On Fast Computation of Nested Cross-Validation

by

Shuyang (Henry) Cao

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Statistics

Waterloo, Ontario, Canada, 2026

© Shuyang (Henry) Cao 2026

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Cross-validation is a resampling procedure that provides a point estimate of prediction error. Uncertainty in this estimate is challenging to quantify. Nested Cross-validation [4] addresses this by providing a prediction interval. However, computing this interval can be infeasible as it requires running the cross-validation procedure and refitting the model an extraordinary number of times. Under penalized regression, we provide a fast and efficient way to compute the prediction interval with a single model fit. We give a thorough analysis of our proposed method, from the theoretical property of its core computation technique to its implementation and complexity. Our method performs best when the number of folds scales with sample size and reduce the computation by $O(n)$, where n is the sample size. For application, we evaluate nested cross-validation on tuning parameters for signal regression models, comparing results with restricted maximum likelihood and other cross-validation based procedures.

Acknowledgements

I would like to start by thanking my supervisor Prof. Alex Stringer. This thesis is largely based upon a research project I worked under his supervision and a paper we co-write under preparation. He took me as an undergraduate research assistant at early 2024, and I have been working with him for more than two years and a half. His kindness, patience and wonderful guidance has been the biggest encouragement for me to grow as a researcher. Preparing this thesis brought me back to those days of coming to his office for weekly meetings since 2024. I start from knowing nothing about research and academic writing to the point of finishing this thesis. This would not be possible without his support and his belief on me. I will forever remember those good memories of our early morning meetings. So many times I was frustrated about research during the week, and somehow his encouragement magically boosted my energy after the meeting. Thank you.

I also want to thank Prof. Jeffrey Negrea for serving as my committee member and taking time to read this thesis. Prof. Negrea has been another important figure in my short academic career. I took his stochastic processes course at the end of my undergrad, during which I was at a low point of my passion for math. I learned so many interesting stuffs from his course, and his passion for mathematics made me start to believe in myself again. During my time as a graduate student, his wisdom and inspiring spirit continue every time I chatted with him. He will be someone I learn from whether or not I continue my research career.

There are other faculty members I want to express my gratitude. Thank Prof. Mufan Li for also serving as my committee member and spending time reviewing this thesis. Thank Prof. Aukosh Jagannath for teaching the high dimensional probability course and letting me learn so much eye-opening knowledge. Thank Prof. Greg Rice for many wonderful chats during my time as his TA. Thank Prof. Glen McGee for his constant guidance on statistics. His course on linear models was the reason I fell into statistics during undergraduate. After taking three courses with him both as an undergrad and as a master student, his insights on applied statistics have always been influential to me and motivate me to find my own place in statistics. Last, I want to thank Prof. Surya Banerjee for many interesting chats in his office and the books we exchange for reading.

I also want to thank some of my peers. Thank Nigel for being my informal math coach all the time. Specifically, his feedback on Theorem 3 of this work help me to make the proof better. Thank Tianyi for always being a good mentor when I am struggling with research. His advice shaped many parts of this work. Thank Jan for being a good friend. During the time we took a class together, his support was beyond what I would ever wish

for. All those late night memories of working on math with him in the graduate lounge will never be forgotten. Thank Keeley for many insightful chats we had. Even if our friendship began at undergrad, it was during our master program that I get to know you deeper. Your friendship made my time in this program more rewarding. I also want to extend my thanks to Ethan, Maruhan and Andrei, who have been my supportive friends since early days of undergrad.

Finally, I want to thank my mom for her unconditional love. The past several years have been rough for me in many aspects of life. Her support was the light that carried me through the darkest moments. Her love taught me how to treat the world with a gentle heart, even when life hits me hard.

Memory is a different form of life. All the people I mentioned above brought me so many warm memories. You are and will always be part of my life. I appreciate you for being who you are and your kindness. Thank you.

Table of Contents

Author's Declaration ii

Abstract iii

Acknowledgements iv

List of Figures viii

List of Tables ix

1 Introduction 1

1.1 Overview . . . 1
1.2 Contribution . . . 1
1.3 Organization . . . 2

2 Background 3

2.1 Notations . . . 3
2.2 Predictive modeling . . . 4
2.3 Cross-validation . . . 4
2.4 Literature Review . . . 5
2.5 Nested Cross-validation . . . 6
2.6 Penalized regression . . . 8
2.7 Computational Challenge . . . 8

3 Main Results **9**

3.1 Theory . . . 10

3.1.1 Core Technique . . . 10

3.1.2 Invertibility . . . 11

3.2 Computation . . . 16

3.3 Implementation and Complexity . . . 18

3.3.1 Proposed Formula . . . 18

3.3.2 Refitting the Model . . . 19

3.3.3 An Alternative Implementation . . . 20

3.3.4 Compare Asymptotic Complexity . . . 22

3.3.5 Compare Finite Sample Implementation . . . 25

3.4 Extension to Ultra High Dimension Regression . . . 28

4 Application **31**

4.1 Model: Functional Principal Components . . . 31

4.2 Simulation . . . 32

4.2.1 REML . . . 33

4.2.2 Simulation Details . . . 33

4.2.3 Results . . . 34

References **37**

APPENDICES **40**

A Preliminaries on Numerical Linear Algebra **41**

B Convex Optimization **43**

C Background of Ultra High Dimension Ridge Regression **46**

List of Figures

3.1 Change of log scale relative computation time for fitting an additive model. Relative computation means the computation time of refitting the model (slow) over the computation time of NCV formula (fast). Regime corresponds to $K = O(n)$, p fixed in Table 3.4, meaning n increases and p keeps fixed. 23

3.2 Change of log scale relative computation time for fitting an additive model. Relative computation means the computation time of refitting the model (slow) over the computation time of NCV formula (fast). Regime corresponds to $K = O(n)$, $p = O(n^{\alpha})$ in Table 3.4, meaning n increases and p also increases with n. 24

3.3 Change of relative computation time (refitting model / NCV formula in Proposition 1) with the increase of K and varying pair of (n, p). Horizontal dotted line is 1, meaning computation time of both is equal. Vertical dotted line is the predicted K^*. Relative computation time is expected to rise above 1 when K is bigger than predicted K^* and vice versa. 27

4.1 Boxplot of prediction errors with results of tuning parameters. NCV point estimates with prediction interval are shown as solid lines. Choices of tuning parameters that give the lowest GCV score, LOO score and that maximize REML are shown as vertical dotted lines. Each row corresponds to one choice of principal components, hence one variance proportion. 35

4.2 Plot of negative restricted maximum likelihood. Vertical dotted line indicates the value of tuning parameter that minimizes the negative REML curve. Each row corresponds to one choice of principal components, hence one variance proportion. 36

List of Tables

3.1 Step-wise computational complexity and FLOP count for computing $\widehat{r}_{jk}^{\text{te}}$ with proposed formula. Total FLOP count and complexity are also reported. 19

3.2 Step-wise computational complexity and FLOP count for computing $\widehat{r}_{jk}^{\text{te}}$ with refitting models. Total FLOP count and complexity are also reported. 20

3.3 Step-wise computational complexity and FLOP count for computing $\widehat{r}_{jk}^{\text{te}}$ with proposed formula, but an alternative implementation using Woodbury. Total FLOP count and complexity are also reported. 21

3.4 Asymptotic complexity required to compute the nested cross-validation prediction interval by model refitting (Eq. (3.20) and using Proposition 1 with and without the Woodbury identity (Eq. (3.21)). Shown are the four asymptotic regimes: K-fold (K fixed) and leave-k-out ($K = O(n)$) nested cross-validation, as well as low-dimensional (p fixed) and high-dimensional ($p = O(n^\alpha)$ for $1 \geq \alpha > 0$) penalized regression. 22

3.5 Step-wise computational complexity and FLOP count for computing $\widehat{r}_{jk}^{\text{te}}$ with proposed formula. Model is ridge regression in ultra high dimension ($n < p$). Total FLOP count and complexity are also reported. 29

3.6 Step-wise computational complexity and FLOP count for computing $\widehat{r}_{jk}^{\text{te}}$ with refitting models. Model is ridge regression in ultra high dimension ($n < p$). Total FLOP count and complexity are also reported. 30

3.7 Asymptotic complexity required to compute the nested cross-validation prediction interval by model refitting and using Proposition 1. Model is ridge regression in ultra high dimension ($n < p$). Shown are the four asymptotic regimes: K-fold (K fixed) and leave-k-out ($K = O(n)$) nested cross-validation, as well as p fixed and $p = O(n^\alpha)$ for $\alpha > 1$. 30

A.1 FLOP Count of Vector and Matrix Operations [16, p. 12] 41

Chapter 1

Introduction

1.1 Overview

Predictive modeling is a fundamental task in modern science. Models are built using an available set of training data consisting of input-output pairs, with the goal of minimizing prediction error on future observations that have not yet been realized. However, estimating the prediction error of a model fit to a set of training data using only that training data is challenging, and established methods based on cross-validation and covariance penalties provide only point estimates of prediction error. Any decisions based on these criteria therefore ignore the potentially large uncertainty in the prediction error. Nested Cross-Validation was shown by Bates et al. [4] to provide prediction intervals for the prediction error, mitigating this problem, but their algorithm requires refitting the predictive model a lot of times leading to intense computational burden even in simple problems. In this work, we tackle this problem for penalized regression models and develop a faster way of computing nested cross-validation.

1.2 Contribution

We provide an approach for fast computation of nested cross-validation for penalized regression models. To elaborate, a theoretical analysis of a computation technique we leverage is conducted. This technique is referred as the leave-several-out formula (Lemma 1) in the literature. Then we apply this technique to derive a closed form formula of nested cross-validation. Our formula requires only a single model fit on the whole training data,

whereas the original approach requires refitting the model on subset of the whole data many times. In addition, we implement our formula and compare its performance with the old procedure of refitting the model. Both asymptotic complexity and finite sample performance are considered. At last, we extend our result to ridge regression in ultra high dimension. For applications, we take advantage of our fast computation to examine the performance of nested cross-validation for tuning parameters of functional principal component model.

1.3 Organization

We start Chapter 2 with notations and crucial background knowledge, including 2.1, 2.2, 2.3 and 2.6. Topics include predictive modeling, cross-validation and penalized regression. In section 2.4, we conduct a literature review on past ideas of predictive modeling, cross-validation and uncertainty quantification. Then we introduce nested cross-validation and explain briefly its procedure in section 2.5. Section 2.7 provides the motivation of our work.

Chapter 3 demonstrates main results of our work. Section 3.1 contains a theoretical analysis of the computation technique we use. Section 3.2 shows how we derive a closed form formula for nested cross-validation. In section 3.3, we implement our formula and delve into whether empirically it is faster than the existing method. Our results can be extended to ridge regression in ultra high dimension, which is shown in section 3.4.

Chapter 4 compares nested cross-validation with restricted maximum likelihood and other cross-validation type procedure on tuning parameters. This is an application of our work, as without our contribution, tuning parameters with nested cross-validation would be much more computationally burdensome for penalized regression models.

Chapter 2

Background

2.1 Notations

Let $\mathcal{X} \subseteq \mathbb{R}^p$ and $\mathcal{Y} \subseteq \mathbb{R}$. We observe training data $\mathcal{T} = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ consisting of inputs X_i and outputs Y_i where each (X_i, Y_i) are independently distributed according to a joint distribution $\mathbb{P}_{XY}$ on $\mathcal{X} \times \mathcal{Y}$. When appropriate we collect the samples from $\mathcal{T}$ into $X \in \mathbb{R}^{n \times p}$ and $y \in \mathbb{R}^n$. For any $\mathcal{I} \subseteq \{1, \ldots, n\}$, we let $X_{\mathcal{I}\mathcal{I}}$ and $X_{(-\mathcal{I})(-\mathcal{I})}$ denote the matrix obtained by either including or excluding indices in $\mathcal{I}$ from matrix X, respectively. Note that $X_{\mathcal{I}\mathcal{I}}$ (or $X_{(-\mathcal{I})(-\mathcal{I})}$) is a submatrix by subsetting both rows and columns for indices in (or not in) $\mathcal{I}$. When subsetting only on rows or columns, we denote it as $X_{\mathcal{I}\cdot}$ or $X_{\cdot\mathcal{I}}$ respectively. A similar definition applies for vector y, and we let $\mathcal{T}(\mathcal{I}) = \{(X_i, Y_i)\}_{i \in \mathcal{I}}$. A prediction function is $f : \mathcal{X} \to \mathcal{Y}$ from some space $\mathcal{F}$. A learning algorithm $\mathcal{A}_\lambda(\cdot) : (\mathcal{X} \times \mathcal{Y})^n \to \mathcal{F}$ takes in training data and (optionally) a hyperparameter $\lambda \in \mathbb{R}^d$ and returns a prediction function $\mathcal{A}_\lambda(\mathcal{T}) = \widehat{f}_\lambda \in \mathcal{F}$. We use the common shorthand $y = f(X)$ to indicate the n-dimensional vector $y = (f(X_{1,\cdot}), \ldots, f(X_{n,\cdot}))^T$. Predictions $\widehat{y} = \widehat{f}_\lambda(X)$ returned by $\widehat{f}_\lambda$ will incur error compared to future observed data, and this is measured by a non-negative loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+ \cup \{0\}$ that satisfies $\ell(y, \widehat{y}) = 0$ if and only if $y = \widehat{y}$. In practice, $\mathcal{A}_\lambda$ is run by minimizing the training error (or empirical risk), defined as $\mathcal{R}_{\text{tr}}(\mathcal{A}_\lambda; \mathcal{T}) = \frac{1}{n} \sum_{i=1}^n \ell\big(\widehat{f}_\lambda(x_i), y_i\big)$.

2.2 Predictive modeling

Predictive modeling means the evaluation of a fitted model is based on its performance of predictions on future data. Leo Breiman [10], an influential advocate of predictive modeling, argues for the importance of prediction accuracy as a more flexible criterion to consider a larger class of learning algorithms.

This motivates the notion of generalization, meaning how well the performance of a fitted model generalizes beyond its training data. Generalization error of a learning algorithm and its training set is defined as:

$$\mathcal{R}(\mathcal{A}_\lambda; \mathcal{T}) = \mathbb{E}\left[\ell\left\{\widehat{f}_\lambda(X^*), Y^*\right\} \middle| \mathcal{T}\right].$$

Here $\widehat{f}_\lambda = \mathcal{A}_\lambda(\mathcal{T})$ and $(X^*, Y^*) \sim \mathbb{P}_{XY}$ is a new observation not observed in $\mathcal{T}$ but generated from the same joint distribution. The goal of predictive modeling is to use the observed data $\mathcal{T}$ to find some $\widehat{f}_\lambda \in \mathcal{F}$ that incurs low generalization error. The generalization error is a random variable that is a function of $\mathcal{T}$ and $\mathcal{A}_\lambda$. Note that in the learning theory literature, generalization error is also called risk or true risk. Throughout this work, we use generalization error or prediction error interchangeably. The average generalization error is

$$\mathcal{E}(\mathcal{A}_\lambda) = \mathbb{E}\left\{\mathcal{R}\left(\mathcal{A}_\lambda; \mathcal{T}\right)\right\},$$

and is a fixed, constant number that depends on the learning algorithm $\mathcal{A}_\lambda$. Both $\mathcal{E}(\mathcal{A}_\lambda)$ and $\mathcal{R}(\mathcal{A}_\lambda; \mathcal{T})$ depend on the hyperparameter λ through $\mathcal{A}_\lambda$. The hyperparameter λ, if present, may be chosen by minimizing an estimate of $\mathcal{E}(\mathcal{A}_\lambda)$ or $\mathcal{R}(\mathcal{A}_\lambda; \mathcal{T})$ over λ.

Estimation of $\mathcal{E}\{\mathcal{A}_\lambda\}$ and prediction of $\mathcal{R}(\mathcal{A}_\lambda; \mathcal{T})$ using only $\mathcal{T}$ are both challenging tasks. Hence, on one hand, past attempts of model selection try to minimize the difference between $\mathcal{R}_{\text{tr}}(\mathcal{A}_\lambda; \mathcal{T})$ and $\mathcal{R}(\mathcal{A}_\lambda; \mathcal{T})$. Classic examples include Mallow's C_p [26] and AIC [1]. On the other hand, methods like cross-validation (CV) provide a direct estimate of $\mathcal{E}\{\mathcal{A}_\lambda\}$ or $\mathcal{R}(\mathcal{A}_\lambda; \mathcal{T})$. All of them fall into the spectrum of predictive modeling.

2.3 Cross-validation

Cross-validation is a resampling technique that estimates $\mathcal{E}(\mathcal{A}_\lambda)$ or $\mathcal{R}(\mathcal{A}_\lambda; \mathcal{T})$ by iteratively applying $\mathcal{A}_\lambda$ to subsets of $\mathcal{T}$ and then averaging the resulting estimates. For some $K \in \mathbb{N}$ such that $n/K \in \mathbb{N}$, K-fold cross-validation includes four steps. First, we partition $\mathcal{I} = \{1, \cdots, n\}$ into pairwise disjoint folds, denoted as $\mathcal{I}_1, ..., \mathcal{I}_K$ of $\{1, ..., n\}$. All of them

are of equal size n/K. Second, for each fold $\mathcal{I}_k$, we fit the fold on $(X_i, Y_i)_{i\in\mathcal{I}\setminus\mathcal{I}_k}$ to get $\mathcal{A}_\lambda((X_i, Y_i)_{i\in\mathcal{I}\setminus\mathcal{I}_k}) = \widehat{f}_\lambda^{-k}$. Then we validate $\widehat{f}_\lambda^{-k}$ on $(X_i, Y_i)_{i\in\mathcal{I}_k}$ to get a validation score $(K/n)\sum_{i\in\mathcal{I}_k}\ell(Y_i, \hat{f}_\lambda^{-k}(X_i))$. Finally, we average all validation score to get

$$\widehat{\mathcal{E}}_K^{\mathrm{CV}}(\mathcal{A}_\lambda) = \frac{1}{n}\sum_{i=1}^{n}\ell\left\{Y_i, \widehat{f}_\lambda^{-\mathcal{K}(i)}(X_i)\right\}, \tag{2.1}$$

where $\mathcal{K}(i) \in \{1, \ldots, K\}$ denotes the index of the partition to which y_i belongs. We frequently refer (2.1) as CV error or CV (point) estimate.

2.4 Literature Review

Cross-validation has been recognized as a good method for predictive modeling [14, 2]. This section aims to give an overview of critical ideas on theoretical developments of CV, so that we understand how nested cross-validation (NCV) [4], the methodology we will focus on for this thesis, fits in the literature. There are two goals for the theory of CV. First, we want to provide theoretical justifications of CV error as an estimate for $\mathcal{E}\{\mathcal{A}_\lambda\}$ or $\mathcal{R}(\mathcal{A}_\lambda; \mathcal{T})$. Second, if we have a good understanding of CV point estimate, we want to compute standard errors for uncertainty quantification.

