{\beta}}]^2 \\ &= Var[Y_i] + Var[\boldsymbol{X_i}\hat{\boldsymbol{\beta}}] - 2cov(Y_i, \boldsymbol{X_i}\hat{\boldsymbol{\beta}}) + (E[Y_i] - E[\boldsymbol{X_i}\hat{\boldsymbol{\beta}}])^2 \\ E[(Y'_i - \boldsymbol{X_i}\hat{\boldsymbol{\beta}})^2] &= Var[Y'_i - \boldsymbol{X_i}\hat{\boldsymbol{\beta}}] + E[Y'_i - \boldsymbol{X_i}\hat{\boldsymbol{\beta}}]^2 \\ &= Var[Y'_i] + Var[\boldsymbol{X_i}\hat{\boldsymbol{\beta}}] - 2cov(Y'_i, \boldsymbol{X_i}\hat{\boldsymbol{\beta}}) + (E[Y'_i] - E[\boldsymbol{X_i}\hat{\boldsymbol{\beta}}])^2 \end{aligned}$$

Because $\boldsymbol{Y}$ and $\boldsymbol{Y'}$ ard IDD, so $cov(Y'_i, \boldsymbol{X_i}\hat{\boldsymbol{\beta}}) = 0$ and we have

$$\begin{aligned} E[(Y'_i - \boldsymbol{X_i}\hat{\boldsymbol{\beta}})^2] &= Var[Y_i] + Var[\boldsymbol{X_i}\hat{\boldsymbol{\beta}}] + (E[Y_i] - E[\boldsymbol{X_i}\hat{\boldsymbol{\beta}}])^2 \\ &= E[(Y_i - \boldsymbol{X_i}\hat{\boldsymbol{\beta}})^2] + 2cov(Y_i, \boldsymbol{X_i}\hat{\boldsymbol{\beta}}) \end{aligned}$$
$$\Rightarrow E[n^{-1}\sum_{i=1}^{n}(Y'_i - \boldsymbol{X_i}\hat{\boldsymbol{\beta}})^2] = E[n^{-1}\sum_{i=1}^{n}(Y_i - \boldsymbol{X_i}\hat{\boldsymbol{\beta}})^2] + 2n^{-1}\sum_{i=1}^{n}cov(Y_i, \boldsymbol{X_i}\hat{\boldsymbol{\beta}}) \quad (2.26)$$

Because $\hat{\boldsymbol{Y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}}$ and $\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}$, so $\hat{\boldsymbol{Y}} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y} = \boldsymbol{HY}$, where $\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$

$$cov(\boldsymbol{Y}, \boldsymbol{X}\hat{\boldsymbol{\beta}}) = cov(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = cov(\boldsymbol{Y}, \boldsymbol{HY}) = \boldsymbol{H}cov(\boldsymbol{Y}, \boldsymbol{Y}) = \boldsymbol{H}Var[\boldsymbol{Y}]$$
$$\Rightarrow cov(\boldsymbol{Y_i}, \boldsymbol{X_i}\hat{\boldsymbol{\beta}}) = \boldsymbol{H_{ii}}Var[\boldsymbol{Y}] \quad (2.27)$$

Combine Equation 2.26 and 2.27:

$$\begin{aligned} E[n^{-1}\sum_{i=1}^{n}(Y_i' - \boldsymbol{X_i}\hat{\boldsymbol{\beta}})^2] &= E[n^{-1}\sum_{i=1}^{n}(Y_i - \boldsymbol{X_i}\hat{\boldsymbol{\beta}})^2] + 2n^{-1}\sum_{i=1}^{n}\boldsymbol{H_{ii}}Var[\boldsymbol{Y}] \\ &= E[n^{-1}\sum_{i=1}^{n}(Y_i - \boldsymbol{X_i}\hat{\boldsymbol{\beta}})^2] + 2n^{-1}\sum_{i=1}^{n}\boldsymbol{trH_{ii}}Var[\boldsymbol{Y}] \\ &= E[n^{-1}\sum_{i=1}^{n}(Y_i - \boldsymbol{X_i}\hat{\boldsymbol{\beta}})^2] + 2n^{-1}\sum_{i=1}^{n}(p+1)\boldsymbol{\sigma^2} \end{aligned} \tag{2.28}$$

Ideally, adequate subset of predictors should not only has low MSE of in-sample date, but also minimize the difference between $E[MSE_{in}]$ and $E[MSE_{out}]$, so Mallows's *Cp* is defined as

$$\begin{aligned} Cp &= E[n^{-1}\sum_{i=1}^{n}(Y_i\boldsymbol{X_i}\hat{\boldsymbol{\beta}})^2] + 2n^{-1}\sum_{i=1}^{n}(p+1)\hat{\boldsymbol{\sigma}}^{\boldsymbol{2}} \\ &= MSE + 2n^{-1}\hat{\boldsymbol{\sigma}}^{\boldsymbol{2}}(p+1) \end{aligned}$$

where MSE is mean squared error of current training data, n is the number of data point, p is the number of predictors and $\hat{\boldsymbol{\sigma}}^{\boldsymbol{2}}$ is the estimated variance of stochastic term $\boldsymbol{\varepsilon}$ using full model (with all predictors).

$$\boldsymbol{\sigma} \approx \hat{\boldsymbol{\sigma}} = \frac{RSS_{full}}{n - P - 1} \tag{2.29}$$

where RSS_{full} is residual sum of squares of full model, n is the number of data point, P is the total number of predictors. The corresponding derivation can be found in Chapter 3 of [9].

The best adequate subset model can minimize *Cp*.

Leave-one-out Cross-Validation (LOOCV)

Ideally, the trained model can predict new data. In order to test the ability to predict new data, we leave one data point out of the dataset to train model and use trained model to predict the missing data point. The sum of error of prediction is referred to a new criterion: LOOCV, denoted by Λ.

$$\Lambda = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \lambda_i)^2 \tag{2.30}$$

where n is the number of data points, Y_i is the true value of missing data and λ_i is the predicted value of missing data. Based on LOOCV, we want to find the subset model that minimize the Λ. Formula 2.30 requires to train model n times to calculate Λ once, which is extremely time-consuming. Section 5.5 of [31] shows a fast way to calculate Λ:

$$\Lambda = \frac{1}{n}\sum_{i=1}^{n}(\frac{e_i}{1 - H_{ii}})^2 \tag{2.31}$$

where e_i is the residual when predicting the i-th data point using the model trained by all data points. H_{ii} is the i-th diagonal value of $\boldsymbol{H}$, where $\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$.

k-Fold Cross-Validation

Similar to LOOCV, this method equally divide total dataset into k folds, where *k-1* folds are training data and 1 fold is test data. LOOCV is a special case of k-Fold Cross-Validation when k is equal to n. Each fold of test data has a MSE and this method goes though k-fold test data to calculate averaged MSE as a select criterion. The goal is selecting adequate subset model that minimize the averaged MSE.

Akaike information criterion (AIC):
Akaike information criterion (AIC) is proposed by Hirotugu Akaike in 1973 [32]. AIC is a criterion for balancing complexity and quality of fit of a model. AIC is based on information theory and derivation is complicated. AIC is defined as

$$AIC = n \log MSE + 2p$$

where n is the number of training data points, MSE is mean squared error and p is the number of predictors. This equation has the same form as the form of Mallows's *Cp*: "evaluation + penalty", adding a new predictor to the model increases the quality of fit and penalty at the same time. The goal of feature selection is finding the subset model that minimize AIC value.

Bayesian information criterion (BIC)

Another similar criterion is called Bayesian information criterion (BIC) proposed by [33]. BIC is defined as

$$BIC = n \log MSE + p \log n.$$

BIC has the same form as the form of AIC and the model selected by BIC criterion tends to minimize BIC. Because BIC has stricter penalty term than AIC ($\log n > 2$), the number of predictors selected by BIC is less than the number of predictors selected by AIC.

2.2 Principal component regression (PCR)

Principal component regression (PCR) is based on Principal component analysis (PCA). PCR regresses response variable on the principal components (PCs) generated by PCA rather than the original data. Because our original database has high degree of multicollinearity, so directly training the model by original data will result in large variance of estimated $\boldsymbol{\beta}$ and therefore we cannot precisely estimate model coefficients (the reason is shown in Section 2.1.4). We can eliminate the predictors with high VIF to overcome high-degree multicollinearity problem, but this does not fulfill the **Goal 1**. By using PCR, the solution is

1. Applying PCA to original data to get the same number of PCs as the number of predictors in original data
2. perform feature selection on these PCs to get an adequate subset model
3. train selected model to estimate model coefficients
4. map the estimated coefficients associated with the selected PCs back to the original data

In this section, we are going to explain the theory behind PCA.

Intuitively, PCA maps the original data to the new coordinate that maximizes the variance of projection of the data and uses the value of projection instead of the original data as independent variables in the regression model. Figure 2.1 is a simple example of PCA, which shows the original data (white circle) and its projection (blue dot) on its principal direction (green line) in the case of two-dimensional data. In practice, the new coordinates are derived by calculating the eigenvector corresponding to the biggest eigenvalue of original data matrix.

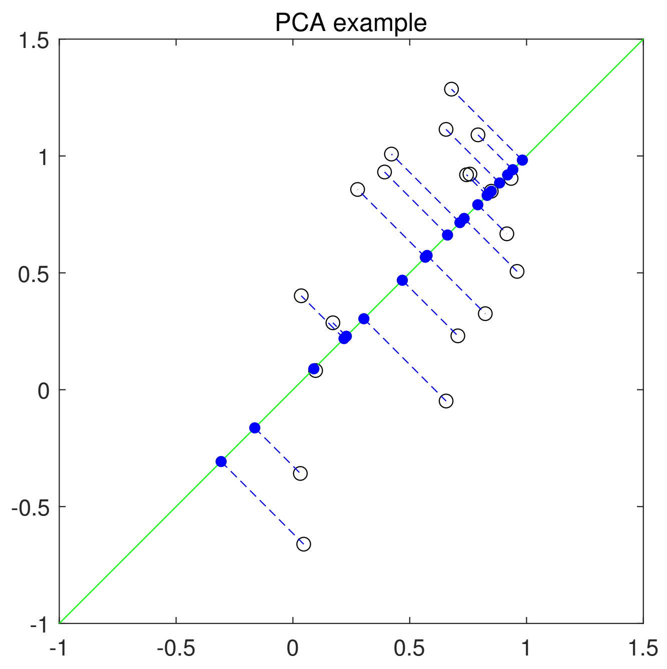

Figure 2.1: This a PCA example in the case of 2-dimension data. The white circles are the original data points, the blue dots are the projections of original data to the green line, and the green line represents the direction of the primary principal component, in which the lengths between point $(0, 0)$ and blue dots have the largest variance.

