NOVEL APPROACHES IN CLASSIFICATION ERROR ESTIMATION, PREDICTING GENERALIZATION IN DEEP LEARNING, AND HYBRID COMPARTMENTAL MODELS

A Dissertation

by

PARISA GHANE

Submitted to the Graduate and Professional School of
Texas A&M University
In partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Ulisses Braga-Neto
Committee Members,	Nicolaas EP Deutz
	Ivan Ivanov
	Yang Shen
Head of Department,	Miroslav Begovic

August 2022

Major Subject: Electrical Engineering

ABSTRACT

In data-poor environments, it may not be possible to set aside a large enough test data set to produce accurate test-set error estimates. On the other hand, in modern classification applications where training is time and resource intensive, as when training deep neural networks, classification error estimators based on resampling, such as cross-validation and bootstrap, are too computationally expensive, since they require training tens or hundreds of classifiers on resampled versions of the training data. The alternative in this case is to train and test on the same data, without resampling, i.e., to use resubstitution-like error estimators. Here, a family of generalized resubstitution classifier error estimators are proposed and their performance in various scenarios is investigated. This family of error estimators is based on empirical measures. The plain resubstitution error estimator corresponds to choosing the standard empirical measure that puts equal probability mass over each training points. Other choices of empirical measure lead to bolstered resubstitution, posterior-probability, Bayesian error estimators, as well as the newly proposed bolstered posterior-probability error estimators.

Empirical results of this dissertation suggest that the generalized resubstitution error estimators are particularly useful in the presence of small sample size for various classification rules. In particular, bolstering led to remarkable improvement in error estimation in the majority of experiments on traditional classifiers as well as modern deep neural networks. Bolstering is a type of data augmentation that systematically generates meaningful samples, primarily through data-driven bolstering parameters. The bolstering parameter for low to average dimensional data was defined based on the Euclidean distance between samples in each class. But Euclidean distance between images is not straightforward and semantically meaningful. Hence, for experiments with image data, parameters of data augmentation were selected in a different fashion. I introduce three approaches to image augmentation, among which weighted augmented data combined with the posterior probability was most effective in predicting the generalization gap in deep learning.

For the study of protein turn over, I propose hybrid compartmental models (HCM), that are

useful for multi-substrate experiments. Unlike the conventional compartmental models, HCM starts with a partially specified structure for tracer models, estimates the tracer parameters given the data, and finally determines the details of model's structure by choosing the most physiologically meaningful tracee model among the resulting alternative tracee models. The parameters in the alternatives tracee models are computed by simple mathematical operations on tracer parameters. The proposed HCM was employed to estimate kinetics of Phenylalanine and Tyrosine using tracer-tracee-ratio (TTR) data. Results show that HCM tracer model was able to fit the TTR-time data points, and the best tracee model was selected by comparing the alternative tracee models' parameters with those reported in the literature.

DEDICATION

To my parents and my grandmother.

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my academic advisor, Dr. Ulisses Braga-Neto, for his unconditional support, guidance, teaching, and encouragement throughout this dissertation. Working under his supervision was a great honor for me. I would also like to thank my supervisor, Dr. Nicolaas EP Deutz, for providing guidance and feedback throughout development of compartmental models. He was more than generous with his precious time and knowledge. I express my deepest appreciation to Dr. Ivan Ivanov and Dr. Yang Shen, for their encouragements and constructive comments. I owe a special debt of gratitude to my dear parents for making the ultimate sacrifice and for bearing with me during my absence as well as my sister for motivating me throughout my entire graduate and professional school. Also, special thanks go to my friends who have been there cheerfully throughout my doctoral studies.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor Ulisses Braga-Neto and Professor Yang Shen of the Department of Electrical and Computer Engineering and Professor Nicolaas Deutz of the Department of Health and Kinesiology and Professor Ivan Ivanov of the Department of Biomedical Engineering.

The analyses depicted in Chapter 2 were conducted in part by Dr. Ulisses Braga-Neto. The data analyzed for Chapter 4 was provided by Professor Nicolaas Deutz.

All other work conducted for the dissertation was completed by the student independently.

Funding Sources

This graduate study was supported by graduate assistantship from department of electrical and computer engineering at Texas A&M University.

NOMENCLATURE

GR	Generalize Resubstitution
CV	Cross-Validation
VC	Vapnik–Chervonenkis
DNN	Deep Neurtal Network
CNN	Convolutional Neural Network
SVM	Support Vector Machines
RBF	Radial Basis Function
kNN	k Nearest Neighbors
CART	Classification and Regression Tree
RMS	Root Mean Square
resub	Resubstitution Error Estimator
bolster	Bolstered Error Estimator
PP	Posterior Probability
boot	Bootstrap
CNN	Convolutional Neural Network
Acc	Accuracy
Var	Variance
PGDL	Predicting Generalization in Deep Learning
CMI	Conditional Mutual Information
HCM	Hybrid Compartmental Model
CCM	Conventional Compartmental model
EC	Extra-Cellular

IC	Inter-Cellular
TTR	Tracer-Tracee Ratio
F_{ij}	Flux to pool i from pool j
$F_{s_1 \to s_2}$	Flux from substrate 1 to substrate 2
k_{ij}	Tracer fractional rate to pool i from pool j
$k_{s_1 \to s_2}$	Tracer fractional rate from substrate 1 to substrate 2
Q_j	Pool j size
S_j	Substrate j
U_{IC}	de-novo production into Inter-cellular pool