We start with the first goal, which is CV consistency. In other words, when using CV as a model selection criterion, how close is CV from $\mathcal{E}\{\mathcal{A}_\lambda\}$ or $\mathcal{R}(\mathcal{A}_\lambda; \mathcal{T})$. There has been two main directions. The first direction leverages empirical process theory to justify the usage of CV. The idea is to express the CV error with the empirical measure of the data, and then apply uniform concentration inequalities to establish the finite-sample oracle optimality of the CV-selected model [34, 23]. Note that this line of work has connections with other model selection methods such as jackknife and bootstrap, which frequently uses empirical process theory to build up consistency results [33].

The second direction makes assumptions on algorithm stability. Note that stability is a concept that connects a wide range of topics in statistics [36]. Here stability in CV refers to the learning algorithm $\mathcal{A}_\lambda$. Informally, stability assumption aims to give a uniform bound on $\mathcal{A}_\lambda$, meaning if we perturb the training set $\mathcal{T}$ to become $\mathcal{T}'$, we expect $\mathcal{A}_\lambda(\mathcal{T}')$ to not change drastically. Stability assumption was initially proposed by Bousquet and Elisseeff [7] to justify consistency of CV error as an estimate of $\mathcal{R}(\mathcal{A}_\lambda; \mathcal{T})$, and then a line of works follow [21, 22, 11, 8]. Stronger results are given by Austern and Zhou [3] and Bayle et al. [5], in which they both introduce central limit theorem (CLT) for CV estimate with

stability assumptions. Their differences lie in where the CLT is centered. We will come back to their work when it comes to uncertainty quantification. Lei [24] summarizes the theory of CV from the lens of stability. Bates et al. [4] study what CV is estimating and do not rely on stability assumptions. They conclude that under linear model and square loss, CV is estimating $\mathcal{E}(\mathcal{A}_\lambda)$, rather than $\mathcal{R}(\mathcal{A}_\lambda; \mathcal{T})$.

While cross-validation is highly general, imposing no restrictions on $\mathcal{A}_\lambda$ or $\mathbb{P}$, it only returns a point estimate. Uncertainty quantification of CV, specifically a computable prediction interval, has been studied in the literature. Disputes on consistency of CV point estimate carry on to uncertainty quantification. Based on different theories of what CV point estimate is estimating, different standard error estimators are proposed. Bengio and Grandvalet [6] shows that there is no unbiased estimator of CV standard error with a single run of CV. This means that any uncertainty quantification of CV requires more samplings to be done. Later, the first CLT of CV estimate is constructed in [13], but it remains as theoretical exploration and does not offer a computable standard error estimator. Consider the K-fold CV setting defined in section 2.3 and we denote the training set after leaving out fold $\mathcal{I}_k$ as $\mathcal{T}^{-k} = (X_i, Y_i)_{i \in \{1,\cdots,n\} \setminus \mathcal{I}_k}$. Bayle et al. [5] proposes a CLT that is centered on $(1/K) \sum_{k=1}^{K} \mathcal{R}(\mathcal{A}_\lambda; \mathcal{T}^{-k})$, and Austern and Zhou [3] introduces a CLT centered on $\mathcal{E}(\mathcal{A}_\lambda)$. Both provide a computable estimator for the standard error. Bates et al. [4] develop nested cross-validation for a prediction interval centered on $\mathcal{R}(\mathcal{A}_\lambda; \mathcal{T})$, which we will elaborate in section 2.5.

2.5 Nested Cross-validation

Nested cross-validation computes a prediction interval for $\mathcal{R}\{\mathcal{A}_\lambda; \mathcal{T}\}$ by resampling the entire cross-validation procedure. This involves running cross-validation many times, with each run of cross-validation requiring many model fits. The result is uncertainty quantification for an estimate of generalization error $\mathcal{R}\{\mathcal{A}_\lambda; \mathcal{T}\}$, at substantial computational cost.

Fundamental to cross-validation and its nested extension is the notion of a train-test split. Let $\mathcal{I}_{\text{tr}} \cup \mathcal{I}_{\text{te}} = \{1, \ldots, n\}$ and $\mathcal{I}_{\text{tr}} \cap \mathcal{I}_{\text{te}} = \emptyset$. Then let $\mathcal{T}_{\text{tr}} = (X_i, Y_i)_{i \in \mathcal{I}_{\text{tr}}}$ and $\mathcal{T}_{\text{te}} = (X_i, Y_i)_{i \in \mathcal{I}_{\text{te}}}$ be a partition of $\mathcal{T}$ into a training and test set, and let $\widehat{f}_\lambda = \mathcal{A}_\lambda(\mathcal{T}_{\text{tr}})$. Then

$$\widehat{\mathcal{R}}_{\text{split}} = \frac{1}{|\mathcal{T}_{\text{te}}|} \sum_{i \in \mathcal{I}_{\text{te}}} \ell\left(Y_i, \widehat{f}_\lambda(X_i)\right) \tag{2.2}$$

satisfies $\mathbb{E}(\widehat{\mathcal{R}}_{\text{split}} | \mathcal{T}_{\text{tr}}) = \mathcal{R}\{\mathcal{A}_\lambda; \mathcal{T}_{\text{tr}}\}$. Let $\widehat{\mathcal{R}}(\mathcal{T}_{\text{tr}})$ be any estimator of the generalization error

$\mathcal{R}\{\mathcal{A}_\lambda; \mathcal{T}_{\text{tr}}\}$ that depends on $\mathcal{T}_{\text{tr}}$ but not $\mathcal{T}_{\text{te}}$. Nested cross-validation decomposes the mean squared difference between $\widehat{\mathcal{R}}(\mathcal{T}_{\text{tr}})$ and generalization error on the training set,

$$\mathcal{P}(\mathcal{A}_\lambda; \mathcal{T}_{\text{tr}}) = \mathbb{E}\left[\left(\widehat{\mathcal{R}}(\mathcal{T}_{\text{tr}}) - \mathcal{R}\{\mathcal{A}_\lambda; \mathcal{T}_{\text{tr}}\}\right)^2\right],$$

as

$$\mathcal{P}(\mathcal{A}_\lambda; \mathcal{T}_{\text{tr}}) = \mathbb{E}\left[\left(\widehat{\mathcal{R}}(\mathcal{T}_{\text{tr}}) - \widehat{\mathcal{R}}_{\text{split}}\right)^2\right] - \mathbb{E}\left[\left(\widehat{\mathcal{R}}_{\text{split}} - \mathcal{R}\{\mathcal{A}_\lambda; \mathcal{T}_{\text{tr}}\}\right)^2\right]$$

[4, Lemma 1]. The two terms on the right-hand side are estimated by applying cross-validation to further resampled datasets. Partition the original data into $K \in \mathbb{N}$ folds $\mathcal{T}(\mathcal{I}_1), \ldots, \mathcal{T}(\mathcal{I}_K)$ as in cross-validation (Section 2.3). For $j = 1, \ldots, K$, run a $(K-1)$-fold cross-validation on data $\mathcal{T}(\mathcal{I}_{-j}) = \mathcal{T}(\cup_{i \neq j} \mathcal{I}_i)$ and obtain the error estimate $\widehat{e}_j^{\text{CV}} = \widehat{\mathcal{E}}_{K-1}^{\text{CV}}(\mathcal{A}_\lambda)$ via Equation (2.1). For each j, further compute $\widehat{f}_{\lambda,j} = \mathcal{A}_\lambda\{\mathcal{T}(\mathcal{I}_{-j})\}$ which leads to holdout/test error estimates:

$$\begin{aligned} \widehat{e}_j^{\text{te}} &= \frac{1}{|\mathcal{I}_j|} \sum_{i \in \mathcal{I}_j} \ell\left\{Y_i, \widehat{f}_{\lambda,j}(X_i)\right\}, \\ \widehat{v}_j^{\text{te}} &= \frac{1}{|\mathcal{I}_j| - 1} \sum_{i \in \mathcal{I}_j} \left[\ell\left\{Y_i, \widehat{f}_{\lambda,j}(X_i)\right\} - \widehat{e}_j^{\text{te}}\right]^2. \end{aligned} \tag{2.3}$$

Each $\widehat{e}_j^{\text{te}}$ is an unbiased estimate of $\mathcal{R}[\mathcal{A}_\lambda\{\mathcal{T}(\mathcal{I}_{-j})\}; \mathcal{T}(\mathcal{I}_{-j})]$ and $\widehat{v}_j^{\text{te}}$ is an unbiased estimate of the variance of $\widehat{e}_j^{\text{te}}$. Nested cross-validation estimates $\mathcal{R}\{\mathcal{A}_\lambda; \mathcal{T}_{\text{tr}}\}$ by $\widehat{\mathcal{R}}\{\mathcal{A}_\lambda; \mathcal{T}_{\text{tr}}\} = K^{-1}(\widehat{e}_1^{\text{CV}} + \cdots + \widehat{e}_K^{\text{CV}})$ and $\mathcal{P}(\mathcal{A}_\lambda; \mathcal{T})$ by

$$\widehat{\mathcal{P}}(\mathcal{A}_\lambda; \mathcal{T}) = \frac{1}{K} \sum_{j=1}^{K} \left[\{\widehat{e}_j^{\text{CV}} - \widehat{e}_j^{\text{te}}\}^2\right] - \frac{\widehat{v}_j^{\text{te}}}{|\mathcal{I}_j|}. \tag{2.4}$$

Along with an appropriate bias correction term $\widehat{\mathcal{B}}$, the nested cross-validation $(1-\alpha)100\%$ prediction interval for $\mathcal{R}\{\mathcal{A}_\lambda; \mathcal{T}_{\text{tr}}\}$ is therefore

$$\widehat{\mathcal{R}}\{\mathcal{A}_\lambda; \mathcal{T}_{\text{tr}}\} - \widehat{\mathcal{B}} \pm z_{1-\alpha/2} \widehat{\mathcal{P}}(\mathcal{A}_\lambda; \mathcal{T})^{1/2}, \tag{2.5}$$

where z_p is the p-th percentile of the standard Normal distribution.

Bates et al. [4] further suggest that the estimation of $\widehat{\mathcal{P}}(\mathcal{A}_\lambda; \mathcal{T})$ may be improved by repeating the entire nested cross-validation procedure some number $L \in \mathbb{N}$ times and using the average of the $\widehat{\mathcal{P}}(\mathcal{A}_\lambda; \mathcal{T})$ obtained from each run to form the prediction interval (2.5). They suggest $L = 200$ although limited explanation is provided.

2.6 Penalized regression

Penalized regression defines $\mathcal{A}_\lambda(\mathcal{T}) = \widehat{f}_\lambda$ where $\widehat{f}_\lambda(x) = x^T(X^TX + S_\lambda)^{-1}X^Ty$. The matrix $S_\lambda \in \mathbb{R}^p$ is called a penalty matrix and is model-specific. Often, $\lambda \geq 0$ is a scalar and $S_\lambda = \lambda S$ for a fixed matrix S. The vector of predictions on the training set is a linear smooth of the training inputs, $\widehat{f}_\lambda(X) = H(\lambda)y$, with smoothing or "hat" matrix $H(\lambda) = X(X^TX + \lambda S)^{-1}X^T$. This covers ridge regression, nonlinear regression via penalized splines [17], and some functional and signal regression models including functional principle components regression [29, 30]. In chapter 4, we investigate the use of our proposed methods on hyperparameter selection for functional principle component regression models.

The hyperparameter $\lambda \in \mathbb{R}^d$ typically controls fit to the training data versus generalization to new data. Higher values can yield simpler fitted models with worse fit to the training data but a smaller gap between performance on the training versus new data. Lower values can yield complex models with excellent fit to the training data but a large gap between performance on the training versus new data. Choosing a value of λ that balances fit and prediction is therefore a critical step in fitting a penalized regression model. The presence of λ defines an error curve since $\mathcal{R}(\mathcal{A}_\lambda; \mathcal{T})$ and $\mathcal{E}(\mathcal{A}_\lambda)$ now depend on λ. The value of λ must be chosen and one option is to choose it to minimize an estimate of $\mathcal{R}(\mathcal{A}_\lambda; \mathcal{T})$ or $\mathcal{E}(\mathcal{A}_\lambda)$.

2.7 Computational Challenge

Cross-validation requires fitting the model K times and is generally computationally burdensome. Nested cross-validation further exacerbates this computational burden by requiring $K - 1$ additional model fits per fold and then repeating the entire procedure L times. The total complexity after removing some redundant computations is $LK(K+1)/2$ model fits. This represents an extraordinary computational burden, rendering nested cross-validation impractical even in simple models. For example, running 5-fold nested cross-validation using the suggested $L = 200$ requires 3000 model fits. Further, when the goal of predicting generalization error is to do so as a function of the hyperparameter λ and then find the value $\widehat{\lambda}$ that minimizes the estimate, the entire nested cross-validation procedure will have to be repeated for each value of λ that is tried. The goal of this work is to find a faster way of computing NCV under the setting of penalized regression and square loss.

Chapter 3

Main Results

The methods proposed in this work apply to the square loss, $\ell(y, \widehat{y}) = n^{-1} \|y - \widehat{y}\|_2^2$. In addition, we always assume $\lambda > 0$ for the penalized regression model. It is well-known that in penalized regression with training-set predictions $\widehat{y} = \widehat{f}_\lambda(X) = H(\lambda)y$ and square loss, the K-fold cross-validation estimate can be computed using only a single model fit and some matrix algebra:

$$\widehat{\mathcal{E}}_K^{\mathrm{CV}}(\mathcal{A}_\lambda) = \frac{1}{n} \sum_{j=1}^{K} (y_{\mathcal{I}_j} - \widehat{y}_{\mathcal{I}_j})^T (I - H(\lambda)_{\mathcal{I}_j \mathcal{I}_j})^{-2} (y_{\mathcal{I}_j} - \widehat{y}_{\mathcal{I}_j}). \tag{3.1}$$

Eq. (3.1) uses the well-known residual identity (or leave-several out formula) [35, p. 257],

$$y_{\mathcal{I}_j} - \widehat{f}_{\lambda,j}\left(X_{\mathcal{I}_j \cdot}\right) = \left(I - H(\lambda)_{\mathcal{I}_j \mathcal{I}_j}\right)^{-1} \left(y_{\mathcal{I}_j} - \widehat{y}_{\mathcal{I}_j}\right), \tag{3.2}$$

which holds for penalized regression predictions. The case $K = n$ is called leave-one-out cross-validation and has received special attention in the literature. For penalized regression, the one-step formula simplifies considerably:

$$\widehat{\mathcal{E}}_n^{\mathrm{CV}}(\mathcal{A}_\lambda) = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{y_i - x_i^T \widehat{\beta}}{1 - H(\lambda)_{ii}} \right)^2 .$$

The diagonal element $H(\lambda)_{ii}$ is called the influence of y_i on $\widehat{f}_\lambda$. (3.2) provides us with the tool for faster computation of NCV. However, questions remain regarding how it should be applied and its performance. Answering those questions would be the goal of this section.

We outline the structure of this section. First, section 3.1 begins with a detailed theoretical analysis of (3.2), referred as the leave-several-out formula. In particular, we give its proof and analyze its invertibility in section 3.1.1 and section 3.1.2 respectively. Next, in section 3.2, we leverage this technique for computation of NCV. We obtain a closed form formula of NCV requiring only a single model fit. We also show the full derivation of our formula and connect it to 2.5. After that, we analyze the implementation of our formula and its performance in section 3.3. Specifically, both asymptotic complexity and finite sample analysis are considered. At last, we extend our results to ultra high dimensional regression in section 3.4.

3.1 Theory

The residual identity (3.2), also known as the leave-several-out formula, is the core of our computation technique, hence it bears importance to study its theoretical properties before applying it to NCV. First, we will state it explicitly and prove it in section 3.1.1. In the literature, this technique is well-known and the proof is usually omitted. However, we did not find a source that proves it in our setting, a penalized regression with a general penalty matrix $S \in \mathbb{R}^{p\times p}$. Hence, we state it in Lemma 1 and prove it.

Second, we study the invertibility of the matrix $I - H(\lambda)_{\mathcal{I}_k\mathcal{I}_k}$ in section 3.1.2. This problem matters for the validity of Lemma 1. When $H(\lambda)$ is the hat matrix and we obtain its submatrix as $H(\lambda)_{\mathcal{I}_k\mathcal{I}_k}$, it is unclear when $I - H(\lambda)_{\mathcal{I}_k\mathcal{I}_k}$ is invertible. In Theorem 3, we characterize its conditions of invertibility. Note that for Lemma 1, we assume $I - H_{\mathcal{I}\mathcal{I}}$ is always invertible and leave its examination to Theorem 3.