Mathematically, like Equation 2.1, define our standardized original dataset as $\boldsymbol{D_{n\times p}}$:

$$\boldsymbol{D_{n\times p}} = \begin{bmatrix} X_1^1 & X_1^2 & ... & X_1^p \\ X_2^1 & X_2^2 & ... & X_2^p \\ X_3^1 & X_3^2 & ... & X_3^p \\ ... & ... & ... & ... \\ X_n^1 & X_n^2 & ... & X_n^p \end{bmatrix} \tag{2.32}$$

where n is the number of observed data points, p is the number predictors and X_n^p represents the n-th observed value of p-th predictors with zero mean. The projection of $\boldsymbol{D}$ to a direction $\boldsymbol{d}_{p\times 1}$ is

$$\boldsymbol{P}_{n\times 1} = \boldsymbol{D}_{n\times p} \cdot \boldsymbol{d}_{p\times 1} \tag{2.33}$$

where $\boldsymbol{D}_{n\times p}$ represents original data, P_n is the n-th element in $\boldsymbol{P}_{n\times 1}$, representing the length of projection of n-th data point $\boldsymbol{D_n}$ on direction $\boldsymbol{d}_{p\times 1}$, and $\boldsymbol{d}_{p\times 1}$ is the a direction of a p-dimension space with $\|d\|^2 = 1$. Because of $E[D] = \boldsymbol{0}_{p\times 1}$, $\boldsymbol{P}$ has zero mean. The first principal component is defined as the direction that maximizes the variance of corresponding projection, and we can find such direction by solving following optimization problem.

$$\begin{aligned} \boldsymbol{d}_{prime} &= \underset{\boldsymbol{d}}{\operatorname{argmax}} Var[\boldsymbol{P}] = \underset{\boldsymbol{d}}{\operatorname{argmax}}\ n^{-1}\sum_{i=1}^{n} P_i^2 \\ &= \underset{\boldsymbol{d}}{\operatorname{argmax}}\ n^{-1}(\boldsymbol{P}^T \cdot \boldsymbol{P}) = \underset{\boldsymbol{d}}{\operatorname{argmax}}\ \boldsymbol{d}^T \frac{\boldsymbol{D}^T\boldsymbol{D}}{n}\boldsymbol{d} \end{aligned} \tag{2.34}$$

Because each column in $\boldsymbol{D}_{n\times p}$ has zero mean, so $\frac{\boldsymbol{D}_{n\times p}^T \cdot \boldsymbol{D}_{n\times p}}{n}$ is variance-covariance matrix of $\boldsymbol{D}_{n\times p}$ (one benefit of centering the data to zero mean, discussed in Section 3.1.2). Let σ denotes variance-covariance matrix of $\boldsymbol{D}_{n\times p}$, and we have: $Var[\boldsymbol{P}] = \boldsymbol{d}^T\sigma\boldsymbol{d}$, which means variance of projections are the eigenvalues of variance-covariance matrix of $\boldsymbol{D}_{n\times p}$. So projection matrix with top **k** largest variance, $\boldsymbol{P}_{n\times k}$ is a linear transform of original data, $\boldsymbol{D}_{n\times p}$, and k eigenvectors with top k largest eigenvalues as k directions, $\boldsymbol{d}_{p\times k}$:

$$\boldsymbol{P}_{n\times k} = \boldsymbol{D}_{n\times p} \cdot \boldsymbol{d}_{p\times k}$$

We can get the largest variance by calculating the eigenvector with the largest eigenvalue. n-dimension database will give us n PC and direction pairs, and these PCs are orthogonal to each other, so we can apply feature selection algorithms on them without worrying about multicollinearity problem.

After selecting subset of PCs and training the selected subset model, we can map the coefficients back to the original data to get the full coefficients. Let $\boldsymbol{PC_k}$ denote selected k PCs, $\hat{\boldsymbol{\beta}}_{\boldsymbol{PC}}$ denote corresponding estimated coefficients, and $\boldsymbol{d_k}$ denote corresponding directions:

$$\begin{aligned} &\hat{\boldsymbol{Y}} = \boldsymbol{PC_k} \cdot \hat{\boldsymbol{\beta}}_{\boldsymbol{PC}} \\ \xrightarrow{\boldsymbol{PC_k}=\boldsymbol{D}_{n\times p}\cdot\boldsymbol{d}_{p\times k}} &\hat{\boldsymbol{Y}} = \boldsymbol{D}_{n\times p} \cdot \underbrace{\boldsymbol{d}_{p\times k} \cdot \hat{\boldsymbol{\beta}}_{\boldsymbol{PC}}}_{\hat{\boldsymbol{\beta}}\text{ (full coefficients)}} \end{aligned} \tag{2.35}$$

Equation 2.35 has the same form as Equation 2.7 and $\hat{\boldsymbol{\beta}} = \boldsymbol{d}_{p\times k} \cdot \hat{\boldsymbol{\beta}}_{\boldsymbol{PC}}$ is the estimated coefficients of full model.

2.3 Partial least squares regression (PLSR)

Partial least squares regression (PLSR) builds a linear regression model by projecting predictors and response variables to a new direction. PLSR can be traced back to Herman Ole Andreas Wold in 1966 and is introduced for removing multicollinearity problem at the beginning [34].

Recall Equation 2.34, the first principal component in the perspective of PCA is defined as the projection to the direction that maximizes the variance of projections of original data to a sub direction. However the most principal component in the perspective of PLSR is defined as the projection to the direction that maximizes the absolute value of covariance of projections of original data and the response variable. Because "principal component" has different definitions for PLSR and PCR, PLSR_PC refers to principal component of PLSR and PCA_PC refers to principal component of PCA. Let $\boldsymbol{P}$ represents a projection of the original data. The covariance of $\boldsymbol{P}$ and response variable $\boldsymbol{Y}$ can be calculated as:

$$\begin{aligned} cov(\boldsymbol{P}, \boldsymbol{Y}) &= n^{-1} \sum_{i=1}^{n} (P_i - \overline{\boldsymbol{P}})(Y_i - \overline{\boldsymbol{Y}}) \\ \xrightarrow{\overline{\boldsymbol{Y}}=0, \overline{\boldsymbol{P}}=0} cov(\boldsymbol{P}, \boldsymbol{Y}) &= n^{-1} \sum_{i=1}^{n} P_i Y_i = n^{-1} \boldsymbol{P}^T \cdot \boldsymbol{Y} \end{aligned}$$

Because $cov(\boldsymbol{P}, \boldsymbol{Y})$ can be positive and negative, the prime direction is intended to maximize the $|cov(\boldsymbol{P}, \boldsymbol{Y})|$ or $cov^2(\boldsymbol{P}, \boldsymbol{Y})$.

$$cov^2(\boldsymbol{P}, \boldsymbol{Y}) = n^{-2} \boldsymbol{P}^T \boldsymbol{Y} \boldsymbol{Y}^T \boldsymbol{P} \tag{2.36}$$

Combining with Equation 2.33, Equation 2.36 can be rewritten as:

$$cov^2(\boldsymbol{P}, \boldsymbol{Y}) = n^{-2} \boldsymbol{d}^T \boldsymbol{D}^T \boldsymbol{Y} \boldsymbol{Y}^T \boldsymbol{D} \boldsymbol{d} \tag{2.37}$$

Therefore $cov^2(\boldsymbol{P}, \boldsymbol{Y})$ is maximized when $\boldsymbol{d}$ is the eigenvector corresponding to the largest eigenvalue of matrix $\boldsymbol{D}^T \boldsymbol{Y} \boldsymbol{Y}^T \boldsymbol{D}$, where $\boldsymbol{D}$ is defined in 2.32 and $\boldsymbol{Y}$ is the samples of CPU load in matrix form. Let $\boldsymbol{d}_1$ denotes the most principal direction, $\boldsymbol{d}_2$ denotes the second most principal direction, and $\boldsymbol{d}_n$ denotes the n-th most principal direction, $\boldsymbol{d}_n$ is the eigenvector corresponding to the n-th largest eigenvalue of matrix $\boldsymbol{D}^T \boldsymbol{Y} \boldsymbol{Y}^T \boldsymbol{D}$.

When $\boldsymbol{d}_1$ is available, the projection, $\boldsymbol{P}_1$, on $\boldsymbol{d}_1$ can be calculated using Equation 2.33:

$$\begin{aligned} \boldsymbol{P}_1 &= \boldsymbol{D} \cdot \boldsymbol{d}_1 \\ \boldsymbol{P}_n &= \boldsymbol{D} \cdot \boldsymbol{d}_n \end{aligned} \tag{2.38}$$

Then we regress the response variable $\boldsymbol{Y}$ on $\boldsymbol{P}_1$ and get a coefficient denoted by b_1 by using the OLS method that discussed in Section 2.1.2:

$$\boldsymbol{Y} = b_1\boldsymbol{P}_1 + \boldsymbol{E}_1$$

where $\boldsymbol{E}_1$ is the error vector that will be minimized by adding more projections: $\boldsymbol{P}_2$, $\boldsymbol{P}_3$, ..., $\boldsymbol{P}_n$. Because $\boldsymbol{P}_1$ is just the projection with one dimension and can not represent the original data with p dimensions, in order to derive the second most prime project $\boldsymbol{P}_2$, the first step is to calculate the residual matrix in original $\boldsymbol{D}$ that cannot be linearly represented by $\boldsymbol{P}_1$:

$$\boldsymbol{D} = \boldsymbol{P}_1\boldsymbol{u}_1 + \boldsymbol{R}_1 \tag{2.39}$$

where $\boldsymbol{R}_1$ is the residual matrix. Equation 2.39 is the regression problem with multiply response variables and single predictor, which is in the scope of multivariate multiply regression and not discussed in the Theory chapter, but it is well discussed in [35]. $\boldsymbol{u}_1$ is

$$\boldsymbol{u}_1 = (\boldsymbol{P}_1^T\boldsymbol{P}_1)^{-1}\boldsymbol{P}_1^T\boldsymbol{D} \tag{2.40}$$

Then replace $\boldsymbol{D}$ in Equation 2.37 and Equation 2.38 by $\boldsymbol{R}_1$ to get $\boldsymbol{d}_2$ and $\boldsymbol{P}_2$:

$$\boldsymbol{P}_2 = \boldsymbol{R}_1 \cdot \boldsymbol{d}_2 \tag{2.41}$$

Regress $\boldsymbol{E}_1$ on $\boldsymbol{P}_2$ to compensate the error $\boldsymbol{E}_1$ and get coefficient b_2 associated with $\boldsymbol{P}_2$:

$$\boldsymbol{E}_1 = b_2\boldsymbol{P}_2 + \boldsymbol{E}_2$$

Replace $\boldsymbol{D}$ and $\boldsymbol{P}_1$ in Equation 2.39 by $\boldsymbol{R}_1$ and $\boldsymbol{P}_2$ respectively to get a new residual matrix $\boldsymbol{R}_2$:

$$\boldsymbol{R}_1 = \boldsymbol{P}_2\boldsymbol{u}_2 + \boldsymbol{R}_2 \tag{2.42}$$

Repeat above process recursively to get b_3, b_4, ... , b_n; $\boldsymbol{E}_3$, $\boldsymbol{E}_4$, ..., $\boldsymbol{E}_n$ and $\boldsymbol{P}_3$, $\boldsymbol{P}_4$, ..., $\boldsymbol{P}_n$. In total, original data $\boldsymbol{D}$ and $\boldsymbol{Y}$ can be expressed as:

$$\begin{aligned}\boldsymbol{D} &= \boldsymbol{u}_1\boldsymbol{P}_1 + \boldsymbol{u}_2\boldsymbol{P}_2 + ... + \boldsymbol{u}_p\boldsymbol{P}_p + \boldsymbol{R}_p\\ \boldsymbol{Y} &= b_1\boldsymbol{P}_1 + b_2\boldsymbol{P}_2 + ... + b_p\boldsymbol{P}_p + \boldsymbol{E}_p\\ &= \boldsymbol{P}\cdot\boldsymbol{B}(\text{ in matrix form})\end{aligned} \tag{2.43}$$

where $\boldsymbol{B} = \left[b_1, b_2, ..., b_p\right]^T$ and $\boldsymbol{P} = \left[\boldsymbol{P}_1, \boldsymbol{P}_2, ..., \boldsymbol{P}_p\right] = \boldsymbol{D}\cdot\left[\boldsymbol{d}_1, \boldsymbol{d}_2, ..., \boldsymbol{d}_p\right]$. Equation 2.43 can be rewritten as:

$$\begin{aligned}\boldsymbol{Y} &= \boldsymbol{D}\cdot\left[\boldsymbol{d}_1, \boldsymbol{d}_2, ..., \boldsymbol{d}_p\right]\cdot\boldsymbol{B}\\ &= \boldsymbol{D\beta}\end{aligned}$$

Because $\boldsymbol{P}_n$ has weaker covariance with $\boldsymbol{Y}$ than $\boldsymbol{P}_{n-1}$ does, the quality of fit increases slower and slower when more PLSR_PCs are included, which agrees with the result in Section 4.5. So in practice, it is feasible to truncate the tail PLSR_PCs with losing small quality of fit.