3.1.1 Core Technique

Lemma 1. *We consider a penalized regression setting. Let $H(\lambda) \in \mathbb{R}^{n\times n}$ be the hat matrix, $\mathcal{I} \subseteq \{1, \cdots, n\}$ and $I \in \mathbb{R}^{|\mathcal{I}|\times|\mathcal{I}|}$ be the identity matrix. We follow the same submatrix and vector notations given in section 2.1. Let $\hat{y}$ be the predicted y from the full model fit. In particular, $\hat{y} = H(\lambda)y$. Let $\hat{y}^{-\mathcal{I}}$ be the predicted y by fitting the penalized regression on data that are not in $\mathcal{I}$, namely $(X_{-\mathcal{I}.}, y_{-\mathcal{I}})$. Then*

$$\left(I - H(\lambda)_{\mathcal{I}\mathcal{I}}\right)^{-1}\left(y_{\mathcal{I}} - \hat{y}_{\mathcal{I}}\right) = y_{\mathcal{I}} - \hat{y}_{\mathcal{I}}^{-\mathcal{I}}. \tag{3.3}$$

Proof. We adapt a proof from [35], [18] and [15]. When fitting a penalized regression model

on $(X_{-\mathcal{I}\cdot}, y_{-\mathcal{I}})$, we want to solve the optimization problem as below

$$\min_{\beta} \quad (y_{-\mathcal{I}} - X_{-\mathcal{I}\cdot}\beta)^T (y_{-\mathcal{I}} - X_{-\mathcal{I}\cdot}\beta) + \lambda\beta^T S\beta, \tag{3.4}$$

where $\hat{\beta}_\lambda^{-\mathcal{I}} = (X_{-\mathcal{I}\cdot}^T X_{-\mathcal{I}\cdot} + \lambda S)^{-1} X_{-\mathcal{I}\cdot}^T y_{-\mathcal{I}}$ is the fitted coefficient of penalized regression on $(X_{-\mathcal{I}\cdot}, y_{-\mathcal{I}})$. Now let $\hat{y}_{\mathcal{I}}^{-\mathcal{I}} = X_{\mathcal{I}\cdot}\hat{\beta}_\lambda^{-\mathcal{I}}$. If we change (3.4) to

$$\min_{\beta} \quad (y_{-\mathcal{I}} - X_{-\mathcal{I}\cdot}\beta)^T (y_{-\mathcal{I}} - X_{-\mathcal{I}\cdot}\beta) + \left(\hat{y}_{\mathcal{I}}^{-\mathcal{I}} - X_{\mathcal{I}\cdot}\hat{\beta}_\lambda^{-\mathcal{I}}\right)^T \left(\hat{y}_{\mathcal{I}}^{-\mathcal{I}} - X_{\mathcal{I}\cdot}\hat{\beta}_\lambda^{-\mathcal{I}}\right) + \lambda\beta^T S\beta, \tag{3.5}$$

the solution would still be the same. Observe that (3.5) is equivalent to fitting a penalized regression on X and $(\hat{y}_{\mathcal{I}}^{-\mathcal{I}}, y_{-\mathcal{I}})$. Since we fit it on X, the hat matrix is $H(\lambda) = X(X^TX + \lambda S)^{-1}X^T$. Let $\hat{y} = H(\lambda)y$, the predicted y for fitting the model on (X, y). Observe that by (3.5) and the definition of $\hat{y}_{\mathcal{I}}^{-\mathcal{I}}$, we can look at the entries in $\mathcal{I}$ of the predicted values and then get

$$H(\lambda)_{\mathcal{I}\cdot} \begin{pmatrix} \hat{y}_{\mathcal{I}}^{-\mathcal{I}} \\ y_{-\mathcal{I}} \end{pmatrix} = H(\lambda)_{\mathcal{I}\mathcal{I}}\hat{y}_{\mathcal{I}}^{-\mathcal{I}} + H(\lambda)_{\mathcal{I}(-\mathcal{I})}y_{-\mathcal{I}} = \hat{y}_{\mathcal{I}}^{-\mathcal{I}}.$$

It follows that

$$\begin{aligned}
& H(\lambda)_{\mathcal{I}\mathcal{I}}\hat{y}_{\mathcal{I}}^{-\mathcal{I}} + H(\lambda)_{\mathcal{I}\mathcal{I}}y_{\mathcal{I}} + H(\lambda)_{\mathcal{I}(-\mathcal{I})}y_{-\mathcal{I}} - H(\lambda)_{\mathcal{I}\mathcal{I}}y_{\mathcal{I}} = \hat{y}_{\mathcal{I}}^{-\mathcal{I}} \\
\Rightarrow \quad & \hat{y}_{\mathcal{I}} + H(\lambda)_{\mathcal{I}\mathcal{I}}\hat{y}_{\mathcal{I}}^{-\mathcal{I}} - H(\lambda)_{\mathcal{I}\mathcal{I}}y_{\mathcal{I}} = \hat{y}_{\mathcal{I}}^{-\mathcal{I}} \\
\Rightarrow \quad & -\hat{y}_{\mathcal{I}}^{-\mathcal{I}} - H(\lambda)_{\mathcal{I}\mathcal{I}}\left(y_{\mathcal{I}} - \hat{y}_{\mathcal{I}}^{-\mathcal{I}}\right) = -\hat{y}_{\mathcal{I}} \\
\Rightarrow \quad & y_{\mathcal{I}} - \hat{y}_{\mathcal{I}}^{-\mathcal{I}} - H(\lambda)_{\mathcal{I}\mathcal{I}}\left(y_{\mathcal{I}} - \hat{y}_{\mathcal{I}}^{-\mathcal{I}}\right) = y_{\mathcal{I}} - \hat{y}_{\mathcal{I}} \\
\Rightarrow \quad & \left(I - H(\lambda)_{\mathcal{I}\mathcal{I}}\right)\left(y_{\mathcal{I}} - \hat{y}_{\mathcal{I}}^{-\mathcal{I}}\right) = y_{\mathcal{I}} - \hat{y}_{\mathcal{I}} \\
\Rightarrow \quad & y_{\mathcal{I}} - \hat{y}_{\mathcal{I}}^{-\mathcal{I}} = \left(I - H(\lambda)_{\mathcal{I}\mathcal{I}}\right)^{-1}\left(y_{\mathcal{I}} - \hat{y}_{\mathcal{I}}\right)
\end{aligned}$$

as desired. □

3.1.2 Invertibility

Lemma 1 has been used in the literature [35, p. 257], while it is never clear when $I - H(\lambda)_{\mathcal{I}\mathcal{I}}$ is invertible. In Theorem 3, we show that when $I - H(\lambda)_{\mathcal{I}\mathcal{I}}$ is always invertible, and when it is not, we characterize its invertibility with a necessary and sufficient condition. Before we state the result, we mention several facts about $S \in \mathbb{R}^{p\times p}$ that will be useful later. Note that under the penalized regression setting, the penalty matrix S is at least positive semidefinite. Further, since S is symmetric, spectral decomposition gives us $S = U\Lambda U^T$

where U is an orthogonal matrix formed by eigenvectors. Let $D \subseteq \{1, ..., p\}$ be the set of indices that S has strictly positive eigenvalues. In particular, we can write Λ as

$$\Lambda = \begin{pmatrix} 0_{(p-|D|)\times(p-|D|)} & 0 \\ 0 & \Lambda_D \end{pmatrix}$$

such that $\Lambda_D \succ 0$. It follows that

$$S = \begin{pmatrix} U_{\cdot(-D)} & U_{\cdot D} \end{pmatrix} \begin{pmatrix} 0_{(p-|D|)\times(p-|D|)} & 0 \\ 0 & \Lambda_D \end{pmatrix} \begin{pmatrix} U^T_{\cdot(-D)} \\ U^T_{\cdot D} \end{pmatrix} = \begin{pmatrix} 0_{(p-|D|)\times(p-|D|)} & 0 \\ 0 & S_D \end{pmatrix} \tag{3.6}$$

where $S_D = U_{\cdot D}\Lambda_D U^T_{\cdot D} \in \mathbb{R}^{|D|\times|D|}$.

We use ν to denote eigenvalues in general. For any $n \times n$ symmetric matrix A, we write its eigenvalues in increasing order as $\nu_1(A) \leq \nu_2(A) \leq \cdots \leq \nu_n(A)$. For the sake of completeness, we state two theorems that will be useful for our proof.

Theorem 1 (Cauchy Interlacing)**.** *[20, p. 242] Let $B \in \mathbb{R}^{n\times n}$ be symmetric, let $y \in \mathbb{R}^n$ and $a \in \mathbb{R}$ be given, and let*

$$A = \begin{pmatrix} B & y \\ y^T & c \end{pmatrix} \in \mathbb{R}^{(n+1)\times(n+1)}.$$

Then

$$\nu_1(A) \leq \nu_1(B) \leq \nu_2(A) \leq \cdots \leq \nu_n(A) \leq \nu_n(B) \leq \nu_{n+1}(A).$$

Remark 1. *We will use Theorem 1 by letting $B = I - H(\lambda)_{\mathcal{I}\mathcal{I}}$. Note that if $B = I - H(\lambda)_{\mathcal{I}\mathcal{I}}$ and we keep expanding dimension like in Theorem 1, we will get $P(I - H(\lambda))P^T$ where $P \in \mathbb{R}^{n\times n}$ is some permutation matrix. A useful fact is that $I - H(\lambda)$ and $P(I - H(\lambda))P^T$ share the same eigenvalues [20, p. 58]. Then depending on the dimension difference of $n - |\mathcal{I}|$, we may need to apply Theorem 1 iteratively to get a lower bound on the eigenvalue of $I - H(\lambda)_{\mathcal{I}\mathcal{I}}$.*

The second one is adapted from [20], which is useful for analyzing the eigenvalues of $H(\lambda)$.

Theorem 2. *[20, p. 65] Suppose $A \in \mathbb{R}^{p\times n}$ and $B \in \mathbb{R}^{n\times p}$ with $p \leq n$. Then the n eigenvalues of BA are the p eigenvalues of AB with $n - p$ zeroes.*

Now we state our main result and provide its proof.

Theorem 3. *Suppose* $\lambda > 0$, $X \in \mathbb{R}^{n\times p}$ *with* $\text{rank}(X) = p$ *and* $H(\lambda) = X(X^TX + \lambda S)^{-1}X^T \in \mathbb{R}^{n\times n}$. *We carry the submatrix notations given in section 2.1. When the penalty matrix* S *is positive definite,* $I - H(\lambda)_{\mathcal{I}\mathcal{I}}$ *is always invertible. When* S *is positive semidefinite but not positive definite,* $I - H(\lambda)_{\mathcal{I}\mathcal{I}}$ *is invertible if and only if* $\text{Null}\left(X_{(-\mathcal{I})(-D)}\right) = \{0\}$.

Proof. We first assume $S \succ 0$. Since $\text{rank}(X) = p$ and $S \succ 0$, $X^TX + \lambda S$ is invertible. Since $X^TX \succeq 0$, $S \succ 0$ and $\lambda > 0$, $X^TX + \lambda S \succ 0$, hence $(X^TX + \lambda S)^{-1} \succ 0$. Let $y \in \mathbb{R}^n$. Notice that

$$y^TH(\lambda)y = y^TX\left(X^TX + \lambda S\right)^{-1}X^Ty = (X^Ty)^T\left(X^TX + \lambda S\right)^{-1}(X^Ty) \geq 0.$$

where the last inequality can be seen through a Cholesky decomposition of positive definite $(X^TX + \lambda S)^{-1}$. Note that the last inequality has to be $\geq$ because we can choose $y \in \text{Null}(X^T)$ and if $\text{rank}(X) = p < n$, the existence of non-trivial $y \in \text{Null}(X^T)$ is guaranteed. Thus, $H(\lambda) \succeq 0$. It follows that for any eigenvalue of $H(\lambda)$, denoted as $\nu_i(H(\lambda))$ for $i = 1, \cdots, n$, $\nu_i(H(\lambda)) \geq 0$. Next, we want to upper bound $\nu_i(H(\lambda))$. By Theorem 2, it is equivalent to consider $H(\lambda)' = X^TX(X^TX + \lambda S)^{-1}$. By what we already proved, $\nu \geq 0$. Let $a \in \mathbb{R}^p$ be an eigenvector of $H(\lambda)'$ with its corresponding eigenvalue ν. Then we have

$$H(\lambda)'a = X^TX(X^TX + \lambda S)^{-1}a = \nu a. \tag{3.7}$$

We define $u = (X^TX + \lambda S)^{-1}a$. Note that $u \neq 0$ since a is an eigenvector and $(X^TX + \lambda S)^{-1} \succ 0$. Then (3.7) becomes

$$(X^TX)u = \nu(X^TX + \lambda S)u. \tag{3.8}$$

By multiplying both sides of (3.8) with u, we reach

$$u^T(X^TX)u = \nu u^T(X^TX + \lambda S)u = \nu u^T(X^TX)u + \nu u^T(\lambda S)u.$$

Rearrange and we get

$$(1 - \nu)u^T(X^TX)u = \nu u^T(\lambda S)u. \tag{3.9}$$

Note that obviously $X^TX \succeq 0$. Since $\text{rank}(X^TX) = p$ and $X^TX \in \mathbb{R}^p$, $X^TX \succ 0$. Also, $u^T(\lambda S)u > 0$ by $\lambda S \succ 0$. Recall $\nu \geq 0$. If $1 - \nu \leq 0$, $(1 - \nu)u^T(X^TX)u \leq 0$ and $\nu u^T(\lambda S)u \geq 0$. We must have $(1 - \nu)u^T(X^TX)u = \nu u^T(\lambda S)u = 0$. Since $u^T(\lambda S)u > 0$, we must have $\nu = 0$. Since $u^TX^TXu > 0$, we get $1 - \nu = 0$, thus $\nu = 1 \neq 0$. Then we have a contradiction. Thus, we must have $1 - \nu > 0$ and then $\nu < 1$. We conclude that $0 \leq \nu_i(H(\lambda)) < 1$ for $i = 1, \cdots, n$. It follows that $0 < \nu_i(I_n - H(\lambda)) \leq 1$ for $i = 1, \cdots, n$. With iteratively applying Theorem 1, this gives us

$$0 < \nu_n(I_n - H(\lambda)) \leq \nu_{|\mathcal{I}|}(I - H(\lambda)_{\mathcal{I}\mathcal{I}}).$$

Hence, $I - H(\lambda)_{\mathcal{I}\mathcal{I}} \succ 0$ and it must be invertible.

Next, we assume $S \succeq 0$ but not $S \succ 0$. We want to characterize its condition of invertibility. We start by noting a result from fitting penalized regression, which is

$$\widehat{\beta} = \arg\min_{\beta} \left\{ \|y - X\beta\|_2^2 + \lambda \beta^T S \beta \right\} = \left(X^T X + \lambda S \right)^{-1} X^T y.$$

It follows that

$$\begin{aligned}
\min_{\beta} \left\{ \|y - X\beta\|_2^2 + \lambda \beta^T S \beta \right\} &= \left\| y - X\widehat{\beta} \right\|_2^2 + \lambda \widehat{\beta}^T S \widehat{\beta} \\
&= y^T y - 2 y^T X \widehat{\beta} + \widehat{\beta}^T X^T X \widehat{\beta} + \lambda \widehat{\beta}^T S \widehat{\beta} \\
&= y^T y - 2 y^T X \widehat{\beta} + \widehat{\beta}^T \left(X^T X + \lambda S \right) \widehat{\beta}. \\
&= y^T y - 2 y^T X \widehat{\beta} + \widehat{\beta}^T \left(X^T X + \lambda S \right) \left(X^T X + \lambda S \right)^{-1} X^T y \\
&= y^T y - 2 y^T X \widehat{\beta} + \widehat{\beta}^T X^T y \\
&= y^T y - y^T X \widehat{\beta} \\
&= y^T \left(y - X\widehat{\beta} \right) \\
&= y^T (I_n - H(\lambda)) y.
\end{aligned} \tag{3.10}$$

Note that (3.10) holds for any data vector $y \in \mathbb{R}^n$, hence it enables us to study $y^T(I_n - H(\lambda))y$ in the lens of an optimization problem.

Now we return to our if and only if statement. We start with the backwards direction, and we proceed by proving its contrapositive. That means we assume $I - H(\lambda)_{\mathcal{I}\mathcal{I}}$ is not invertible, and we want to show $\text{Null}(X_{(-\mathcal{I})(-D)}) \neq \{0\}$. Note that if $S \succeq 0$, we can return to (3.9) to look at the upper bound of ν. $\text{rank}(X) = p$ still gives $u^T(X^T X)u > 0$. If $1 - \nu < 0$, the LHS of (3.9) is negative, while the RHS is non-negative, hence a contradiction. Thus, $\nu \leq 1$. Then $I_n - H(\lambda) \succeq 0$. By iteratively applying Theorem 1 again, $I - H(\lambda)_{\mathcal{I}\mathcal{I}} \succeq 0$. Since $I - H(\lambda)_{\mathcal{I}\mathcal{I}}$ is not invertible, there must exist an eigenvector $a \in \mathbb{R}^{|\mathcal{I}|}$ such that $a^T(I - H(\lambda)_{\mathcal{I}\mathcal{I}})a = 0$. Suppose $\mathcal{I} = \{i_1, \cdots, i_{|\mathcal{I}|}\}$ with $i_1 \leq i_2 \leq \cdots \leq i_{|\mathcal{I}|}$. We construct a vector $a' \in \mathbb{R}^n$ such that

$$a'_j = \begin{cases} a_m, & j \in \mathcal{I}, j = i_m, \\ 0, & j \notin \mathcal{I}. \end{cases} \tag{3.11}$$

This gives us $a^T(I - H(\lambda)_{\mathcal{I}\mathcal{I}})a = (a')^T(I_n - H(\lambda))a'$. By (3.10), we have

$$\begin{aligned} 0 = a^T(I - H(\lambda)_{\mathcal{I}\mathcal{I}})a = (a')^T(I_n - H(\lambda))a' &= \min_{\beta} \left\{ \|a' - X\beta\|_2^2 + \lambda\beta^T S\beta \right\} \\ &= \min_{\beta} \left\{ \|a' - X\beta\|_2^2 + \lambda\beta_D^T S_D \beta_D \right\}. \end{aligned} \tag{3.12}$$

where the last equality follows from (3.6). Note that $S_D \succ 0$ from our discussion on (3.6). Since the optimization in (3.12) attains 0, we must have $\beta_D = 0$ and $a' = X\beta$. Note that $X\beta = X_{\cdot(-D)}\beta_{-D} + X_{\cdot D}\beta_D$. Then we get $a' = X_{\cdot(-D)}\beta_{-D}$. Recall the construction of a' in (3.11), we have $a = X_{\mathcal{I}(-D)}\beta_{-D}$ and $0 = X_{(-\mathcal{I})(-D)}\beta_{-D}$. Since a is an eigenvector, $a \neq 0$ by definition. Then $\beta_{-D} \neq 0$. Hence, $\text{Null}(X_{(-\mathcal{I})(-D)}) \neq \{0\}$ as desired.

We are left with the forward direction. Like the backward direction, we proceed with a proof by contrapositive. Assume $\text{Null}(X_{(-\mathcal{I})(-D)}) \neq \{0\}$ and we want to show $I - H(\lambda)_{\mathcal{I}\mathcal{I}}$ is not invertible. Since $I - H_{\mathcal{I}\mathcal{I}} \succeq 0$ from middle steps of the forward direction, it is equivalent to construct an eigenvector a such that $a^T(I - H(\lambda)_{\mathcal{I}\mathcal{I}})a = 0$. Let $0 \neq \beta_{-D} \in \text{Null}\left(X_{(-\mathcal{I})(-D)}\right)$. We construct a as $a = X_{\mathcal{I}(-D)}\beta_{-D}$. We need to justify $a \neq 0$ so that it is a valid eigenvector. If $a = 0$, then $\beta_{-D} \in \text{Null}\left(X_{\mathcal{I}(-D)}\right)$. It follows that $\beta_{-D} \in \text{Null}\left(X_{\cdot(-D)}\right)$. Since β_{-D} is non-trivial, we know $\dim\left(\text{Null}\left(X_{\cdot(-D)}\right)\right) > 0$. However, since $\text{rank}(X) = p$, we have $\text{rank}\left(X_{\cdot(-D)}\right) = p - |D|$. By the Rank-Nullity theorem, we must have $\dim\left(\text{Null}\left(X_{\cdot(-D)}\right)\right) = 0$. Hence, we have a contradiction and $a \neq 0$. Next, we use again the construction of a' in (3.11) and (3.10) to consider

$$a^T(I - H(\lambda)_{\mathcal{I}\mathcal{I}})a = (a')^T(I_n - H(\lambda))a' = \min_{\beta} \left\{ \|a' - X\beta\|_2^2 + \lambda\beta_D^T S_D \beta_D \right\}. \tag{3.13}$$

If we can argue the minimum value of the optimization in (3.13) attains 0, we are done. Let

$$\beta = \begin{pmatrix} \beta_{-D} \\ 0_{|D|} \end{pmatrix}.$$

Then $a' - X\beta = 0$ by construction of a and a', and $\lambda\beta_D^T S_D \beta_D = 0$ by construction of β. Since $\|a' - X\beta\|_2^2 \geq 0$ and $\lambda\beta_D^T S_D \beta_D \geq 0$, we know our constructed β attains the minimum value of the optimization in (3.13), which is 0. By (3.13) again, $a^T(I - H(\lambda)_{\mathcal{I}\mathcal{I}})a = 0$. Thus, $I - H(\lambda)_{\mathcal{I}\mathcal{I}}$ is not invertible. The forward direction is also completed. □

Remark 2. *Theorem 3 can be seen as a checking criterion for invertibility of $I - H(\lambda)_{\mathcal{I}\mathcal{I}}$. In a K-fold CV setting, we can first check whether any fold $\mathcal{I}_k$ violates the invertibility condition. If it does, we can resample K folds. Empirically, $I - H(\lambda)_{\mathcal{I}\mathcal{I}}$ tends to be invertible with high probability. Proving it would be of interest for future work.*

3.2 Computation

After studying validity of Lemma 1 in section 3.1, we apply the technique to NCV. From now on, we assume matrix form of $I - H_{\mathcal{I}\mathcal{I}}$ is always invertible, hence positive definite from proof of Theorem 3.

Our main contribution is to provide a closed-form formula of computing NCV for penalized regression models. We give the formula in Proposition 1. Let $M_{n,K} \in \mathbb{R}^{\frac{2n}{K} \times \frac{2n}{K}}$ be the matrix obtained by setting the first n/K diagonal entries of the identity matrix of dimension $2n/K$ to be zero.

Proposition 1. *Fix $L, L_B \in \mathbb{N}$. For $l = 1, \cdots, L$, we denote the sampled folds as $\mathcal{I}_1^l, \cdots, \mathcal{I}_K^l$. Similarly, we denote folds $\mathcal{I}_1^{l_b}, \cdots, \mathcal{I}_K^{l_b}$ for $l_b = 1, \cdots, L_B$. The nested cross-validation point estimate of prediction error is*

$$\widehat{\mathcal{R}}\{\mathcal{A}_\lambda; \mathcal{T}_{tr}\} = \frac{1}{Ln(K-1)} \sum_{l=1}^{L} \sum_{(k,j),k<j} \left(y_{I_{jk}^l} - \widehat{y}_{\mathcal{I}_{jk}^l}\right)^T \left(I - H(\lambda)_{\mathcal{I}_{jk}^l \mathcal{I}_{jk}^l}\right)^{-2} \left(y_{I_{jk}^l} - \widehat{y}_{\mathcal{I}_{jk}^l}\right), \tag{3.14}$$

and the bias correction term is

$$\begin{aligned} \widehat{\mathcal{B}} = &\left(1 + \frac{K-2}{K}\right) \\ &\left(\widehat{\mathcal{R}}\{\mathcal{A}_\lambda; \mathcal{T}_{tr}\} - \frac{1}{nL_B} \sum_{l_b=1}^{L_B} \sum_{j=1}^{K} \left(y_{\mathcal{I}_j^{l_b}} - \widehat{y}_{\mathcal{I}_j^{l_b}}\right)^T \left(I - H(\lambda)_{\mathcal{I}_j^{l_b} \mathcal{I}_j^{l_b}}\right)^{-2} \left(y_{\mathcal{I}_j^{l_b}} - \widehat{y}_{\mathcal{I}_j^{l_b}}\right)\right). \end{aligned} \tag{3.15}$$

The nested cross-validation estimate of the mean-squared error is

$$\begin{aligned} \widehat{\mathcal{P}}(\mathcal{A}_\lambda; \mathcal{T}) = &\frac{1}{KL} \sum_{l=1}^{L} \sum_{j=1}^{K} \Bigg\{ \\ &\left[\frac{K}{n(K-1)} \sum_{k \neq j} \left(y_{\mathcal{I}_{jk}^l} - \widehat{y}_{\mathcal{I}_{jk}^l}\right)^T \left(I - H(\lambda)_{\mathcal{I}_{jk}^l \mathcal{I}_{jk}^l}\right)^{-1} M_{n,K} \left(I - H(\lambda)_{\mathcal{I}_{jk}^l \mathcal{I}_{jk}^l}\right)^{-1} \left(y_{\mathcal{I}_{jk}^l} - \widehat{y}_{\mathcal{I}_{jk}^l}\right) \right. \\ &\left. - \frac{K}{n} \left(y_{\mathcal{I}_j^l} - \widehat{y}_{\mathcal{I}_j^l}\right)^T \left(I - H(\lambda)_{\mathcal{I}_j^l \mathcal{I}_j^l}\right)^{-2} \left(y_{\mathcal{I}_j^l} - \widehat{y}_{\mathcal{I}_j^l}\right)\right]^2 - \\ &\frac{\sum_v \left(\left(\left(I - H(\lambda)_{\mathcal{I}_j^l \mathcal{I}_j^l}\right)^{-1} \left(y_{\mathcal{I}_j^l} - \widehat{y}_{\mathcal{I}_j^l}\right)\right)_v^2 - \frac{1}{|\mathcal{I}_j^l|} \sum_u \left(\left(I - H(\lambda)_{\mathcal{I}_j^l \mathcal{I}_j^l}\right)^{-1} \left(y_{\mathcal{I}_j^l} - \widehat{y}_{\mathcal{I}_j^l}\right)\right)_u^2\right)^2}{(|\mathcal{I}_j^l| - 1)|\mathcal{I}_j^l|} \Bigg\}, \end{aligned} \tag{3.16}$$

The formulas in Proposition 1 compute the entire prediction interval (2.5) using only a single model fit to the whole data $\mathcal{T}$. An investigation of the computational complexity of this formula and the associated implementation details are given in Section 3.3.

Proof of Proposition 1. By looking at (3.14) and (3.16), it is sufficient to derive our formula for any l and then average the results for all $l = 1, \cdots, L$. We apply Eq. (3.2) to fold $\mathcal{I}_k^l$ and pair of folds $\mathcal{I}_k^l \cup \mathcal{I}_j^l$ with $k \neq j$ to obtain

$$\widehat{r}_j^{\text{te}} \equiv y_{\mathcal{I}_j^l} - \widehat{f}_{\lambda,j}\left(X_{\mathcal{I}_j^l \cdot}\right) = \left(I - H(\lambda)_{\mathcal{I}_j^l \mathcal{I}_j^l}\right)^{-1}\left(y_{\mathcal{I}_j^l} - \widehat{y}_{\mathcal{I}_j^l}\right),$$
$$\widehat{r}_{jk}^{\text{te}} = y_{\mathcal{I}_j^l \cup \mathcal{I}_k^l} - \widehat{f}_{\lambda,jk}\left(X_{\mathcal{I}_j^l \cup \mathcal{I}_k^l \cdot}\right) = \left(I - H(\lambda)_{\mathcal{I}_{jk}^l \mathcal{I}_{jk}^l}\right)^{-1}\left(y_{\mathcal{I}_{jk}^l} - \widehat{y}_{\mathcal{I}_{jk}^l}\right).$$

For each $j \in \{1, \ldots, K\}$, by (2.3) we have

$$\widehat{e}_j^{\text{te}} = \frac{K}{n}\left(y_{\mathcal{I}_j} - \widehat{y}_{\mathcal{I}_j}\right)\left(I - H(\lambda)_{\mathcal{I}_j \mathcal{I}_j}\right)^{-2}\left(y_{\mathcal{I}_j} - \widehat{y}_{\mathcal{I}_j}\right). \tag{3.17}$$

Recall in Section 2.5, $\widehat{e}_j^{\text{CV}}$ is the result of performing a $(K-1)$-fold CV on data $\mathcal{T}(\cup_{i \neq j} \mathcal{I}_i)$. Based on the CV procedure, for $k \in \{1, \ldots, j-1, j+1, \ldots, K\}$, we fit the model on data $\mathcal{T}(\cup_{i \notin \{j,k\}} \mathcal{I}_i)$ to get $\widehat{f}_{\lambda,jk} = \mathcal{A}_\lambda(\mathcal{T}(-\mathcal{I}_j \cup \mathcal{I}_k))$ and use it to predict $y_{\mathcal{I}_k}$. However, since we run a CV procedure to get $\widehat{e}_j^{\text{CV}}$, we need to aggregate errors corresponding to only fold $\mathcal{I}_k$ but not $\mathcal{I}_j$, thus we define

$$\widehat{r}_{jk'}^{\text{te}} = M_{n,K}\widehat{r}_{jk}^{\text{te}}$$

where $M_{n,K} \in \mathbb{R}^{\frac{2n}{K} \times \frac{2n}{K}}$ is a matrix modified from an identity matrix of dimension $2n/K$ by replacing the first n/K diagonals with 0. Thus, M extracts the last n/K entries of a dimension $2n/K$ vector. Note that $M_{n,K}^2 = M_{n,K}$ and when implementing the formula on a computer, the mapping of $M_{n,K}$ can be replaced by directly subsetting entries of an array. It follows that

$$\begin{aligned}\widehat{e}_j^{\text{CV}} &= \frac{K}{n(K-1)}\sum_{k \neq j}\left(\widehat{r}_{jk'}^{\text{te}}\right)^T \widehat{r}_{jk'}^{\text{te}} \\ &= \frac{K}{n(K-1)}\sum_{k \neq j}\left(y_{I_{jk}^l} - \widehat{y}_{\mathcal{I}_{jk}^l}\right)^T \left(I - H(\lambda)_{\mathcal{I}_{jk}^l \mathcal{I}_{jk}^l}\right)^{-1} M_{n,K}\left(I - H(\lambda)_{\mathcal{I}_{jk}^l \mathcal{I}_{jk}^l}\right)^{-1}\left(y_{I_{jk}^l} - \widehat{y}_{\mathcal{I}_{jk}^l}\right).\end{aligned} \tag{3.18}$$

Further, notice that in (2.3), $\widehat{v}_j^{\text{te}}$ can be seen as the variance of $\widehat{r}_j^{\text{te}}$. It follows that by the definition of variance and combining (3.17),(3.18) and (2.4), we get (3.16). Next, recall

in section 2.5, $\widehat{\mathcal{R}}\{\mathcal{A}_\lambda; \mathcal{T}_{\text{tr}}\}$ is the average of all of $\widehat{e}_j^{\text{CV}}$ for $j \in \{1, \ldots, K\}$. Notice that the summation of $\widehat{e}_1^{\text{CV}} + \ldots + \widehat{e}_K^{\text{CV}}$ is equivalent to summing $\widehat{r}_{jk}^{\text{te}}$ for all $k < j$. Then (3.14) quickly follows. Now we are left to justify the bias correction term (3.15). From previous NCV implementation by [4], $\widehat{\mathcal{B}}$ requires an additional L_b times run of K-fold CV and aggregate all the CV errors. For $l_b \in \{1, \cdots, L_b\}$, we denotes its CV folds as $\mathcal{I}_1^{l_b}, \cdots, \mathcal{I}_K^{l_b}$. Note that the sum of square errors for one fold is $(y_{\mathcal{I}_j^{l_b}} - \widehat{y}_{\mathcal{I}_j^{l_b}})^T (I - H(\lambda)_{\mathcal{I}_j^{l_b}\mathcal{I}_j^{l_b}})^{-2} (y_{\mathcal{I}_j^{l_b}} - \widehat{y}_{\mathcal{I}_j^{l_b}})$. Then we aggregate it for all K folds, and further aggregate for all L_B times of running CV. This gives us the difference term taken with respect to $\widehat{\mathcal{R}}\{\mathcal{A}_\lambda; \mathcal{T}_{\text{tr}}\}$ in (3.15). The $1 + (K-2)/K$ scaling in (3.15) follows directly from Appendix C [4]. (3.16), (3.14) and (3.15) together give a closed-form formula of (2.5) as desired. □

3.3 Implementation and Complexity

We give full details on the implementation and complexity of our formula in Proposition 1. We consider the setting $n \geq p$. The main computation hurdle lies in

$$\widehat{r}_{jk}^{\text{te}} = \left(I - H(\lambda)_{\mathcal{I}_{jk}\mathcal{I}_{jk}}\right)^{-1} \left(y_{\mathcal{I}_{jk}} - \widehat{y}_{\mathcal{I}_{jk}}\right). \tag{3.19}$$

(3.19) highlights the main difference between NCV and CV, because usual K-fold CV needs to compute (3.19) K times, while NCV needs to compute it for $(LK^2 + LK)/2$ times. If we choose to refit the model, computing $\widehat{r}_{jk}^{\text{te}}$ costs $O(np^2 + p^3)$, which is the usual complexity of fitting a penalized regression. Hence, the complexity of refitting the model is $O(LK^2np^2 + LK^2p^3)$. Next, we give a detailed comparison of two ways of computing (3.19). One uses refitting the model and one implements our formula. Note that floating point operations (FLOP) counting is a common way to evaluate the number of matrix operations in an algorithm. For background on numerical linear algebra, please refer to Appendix A.

3.3.1 Proposed Formula

We first derive a mathematical expression of (3.19) for the sake of implementation and then give complexity and the number of FLOPs for all steps. Observe that by the form of $H(\lambda)$ in penalized regression, we have

$$\left(I - H(\lambda)_{\mathcal{I}_{jk}\mathcal{I}_{jk}}\right)^{-1} \left(y_{\mathcal{I}_{jk}} - \widehat{y}_{\mathcal{I}_{jk}}\right) = \left(I - X_{\mathcal{I}_{jk}\cdot} \left(X^TX + \lambda S\right)^{-1} X_{\mathcal{I}_{jk}\cdot}^T\right)^{-1} \left(y_{\mathcal{I}_{jk}} - \widehat{y}_{\mathcal{I}_{jk}}\right)$$

It follows that we can compute $\left(X^TX+\lambda S\right)^{-1}$ once and store this $p\times p$ matrix. Next, we outline the FLOPs and complexity for in each step shown in Table 3.1. Note that the last line of the table follows from the need to compute $\widehat{r}^{\text{te}}_{jk}$ for $O(LK^2)$ times

Table 3.1: Step-wise computational complexity and FLOP count for computing $\widehat{r}^{\text{te}}_{jk}$ with proposed formula. Total FLOP count and complexity are also reported.

Computation Step	**FLOP Count**	**Complexity**
$\left(X^TX+\lambda S\right)^{-1}X^T_{\mathcal{I}_{jk}\cdot}$	$2p^2(2n/K)$	$O(p^2(n/K))$
$X_{\mathcal{I}_{jk}\cdot}\left(X^TX+\lambda S\right)^{-1}X^T_{\mathcal{I}_{jk}\cdot}$	$2p(2n/K)^2$	$O(p(n/K)^2)$
$I-X_{\mathcal{I}_{jk}\cdot}\left(X^TX+\lambda S\right)^{-1}X^T_{\mathcal{I}_{jk}\cdot}$	$(2n/K)^2$	$O((n/K)^2)$
$\left(I-X_{\mathcal{I}_{jk}\cdot}\left(X^TX+\lambda S\right)^{-1}X^T_{\mathcal{I}_{jk}\cdot}\right)^{-1}$	$\frac{1}{3}(2n/K)^3+2(2n/K)^2$	$O((n/K)^3)$
$\left(I-X_{\mathcal{I}_{jk}\cdot}\left(X^TX+\lambda S\right)^{-1}X^T_{\mathcal{I}_{jk}\cdot}\right)^{-1}(y_{\mathcal{I}_{jk}}-\hat{y}_{\mathcal{I}_{jk}})$	$2(2n/K)^2$	$O((n/K)^2)$
Total Number of FLOPs	$2p^2(2n/K)+2p(2n/K)^2+\frac{1}{3}(2n/K)^3+5(2n/K)^2$	
Total Complexity of $\widehat{r}^{\text{te}}_{jk}$	$O\left(n^3/K^3+n^2p/K^2+np^2/K\right)$	
Total Complexity	$O\left(Ln^3/K+Ln^2p+LKnp^2\right)$	

3.3.2 Refitting the Model

If we refit the model to compute $\widehat{r}^{\text{te}}_{jk}$, we also need to do that for $(LK^2+LK)/2$ times. Each time we fit the model on $(X_i,Y_i)_{i\in\{1,\cdots,n\}\setminus\mathcal{I}_{jk}}$, and then compute the fitted regression coefficient as

$$\hat{\beta}=\left(X^T_{(-\mathcal{I}_{jk})\cdot}X_{(-\mathcal{I}_{jk})\cdot}+\lambda S\right)^{-1}X^T_{(-\mathcal{I}_{jk})\cdot}y_{-\mathcal{I}_{jk}}.$$

It follows that

$$\widehat{r}^{\text{te}}_{jk}=y_{\mathcal{I}_{jk}}-X_{\mathcal{I}_{jk}\cdot}\hat{\beta}=y_{\mathcal{I}_{jk}}-X_{\mathcal{I}_{jk}\cdot}\left(X^T_{(-\mathcal{I}_{jk})\cdot}X_{(-\mathcal{I}_{jk})\cdot}+\lambda S\right)^{-1}X^T_{(-\mathcal{I}_{jk})\cdot}y_{-\mathcal{I}_{jk}}. \tag{3.20}$$

Next, we give the FLOP count and complexity of 3.20 in Table 3.2.

Table 3.2: Step-wise computational complexity and FLOP count for computing $\widehat{r}_{jk}^{\text{te}}$ with refitting models. Total FLOP count and complexity are also reported.