As mentioned, PLSR can overcome the multicollinearity problem. In the following, the relationship between the k-th PLSR_PC ($\boldsymbol{P}_k$) and the $k+m$-th PLSR_PC ($\boldsymbol{P}_{k+m}$) will be derived. Combing Equation 2.42 and 2.41, $\boldsymbol{P}_k$ can be written as a function of $\boldsymbol{P}_{k-1}$, where $k \geq 2$:

$$\begin{aligned}\boldsymbol{P}_k &= \boldsymbol{R}_{k-1} \cdot \boldsymbol{d}_k \\ \xrightarrow{\boldsymbol{R}_{k-1}=\boldsymbol{R}_{k-2}-\boldsymbol{P}_{k-1}\boldsymbol{u}_{k-1}} &= (\boldsymbol{R}_{k-2} - \boldsymbol{P}_{k-1}\boldsymbol{u}_{k-1}) \cdot \boldsymbol{d}_k \\ \xrightarrow{\boldsymbol{R}_{k-2}=\boldsymbol{P}_{k-1}\cdot\boldsymbol{d}_{k-1}^{-1}} &= (\boldsymbol{P}_{k-1} \cdot \boldsymbol{d}_{k-1}^{-1} - \boldsymbol{P}_{k-1}\boldsymbol{u}_{k-1}) \cdot \boldsymbol{d}_k \\ &= \boldsymbol{P}_{k-1} \cdot (\boldsymbol{d}_{k-1}^{-1} - \boldsymbol{u}_{k-1}) \cdot \boldsymbol{d}_k\end{aligned}$$

So the dot product of $\boldsymbol{P}_k^T$ and $\boldsymbol{P}_{k+m}$ can be recursively written as a function of $\boldsymbol{P}_k^T\boldsymbol{P}_{k+1}$, where $k \geq 2$ and $m \geq 1$:

$$\begin{aligned}\boldsymbol{P}_k^T\boldsymbol{P}_{k+m} &= \boldsymbol{P}_k^T\boldsymbol{P}_{k+m-1}(\boldsymbol{d}_{k+m-1}^{-1} - \boldsymbol{u}_{k+m-1})\boldsymbol{d}_{k+m} \\ &= \boldsymbol{P}_k^T\boldsymbol{P}_{k+m-2}(\boldsymbol{d}_{k+m-2}^{-1} - \boldsymbol{u}_{k+m-2})\boldsymbol{d}_{k+m-1}(\boldsymbol{d}_{k+m-1}^{-1} - \boldsymbol{u}_{k+m-1})\boldsymbol{d}_{k+m} \\ \cdots\cdots\cdots \\ &= \boldsymbol{P}_k^T\boldsymbol{P}_{k+1}\prod_{i=1}^{m-1}(\boldsymbol{d}_{k+m-i}^{-1} - \boldsymbol{u}_{k+m-i})\boldsymbol{d}_{k+m-i+1} \qquad (2.44)\end{aligned}$$

The product of $\boldsymbol{P}_k^T$ and $\boldsymbol{P}_{k+1}$ is:

$$\begin{aligned}\boldsymbol{P}_k^T\boldsymbol{P}_{k+1} &= \boldsymbol{P}_k^T(\boldsymbol{R}_{k-1} - \boldsymbol{P}_k\boldsymbol{u}_k) \\ \xrightarrow{\boldsymbol{u}_k=(\boldsymbol{P}_k^T\boldsymbol{P}_k)^{-1}\boldsymbol{P}_k^T\boldsymbol{R}_{k-1}} &= \boldsymbol{P}_k^T\boldsymbol{R}_{k-1} - \boldsymbol{P}_k^T\boldsymbol{P}_k(\boldsymbol{P}_k^T\boldsymbol{P}_k)^{-1}\boldsymbol{P}_k^T\boldsymbol{R}_{k-1} \\ &= \boldsymbol{P}_k^T\boldsymbol{R}_{k-1} - \boldsymbol{P}_k^T\boldsymbol{R}_{k-1} = 0 \qquad (2.45)\end{aligned}$$

Equation 2.45 means that any two contiguous PLSR_PCs ($\boldsymbol{P}_k$ and $\boldsymbol{P}_{k+1}$) are orthogonal. Combining with Equation 2.44, the dot product of $\boldsymbol{P}_k^T$ and $\boldsymbol{P}_{k+m}$ is zero for $k \geq 2$ and $m \geq 1$, which means all the PLSR_PCs are orthogonal to each other. So the multicollinearity does not exist in Equation 2.43.

2.4 Ridge regression

As mentioned in Section 2.1.4, when the degree of multicollinearity is high in the dataset, the variance of estimated coefficients increases dramatically. The error of estimates can be decomposed to the error caused by bias and the error caused by variance. For OLS, although the estimated coefficients, $\hat{\boldsymbol{\beta}}$, is unbiased estimates of model coefficients (**Property 6**), $\boldsymbol{\beta}$, the large variance of $\hat{\boldsymbol{\beta}}$ caused by high degree of multicollinearity still results in large deviation away from the true values. Figure 2.2 is an intuitive example of relationship between error and variance-bias.

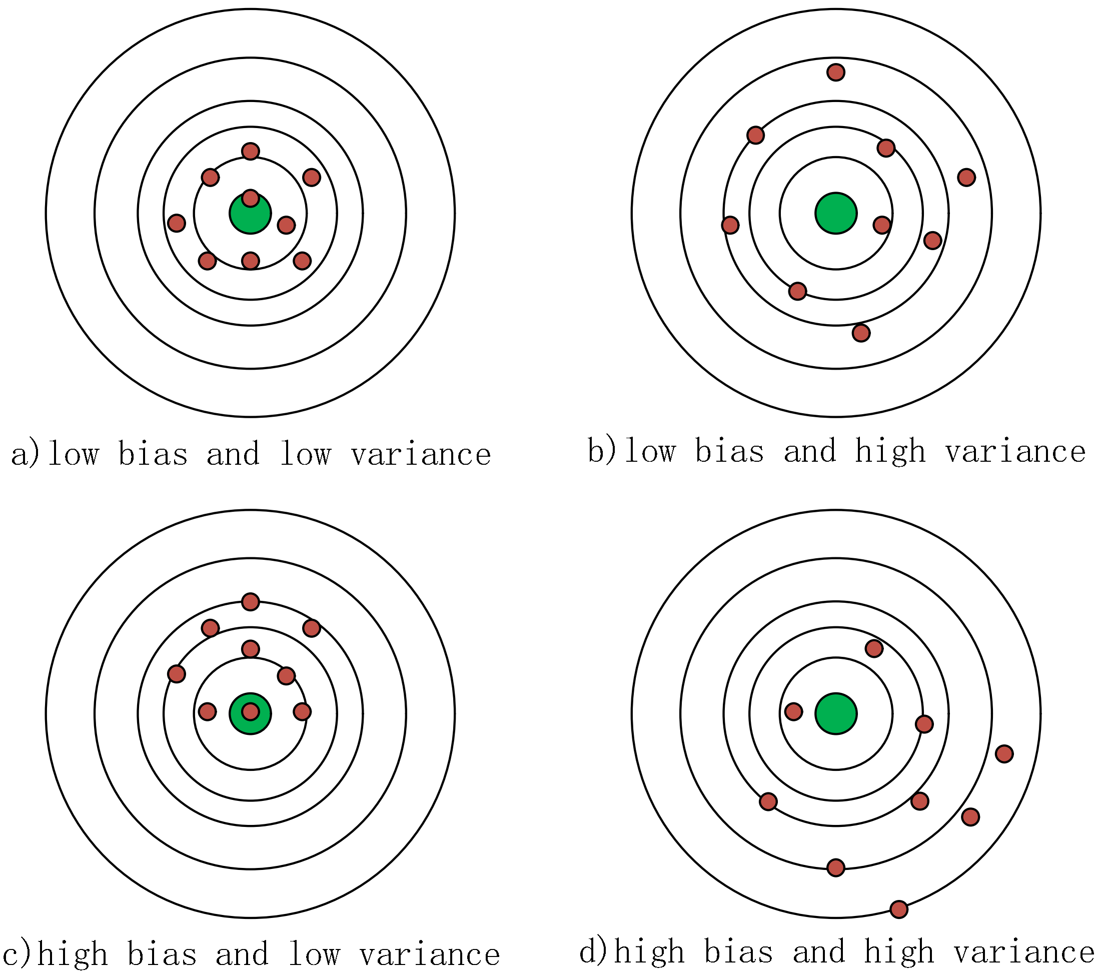
a) low bias and low variance
b) low bias and high variance
c) high bias and low variance
d) high bias and high variance

Figure 2.2: The green dots in the figures represent the true value we try to estimate. The red dots represent the distribution of the estimates. Figure a) has the lowest error than the errors in the other 3 figures, because it has low bias and low variance. Figure b) and c) have higher error than the error in Figure a), because either bias or variance increases. Figure d) has the highest error than the errors in the other 3 figures, because it has large bias and variance.

Ridge regression overcomes the multicollinearity problem by introducing penalty term to the loss function (Equation 2.9) used in OLS [36]:

$$L_{ridge} = \underbrace{\sum_{i=1}^{n}(Y_i - X_i\hat{\boldsymbol{\beta}})^2}_{\text{error term}} + \underbrace{\lambda \left\|\hat{\boldsymbol{\beta}}\right\|^2}_{\text{penalty term}} \tag{2.46}$$

where the error term is exactly the same as the OLS loss function (Equation 2.9), λ in penalty term controls the degree of the penalty. By tuning the λ, the variance and bias is balanced, and therefore the error is improved. Similar to the derivation of Equation 2.12, the coefficients can be derived by minimizing the loss function

L_{ridge}:

$$
\begin{aligned}
\hat{\boldsymbol{\beta}}^{ridge} &= \underset{\hat{\beta}}{\operatorname{argmin}} L_{ridge} \\
&= \underset{\hat{\beta}}{\operatorname{argmin}} \sum_{i=1}^{n} (Y_i - X_i\hat{\boldsymbol{\beta}})^2 + \lambda \left\|\hat{\boldsymbol{\beta}}\right\|^2 \\
&\text{subject to } \lambda \geq 0
\end{aligned}
\tag{2.47}
$$

This minimization problem can be solved by setting the partial derivative of L_{ridge} with respect to $\hat{\boldsymbol{\beta}}$ to zero:

$$
\begin{aligned}
&\frac{\partial L}{\partial \hat{\boldsymbol{\beta}}^{ridge}} 2\boldsymbol{X^T X}\hat{\boldsymbol{\beta}}^{ridge} - 2\boldsymbol{X^T Y} + 2\lambda\hat{\boldsymbol{\beta}}^{ridge} = 0 \\
\Rightarrow &(\boldsymbol{X^T X} + \lambda \boldsymbol{I})\hat{\boldsymbol{\beta}}^{ridge} == \boldsymbol{X}^T\boldsymbol{Y} \\
\Rightarrow &\hat{\boldsymbol{\beta}}^{ridge} == (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{Y}
\end{aligned}
\tag{2.48}
$$

Skip tedious mathematical derivation [37], the expectation ($E[\hat{\boldsymbol{\beta}}^{ridge}]$) and variance ($Var[\hat{\boldsymbol{\beta}}^{ridge}]$) of $\hat{\boldsymbol{\beta}}^{ridge}$ are:

$$E[\hat{\boldsymbol{\beta}}^{ridge}] = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} \tag{2.49}$$

$$Var[\hat{\boldsymbol{\beta}}^{ridge}] = \sigma^2[\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}]^{-1}\boldsymbol{X}^T\boldsymbol{X}\{[\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}]^{-1}\}^T \tag{2.50}$$

Obviously, for $\lambda > 0$, $E[\hat{\boldsymbol{\beta}}^{ridge}]$ is not equal to $\boldsymbol{\beta}$, so the ridge regression is unbiased estimate. Furthermore, when the λ increases, $E[\hat{\boldsymbol{\beta}}^{ridge}]$ deviates further away from $\hat{\boldsymbol{\beta}}^{ridge}$, and $Var[\hat{\boldsymbol{\beta}}^{ridge}]$ decreases. Hence, tuning the λ can balance the bias-variance trade-off to minimize the error. Figure 2.3 intuitively illustrates how λ balances the bias-variance trade-off and decreases the error.