Computation Step	**FLOP Count**	**Complexity**
$X^T_{(-\mathcal{I}_{jk})\cdot} X_{(-\mathcal{I}_{jk})\cdot}$	$2p^2(n-2n/K)$	$O(p^2(n-n/K))$
$X^T_{(-\mathcal{I}_{jk})\cdot} X_{(-\mathcal{I}_{jk})\cdot} + \lambda S$	p^2	$O(p^2)$
$\left(X^T_{(-\mathcal{I}_{jk})\cdot} X_{(-\mathcal{I}_{jk})\cdot} + \lambda S\right)^{-1}$	$\frac{1}{3}p^3 + 2p^2$	$O(p^3)$
$X^T_{(-\mathcal{I}_{jk})\cdot} y_{-\mathcal{I}_{jk}}$	$2p(n-2n/K)$	$O(p(n-n/K))$
$\left(X^T_{(-\mathcal{I}_{jk})\cdot} X_{(-\mathcal{I}_{jk})\cdot} + \lambda S\right)^{-1} X^T_{(-\mathcal{I}_{jk})\cdot} y_{-\mathcal{I}_{jk}}$	$2p^2$	$O(p^2)$
$X_{\mathcal{I}_{jk}\cdot}\hat{\beta}$	$2p(2n/K)$	$O(pn/K)$
$y_{\mathcal{I}_{jk}} - X_{\mathcal{I}_{jk}\cdot}\hat{\beta}$	$2n/K$	$O(n/K)$
Total Number of FLOPs	$2p^2(n-2n/K) + \frac{1}{3}p^3 + 5p^2 + 2pn + 2n/K$	
Total Complexity of $\widehat{r}_{jk}^{\text{te}}$	$O\left(np^2 + p^3\right)$	
Total Complexity	$O\left(LK^2np^2 + LK^2p^3\right)$	

3.3.3 An Alternative Implementation

Our proposed formula has an alternative implementation using Woodbury matrix identity Suppose $H(\lambda) = X\left(X^TX + \lambda S\right)^{-1}X^T$ Notice that by the Woodbury matrix identity,

$$\left(I - H(\lambda)_{\mathcal{I}_{jk}\mathcal{I}_{jk}}\right)^{-1} = I + X_{\mathcal{I}_{jk}\cdot}\left(X^T_{(-\mathcal{I}_{jk})\cdot} X_{(-\mathcal{I}_{jk})\cdot} + \lambda S\right)^{-1} X^T_{\mathcal{I}_{jk}\cdot}.$$

It follows that

$$\widehat{r}_{jk}^{\text{te}} = \left(y_{\mathcal{I}_{jk}} - \widehat{y}_{\mathcal{I}_{jk}}\right) + X_{\mathcal{I}_{jk}\cdot}\left(X^T_{(-\mathcal{I}_{jk})\cdot} X_{(-\mathcal{I}_{jk})\cdot} + \lambda S\right)^{-1} X^T_{\mathcal{I}_{jk}\cdot}\left(y_{\mathcal{I}_{jk}} - \widehat{y}_{\mathcal{I}_{jk}}\right). \tag{3.21}$$

Similar to before, we give complexity of each step for computing (3.21) in Table 3.3. Looking at Table 3.2 and Table 3.3, we see that the complexity of Woodbury implementation recovers the complexity of refitting the model. Hence, Woodbury implementation guarantees that our formula will never be worse than refitting the model. Regarding the number of FLOPs, the only difference lies in $2p(2n/K)$ for Woodbury implementation and $2pn$ for refitting the model. Since usually $2n/K < n$, we conclude that our proposed formula will always be slightly faster than refitting the model.

Table 3.3: Step-wise computational complexity and FLOP count for computing $\widehat{r}_{jk}^{\text{te}}$ with proposed formula, but an alternative implementation using Woodbury. Total FLOP count and complexity are also reported.

Computation Step	**FLOP Count**	**Complexity**
$X^T_{-\mathcal{I}_{jk}\cdot}X_{-\mathcal{I}_{jk}\cdot}$	$2p^2(n-2n/K)$	$O(p^2(n-n/K))$
$X^T_{-\mathcal{I}_{jk}\cdot}X_{-\mathcal{I}_{jk}\cdot}+\lambda S$	$2p^2$	$O(p^2)$
$(X^T_{-\mathcal{I}_{jk}\cdot}X_{-\mathcal{I}_{jk}\cdot}+\lambda S)^{-1}$	$\frac{1}{3}p^3+2p^2$	$O(p^3)$
$X^T_{\mathcal{I}_{jk}\cdot}\left(y_{\mathcal{I}_{jk}}-\widehat{y}_{\mathcal{I}_{jk}}\right)$	$2p(2n/K)$	$O(pn/K)$
$(X^T_{-\mathcal{I}_{jk}\cdot}X_{-\mathcal{I}_{jk}\cdot}+$ $\lambda S)^{-1}X^T_{\mathcal{I}_{jk}\cdot}\left(y_{\mathcal{I}_{jk}}-\widehat{y}_{\mathcal{I}_{jk}}\right)$	$2p^2$	$O(p^2)$
$X_{\mathcal{I}_{jk}\cdot}(X^T_{-\mathcal{I}_{jk}\cdot}X_{-\mathcal{I}_{jk}\cdot}+$ $\lambda S)^{-1}X^T_{\mathcal{I}_{jk}\cdot}\left(y_{\mathcal{I}_{jk}}-\widehat{y}_{\mathcal{I}_{jk}}\right)$	$2p(2n/K)$	$O(pn/K)$
$\left(y_{\mathcal{I}_{jk}}-\widehat{y}_{\mathcal{I}_{jk}}\right)+$ $X_{\mathcal{I}_{jk}\cdot}\left(X^T_{-\mathcal{I}_{jk}\cdot}X_{-\mathcal{I}_{jk}\cdot}+\lambda S\right)^{-1}X^T_{\mathcal{I}_{jk}\cdot}\left(y_{\mathcal{I}_{jk}}-\widehat{y}_{\mathcal{I}_{jk}}\right)$	$2n/K$	$O(n/K)$
Total Number of FLOPs	$2p^2(n-2n/K)+\frac{1}{3}p^3+5p^2$ $+2p(2n/K)+2n/K$	
Total Complexity of $\widehat{r}_{jk}^{\text{te}}$	$O\left(np^2+p^3\right)$	
Total Complexity	$O\left(LK^2np^2+LK^2p^3\right)$	

3.3.4 Compare Asymptotic Complexity

With Table 3.1, Table 3.2 and Table 3.3, we compare them in different regime of n, p, K and summarize our results in Table 3.4. Observe in Table 3.4 that when p scales with n^α or p is fixed, the one-step formula without Woodbury reduces the computational complexity by order $O(n)$. The empirical evaluation shown in Figure 3.1 and Figure 3.2 align with our theoretical results.

Table 3.4: Asymptotic complexity required to compute the nested cross-validation prediction interval by model refitting (Eq. (3.20) and using Proposition 1 with and without the Woodbury identity (Eq. (3.21)). Shown are the four asymptotic regimes: K-fold (K fixed) and leave-k-out ($K = O(n)$) nested cross-validation, as well as low-dimensional (p fixed) and high-dimensional ($p = O(n^\alpha)$ for $1 \geq \alpha > 0$) penalized regression.

Regime	Refitting Model	NCV formula, Woodbury	NCV formula, no Woodbury
K, p fixed	$O\left(K^2np^2 + K^2p^3\right)$	$O\left(K^2np^2 + K^2p^3\right)$	$O\left(n^3/K + n^2p + Knp^2\right)$
$K = O(n)$, p fixed	$O\left(n^3p^2\right)$	$O\left(n^3p^2\right)$	$O\left(n^2p^2\right)$
K fixed, $p = O(n^\alpha)$	$O\left(K^2n^{1+2\alpha}\right)$	$O\left(K^2n^{1+2\alpha}\right)$	$O(n^3/K + n^{2+\alpha} + Kn^{1+2\alpha})$
$K = O(n)$, $p = O(n^\alpha)$	$O\left(n^{3+2\alpha}\right)$	$O\left(n^{3+2\alpha}\right)$	$O\left(n^{2+2\alpha}\right)$

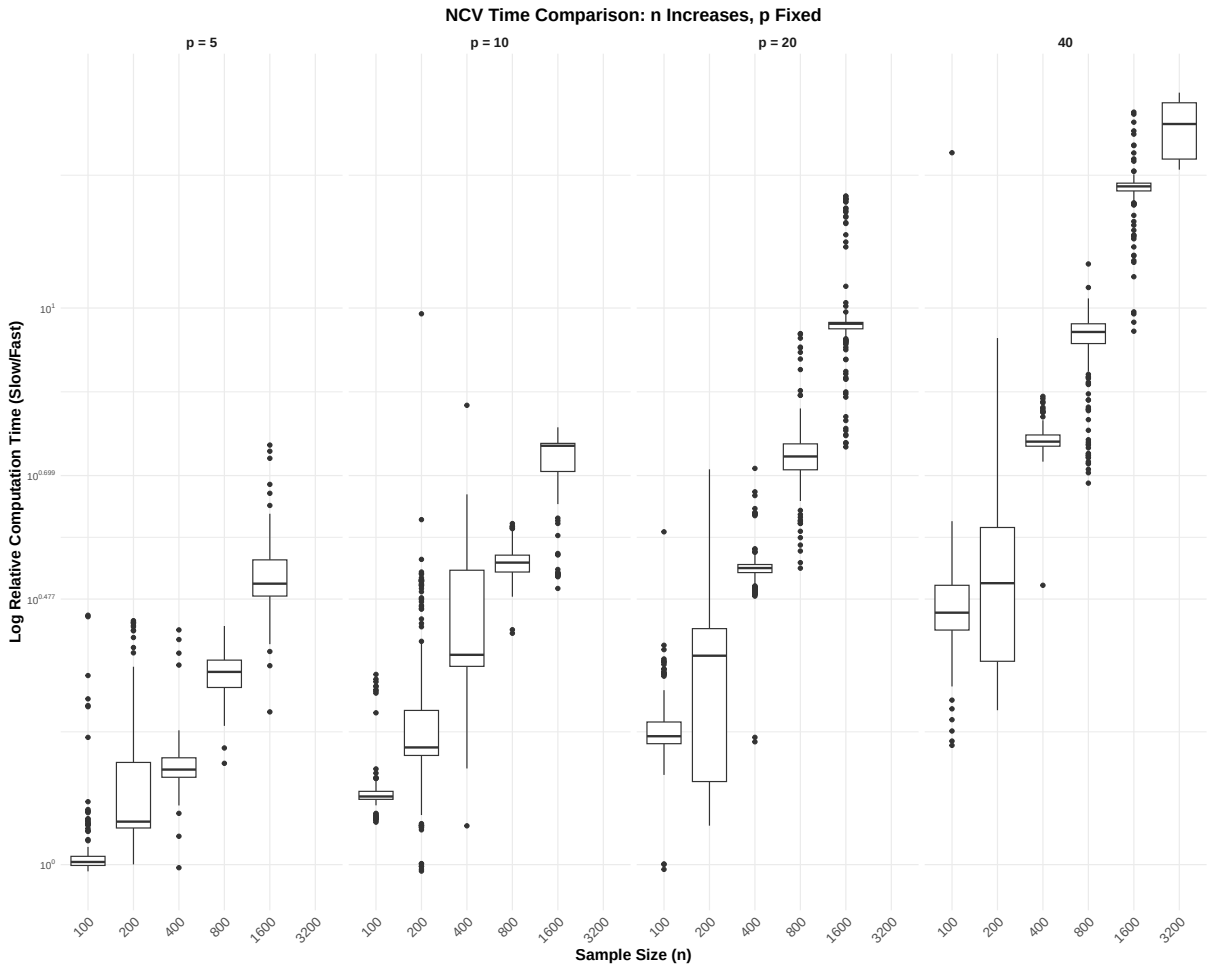


Figure 3.1: Change of log scale relative computation time for fitting an additive model. Relative computation means the computation time of refitting the model (slow) over the computation time of NCV formula (fast). Regime corresponds to $K = O(n)$, p fixed in Table 3.4, meaning n increases and p keeps fixed.

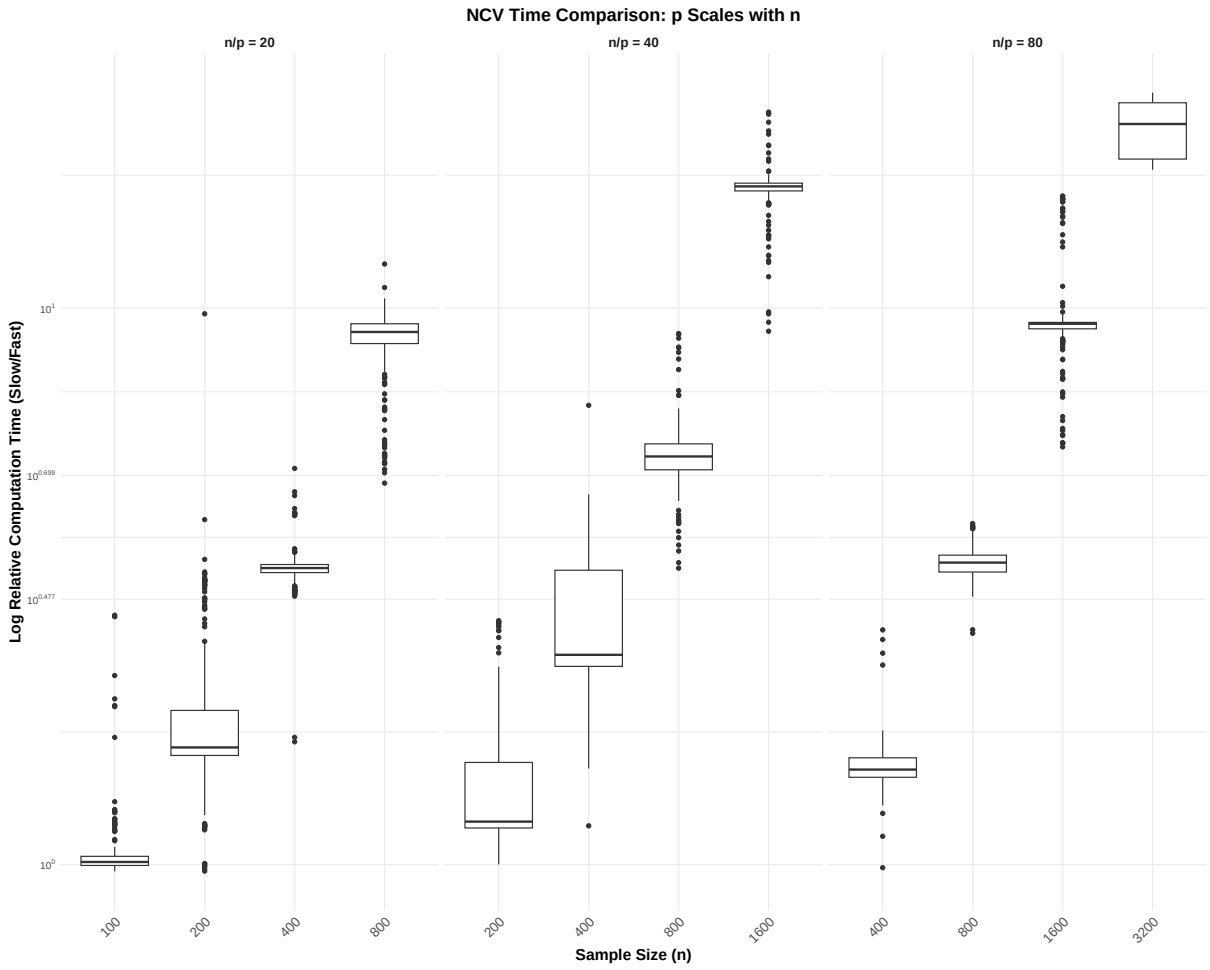


Figure 3.2: Change of log scale relative computation time for fitting an additive model. Relative computation means the computation time of refitting the model (slow) over the computation time of NCV formula (fast). Regime corresponds to $K = O(n)$, $p = O(n^\alpha)$ in Table 3.4, meaning n increases and p also increases with n.

3.3.5 Compare Finite Sample Implementation

The actual values of n, p, and K determine which of the three methods compared in Table 3.4 should be applied to a given dataset and model. The same accounting of floating point operations that gives Table 3.4 yields the following result.

Proposition 2. *Fix n and p. The one-step formula for the nested cross-validation prediction interval given in Proposition 1 requires less floating point operations than refitting the model whenever $K > K^*$, where K^* is the smallest root in $[2, n]$ of the quartic polynomial $f(K) = \alpha_{n,p}^{(0)} + \alpha_{n,p}^{(1)}K + \alpha_{n,p}^{(2)}K^2 + \alpha_{n,p}^{(3)}K^3 + \alpha_{n,p}^{(4)}K^4$ with coefficients*

$$\begin{aligned}
\alpha_{n,p}^{(0)} &= 6n^3, \\
\alpha_{n,p}^{(1)} &= -8n^3 + 12pn^2 + 30n^2, \\
\alpha_{n,p}^{(2)} &= -24pn^2 - 60n^2, \\
\alpha_{n,p}^{(3)} &= -18p^2n + p^3 + 15p^2 + 6n + 6pn, \\
\alpha_{n,p}^{(4)} &= 6p^2n + p^3 + 15p^2 + 6pn.
\end{aligned}$$

Given a choice of K, a dataset of size n, and a model of size p, Proposition 2 is used in practice to decide whether to perform nested cross-validation by refitting the model or by using the one-step formula. Although it appears to require finding the root(s) of a quartic polynomial, recalling that only values of K for which $n/K \in \mathbb{N}$ are acceptable, we simply compute $f(K)$ for all such divisors of n to find the point where it crossed zero. Empirically, K^* can be approximated by a numerical root solver or plotting $f(K)$ directly. Figure 3.3 reports measured relative computational times between Proposition 1 and model refitting in which the value of K^* correctly predicts the point at which Proposition 1 becomes faster. We are left with the justification of Proposition 2.

Proof of Proposition 2. We first outline our strategy. Given n, p, K, we want to count the FLOPs for implementing our proposed formula (Proposition 1) and the FLOPs for the old method, which is refitting the model. Note that FLOP count of computing $\widehat{r}_{jk}^{\text{te}}$ for these two methods are in Table 3.1 and Table 3.2. Both will give us a polynomial of K with coefficients depending on n and p. Equate them and rearrange to get $f(K)$.