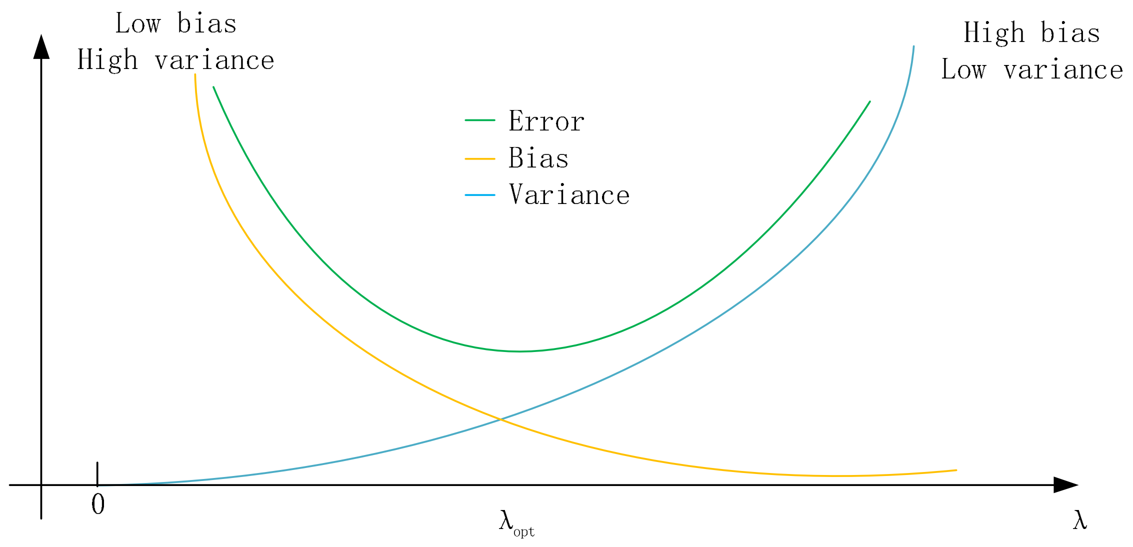

Figure 2.3: When λ is zero, the bias is zero, which is the case of OLS estimation. Equation 2.49 and 2.50 shows that when λ increases, bias increases and variance decreases. Hence, the λ_{opt} that minimizes the error is aimed.

K-fold cross validation mentioned in Section 2.1.5 is widely used for choosing λ [37], which chooses the λ that minimizes the overall MSE.

But because 1) the contribution of each signal to the CPU load are required in this project; 2) the training data set is huge - a 237912×14000 matrix, instead of original data, principal components is processed by ridge regression method.

3
Methods

Section 4.2 shows high degree of multicollinearity exists in our data, and precise estimates of the effect of each signal on CPU load (interpretability) is key, not only predictive power, eliminating signal with high VIF can solve multicollinearity problem, but we are not be able to estimate the effect of each signal on CPU. In this chapter we present methods that overcome multicollinearity problem without losing interpretability.

3.1 Data pre-processing

3.1.1 Data construction

The original sampled at time t is consist of 1400 signals (denoted as $\boldsymbol{X}^t$) and the CPU load signal (denoted as $\boldsymbol{Y}^t$). As mentioned in Section 1.2.3, $\boldsymbol{X}^t$ is not causal to $\boldsymbol{Y}^t$, but $\boldsymbol{X}^{t-1}$, $\boldsymbol{X}^{t-2}$, $\boldsymbol{X}^{t-3}$...$\boldsymbol{X}^{t-10}$ are causal to $\boldsymbol{Y(t)}$, so it is necessary to construct new data before training the model. Because the length of effect window is assumed to be 250 ms, the current CPU load depends on the previous 10 periods samples. Table 3.1 illustrates the structure of the sampled original data. Table 3.2 illustrates the structure of the newly constructed data. Each $\boldsymbol{X}^t$ has 1400 signals/predictors, and the past 10 samples are included into the model, so the size of the reconstructed data is 10 times increased: 475826×14000. In order to verify the trained model, the data is randomly divided into training set (50%) and test set (50%): each sample has 50% probability of being test data and 50% probability of being training data.

Table 3.1: Sampled original data

sampled signals	CPU load
$\boldsymbol{X}^0$	$\boldsymbol{Y}^0$
$\boldsymbol{X}^1$	$\boldsymbol{Y}^1$
...	...
$\boldsymbol{X}^{n-1}$	$\boldsymbol{Y}^{n-1}$

Table 3.2: Newly constructed data (n-PPS represents the n-periods previous samples)

Current Samples	1-PPS	2-PPS	...	9-PPS	CPU load
$\boldsymbol{X}^9$	$\boldsymbol{X}^8$	$\boldsymbol{X}^7$	...	$\boldsymbol{X}^0$	$\boldsymbol{Y}^{10}$
$\boldsymbol{X}^{10}$	$\boldsymbol{X}^9$	$\boldsymbol{X}^8$	...	$\boldsymbol{X}^1$	$\boldsymbol{Y}^{11}$
...	...		...		...
$\boldsymbol{X}^{n-2}$	$\boldsymbol{X}^{n-3}$	$\boldsymbol{X}^{n-4}$	...	$\boldsymbol{X}^{n-11}$	$\boldsymbol{Y}^{n-1}$

3.1.2 Data Standardization

Because signals have different units and are measured at different scales, estimated coefficients expand or shrink at the same rate as signal units change. Evaluating all the signals in the same scale is a straightforward way to determine the importance of each signal by comparing their associated coefficients, and it avoids the situation where some coefficients are extremely large whereas some are extremely small. So before performing further operation on data, we standardize the data to center the data to zero mean and rescale them to have unit variance. The other reason we need to center the data is that when derive PCA equation in Section 2.2, if each column in $\boldsymbol{D_{n\times p}}$ has zero mean, then variance-covariance matrix of $\boldsymbol{D_{n\times p}}$ is $\frac{\boldsymbol{D}_{n\times p}\cdot\boldsymbol{D}^T_{n\times p}}{n}$:

$$Var[\boldsymbol{D_{n\times p}}] = \begin{bmatrix} Var[\boldsymbol{D}_{n\times 1},\boldsymbol{D}_{n\times 1}] & Var[\boldsymbol{D}_{n\times 1},\boldsymbol{D}_{n\times 2}] & ... & Var[\boldsymbol{D}_{n\times 1},\boldsymbol{D}_{n\times p}] \\ Var[\boldsymbol{D}_{n\times 2},\boldsymbol{D}_{n\times 1}] & Var[\boldsymbol{D}_{n\times 2},\boldsymbol{D}_{n\times 2}] & ... & Var[\boldsymbol{D}_{n\times 2},\boldsymbol{D}_{n\times p}] \\ ... & ... & ... & ... \\ Var[\boldsymbol{D}_{n\times p},\boldsymbol{D}_{n\times 1}] & Var[\boldsymbol{D}_{n\times p},\boldsymbol{D}_{n\times 2}] & ... & Var[\boldsymbol{D}_{n\times p},\boldsymbol{D}_{n\times p}] \end{bmatrix}_{p\times p}$$

because of

$$\begin{aligned} Var[\boldsymbol{D}_{n\times p},\boldsymbol{D}_{n\times k}] &= \frac{1}{n}\sum_{i=1}^{n}(D_{i\times p}-E[D_{n\times p}])(D_{i\times k}-E[D_{n\times k}]) \\ \xrightarrow{E[D_{n\times p}]=E[D_{n\times k}]=0} &= \frac{1}{n}\sum_{i=1}^{n}D_{i\times p}D_{i\times k} = \sum_{i=1}^{n}\boldsymbol{D}_{n\times p}\cdot\boldsymbol{D}_{n\times k} \end{aligned}$$

therefore

$$\begin{aligned} Var[\boldsymbol{D_{n\times p}}] &= \frac{1}{n}\begin{bmatrix} \boldsymbol{D}_{n\times 1}\cdot\boldsymbol{D}_{n\times 1} & \boldsymbol{D}_{n\times 1}\cdot\boldsymbol{D}_{n\times 2} & ... & \boldsymbol{D}_{n\times 1}\cdot\boldsymbol{D}_{n\times p} \\ \boldsymbol{D}_{n\times 2}\cdot\boldsymbol{D}_{n\times 1} & \boldsymbol{D}_{n\times 2}\cdot\boldsymbol{D}_{n\times 2} & ... & \boldsymbol{D}_{n\times 2}\cdot\boldsymbol{D}_{n\times p} \\ ... & ... & ... & ... \\ \boldsymbol{D}_{n\times p}\cdot\boldsymbol{D}_{n\times 1} & \boldsymbol{D}_{n\times p}\cdot\boldsymbol{D}_{n\times 2} & ... & \boldsymbol{D}_{n\times p}\cdot\boldsymbol{D}_{n\times p} \end{bmatrix}_{p\times p} \\ &= \frac{1}{n}\boldsymbol{D}_{n\times p}\cdot\boldsymbol{D}^T_{n\times p} \end{aligned}$$

After standardization, the new value we have is

$$\widetilde{D^p_n} = \frac{D^p_n - E[\boldsymbol{D}^p]}{std[\boldsymbol{D}^p]}$$

where $E[\boldsymbol{D}^p]$ and $std[\boldsymbol{D}^p]$ are the mean and standard deviation of the p-th signal, $\boldsymbol{D}^p$, respectively. Another benefit of standardization is that the intercept of model will disappear. Recall **Property 4** mentioned in Section 2.1.3, trained model goes through the means of dataset, $(\overline{\boldsymbol{X}}, \overline{\boldsymbol{Y}})$. Because $\overline{\boldsymbol{X}}$ and $\overline{\boldsymbol{Y}}$ are zero after standardization, the intercept term, β_0, is zero. We can also convert the standardized coefficients $(\widetilde{\beta_p})$ back to the coefficients that are associate with original data (β_p):

$$\beta_j = \widetilde{\beta_j}\frac{std[\boldsymbol{Y}]}{std[\boldsymbol{D}^j]} \qquad \beta_0 = \bar{\boldsymbol{Y}} - \sum_{i=1}^{j=p} \beta_j \overline{\boldsymbol{D}^j}$$

where:

$\widetilde{\beta_j}$:	the coefficients associated with standardized signal, $\widetilde{\boldsymbol{D}^p}$.
β_j :	the coefficients associated with j-th original signal,$\boldsymbol{D}^j$.
β_0 :	intercept of the model trained by original data
$\overline{\boldsymbol{D}^j}$:	mean value of j-th original signal,$\boldsymbol{D}^j$.

3.2 Highly correlated signal pairs

Variance-covariance matrix of standardized data shows that some signal pairs have Pearson correlation coefficients equal to 1, and some are highly correlated (Pearson correlation coefficient above 0.9), which the reason that high degree of multicollinearity exists and that we cannot precisely estimate model coefficients by using original data. There are 434 signal pairs that have Pearson correlation coefficients equal to 1, and these are listed in Table A.1.

3.3 Feature selection strategies

3.3.1 Filter and wrapper method

Filter and wrapper method are two common strategies in feature selection. Filter method simply scores each input feature by calculating a single associated statistical factor like hypothesis test (Chapter 18 of [9]) and Pearson correlation coefficient ([38]). Then it sets a threshold value to filter out some features. Figure 3.1 illustrates the flow of filter method. The advantage of filter method is that it is easy to implement and does not need heavy calculation, but the result is not guaranteed to be the best performing sub-set features of our model.

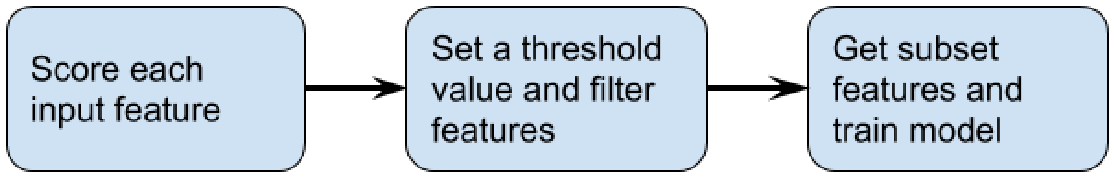
Score each input feature
Set a threshold value and filter features
Get subset features and train model

Figure 3.1: Schematic of filter method

Wrapper method is a recursive strategy that selects features by evaluating and comparing different groups of features. Forward selection and backward elimination are two efficient wrapper methods. They only add or eliminate one input feature each time to or from the current subset based on a evaluation factors that are discussed in Section 2.1.5. Forward selection starts from empty subset, whereas backward elimination starts from full features and they usually give us different results although same evaluation factor is used. In addition, different evaluation factor would also give us different results. Figure 3.2 illustrates the flow of wrapper method. Wrapper method can get more promising result than filter method, but it requires heavy computing and its algorithms are complicated to implement.

Taylor et al. proposed an approach to select features that highly related to the class labels from driver monitoring data to determine the level of driver distraction, in which wrapper method is heavily used [39].

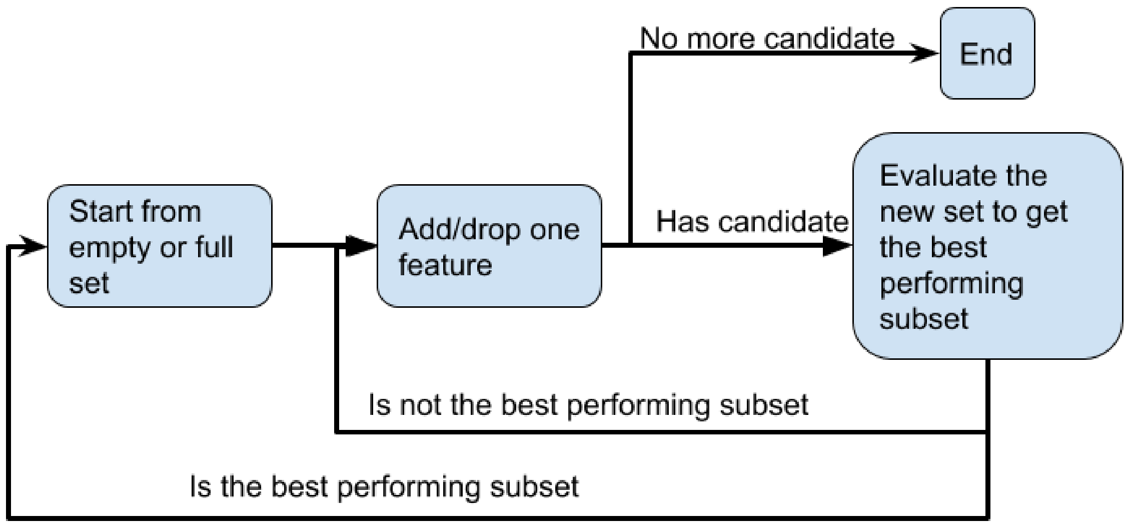
No more candidate
End
Start from empty or full set
Add/drop one feature
Has candidate
Evaluate the new set to get the best performing subset
Is not the best performing subset
Is the best performing subset

Figure 3.2: Schematic of wrapper method

3.3.2 Applying feature selection method on principal components

PCA technique can map n-dimension data to n orthogonal PCs, there are three advantages:

1. Because all the PCs are orthogonal to each other, covariance of any two PCs is zero so multicollinearity does not exist.
2. Sort the n PCs by variance, head PCs have most of total variance and tail PCs have small variance. Therefore we apply feature selection on head PCs that explain large percent of variance, like 95%, rather than on all PCs to avoid massive unnecessary computation.
3. After finishing feature selection on PCs, the sub model coefficients can be map back to the original data to get coefficients of full model.

3.3.3 Proposed methods

By combining feature selection strategies and PCA, the proposed methods are:

Forward selection method:
1. Standardize the original data, $\boldsymbol{D}_{\boldsymbol{n\times p}}$, to get standardized data, $\boldsymbol{X}_{\boldsymbol{n\times p}}$.
2. Divide the data into training data (50%), $\boldsymbol{X}_{tri\times p}$, and test data (50%), $\boldsymbol{X}_{test\times p}$.
3. Apply PCA to $\boldsymbol{X}_{tri\times p}$ to get projection matrix $\boldsymbol{P}_{tri\times k}$, where k is the number of PCA_PCs included that covers a large partition of total variance, like 95%. Python founction for mapping original data to PCA_PCs.
4. Determine a criterion. Refer to Section 2.1.5 the criterion can be the highest R^2_{adj}, the lowest p-value, Cp, AIC, BIC, LOOCV and MSE under k-Fold Cross-Validation.
5. Start from empty subset (no PC is selected), then evaluate each potential PC according to the selected criterion.
6. Add the PC that increases the selected criterion most to subset. Go back to the previous step unit adding new PC does not improve criterion.
7. Map coefficients of PCs back to the original data to get coefficients of full model by Equation 2.35.
8. Map standardized coefficients to un-standardized coefficients.

Backward selection method: Backward elimination method is identical to forward selection except for step 5 and step 6:

5. Start from full PCs, then evaluate each PC according to the selected criterion.
6. Remove the PC that increases the selected criterion most from subset. Go back to the previous step unit removing new PC does not improve criterion.

3.4 Determine the effect window

Because different tasks have different deadline, ideally, the effects of different signals associated with different functions on the CPU load have different length of effect window. But for the problem at hand, the deadline of tasks are not available, instead of assigning different lengths of effect window to different signals, only one effect window is used. The effect window should be the longest deadline among all the tasks, and the following feature selection determine the contribution of different signals at different sampling time to the CPU load. Unfortunately, information about deadline is not available for the moment, so an effect window of 250 ms is used in this thesis, which is an assumption on windows that should be long enough for a practical scenario.

3.5 Result validation

In order to validate the results obtained by using different methods, we divide the reconstructed data into two different sets: 50% of the data are used for training, while the remaining 50% are used for test and validation. Training set is for training the model by using different methods, and the test data is for evaluating the quality of fit of the obtained model. MSE on the test data is calculated as

$$MSE_{test} = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} (Y_{test}^{i} - \hat{Y}^{i})^2$$

where n_{test} is the size of test data, Y_{test}^{i} are the actual value of test data, and $\hat{Y}^{i}$ are the predicted values by feeding the signals from the test data to the trained model.

4
Results

4.1 Continuous CPU load curve

As mentioned in Section 1.2.2, CPU load is calculated by a sliding average window, so the sampled CPU load is time series data, which agrees with the CPU load plotted in Figure 4.1.

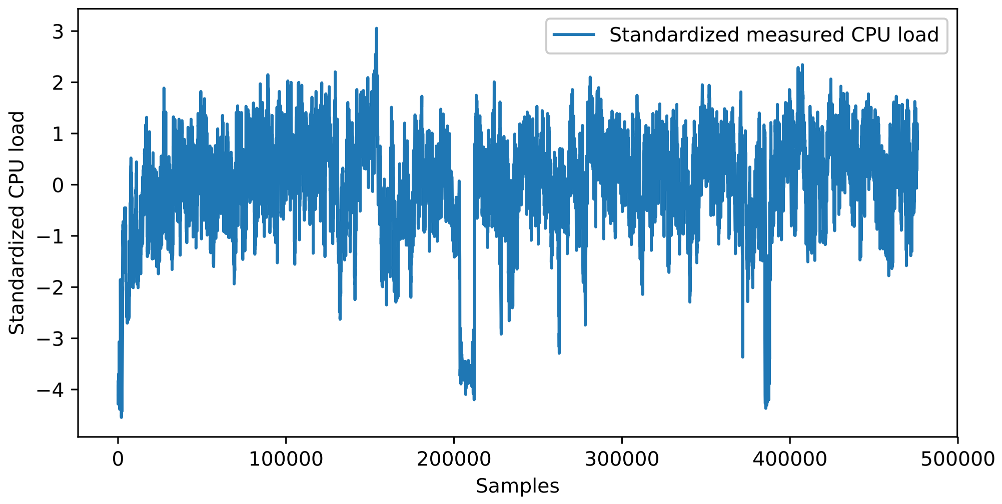

Figure 4.1: Plotted CPU load samples. Y-axis values is standardized CPU load.

4.2 Severe multicollinearity problem

Because there are 1400 kinds of signal in our data, there are $C\binom{1400}{2} = 979300$ signal pairs. Among these signal pairs, Table A.1 shows the highly correlated signal pairs and Table 4.1 shows the number of signal pairs that are located in different Pearson correlation coefficients interval.

Table 4.1: Number of signal pairs with different Pearson correlation coefficient

Pearson correlation coefficient	number of signal pairs
$= 1$	127
≥ 0.99	1477
≥ 0.98	1797
≥ 0.97	2017
≥ 0.96	2292
≥ 0.95	2508
≥ 0.90	3195

As mentioned, high degree multicollinearity exits in original data, and 434 signal pairs have correlation coefficients equal to 1. Even when these signals are removed, the remaining 1227 VIFs associated with each signal are still high and the highest VIF is infinite. Figure 4.2 shows the lowest 800 VIFs.

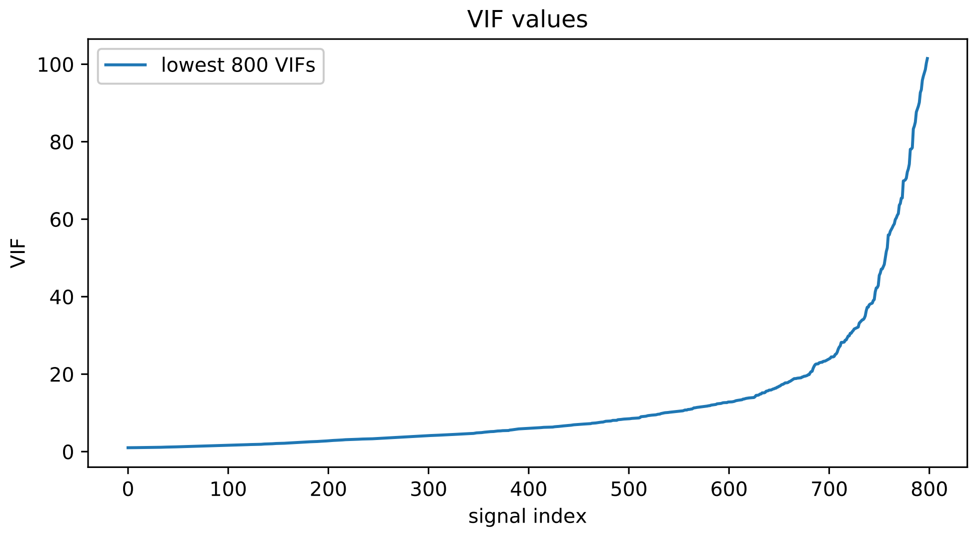

Figure 4.2: Lowest 800 VIFs after removing the signal pairs that have correlation coefficients equal to 1

4.3 Signal preliminary selection

In the reconstructed data, each signal may have delayed effects on the CPU load. In the effect window, the effects of previous signal samples may propagate to the current CPU load. The cross-correlation of signal X_t with CPU load Y_t are defined as:

$$\rho(\tau) = \frac{1}{\sigma_X \sigma_Y} E[(X_{t-\tau} - \mu_X)(Y_t - \mu_Y)]$$

where σ_X and μ_X are mean and standard deviation of X_t, σ_Y and μ_Y are mean and standard deviation of Y_t, and $\tau \in [1, 10]$. Including the past 10 samples of all the signals, there are 14000 predictors in total. By calculating the cross-correlation of the original 1400 signal with the CPU load, 9044 predictors with correlation coefficient below 0.1 are excluded and 4956 predictors are left.