Notice that both Table 3.1 and Table 3.2 uses a pair of folds $\mathcal{I}_{jk}$ setting. When it is for one fold $\mathcal{I}_j$, we replace all of the $2n/K$ in Table 3.1 and Table 3.2 with n/K. From section 2.5 and section 3.2, we need to compute $K(K-1)/2$ double folds residuals $\widehat{r}_{jk}^{\text{te}}$ and

compute K one fold residual $\widehat{r}_k^{\text{te}}$. Thus, the total FLOP count for proposed formula is

$$f_1(K) = \left[2p^2\frac{2n}{K} + 2p\left(\frac{2n}{K}\right)^2 + \frac{1}{3}\left(\frac{2n}{K}\right)^3 + 5\left(\frac{2n}{K}\right)^2\right] \cdot \frac{K(K-1)}{2} + \left[2p^2\frac{n}{K} + 2p\left(\frac{n}{K}\right)^2 + \frac{1}{3}\left(\frac{n}{K}\right)^3 + 5\left(\frac{n}{K}\right)^2\right] \cdot K.$$

The total FLOP count for refitting the model is

$$f_2(K) = \left[2p^2n - 2p^2\frac{2n}{K} + \frac{1}{3}p^3 + 5p^2 + 2pn + \frac{2n}{K}\right] \cdot \frac{K(K-1)}{2} + \left[2p^2n - 2p^2\frac{n}{K} + \frac{1}{3}p^3 + 5p^2 + 2pn + \frac{n}{K}\right] \cdot K.$$

Equate $f_1(K) = f_2(K)$ and rearrange as $f_2(K) - f_1(K)$. We multiply by $6K^2$ on both sides of $f_1(K) = f_2(K)$ to get rid of all the fractions. Then some algebra in between is omitted. We reach

$$\begin{aligned} f(K) =& \alpha_{n,p}^{(0)} + \alpha_{n,p}^{(1)}K + \alpha_{n,p}^{(2)}K^2 + \alpha_{n,p}^{(3)}K^3 + \alpha_{n,p}^{(4)}K^4, \\ \alpha_{n,p}^{(0)} &= 6n^3, \\ \alpha_{n,p}^{(1)} &= -8n^3 + 12pn^2 + 30n^2, \\ \alpha_{n,p}^{(2)} &= -24pn^2 - 60n^2, \\ \alpha_{n,p}^{(3)} &= -18p^2n + p^3 + 15p^2 + 6n + 6pn, \\ \alpha_{n,p}^{(4)} &= 6p^2n + p^3 + 15p^2 + 6pn. \end{aligned}$$

Suppose we have a root $K^* \in [2, n]$ for $f(K)$. When $K > K^*$, $f_2(K) - f_1(K) > 0$, thus $f_1(K) < f_2(K)$, meaning our formula is faster and vice versa. □

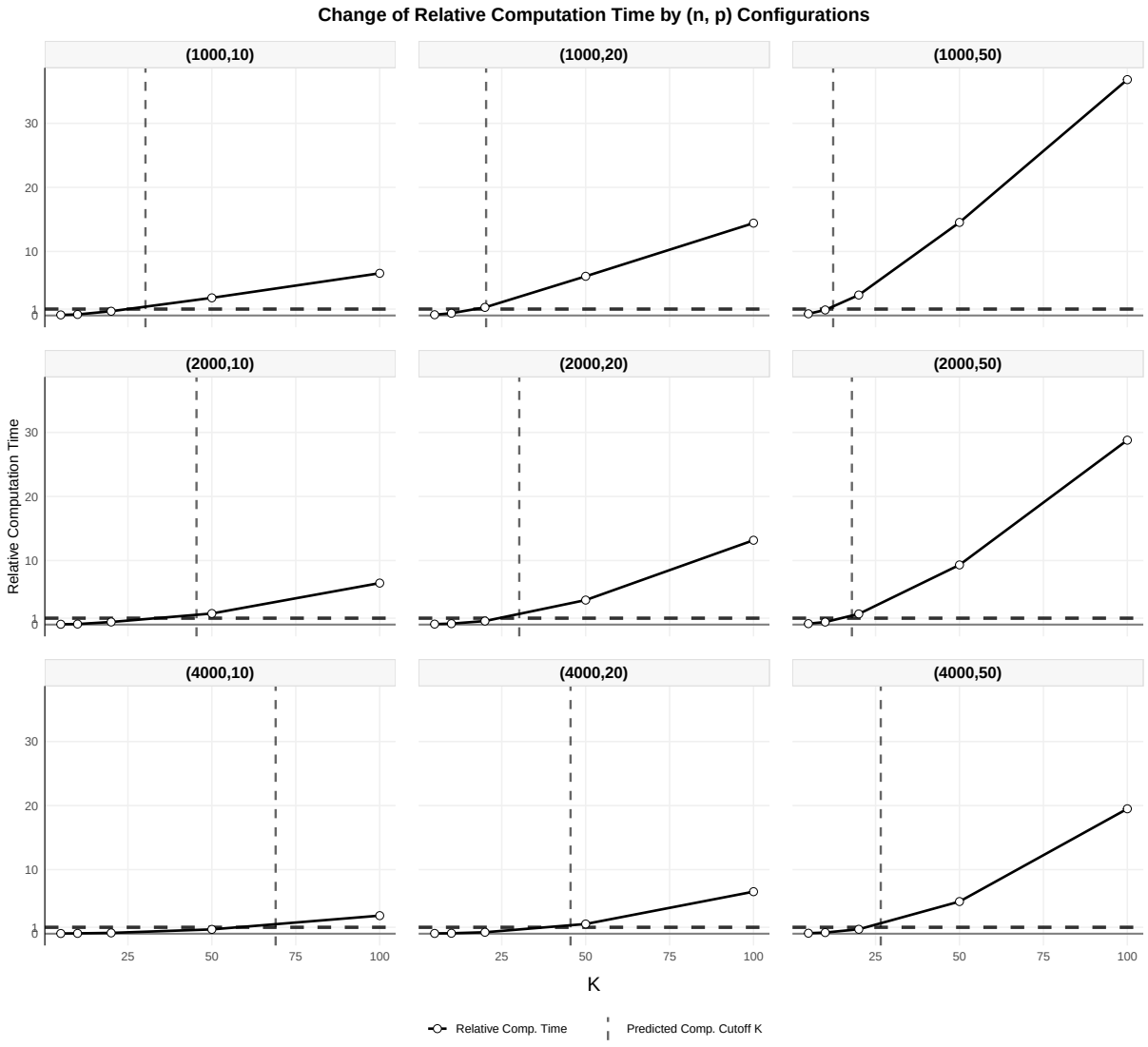


Figure 3.3: Change of relative computation time (refitting model / NCV formula in Proposition 1) with the increase of K and varying pair of (n, p). Horizontal dotted line is 1, meaning computation time of both is equal. Vertical dotted line is the predicted K^*. Relative computation time is expected to rise above 1 when K is bigger than predicted K^* and vice versa.

3.4 Extension to Ultra High Dimension Regression

In this section, we extend our results to the ultra high dimensional setting, meaning $n < p$ for $X \in \mathbb{R}^{n\times p}$. Note that we use the word ultra, so that it differentiates from the high dimension setting of $p = O(n^\alpha)$ for $0 < \alpha \leq 1$ discussed in section 3.3.4. $O(n^\alpha)$ for $\alpha \in (0, 1]$ includes models like penalized splines [32, 12], but they usually remain $n \geq p$. Hence, computationally it is fine as long as $\text{rank}(X) = p$.

When $n < p$ and for a penalty matrix S that is only positive semidefinite such as additive models, the matrix $X^TX + \lambda S$ is no longer invertible, as $\text{rank}(X^TX) = n$ while $X^TX \in \mathbb{R}^{p\times p}$. Thus, most models would require some kind of sparsity penalty. For example, high dimensional additive models [25, 27] needs a LASSO type of penalty, which makes the prediction no longer linear. Then Lemma 1 can no longer be applied.

When $S \succ 0$ and $\lambda > 0$, $X^TX + \lambda S$ remains invertible, hence the hat matrix $H(\lambda)$ still exists, and our technique in Lemma 1 applies. We will consider ridge regression when $n < p$, specifically $S = I$.

Some background knowledge of ridge regression is given in Appendix C. In ultra high dimension, we fit the penalized regression slightly differently, then the form of hat matrix also changes. In particular, by the kernel trick [18], the hat matrix can be written as

$$H(\lambda) = XX^T\left(XX^T + \lambda I_n\right)^{-1}. \tag{3.22}$$

In Appendix C, we prove the equivalence of (3.22) to $H(\lambda) = X(X^TX + \lambda I_p)^{-1}X^T$ in Lemma 4. Further, (3.22) can be motivated by a different formation of ridge regression optimization, shown in Theorem 7. Note that compared with $H(\lambda) = X(X^TX+\lambda I_p)^{-1}X^T$, (3.22) has computational advantage, as it reduces computation from $O(p^3)$ to $O(n^2p)$.

Observe that our formula in Proposition 1 will not change, and the core computation still lies in (3.19). What changes is how we compute the hat matrix, hence the implementation and complexity. Like before, we will give a comparison between our formula and refitting the model like before. For our formula, since we only need to fit the model once for a full model fit, we can compute 3.22 once and store an $n \times n$ matrix. This is fine since we have $n < p$. Then every time we need $H(\lambda)_{\mathcal{I}_{kj}\mathcal{I}_{kj}}$, we directly obtain this sub-matrix. The complete implementation steps with their computation complexity are shown in Table 3.5.

Next, we look at refitting ridge regression when $n < p$. To compute $\widehat{r}^{\text{te}}_{jk}$, refitting the

Table 3.5: Step-wise computational complexity and FLOP count for computing $\widehat{r}^{\text{te}}_{jk}$ with proposed formula. Model is ridge regression in ultra high dimension ($n < p$). Total FLOP count and complexity are also reported.

Computation Step	**FLOP Count**	**Complexity**
$I - H(\lambda)_{\mathcal{I}_{kj}\mathcal{I}_{jk}}$	$(2n/K)^2$	$O((n/K)^2)$
$\left(I - H(\lambda)_{\mathcal{I}_{jk}\mathcal{I}_{jk}}\right)^{-1}$	$\frac{1}{3}(2n/K)^3 + 2(2n/K)^2$	$O((n/K)^3)$
$\left(I - H(\lambda)_{\mathcal{I}_{jk}\mathcal{I}_{jk}}\right)^{-1}\left(y_{\mathcal{I}_{jk}} - \widehat{y}_{\mathcal{I}_{jk}}\right)$	$2(2n/K)^2$	$O((n/K)^2)$
Total Number of FLOPs	$\frac{1}{3}(2n/K)^3 + 5(2n/K)^2$	
Total Complexity of $\widehat{r}^{\text{te}}_{jk}$	$O\left(n^3/K^3\right)$	
Total Complexity	$O\left(Ln^3/K\right)$	

model needs to compute

$$\begin{aligned} \widehat{\beta}^{-\mathcal{I}_{jk}} &= X^T_{(-\mathcal{I}_{jk})\cdot}\left(X_{(-\mathcal{I}_{jk})\cdot}X^T_{(-\mathcal{I}_{jk})\cdot} + \lambda I\right)^{-1} y_{-\mathcal{I}_{jk}}, \\ \widehat{r}^{\text{te}}_{jk} &= y_{\mathcal{I}_{jk}} - X_{\mathcal{I}_{jk}\cdot}\widehat{\beta}^{-\mathcal{I}_{jk}}. \end{aligned} \tag{3.23}$$

From (3.23), we can compute XX^T once and store this $n \times n$ matrix. Then the submatrix $X_{(-\mathcal{I}_{jk})\cdot}X^T_{(-\mathcal{I}_{jk})\cdot}$ can be directly obtained. The full implementation steps and complexity are shown in Table 3.6.

With Table 3.5 and Table 3.6, we compare their complexity in different regime of n, p, K show in Table 3.7. Notice that when $K = O(n)$, our formula reduces the computation by at least $O(n^2)$ for both $p = O(n^\alpha)$ and p is fixed. When K is fixed, our formula has a lower order on K, but may have a higher order on n, thus unclear which is faster in practice.

Table 3.6: Step-wise computational complexity and FLOP count for computing $\widehat{r}^{\text{te}}_{jk}$ with refitting models. Model is ridge regression in ultra high dimension ($n < p$). Total FLOP count and complexity are also reported.

Computation Step	**FLOP Count**	**Complexity**
$X_{(-\mathcal{I}_{jk})\cdot} X^T_{-(\mathcal{I}_{jk})\cdot} + \lambda I$	$2(n-2n/K)^2$	$O((n-2n/K)^2)$
$\left(X_{(-\mathcal{I}_{jk})\cdot} X^T_{-(\mathcal{I}_{jk})\cdot} + \lambda I\right)^{-1}$	$\frac{1}{3}(n-2n/K)^3 + 2(2n/K)^2$	$O((n-2n/K)^3)$
$\left(X_{(-\mathcal{I}_{jk})\cdot} X^T_{-(\mathcal{I}_{jk})\cdot} + \lambda I\right)^{-1} y_{-\mathcal{I}_{jk}}$	$2(n-2n/K)^2$	$O((n-2n/K)^2)$
$X^T_{(-\mathcal{I}_{jk})\cdot} \left(X_{(-\mathcal{I}_{jk})\cdot} X^T_{-(\mathcal{I}_{jk})\cdot} + \lambda I\right)^{-1} y_{-\mathcal{I}_{jk}}$	$2p(n-2n/K)$	$O(p(n-2n/K))$
$X_{\mathcal{I}_{jk}\cdot} \widehat{\beta}^{-\mathcal{I}_{jk}}$	$2p(2n/K)$	$O(pn/K)$
$y_{\mathcal{I}_{jk}} - X_{\mathcal{I}_{jk}\cdot} \widehat{\beta}^{-\mathcal{I}_{jk}}$	$2n/K$	$O(n/K)$
Total Number of FLOPs	$\frac{1}{3}(n-2n/K)^3 + 4pn +$ $6(n-2n/K)^2 + 2n/K$	
Total Complexity of $\widehat{r}^{\text{te}}_{jk}$	$O\left((n-n/K)^3 + pn\right)$	
Total Complexity	$O\left(LK^2(n-n/K)^3 + LK^2pn\right)$	

Table 3.7: Asymptotic complexity required to compute the nested cross-validation prediction interval by model refitting and using Proposition 1. Model is ridge regression in ultra high dimension ($n < p$). Shown are the four asymptotic regimes: K-fold (K fixed) and leave-k-out ($K = O(n)$) nested cross-validation, as well as p fixed and $p = O(n^\alpha)$ for $\alpha > 1$.

Regime	Refitting Model	NCV formula
K, p fixed	$O\left(K^2(n-n/K)^3 + K^2pn\right)$	$O\left(n^3/K\right)$
$K = O(n)$, p fixed	$O\left(n^5 + n^3p\right)$	$O\left(n^2\right)$
K fixed, $p = O(n^\alpha)$	$O\left(K^2(n-n/K)^3 + K^2n^{1+\alpha}\right)$	$O\left(n^3/K\right)$
$K = O(n)$, $p = O(n^\alpha)$	$O\left(n^5 + n^{3+\alpha}\right)$	$O\left(n^2\right)$

Chapter 4

Application

4.1 Model: Functional Principal Components

Let $y_i = \beta_0 + \int_0^1 x_i(t)\omega(t)dt + \epsilon_i, i = 1, \ldots, n$, with $\epsilon_i \sim \mathrm{N}(0, \sigma^2)$ independently. This is the signal regression we start from. The unknown signal parameter is $\omega \in \mathbb{R}^N$. The covariates $x_i(t)$ are usually observed at fixed time points $t_1, \ldots, t_N$. Thus, in matrix and vector form, the model is written as

$$y_i = \beta_0 + x_i\omega + \epsilon_i, \epsilon_i \overset{\text{iid}}{\sim} N(0, 1).$$

Then we do a B-spline expansion on the signal ω, so that $\omega(t) \approx b_1(t)\beta_1 + \cdots + b_A(t)\beta_A$ for some $A \in \mathbb{N}$ leads to writing the the model in discrete form as

$$y_i = \beta_0 + \sum_{j=1}^{N} x_i(t_j) \left\{ \sum_{a=1}^{A} b_a(t_j)\beta_a \right\} + \epsilon_i, i = 1, \ldots, n. \tag{4.1}$$

For further details on signal regression and functional data analysis, see [29]. The unknown parameters are the intercept, β_0, and the spline weights, $\beta = (\beta_1, \ldots, \beta_A)^T$. To estimate the spline weights it is desirable to minimize the penalized least squares criterion:

$$\widehat{\beta}(\lambda) = \underset{\beta \in \mathbb{R}^A}{\arg\min} \left\{ \|y - \beta_0 1_n - XB\beta\|_2^2 + \lambda \beta^T S \beta \right\},$$

where $1_n \in \mathbb{R}^n$ is a vector of 1's, $S \in \mathbb{R}^{A \times A}$ by $S_{ka} = \int_0^1 b_k''(t)b_a''(t)dt$, $X \in \mathbb{R}^{n \times N}$ by $X_{ij} = x_i(t_j)$, and $B \in \mathbb{R}^{N \times A}$ by $B_{ja} = b_a(t_j)$. The penalized least squares criterion is written in vector form has unique solution

$$\widehat{\beta}(\lambda) = \left((XB)^T XB + \lambda S\right)^{-1} (XB)^T y.$$

The corresponding training set predictions are:

$$\widehat{y}(\lambda) = XB\left((XB)^T XB + \lambda S\right)^{-1}(XB)^T y.$$

When the signal is densely observed, the model may be substantially over-parameterized. Dimension reduction is achieved by performing the singular value decomposition $XB = UDV^T$ with $U \in \mathbb{R}^{n\times A}$ a column-orthonormal matrix of left singular vectors, $D = \text{diag}(\delta_1, \ldots, \delta_A)$ a diagonal matrix of ordered singular values, and $V \in \mathbb{R}^{A\times A}$ a square orthonormal matrix of right singular vectors. Some of the singular values may be close to zero, and we retain some number $p \leq A$ of them such that $(\delta_1^2 + \cdots \delta_p^2)/(\delta_1^2 + \cdots \delta_A^2)$ is close to 1. Denote by V_p then first p columns of V. Now the optimization becomes

$$\widehat{\zeta}(\lambda) = \underset{\zeta\in\mathbb{R}^p}{\arg\min}\left\{\|y - \beta_0 1_n - XBV_p\zeta\|_2^2 + \lambda\zeta^T V_p^T S V_p \zeta\right\}.$$

The functional principle component predictions are

$$\widehat{y}_p(\lambda) = (XBV_p)\left[(XBV_p)^T XBV_p + \lambda V_p S V_p\right]^{-1}(XBV_p)^T y.$$

This model is called Functional Principal Component Regression (FPCR) [30]. These are written in the form $\widehat{y}_p(\lambda) = H(\lambda)y$ and so are compatible with Lemma 1 and the formula of Proposition 1. Accordingly, we proceed with nested cross-validation to estimate and quantify uncertainty in the prediction error $\mathcal{R}\{\mathcal{A}_\lambda; \mathcal{T}\}$.

4.2 Simulation

In this simulation, we examine the performance of NCV for tuning parameter selection of FPCR, and take uncertainty with respect to prediction error into consideration. In particular, our study is motivated by [31], in which they compare the performance of smoothing parameter selection between restricted maximum likelihood (REML) and generalized cross-validation (GCV). We provide the background of REML in section 4.2.1. For our simulation on FPCR, we use REML, GCV, and leave-one-out cross-validation (LOO) to select the tuning parameter, and evaluate their performance on uncertainty quantification with NCV. We denote the selected smoothing parameter λ by these methods as $\lambda_{\text{REML}}, \lambda_{\text{GCV}}$ and λ_{LOO} separately.

4.2.1 REML

REML starts by looking at FPCR model in (13) as

$$\begin{aligned} y|\zeta &\sim N(\beta_0 1_n + XBV_A\zeta, \sigma^2 I), \\ \zeta &\sim N(0, (\sigma^2/\lambda)W^{-1}) \end{aligned}$$

where we define

$$W := V_p^T S V_p.$$

REML selects λ by maximizing the restricted log-likelihood

$$l_R(\alpha, \lambda, \sigma|y) = -\frac{1}{2}\{\log|\sigma^2 V_\lambda| + (y-\beta_0 1_n)^T(\sigma^2 V_\lambda)^{-1}(y-\beta_0 1_n) + \log|\sigma^{-2} 1_p^T V_\lambda^{-1} 1_p|\} \tag{4.2}$$

where we define

$$V_\lambda := I_p + \lambda^{-1}(XBV_p)^T W^{-1}(XBV_p).$$

Note that V_λ is invertible since W is invertible and V_p comes from the first p eigenvectors of XB. Given α, λ, the value of σ^2 maximizing (4.2) is

$$\widehat{\sigma}^2_{\alpha,\lambda} = \frac{1}{n-1}(y-\beta_0 1_n)^T V_\lambda^{-1}(y-\beta_0 1_n).$$

It follows that we maximize the profile restricted log-likelihood

$$l_R(\alpha, \lambda|y) = -\frac{1}{2}[\log|V_\lambda| + \log|1_p^T V_\lambda^{-1} 1_p| + (n-1)\log\{(y-\beta_0 1_n)^T V_\alpha^{-1}(y-\beta_0 1_n)\}]. \tag{4.3}$$

Note that since we centered y and X, we know $\alpha = 0$ and (4.3) reduces to

$$l_R(\lambda|y) = -\frac{1}{2}[\log|V_\lambda| + \log|1_p^T V_\lambda^{-1} 1_p| + (n-1)\log\{y^T V_\alpha^{-1} y\}].$$

4.2.2 Simulation Details

We generate $X \in \mathbb{R}^{100\times 100}$, $\omega \in \mathbb{R}^{100}$ and $y \in \mathbb{R}^{100}$ as followed. We start by generating a covariance matrix $\Sigma \in \mathbb{R}^{100\times 100}$ with $\Sigma_{ij} = \exp(-0.1 \cdot |i-j|)$. Then we generate $X \sim N(0, \Sigma)$. Note that we reduce the column sum of X to be zero, since we want the intercept to be zero. We generate time points $t \in \mathbb{R}^{100}$ as $t = (0.01, 0.02, \cdots, 0.99, 1)^T$. We generate ω as $\omega = \sin(t) = (\sin(0.01 \cdot 2\pi), \sin(0.02 \cdot 2\pi), \cdots, \sin(1 \cdot 2\pi))^T$. Lastly, we generate y as $y = X\omega + \epsilon$ with $\epsilon_i \sim N(0, 0.1)$ and $\epsilon = (\epsilon_1, \cdots, \epsilon_{100})$. Note that we also center y to ensure the intercept β_0 is 0.

4.2.3 Results

Figure 4.1 reports the performance of tuning parameters using REML, GCV and LOO. The fitted models with tuning parameter have uncertainty quantified by NCV. Further, Figure 4.1 contain the true prediction errors, so that we can compare the performance of these methods. More clear results of λ_{REML} can be seen in Figure 4.2. We find that λ_{REML} tends to be smaller than λ_{GCV} and λ_{LOO}. However, by looking at the boxplot of prediction error, we see no obvious difference on prediction performance between λ_{REML}, λ_{GCV} and λ_{LOO}. In addition, all of λ_{REML}, λ_{GCV} and λ_{LOO} have large uncertainty shown by NCV prediction interval. Hence, we recommend to use REML for selecting smoothing parameter and use NCV to quantify uncertainty of the selected model.

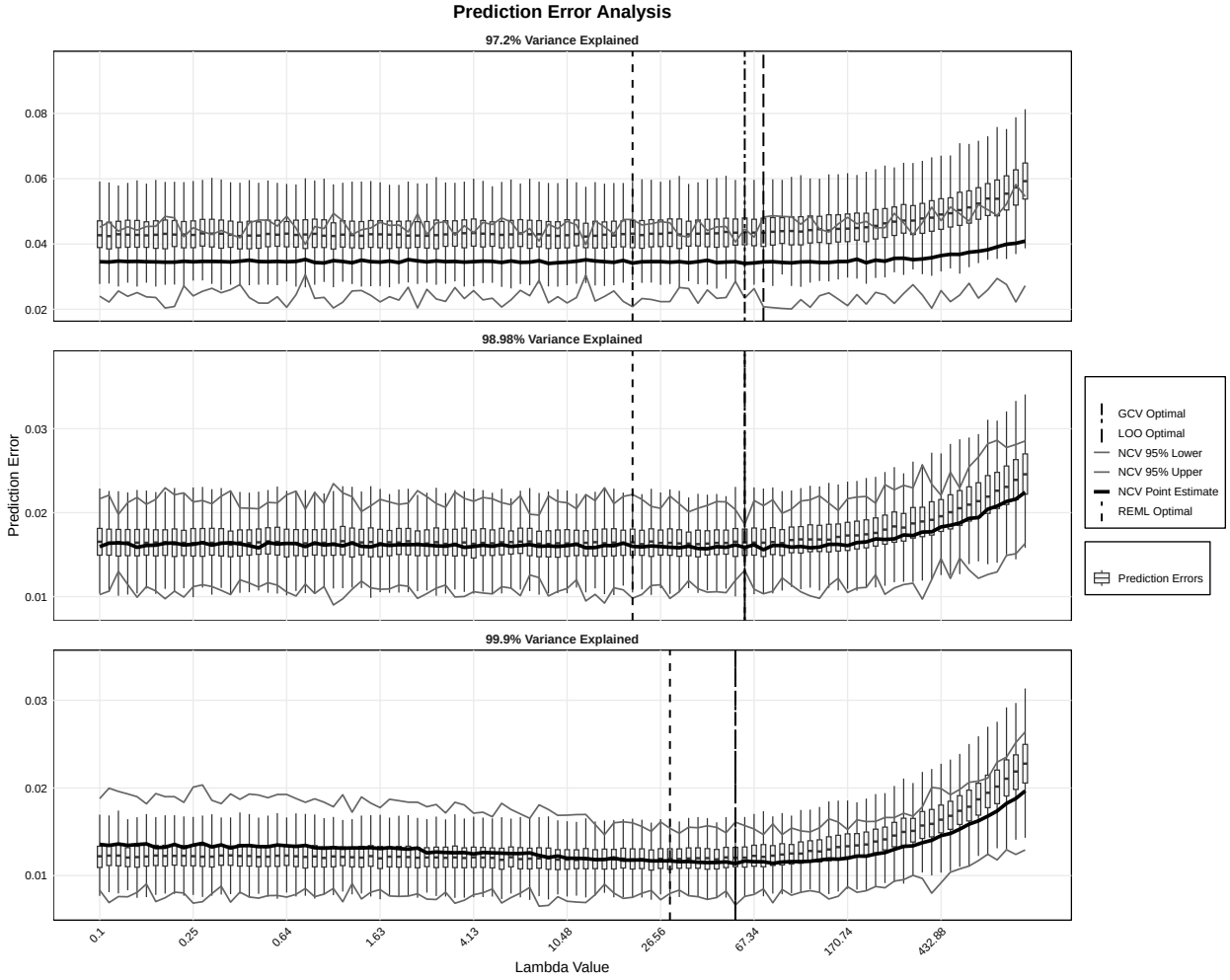


Figure 4.1: Boxplot of prediction errors with results of tuning parameters. NCV point estimates with prediction interval are shown as solid lines. Choices of tuning parameters that give the lowest GCV score, LOO score and that maximize REML are shown as vertical dotted lines. Each row corresponds to one choice of principal components, hence one variance proportion.

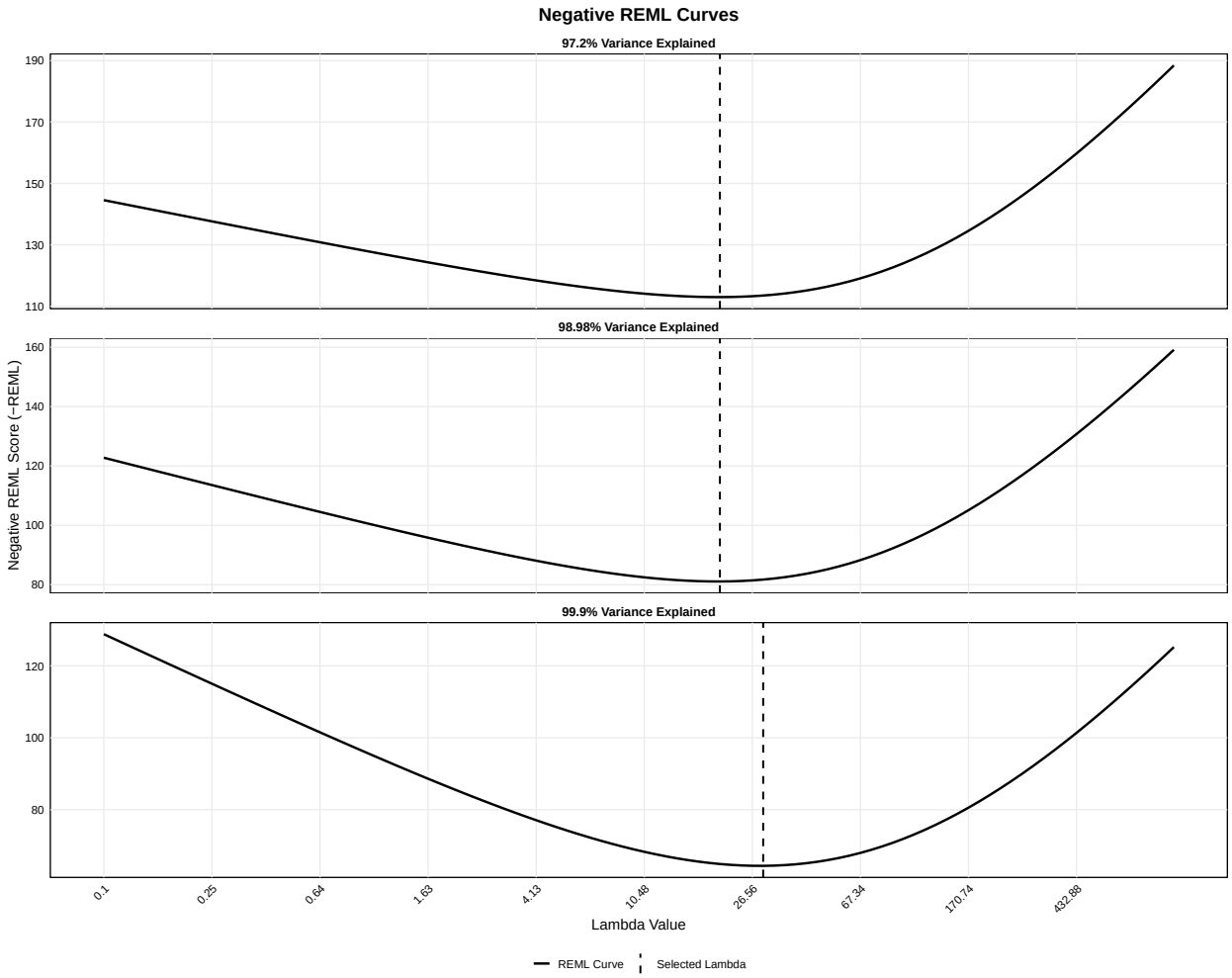


Figure 4.2: Plot of negative restricted maximum likelihood. Vertical dotted line indicates the value of tuning parameter that minimizes the negative REML curve. Each row corresponds to one choice of principal components, hence one variance proportion.

References

[1] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[2] Sylvain Arlot and Alain Celisse. A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4(none):40 – 79, 2010.

[3] Morgane Austern and Wenda Zhou. Asymptotics of cross-validation. *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, 61(4):2804 – 2865, 2025.

[4] Stephen Bates, Trevor Hastie, and Robert Tibshirani. Cross-validation: What does it estimate and how well does it do it? *Journal of the American Statistical Association*, 119(546):1434–1445, 2024.

[5] Pierre Bayle, Alexandre Bayle, Lester Mackey, and Lucas Janson. Cross-validation confidence intervals for test error. In *Conference on Neural Information Processing Systems*, December 2020.

[6] Yoshua Bengio and Yves Grandvalet. No unbiased estimator of the variance of k-fold cross-validation. *Journal of Machine Learning Research*, 5:1089–1105, December 2004.

[7] Olivier Bousquet and André Elisseeff. Stability and generalization. *J. Mach. Learn. Res.*, 2:499–526, March 2002.

[8] Olivier Bousquet, Yegor Klochkov, and Nikita Zhivotovskiy. Sharper bounds for uniformly stable algorithms. In Jacob Abernethy and Shivani Agarwal, editors, *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 610–626. PMLR, 09–12 Jul 2020.

[9] S.P. Boyd and L. Vandenberghe. *Convex Optimization*. Number pt. 1 in Berichte über verteilte messysteme. Cambridge University Press, 2004.

[10] Leo Breiman. Statistical Modeling: The Two Cultures (with comments and a rejoinder by the author). *Statistical Science*, 16(3):199 – 231, 2001.

[11] Alain Celisse and Benjamin Guedj. Stability revisited: new generalisation bounds for the leave-one-out, 2016.

[12] GERDA CLAESKENS, TATYANA KRIVOBOKOVA, and JEAN D. OPSOMER. Asymptotic properties of penalized spline estimators. *Biometrika*, 96(3):529–544, 2009.

[13] Sandrine Dudoit and Mark J. van der Laan. Asymptotics of cross-validated risk estimation in estimator selection and performance assessment. *Statistical Methodology*, 2(2):131 – 154, 2005.

[14] Seymour Geisser. The predictive sample reuse method with applications. *Journal of the American Statistical Association*, 70(350):320–328, 1975.

[15] Gene H. Golub, Michael Heath, and Grace Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223, 1979.

[16] Gene H. Golub and Charles F. Van Loan. *Matrix Computations - 4th Edition*. Johns Hopkins University Press, Philadelphia, PA, 2013.

[17] T.J. Hastie and R.J. Tibshirani. *Generalized Additive Models*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1990.

[18] Trevor Hastie. Ridge regularization: An essential concept in data science. *Technometrics*, 62(4):426–433, 2020. PMID: 36033922.

[19] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning, 2nd Ed.* Springer, New York, 2009.

[20] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 2 edition, 2012.

[21] Satyen Kale, Ravi Kumar, and Sergei Vassilvitskii. Cross-validation and mean-square stability. In *ICS*, pages 487–495, 2011.

[22] Ravi Kumar, Daniel Lokshtanov, Sergei Vassilvitskii, and Andrea Vattani. Near-optimal bounds for cross-validation via loss stability. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 27–35, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[23] Guillaume Lecué and Charles Mitchell. Oracle inequalities for cross-validation type procedures. *Electronic Journal of Statistics*, 6(none):1803 – 1837, 2012.

[24] Jing Lei. A modern theory of cross-validation through the lens of stability, 2025.

[25] Han Liu, Larry Wasserman, John Lafferty, and Pradeep Ravikumar. Spam: Sparse additive models. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.

[26] C. L. Mallows. Some comments on Cp. *Technometrics*, 15(4):661–675, 1973.

[27] Lukas Meier, Sara van de Geer, and Peter Bühlmann. High-dimensional additive modeling. *The Annals of Statistics*, 37(6B):3779 – 3821, 2009.

[28] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012.

[29] J. Ramsay and B.W. Silverman. *Functional Data Analysis*. Springer Series in Statistics. Springer New York, 2006.

[30] Philip T. Reiss and R. Todd Ogden. Functional principal component regression and functional partial least squares. *Journal of the American Statistical Association*, 102(479):984–996, 2007.

[31] Philip T. Reiss and R. Todd Ogden. Smoothing parameter selection for a class of semiparametric linear models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(2):505–523, 2009.

[32] David Ruppert. Selecting the number of knots for penalized splines. *Journal of Computational and Graphical Statistics*, 11(4):735–757, 2002.

[33] J. Shao and D. Tu. *The Jackknife and Bootstrap*. Springer Series in Statistics. Springer New York, 2012.

[34] A.W. van der Vaart, S. Dudoit, and M.J. van der Laan. Oracle inequalities for multi-fold cross validation. *Statistics & Decisions*, 24(3):351–371, 2006. MR2305112.

[35] Simon N. Wood. *Generalized Additive Models: An Introduction with R*. Chapman and Hall/CRC, Boca Raton, FL, 2nd edition, 2017.

[36] Bin Yu. Stability. *Bernoulli*, 19(4):1484 – 1500, 2013.

APPENDICES

Appendix A

Preliminaries on Numerical Linear Algebra

This section aims to give a brief background of numerical linear algebra we need for section 3.3. The primary reference of this section is [16]. Floating point operations (FLOP) is a way to quantify the volume of work in matrix computations. In the numerical linear algebra literature, one FLOP is a basic unit of matrix computations, such as adding, subtraction and multiplication. For example, for some matrices A, B with the same dimension, $A_{ij}+B_{ij}$ is one FLOP. Now for a more concrete example, suppose $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times p}$. We consider the matrix multiplication AB. This needs mnp FLOPs. Table A.1 summarizes the FLOP count for basic matrix and vector operations. Note that we write most of them in the updating value format, because this is what usually happens in practice.

Table A.1: FLOP Count of Vector and Matrix Operations [16, p. 12]

Operation	**Dimension**	**FLOPs**
$\alpha = x^T y$	$x, y \in \mathbb{R}^n$	$2n$
$y = y + \alpha x$	$\alpha \in \mathbb{R},\ x, y \in \mathbb{R}^n$	$2n$
$y = y + Ax$	$A \in \mathbb{R}^{m\times n},\ x \in \mathbb{R}^n,\ y \in \mathbb{R}^m$	$2mn$
$A = A + yx^T$	$A \in \mathbb{R}^{m\times n},\ x \in \mathbb{R}^n,\ y \in \mathbb{R}^m$	$2mn$
$C = C + AB$	$A \in \mathbb{R}^{m\times n},\ B \in \mathbb{R}^{n\times p},\ C \in \mathbb{R}^{m\times p}$	$2mnp$

Next, we give the FLOP count of solving a linear system, as our proposed formula

involves a lot of matrix inversion. Fortunately, the matrices we invert are usually positive definite or enough to assume to be. Regarding why, see section 3.1.2. Inverting a positive definite matrix involves a Cholesky decomposition and then solving two triangular systems. For an n by n matrix, Cholesky decomposition takes $n^3/3$ FLOPs [16, p. 164], and solving either upper or lower triangular system takes n^2 FLOPs [16, p. 106–107]. Thus, together inverting a positive definite matrix takes $n^3/3 + 2n^2$ FLOPs.

Appendix B

Convex Optimization

Content of this section is largely based on [9]. Some of our definitions are not in the most general setting. Our goal is only to provide the necessary background knowledge for section 3.4 and Appendix C. We aim to cover very fundamental concepts. In particular, we will highlight the importance of primal-dual relationship and other core concepts such as the KKT conditions.

Definition 1 (Convex set)**.** *A set C is convex if for any $x, y \in C$ and any θ with $0 \leq \theta \leq 1$, we have $\theta x + (1-\theta)y \in C$.*

Definition 2 (Convex function)**.** *A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if $\mathrm{dom}(f)$ is a convex set and if for all $x, y \in \mathrm{dom}(f)$, and θ with $0 \leq \theta \leq 1$, we have*

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y).$$

An example of a convex function is $f(x) = x^T Q x$ where $Q \succeq 0$. We state it in Lemma 2, which is useful in section 3.4.

Lemma 2. *Let $Q \succeq 0$ and define $f : \mathbb{R}^n \to \mathbb{R}$ as $f(x) = x^T Q x$. Then f is a convex function.*

Proof. Let $x, y \in \mathbb{R}^n$ and $\theta \in [0, 1]$. We consider $\theta f(x) + (1-\theta)f(y) - f(\theta x + (1-\theta)y)$.