4.4 PCR

Figure 4.3 shows that R^2 value benefits from the number of PCA_PCs included into model, but it stops at where the number of PCA_PCs is around 500. We use the PCA_PCs that cover 95% of total variance and perform feature selection methods that are illustrated in Section 3.3 on them. The combinations of 2 feature selection strategies (backward elimination and forward selection) and 4 criteria (R^2_{adj}, AIC, BIC and Mallow's Cp) give rise to 8 results. These results are very similar, so that the 8 curves corresponding to 8 results are overlapped in Figure 4.6.

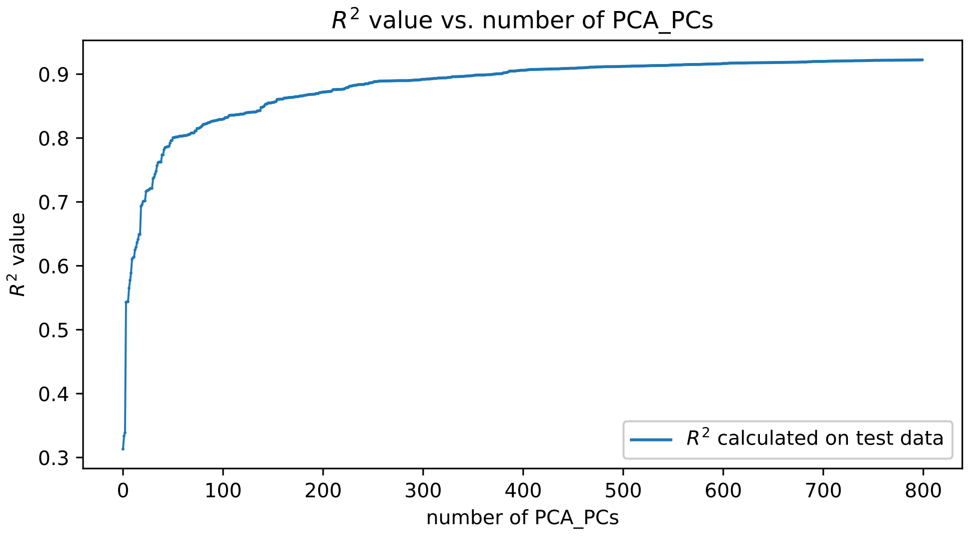

Figure 4.3: R^2 increases as more PCA_PCs are included into model

4.5 PLSR

Technically, the more PCs are included into the model, the higher quality-of-fit (larger R-squared value) is. Figure 4.4 shows that R-squared value increases as more PLSR_PCs are included into the model. In Figure 4.4 R-squared value stop increasing at where the number of PLSR_PCs is around 30. So the 30 PLSR_PCs are selected to our model with MSE_{test} of 0.099, which fulfills **Goal 4**. Figure 4.6 shows coefficients in decreasing order by using PLSR method.

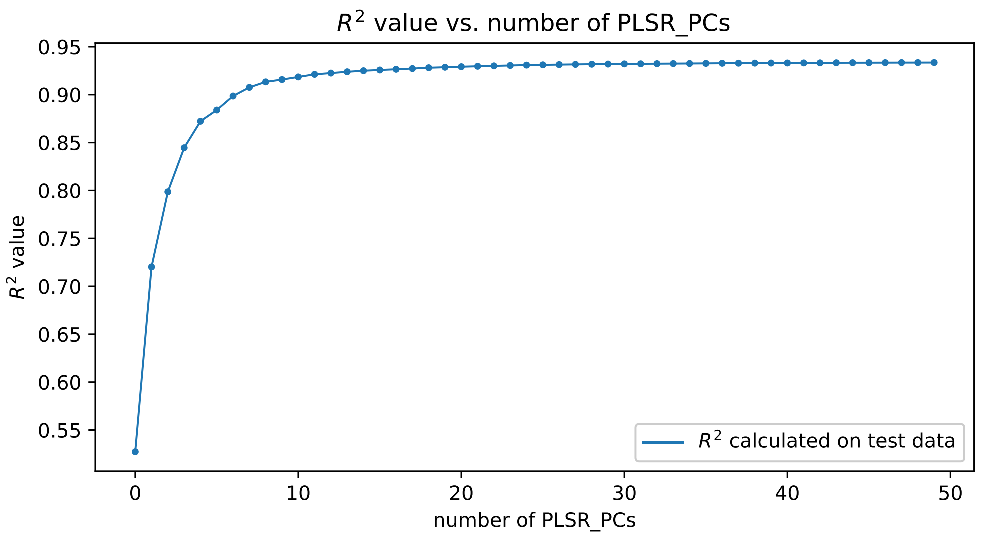

Figure 4.4: R^2 increases as more PLSR_PCs are included into model

4.6 Ridge regression

Recall Equation 2.46 and Figure 2.3, cross validation on MSE is used for choosing the λ that minimizes the MSE. Figure 4.5 shows the trend of MSE when λ changes, and the optimal λ is 1778.52, which gives rise to MSE_{test} of 0.11 on test data. The corresponding result is very similar to the 8 results obtained by using PCR method, and these 9 results are overlapped in Figure 4.6

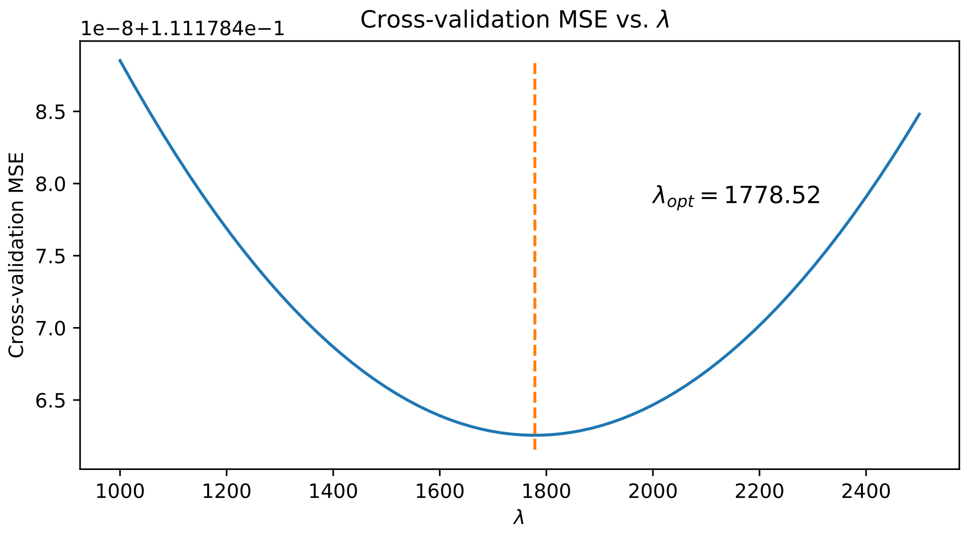

Figure 4.5: Finding the optimal λ by swiping the value of λ

4.7 Training results and validation

In Figure 4.6, contribution coefficients are plotted in decreasing order. The 8 results of PCR method overlap with the result obtained by using Ridge regression, and they have MSE_{test} of around 0.11. The result obtained by using PLSR has MSE_{test} of around 0.099.

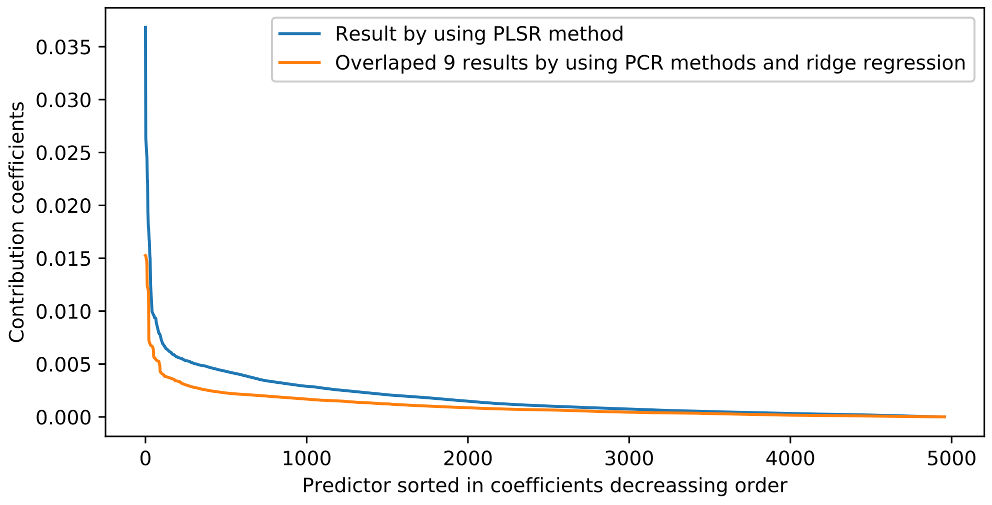

Figure 4.6: Contribution coefficients are plotted in decreasing order.

By feeding the original data to the trained model, the predicted value is obtained. The predicted CPU load and the measured CPU load are plotted in Figure 4.7. The predicted CPU load is roughly as same as the measured CPU load with MSE on the test data of 0.11.

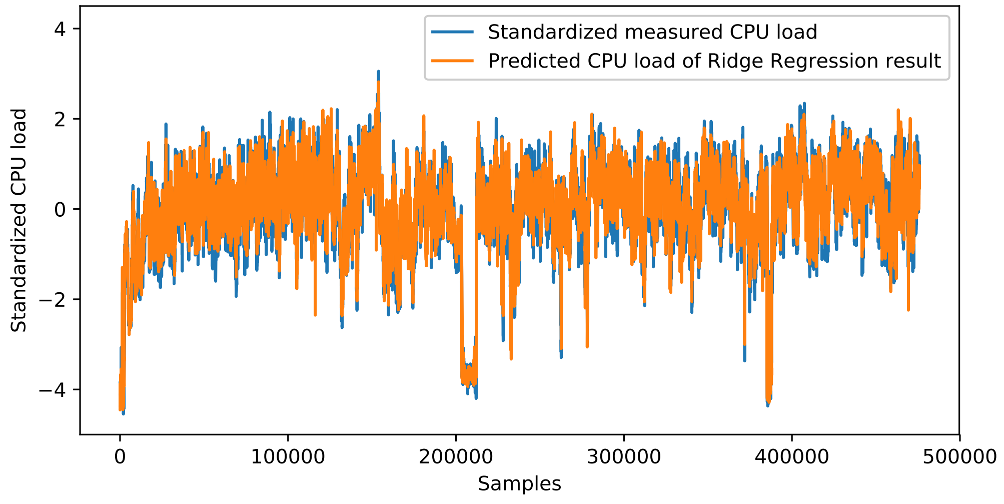

Figure 4.7: Plotted measured CPU load and predicted CPU load. Y-axis is standardized CPU load.

Figure 4.8 shows the scatter plot of predicted CPU load and measured CPU load. With perfect prediction, the points are scattered on a straight line with slope of 1 (orange line). Actually, caused by prediction error, the blue points deviate from the orange line, and the degree of deviation depends on the degree of prediction error.

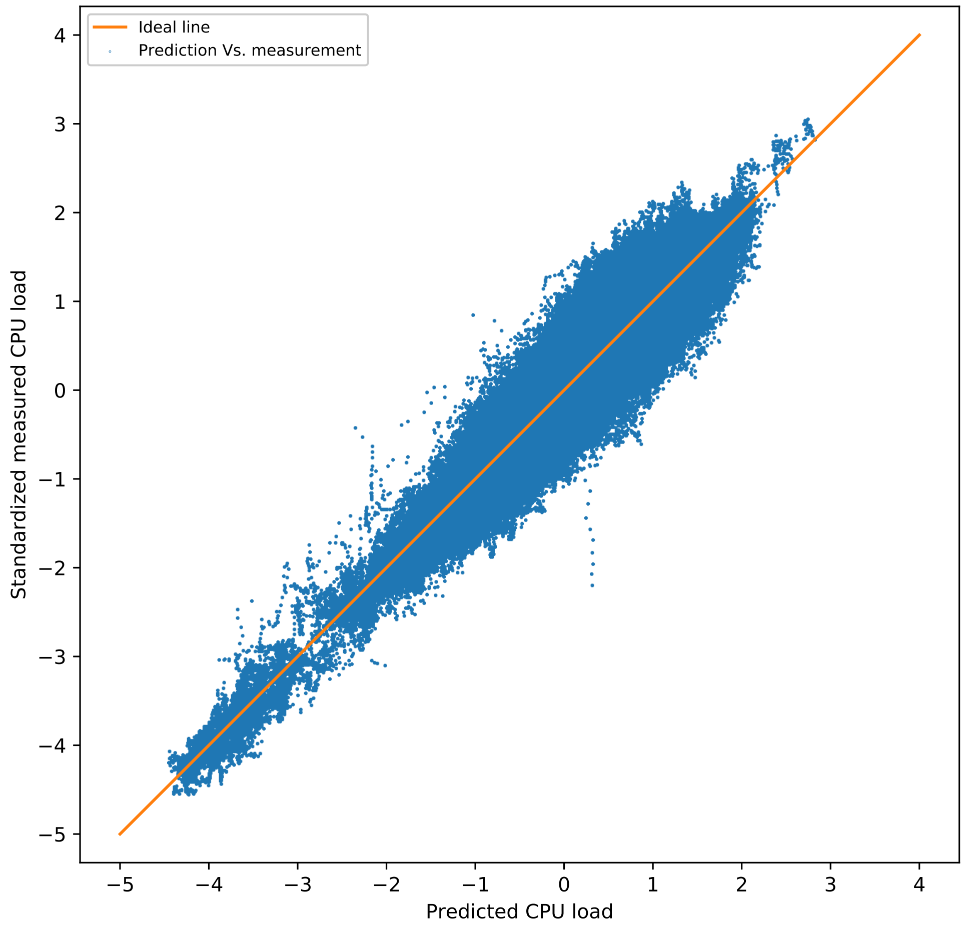

Figure 4.8: Plotted measured CPU load and predicted CPU load. Y-axis is standardized measured CPU load values.

5
Conclusion

In this project, given the sampled input signals to all the software modules that are embedded running on the CPU, a model for predicting CPU load in the real-time system is proposed. Considering how the CPU load is calculated in the real-time system, this model finds relationship between the values of past signals and the CPU busy time for the current calculation of CPU load. Because the interpretability of contribution of each signal to the CPU load is required, this model is based on the following assumptions:

1. Linear relationship between the sampled values of signals and CPU load, which means the signals associated with larger coefficient has higher contribution to the CPU load.
2. Causal relationship between the sampled values of signals and CPU load, which means the current sampled CPU load only depends on the past sampled signals.

After our analysis, 9044 signals can be excluded from the constructed data shown in Table 3.2, because their associated absolute values of standardized correlation coefficients are smaller than 0.1. After the preliminary filtering, 4956 signals are left, and the size of current dataset is 475825×4956. The dataset is standardized next. In order to train the model and validate the quality of the result, the data set is divided into training set (50%) and test set (50%).

In general, OLS is the most efficient method for training a linear regression model. But because high degree of multicollinearity exists in the data, which inflates the variance of estimates, conventional OLS is not feasible for this problem. In order to overcome the multicollinearity problem, PCR, PLSR and ridge regression are used. The ideal model should achieve high quality of fit and meanwhile keep simple according to Occam's razor theory [40]. In order to do so, two feature selection strategies (backward elimination and forward selection) are illustrated in Section 3.3 and some criteria for feature selection are derived in Section 2.1.5.

As the result, PCR method and ridge regression method give rise to the similar results with MSE of around 0.11. By using PLSR method, result with lower MSE is achieved: 0.099. Compared with the current status as described in Section 1.3.4, a new model is proposed: the effects of previous input propagate to the current CPU load. MSE achieved by new model is 0.11, which is less than the MSE of 0.15 mentioned in **Goal 4**. Different feature selection methods that can overcome multicollinearity are proposed, and based on different methods, 10 results are obtained, which fulfills **Goal 1** and **Goal 2**. Unfortunately, **Goal 3** is not fulfilled, but it is remedied by dividing the data to training set and test set.

All the results show that the *signal 1400* has the highest positive contribution to the CPU load. *signal 1400* represents the number of objects on road when cars request steering and more objects require more computation of the CPU, which agrees with the existing result in [20]. The signal with second largest contribution is *signal 973*, which represents the current distance with low confidence for the road geometry. Because functions make decision based on the road geometry, it is highly used in many functions such as auto-breaking and steering, which needs much computational effort [8].

5.1 Weaknesses of the model

First of all, the model is based on the assumption of that the CPU load can be predicted by a linear function of values of input signals to the functions that are running on the CPU. Linearity is assumed because of the requirement of interpretability whereas the actual relationship between the CPU load and the signal values can be extremely complicated and highly non-linear, which is the most important weakness of the model. Second, the effects of signals on the CPU load is assumed a constant in the corresponding effect time. However, the effect can fluctuate because of the CPU scheduling. Third, from real-time systems perspective, the effect window should be the longest deadline among all the tasks. But because the deadline is unknown, the effect window is assumed 250 ms. Then feature selection is performed to select the signals with high contributions to the CPU load further.

5.2 Future work

Using actual effect window instead of 250 ms will improve reliability of the results, which needs further communication with ECU supplier. Because the interpretability of contribution of each signal is required, this model is restricted to the linear relationship. It is possible to increase the quality of fit by introducing polynomial terms and then perform the same training work without losing interpretability. For example, manipulate Table 3.2 to construct quadratic data:

$$\begin{bmatrix} \boldsymbol{X}^{9} & \boldsymbol{X}^{9^2} & \boldsymbol{X}^{8} & \boldsymbol{X}^{8^2} & ... & \boldsymbol{X}^{0} & \boldsymbol{X}^{0^2} & \rightarrow & \boldsymbol{Y}^{10} \\ \boldsymbol{X}^{8} & \boldsymbol{X}^{8^2} & \boldsymbol{X}^{7} & \boldsymbol{X}^{7^2} & ... & \boldsymbol{X}^{1} & \boldsymbol{X}^{1^2} & \rightarrow & \boldsymbol{Y}^{11} \\ ... & ... & ... & ... & ... & ... & ... & \rightarrow & ... \\ \boldsymbol{X}^{n-2} & \boldsymbol{X}^{n-2^2} & \boldsymbol{X}^{n-3} & \boldsymbol{X}^{n-3^2} & ... & \boldsymbol{X}^{n-11} & \boldsymbol{X}^{n-11^2} & \rightarrow & \boldsymbol{Y}^{n} \end{bmatrix}$$

Other regularization methods such as LASSO regression and elastic net can also be used for the training.

As mentioned, interpretability is required in this project. So there is not too many alternatives but to use linear model. Models that can learn the non-linearity between CPU load and signal values and keep interpretable meanwhile are worth studying, which may need to combine the knowledge from hardware level and expertise on machine learning. Because the current CPU load depends on the previous signal values, Recurrent neural network (RNN) may also be an option.

Bibliography

[1] NHTSA. "The Evolution of Automated Safety Technologies". In: (). URL: `https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety` (visited on 04/23/2018).

[2] Peter Valdes-Dapena. "Volvo promises deathproof cars by 2020". In: (). URL: `http://money.cnn.com/2016/01/20/luxury/volvo-no-death-crash-cars-2020/index.html` (visited on 02/25/2018).

[3] *AUTOSAR (AUTomotive Open System ARchitecture)*. URL: `https://www.autosar.org/` (visited on 05/03/2018).

[4] *Recommended Methods and Practices for Timing Analysis and Design within the AUTOSAR Development Process AUTOSAR CP Release 4.3.1*. URL: `https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_TR_TimingAnalysis.pdf` (visited on 05/03/2018).

[5] J. Borenstein and Y. Koren. "Obstacle avoidance with ultrasonic sensors". In: *IEEE Journal on Robotics and Automation* 4.2 (Apr. 1988), pp. 213–218. ISSN: 0882-4967. DOI: `10.1109/56.2085`.

[6] E. Ward and J. Folkesson. "Vehicle localization with low cost radar sensors". In: *2016 IEEE Intelligent Vehicles Symposium (IV)*. June 2016, pp. 864–870. DOI: `10.1109/IVS.2016.7535489`.

[7] J. Berkley. *Automotive LiDAR*. California State University San Marcos, 2016. URL: `https://books.google.se/books?id=HadtAQAACAAJ`.

[8] VOLVO CAR CORPORATION. *internal document*.

[9] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[10] Kenneth P. Burnham and David R. Anderson. *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Vol. 67. Jan. 2002.

[11] Achim Zielesny. "Machine Learning". In: *From Curve Fitting to Machine Learning: An Illustrative Guide to Scientific Data Analysis and Computational Intelligence*. Cham: Springer International Publishing, 2016, pp. 229–406. ISBN: 978-3-319-32545-3. DOI: `10.1007/978-3-319-32545-3_4`. URL: `https://doi.org/10.1007/978-3-319-32545-3_4`.

[12] Simon Rogers and Mark Girolami. *A first course in machine learning*. English. 2. Boca Raton: CRC Press, 2016. ISBN: 1498738486.

[13] Isabelle Guyon and André Elisseeff. "An Introduction to Variable and Feature Selection". In: *J. Mach. Learn. Res.* 3 (Mar. 2003), pp. 1157–1182. ISSN: 1532-4435. URL: `http://dl.acm.org/citation.cfm?id=944919.944968`.

[14] Jundong Li et al. “Feature Selection: A Data Perspective”. In: *ACM Comput. Surv.* 50.6 (Dec. 2017), 94:1–94:45. ISSN: 0360-0300. DOI: 10.1145/3136625. URL: http://doi.acm.org.proxy.lib.chalmers.se/10.1145/3136625.

[15] Girish Chandrashekar and Ferat Sahin. “A survey on feature selection methods”. English. In: *Computers & Electrical Engineering* 40.1 (2014), pp. 16–28. ISSN: 00457906.

[16] R. Rubinstein, A. M. Bruckstein, and M. Elad. “Dictionaries for Sparse Representation Modeling”. In: *Proceedings of the IEEE* 98.6 (Jan. 2010), pp. 1045–1057. ISSN: 00189219. DOI: 10.1109/JPROC.2010.2040551.

[17] Peter Bhlmann and Sara van de Geer. *Statistics for High-Dimensional Data: Methods, Theory and Applications.* 1st. Springer Publishing Company, Incorporated, 2011. ISBN: 9783642201912.

[18] Robert Tibshirani. “Regression shrinkage selection via the LASSO”. In: 73 (June 2011), pp. 273–282.

[19] G E Hinton and R R Salakhutdinov. “Reducing the dimensionality of data with neural networks”. In: *Science* 313.5786 (July 2006), pp. 504–507. DOI: 10.1126/science.1127647. URL: http://www.ncbi.nlm.nih.gov/sites/entrez?db=pubmed&uid=16873662&cmd=showdetailview&indexed=google.

[20] Lars Tornberg. *Private communication.*

[21] Hui Zou and Trevor Hastie. “Regularization and variable selection via the Elastic Net”. In: *Journal of the Royal Statistical Society, Series B* 67 (2005), pp. 301–320.

[22] Lindsay I Smith. “A tutorial on Principal Components Analysis”. In: (2002). [Online; accessed 03-March-2018]. URL: http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf.

[23] MathWorks. *What Is Hardware-In-The-Loop Simulation?* https://se.mathworks.com/help/physmod/simscape/ug/what-is-hardware-in-the-loop-simulation.html.

[24] J Edward Jackson. “A User’s Guide to Principal Components / J.E. Jackson.” In: (Feb. 2018).