Notice that

$$\begin{aligned}
& \theta f(x) + (1-\theta) f(y) - f(\theta x + (1-\theta) y) \\
= \; & \theta x^T Q x + (1-\theta) y^T Q y - \left[\theta^2 x^T Q x + (1-\theta)^2 y^T Q y + 2\theta(1-\theta) x^T Q y\right] \\
= \; & (\theta - \theta^2) x^T Q x + (1-\theta)(1-1+\theta) y^T Q y - 2\theta(1-\theta) x^T Q y \\
= \; & \theta(1-\theta)[x^T Q x + y^T Q y - 2 x^T Q y] \\
= \; & \theta(1-\theta)(x-y)^T Q (x-y) \\
\geq \; & 0
\end{aligned}$$

where the last inequality follows from $Q \succeq 0$ and $\theta, 1-\theta \geq 0$. By Definition 2, f is convex. □

Next, we introduce convex optimization. We start from the primal optimization problem, shown as

$$\begin{aligned}
\min_{x} \quad & f(x) \\
\text{subject to} \quad & f_i(x) \leq 0, \quad i = 1, \ldots, m, \\
& h_j(x) = 0, \quad j = 1, \ldots, p,
\end{aligned} \tag{B.1}$$

where $x \in \mathbb{R}^n$. We denote $\mathcal{D} = \text{dom}(f) \cap_{i=1}^m \text{dom}(f_i) \bigcap \cap_{i=1}^p \text{dom}(h_i)$ as the domain of (B.1) and assume it is non-empty. We also assume $f, f_1, \cdots, f_m, h_1, \cdots, h_p$ are differentiable. If $f, f_1, \cdots, f_m$ are convex functions and $h_i(x) = a_i^T x - b_i$, we say (B.1) is convex. When (B.1) is convex and unconstrained, we can directly check the gradient for the optimal solution, shown in Theorem 4.

Theorem 4. *[9, p. 140] If $f : R^n \to R$ is convex and the problem is unconstrained, then $\nabla f(x) = 0$ is a sufficient and necessary condition for x to be optimal.*

Next, we define and Lagrangian and the dual function, which help to form the dual problem of (B.1). Note that for the rest of the section, we do not require (B.1) to be convex unless stating explicitly.

Definition 3 (Lagrangian)**.** *We define the Lagrangian $L : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ as*

$$L(x, \lambda, \nu) = f(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \tag{B.2}$$

with $\text{dom}(L) = \mathcal{D} \cap \mathbb{R}^m \cap \mathbb{R}^p$. We call λ_i the Lagrange multiplier associated with $f_i(x) \leq 0$ for $i = 1, \cdots, m$, and ν_i the Lagrange multiplier associated with $h_i(x) = 0$ for $i = 1, \cdots, p$.

Definition 4 (Dual function)**.** *The dual function* $g : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ *is*

$$g(\lambda, \nu) = \inf_{x \in \mathcal{D}} L(x, \lambda, \nu). \tag{B.3}$$

Note that for $\lambda \geq 0$, $g(\lambda, \nu)$ is a lower bound of $f(x)$, because $g(\lambda, \nu) = \inf_x L(x, \lambda, \nu) = L(x, \lambda, \nu) \leq f(x)$. We call (λ, ν) with $\lambda \geq 0$ and $(\lambda, \nu) \in \text{dom}(g)$ as dual feasible. Next, we state the dual form of (B.1). The dual form of (B.1) is

$$\begin{aligned} \max_{\lambda, \nu} \quad & g(\lambda, \nu) \\ \text{subject to} \quad & \lambda \geq 0. \end{aligned} \tag{B.4}$$

Let p^* be the optimal value of the prime (B.1) and d^* be the optimal value of the dual (B.4). Then we have $d^* \leq p^*$ since $g(\lambda, \nu)$ is always a lower bound of $f(x)$. This is called weak duality. When we have $d^* = p^*$, we call it strong duality. To check for strong duality when (B.1) is convex, Slater's conditions are usually helpful, as shown in Theorem 5.

Definition 5 (Slater's condition)**.** *Let* relint *denote the relative interior. The Slater's conditions are there exists* $x \in \text{relint}(\mathcal{D})$ *such that*

$$\begin{aligned} f_i(x) < 0, i = 1, \cdots, m, \\ Ax = b. \end{aligned} \tag{B.5}$$

Theorem 5. *[9, p. 226] If (B.1) is convex and Slater's conditions (B.5) hold, strong duality holds.*

Finally, we state the KKT conditions for convex program. It is helpful because when (B.1) is convex, KKT conditions give the necessary and sufficiency conditions for the primal and the dual optimal solutions.

Theorem 6. *[9, p. 244] Let (B.1) be convex. Then strong duality holds with* x^* *is prime optimal and* (λ^*, ν^*) *are dual optimal if and only if the following KKT conditions are satisfied:*

$$\begin{aligned} f_i(x^*) \leq 0, i = 1, \cdots, m, \\ h_i(x^*) = 0, i = 1, \cdots, p, \\ \lambda_i^* \geq 0, i = 1, \cdots, m, \\ \lambda_i^* f_i(x^*) = 0, i = 1, \cdots, m \\ \nabla f(x^*) + \sum_{i=1}^m \lambda_i^* \nabla f_i(x^*) + \sum_{i=1}^p \nu_i^* \nabla h_i(x^*) = 0. \end{aligned} \tag{B.6}$$

Appendix C

Background of Ultra High Dimension Ridge Regression

Let $n < p$, $\lambda > 0$ and $S = I \succ 0$. We assume $\text{rank}(X) = n$. Note that some of the content we cover in this section do not restrict to $n < p$. We include them here, because they help us to motivate (3.22). Since the unconstrained optimization problem of ridge regression remains the same, which is

$$\min_{\beta} \quad \|y - X\beta\|_2^2 + \lambda\beta^T\beta, \tag{C.1}$$

the fitted coefficient, which is also the minimizer of (C.1), is $\widehat{\beta} = (X^TX + \lambda I)^{-1}X^Ty$. This is the same as its form when $n \geq p$.

It follows that the hat matrix $H(\lambda)$ is

$$H(\lambda) = X(X^TX + \lambda I_p)^{-1}X^T. \tag{C.2}$$

The format of hat matrix already implies our technique in Lemma 1 is applicable. However, computationally it is not efficient to invert $X^TX + \lambda I_p \in \mathbb{R}^{p\times p}$, which costs $O(p^3)$ and p can be very large. Since we know $\text{rank}(X^TX) = n < p$, our first goal is to reduce the computation of accessing (C.2).

One thing we can try is singular value decomposition (SVD) on X. This method is mentioned in [19, p. 659]. Let the singular decomposition of X be $X = UDV^T$ and $R = UD$. Lemma (3) will show us how SVD helps us get a solution for (C.1).

Lemma 3. *$\beta = V(R^TR + \lambda I)^{-1}R^Ty$ is a minimizer of (C.1).*

Proof. We first show the objective function of (C.1) is a convex function of β. Then we show $V(R^TR+\lambda I)^{-1}R^Ty$ satisfies the stationarity condition of (C.1). Finally, we conclude that since (C.1) is convex and unconstrained, any solution satisfying the stationarity condition is a minimizer. Let $\theta \in [0,1]$ and $f(\beta) = (y - X\beta)^T(y - X\beta) + \lambda\beta^T\beta$. Let $\beta_1, \beta_2 \in \mathbb{R}^p$. Note since the identity matrix $I_p \succ 0$, by Lemma 2, $\lambda\theta\beta_1^T\beta_1 + \lambda(1-\theta)\beta_2^T\beta_2 \geq \lambda(\theta\beta_1 + (1-\theta)\beta_2)^T(\theta\beta_1 + (1-\theta)\beta_2)$. Hence, if we can show

$$\begin{aligned} & (y - \theta X\beta_1 - (1-\theta)X\beta_2)^T(y - \theta X\beta_1 - (1-\theta)X\beta_2) \\ \leq\ & \theta(y - X\beta_1)^T(y - X\beta_1) + (1-\theta)(y - X\beta_2)^T(y - X\beta_2), \end{aligned} \tag{C.3}$$

we are done. Notice that $y^Ty = \theta_1 y^Ty + (1-\theta)y^Ty$ and $2\theta(1-\theta)\beta_1X^TX\beta_2$ and both sides of (C.3) share $-2\theta y^TX\beta_1, -2(1-\theta)y^TX\beta_2$. Hence, (C.3) reduces to proving

$$\begin{aligned} & (\theta\beta_1 + (1-\theta)\beta_2)^TX^TX(\theta\beta_1 + (1-\theta)\beta_2) \\ \leq\ & \theta\beta_1X^TX\beta_1 + (1-\theta)\beta_2^TX^TX\beta_2. \end{aligned} \tag{C.4}$$

Since obviously $X^TX \succeq 0$, by Lemma 2 (C.4) is true.

Next, we derive the stationarity condition of (C.1). Notice that

$$\begin{aligned} & \frac{\partial}{\partial\beta}\left\{\|y - X\beta\|_2^2 + \lambda\beta^T\beta\right\} \\ =\ & \frac{\partial}{\partial\beta}\left\{y^Ty - 2\beta^TX^Ty + \beta^TX^TX\beta + \lambda\beta^T\beta\right\} \\ =\ & -2X^Ty + 2X^TX\beta + \lambda\beta. \end{aligned}$$

Setting $-2X^Ty + 2X^TX\beta + \lambda\beta = 0$ and we have the stationarity condition as

$$(X^TX + \lambda I_p)\beta = X^Ty. \tag{C.5}$$

Next, we plug in plug in $V(R^TR+\lambda I)^{-1}R^Ty$ and check whether it satisfies (C.5). Observe that $X^TX = VDU^TUDV^T = VD^2V^T$. Then

$$\begin{aligned} & (X^TX + \lambda I_p)V(R^TR + \lambda I_n)^{-1}R^Ty \\ =\ & (VD^2V^TV + \lambda V)(R^TR + \lambda I_n)^{-1}R^Ty \\ =\ & V(D^2 + \lambda I_n)(D^TU^TUD + \lambda I_n)^{-1}R^Ty \\ =\ & V(D^2 + I_n)(D^2 + I_n)^{-1}R^Ty \\ =\ & VR^Ty \\ =\ & X^Ty, \end{aligned}$$

from which we know (C.5) is satisfied. By Theorem 4, $V(R^TR+\lambda I_n)^{-1}R^Ty$ is a minimizer as desired. □

Lemma 3 reduces computation from $O(p^3)$ to $O(n^2p)$, because $R^TR \in \mathbb{R}^{n\times n}$. However, the problem is that we cannot apply our technique in Lemma 1. To elaborate, if we let

$$H(\lambda) = UDV^TV(R^TR+\lambda I_n)R^T = R(R^TR+\lambda I_n)R^T$$

, the problem arises when we do

$$H(\lambda)_{\mathcal{I}\mathcal{I}} = R_{\mathcal{I}\cdot}(R^TR+\lambda I_n)^{-1}R_{\mathcal{I}\cdot}^T,$$

and $R_{\mathcal{I}\cdot} = U_{\mathcal{I}\cdot}D$ does not come from the SVD of $X_{\mathcal{I}\cdot}$. In particular, when we take certain rows of a matrix like $X_{\mathcal{I}\cdot}$, its SVD decomposition is no longer $U_{\mathcal{I}\cdot}DV^T$. We have to do another SVD on $X_{\mathcal{I}\cdot}$ to get $X_{\mathcal{I}\cdot} = \tilde{U}\tilde{D}\tilde{V}^T$. This breaks the usage of Lemma 1, as the power of Lemma 1 is to avoid refitting models, but $X_{\mathcal{I}\cdot} = \tilde{U}\tilde{D}\tilde{V}^T$ is the same as refitting another time.

Instead, by the kernel trick [18], the hat matrix can be written as

$$H(\lambda) = XX^T\left(XX^T+\lambda I_n\right)^{-1}. \tag{C.6}$$

Then two question arise. First, whether (C.6) is equivalent to (C.2). Second, what motivates the alternative form of $H(\lambda)$ in (C.6). We answer them respectively in Lemma 4 and Theorem 7. Note that proof of Theorem 7 can be found in [28, p. 279], but we give a slightly more complete proof based upon framework of convex optimization (Appendix B).

Lemma 4. $H(\lambda) = XX^T\left(XX^T+\lambda I_n\right)^{-1} = X(X^TX+\lambda I_p)^{-1}X^T$.

Proof. Notice that

$$\begin{aligned}
& \left[XX^T(XX^T+\lambda I_n)^{-1} - X(X^TX+\lambda I_p)^{-1}X^T\right](XX^T+\lambda I_n) \\
= \;& XX^T - X(X^TX+\lambda I_p)^{-1}X^T(XX^T+\lambda I_n) \\
= \;& XX^T - X(X^TX+\lambda I_p)^{-1}(X^TXX^T+\lambda X^T) \\
= \;& XX^T - X(X^TX+\lambda I_p)^{-1}(X^TX+\lambda I_p)X^T \\
= \;& XX^T - XX^T \\
= \;& 0.
\end{aligned}$$

Since $XX^T+\lambda I_n \in \mathbb{R}^{n\times n}$ and $\text{rank}(XX^T+\lambda I_n) = n$, we know $\dim(\text{Row}(XX^T+\lambda I_n)) = n$. Thus, we must have

$$0 = XX^T(XX^T + \lambda I_n)^{-1} - X(X^TX + \lambda I_p)^{-1}X^T$$

and $XX^T(XX^T + \lambda I_n)^{-1} = X(X^TX + \lambda I_p)^{-1}X^T$ as desired. □

Remark 3. *Lemma 4 shows that (C.2) and (C.6) are the same quantity. The difference is that (C.6) is computationally more efficient, which costs $O(n^2p)$, while (C.2) costs $O(p^3)$. The reduction is due to not inverting $X^TX + \lambda I_n$, but $XX^T + \lambda I_p$.*

Before we proceed, we give an alternative formation of (C.1) and prove their equivalence. This will be helpful for interpretation of Theorem 7.

Lemma 5. *Let $t > 0$. The ridge optimization problem (C.1) is equivalent to*

$$\begin{aligned} \min_{\beta} \quad & \|y - X\beta\|_2^2, \\ \textit{subject to} \quad & \|\beta\|_2^2 \le t. \end{aligned} \tag{C.7}$$

Proof. We consider (C.7) as the prime problem and want to show its equivalence to (C.1). By Lemma 2 and middle steps of Lemma 3, (C.7) is convex. The Lagrangian of (C.1) is

$$L(\beta, \lambda) = (y - X\beta)^T(y - X\beta) + \lambda(\beta^T\beta - t).$$

Note that Slater's condition holds for (C.7) by taking $\beta = 0$. Hence, strong duality holds. It follows that KKT conditions are both necessary for the optimal solutions of the prime and the dual. Suppose the dual optimal solution is λ^* and the prime optimal solution is β^*. By KKT stationary condition,

$$\begin{aligned} 0 &= \frac{\partial}{\partial\beta}\{L(\beta^*, \lambda^*)\} \\ &= -2X^Ty + 2X^TX\beta^* + 2\lambda^*\beta^*, \end{aligned} \tag{C.8}$$

which is the same as the stationarity condition for (C.1) by taking $\lambda = \lambda^*$. Hence, optimal solution of (C.7) is the same as (C.1) when taking $\lambda = \lambda^*$. If we start from (C.1), we solve it to get β^*. Then let $t = \|\beta^*\|_2^2$ in (C.7) and consider $\lambda^* = \lambda$. Then β^* satisfies (C.8) and $\lambda^*(\beta^* - \|\beta^*\|_2^2) = \lambda^*(\beta^* - t) = 0$, which is the complementary slackness condition of KKT. Hence, β^* is also the optimal solution of (C.7) by sufficiency of KKT. Thus, (C.7) and (C.1) are equivalent. □

Theorem 7. *[28, p. 279] Let $t > 0$. $\beta = X^T(XX^T + \lambda I)^{-1}y$ can be derived from the dual problem of optimization problem below:*

$$\begin{aligned} \min_{\beta,\epsilon} \quad & \epsilon^T\epsilon \\ \textit{subject to} \quad & \beta^T\beta \le t, \\ & \epsilon_i = y_i - x_i^T\beta, \quad i = 1, \dots, n. \end{aligned} \tag{C.9}$$

Proof. We treat (C.9) as the prime problem. Note that (C.9) is convex by Lemma 2. Its Lagrangian is

$$L(\epsilon, \beta, \lambda, \alpha) = \epsilon^T\epsilon + \lambda(\beta^T\beta - t) + \alpha(\epsilon - y + X\beta).$$

By taking $(\beta, \epsilon) = (0_p, y)$, we get $0 < t$ and $y = y - 0_n$. Thus, Slater's conditions hold by definition B.5. By Theorem 6, KKT conditions (B.6) is sufficient and necessary for optimal solutions. Suppose the primal optimal solutions are β^*, ϵ^* and the dual optimal solutions are λ^*, μ^*. Consider stationary conditions of (B.6), and we have

$$\begin{aligned} \frac{\partial}{\partial\beta}L(\epsilon, \beta, \lambda, \alpha) &= 2\lambda\beta + X^T\alpha, \\ \frac{\partial}{\partial\epsilon}L(\epsilon, \beta, \lambda, \alpha) &= 2\epsilon + \alpha. \end{aligned}$$

Thus, we get

$$\begin{aligned} \beta^* &= -\frac{1}{2\lambda^*}X^T\alpha^*, \\ \epsilon^* &= -\frac{1}{2}\alpha^*. \end{aligned} \tag{C.10}$$

With (C.10), we can rewrite $L(\epsilon^*, \beta^*, \lambda^*, \alpha^*)$ as

$$L(\epsilon^*, \beta^*, \lambda^*, \alpha^*) = -\frac{1}{4}(\alpha^*)^T\alpha^* - \frac{1}{4\lambda^*}(\alpha^*)^TXX^T\alpha^* - \lambda^*t - (\alpha^*)^Ty.$$

It follows that by stationary condition of (B.6) again,

$$\frac{\partial}{\partial\alpha}L(\epsilon^*, \beta^*, \lambda^*, \alpha^*) = -\frac{1}{2}\left(I_n + \frac{1}{\lambda^*}XX^T\right)\alpha^* - y = 0.$$

Thus,

$$\alpha^* = -2\left(I_n + \frac{1}{\lambda^*}XX^T\right)^{-1}y.$$

By (C.10), we get

$$\beta^* = -\frac{1}{2\lambda^*} X^T(-2)\left(I_n + \frac{1}{\lambda^*} XX^T\right)^{-1} y = X^T(XX^T + \lambda^* I_n)^{-1} y.$$

□

Remark 4. *Note that (C.9) is obviously equivalent to the alternative formation of ridge regression optimization (C.7). By Lemma 5, we know (C.9), (C.7) and (C.1) all form the optimization of ridge regression.*