[25] Hervé Abdi. “Partial least squares regression and projection on latent structure regression (PLS Regression)”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 2.1 (2010), pp. 97–106. ISSN: 1939-0068. DOI: 10.1002/wics.51. URL: http://dx.doi.org/10.1002/wics.51.

[26] John Aldrich. “R.A. Fisher and the making of maximum likelihood 1912-1922”. In: *Statist. Sci.* 12.3 (Sept. 1997), pp. 162–176. DOI: 10.1214/ss/1030037906. URL: https://doi.org/10.1214/ss/1030037906.

[27] A.W.F. Edwards. *Likelihood.* Cambridge University Press, 1972. URL: https://books.google.se/books?id=8eMIAQAAIAAJ.

[28] Jeffrey Wooldridge. *Introductory Econometrics: A Modern Approach, Fifth Edition.* 5th. Cengage Learning, 2013.

[29] Henry Cohn et al. “Group-theoretic Algorithms for Matrix Multiplication”. In: *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science.* FOCS ’05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 379–388. ISBN: 0-7695-2468-0. DOI: 10.1109/SFCS.2005.39. URL: https://doi.org/10.1109/SFCS.2005.39.

[30] Royal Society (Great Britain). *Proceedings of the Royal Society of London.* v. 58. Taylor & Francis, 1895. URL: `https://books.google.se/books?id=60aL0zlT-90C`.

[31] R.J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice. OTexts: Melbourne, Australia.* 2013. URL: `https://www.otexts.org/fpp` (visited on 04/01/2018).

[32] Hirotogu Akaike. “Information Theory and an Extension of the Maximum Likelihood Principle”. In: *Selected Papers of Hirotugu Akaike.* Ed. by Emanuel Parzen, Kunio Tanabe, and Genshiro Kitagawa. New York, NY: Springer New York, 1998, pp. 199–213. ISBN: 978-1-4612-1694-0. DOI: `10.1007/978-1-4612-1694-0_15`. URL: `https://doi.org/10.1007/978-1-4612-1694-0_15`.

[33] Gideon Schwarz. “Estimating the Dimension of a Model”. In: *Ann. Statist.* 6.2 (Mar. 1978), pp. 461–464. DOI: `10.1214/aos/1176344136`. URL: `https://doi.org/10.1214/aos/1176344136`.

[34] Gregoria Mateos-Aparicio. “Partial Least Squares (PLS) Methods: Origins, Evolution, and Application to Social Sciences”. In: *Communications in Statistics - Theory and Methods* 40.13 (2011), pp. 2305–2317. DOI: `10.1080/03610921003778225`. eprint: `https://doi.org/10.1080/03610921003778225`. URL: `https://doi.org/10.1080/03610921003778225`.

[35] Leo Breiman and Jerome H. Friedman. “Predicting Multivariate Responses in Multiple Linear Regression”. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 59.1 (), pp. 3–54. DOI: `10.1111/1467-9868.00054`. eprint: `https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/1467-9868.00054`. URL: `https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/1467-9868.00054`.

[36] Arthur E. Hoerl and Robert W. Kennard. “Ridge Regression: Biased Estimation for Nonorthogonal Problems”. In: *Technometrics* 12.1 (1970), pp. 55–67. DOI: `10.1080/00401706.1970.10488634`. eprint: `https://www.tandfonline.com/doi/pdf/10.1080/00401706.1970.10488634`. URL: `https://www.tandfonline.com/doi/abs/10.1080/00401706.1970.10488634`.

[37] Wessel N van Wieringen. “Lecture notes on ridge regression”. In: *eprint arXiv:1509.09169* (). DOI: `10.1111/1467-9868.00054`. eprint: `https://arxiv.org/abs/1509.09169`. URL: `https://arxiv.org/abs/1509.09169`.

[38] Wikipedia contributors. *P-value — Wikipedia, The Free Encyclopedia.* [Online; accessed 16-Feb-2018]. 2018. URL: `https://en.wikipedia.org/wiki/P-value`.

[39] Phillip Taylor et al. “Redundant Feature Selection for Telemetry Data”. In: *Agents and Data Mining Interaction.* Ed. by Longbing Cao et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 53–65. ISBN: 978-3-642-55192-5.

[40] Pierre Latouche; Pierre-Alexandre Mattei; Charles Bouveyron; Julien Chiquet. “Combining a relaxed EM algorithm with Occam’s razor for Bayesian variable selection in high-dimensional regression”. In: *Journal of Multivariate Analysis* (). DOI: `10.1016/j.jmva.2015.09.004`.

A
Appendix 1

Table A.1: Signal pairs with correlation coefficient equals to 1.

pair	index1	index2
1	4	56
2	5	1289
3	46	52
4	46	82
5	52	82
6	66	67
7	86	292
8	120	121
9	122	123
10	122	124
11	122	137
12	123	124
13	123	127
14	123	137
15	124	127
16	124	137
17	125	127
18	127	137
19	265	297
20	293	1207
21	295	1208
22	301	1246
23	302	1216
24	303	1217
25	304	1218
26	305	1214
27	306	1215
28	307	1275
29	387	388
30	391	392
31	393	394
32	395	396
44	442	526
45	525	526
46	547	639
47	552	565
48	552	581
49	552	643
50	565	581
51	565	643
52	574	578
53	575	579
54	581	643
55	591	593
56	591	667
57	591	668
58	591	669
59	591	670
60	591	687
61	591	688
62	591	689
63	591	690
64	592	601
65	593	667
66	593	668
67	593	669
68	593	670
69	593	687
70	593	688
71	593	689
72	593	690
73	617	626
74	663	664
75	667	668
86	668	689
87	668	690
88	669	670
89	669	687
90	669	688
91	669	689
92	669	690
93	670	687
94	670	688
95	670	689
96	670	690
97	687	688
98	687	689
99	687	690
100	688	689
101	688	690
102	689	690
103	764	777
104	787	788
105	787	801
106	788	801
107	789	802
108	806	807
109	923	924
110	927	928
111	931	932
112	1101	1102
113	1101	1103
114	1102	1103
115	1188	1189
116	1188	1202
117	1188	1203

Table A.1: Purely correlated signal pairs

pair	index1	index2
33	407	408
34	411	412
35	415	416
36	417	418
37	419	420
38	421	422
39	427	428
40	441	442
41	441	525
42	441	526
43	442	525
76	667	669
77	667	670
78	667	687
79	667	688
80	667	689
81	667	690
82	668	669
83	668	670
84	668	687
85	668	688
118	1188	1212
119	1189	1202
120	1189	1203
121	1189	1212
122	1191	1192
123	1202	1203
124	1202	1212
125	1203	1212
126	1231	1232
127	1281	1282

B
Appendix 2

Plan review

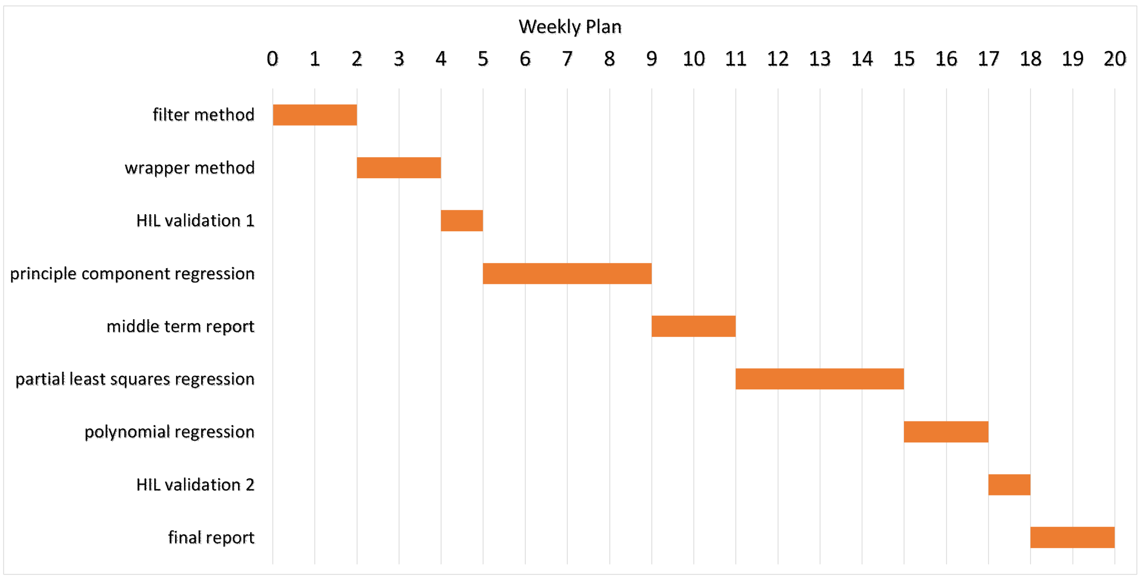

Figure B.1: Initial weekly plan

Figure B.1 is the initial weekly plan. Because the limited resources, HIL validation was rescheduled to be performed out of the scope of this thesis project. As remedy, the data is divided into training set and testing set. There are 14000 signals which are included in the model as predictors. According to 2.2, increase of the number of predictors results in cubic growth of the computation time, so feature selection is very much time-consuming. Polynomial regression multiplies the number of predictors and therefore results in more computational effort, so polynomial regression is not performed either.

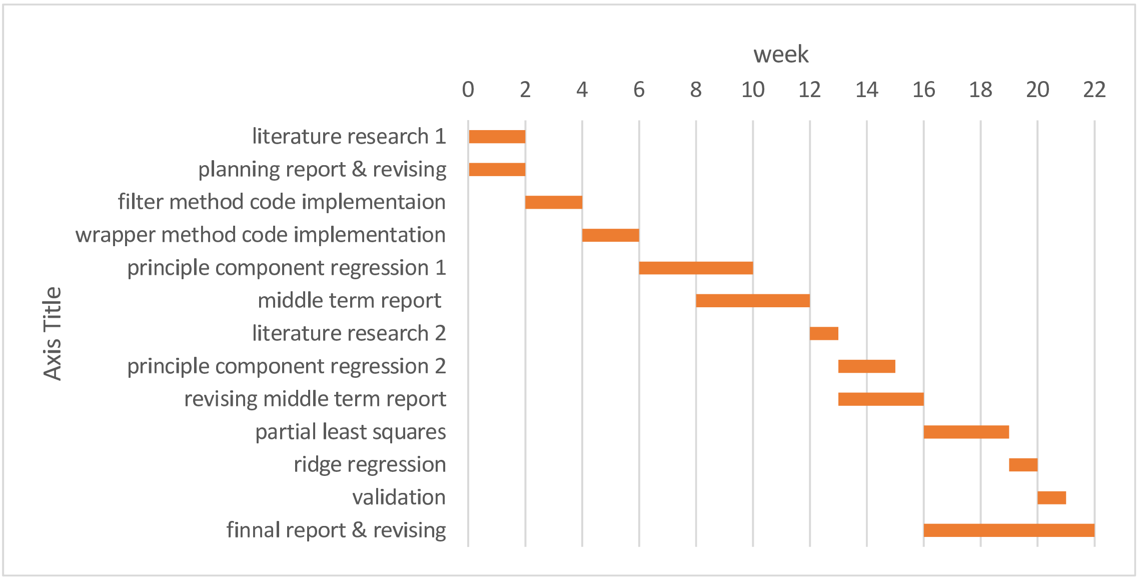

Figure B.2: Actual time line

Figure B.2 shows the actual time line of the project. Compared with initial time plan, Figure B.2 is adjusted. *literature research 1* did basic studies on the research environment, problem and background. Meanwhile a planning report was written and revised. A simple model was implemented and trained using step-wise feature selection on principal component of the original data between week 8 and 12. Because the training processing is time consuming, writing report and training the model are in parallel. After reading AUTOSAR manual in *literature research 2*, the model was improved by taking into account of the effects of past samples on the CPU load. Retrain the model and revise the mid-term report from week 13 to 16. In the rest 6 weeks, partial least squares method and ridge regression method are tried. Then all the models trained by different methods are tested by testing data.