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

1.1 Classification Error Estimation

Given enough training data, good classification algorithms produce classifiers with small error rate on future data, which is also known in machine learning as the *generalization error*. But a classifier is useful only if its generalization error can be stated with confidence. Hence, at a fundamental level, one can only speak of the goodness of a classification algorithm together with an error estimation procedure that produces an accurate assessment of the true generalization error of the resulting classifier. Error estimation for classification has a long history and many different error estimation procedures have been proposed [2, 3, 4, 5, 6]. The subject has recently become a topic of concern in the deep learning community [7]. Error estimators based on resampling, such as cross-validation [8, 9, 10, 11], and bootstrap [12, 13, 14], have long been popular choices of error estimation procedures.

In contemporary classification applications, particularly in the case of deep learning, training can be time and resource intensive [15]. As a result, error estimators based on resampling are no longer a viable choice, since they require training tens or hundreds of classifiers on resampled versions of the training data. It has become instead the norm to use the test-set error, i.e., the error rate on data not used in training, to benchmark classifiers [16]. The test-set error estimator is an unbiased, consistent estimator of the generalization error regardless of the sample size or distribution of the problem [6]. However, this is only true if the test data is truly independent of training, and is not reused in any way [17]. It has been recognized recently that this has not been always the case in image classification using popular benchmarks, where the same public test sets are heavily re-used to measure classification improvement, creating a situation known as "training to the test data" [18]. Strictly speaking, true independent test sets are one-way: they can only be used once. In addition, if training and testing sample sizes are small, the test-set error estimator can display large variance, and become unreliable. All of this means that accurate test-set error

estimation requires cheap access to plentiful labeled data.

The alternative to resampling and test-set error estimation is testing on the training data. The error rate on the training data is known as the *resubstitution* error estimator [19]. This does not require retraining the classifier and is as fast as using a test-set error estimator, but does not assume any separate independent test data. The resubstitution estimator is however usually optimisticaly biased, the more so the more the classification algorithm overfits to the training data. Optimistic bias implies that the difference between resubstitution estimate and the true error, which has been called the "generalization gap" [20], is negative with a high probability. It is key therefore to investigate mechanisms to reduce the bias.

The subject of classification error estimation has a long history and has produced a large body of literature; four main review papers summarize major advances in the field up to 2000 [2, 3, 4, 5]; recent advances in error estimation since 2000 include work on model selection [21], bolstered error estimation [22, 23], feature selection [24, 25, 26, 27], confidence intervals [28, 29, 30], model-based second-order properties [31, 32], and Bayesian error estimators [33, 34]. In this section, we provide a brief review of the basic concepts related to error estimation. A booklength treatment of the topic is provided in [35]; see also [36, 37].

If the distribution of the features and label was known, then one could in principle compute the classification error ε_n by evaluating (2.4). In practice, such knowledge is rarely available, so one employs an *error estimation rule* Ξ_n in order to obtain a classification error estimate

$$\hat{\varepsilon}_n \;=\; \Xi_n(\Psi_n, S_n, \xi)\,, \tag{1.1}$$

where ξ denote *internal random factors* (if any) that represent randomness that is not introduced by the training data S_n; if there are no such internal random factors, the error estimation rule is said to be *nonramdomized*, in which case the error estimate is a determined by the data, otherwise, it is said to be *randomized*, in which case the error estimator is a random variable given the data.

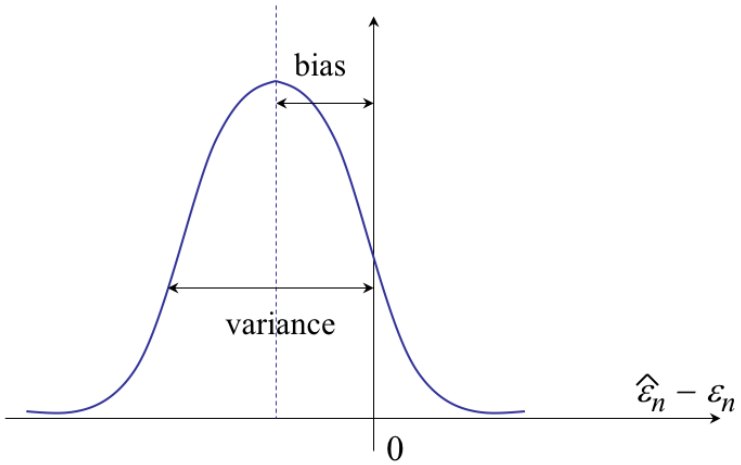


Figure 1.1: Deviation distribution showing bias and variance. The error estimator in this example is optimistically biased.

The *resubstitution rule* Ξ_n^r is an example of nonrandomized error estimation rule:

$$\Xi_n^r(\Psi_n, S_n) \;=\; \frac{1}{n}\sum_{i=1}^{n} I(\Psi_n(S_n)(\mathbf{X}_i) - Y_i)\,. \tag{1.2}$$

Note that the dependence of $\hat{\varepsilon}_n$ on Ψ_n in (1.1) makes explicit the fact that, while the error estimation rule Ξ_n may be fixed (e.g., resubstitution), the properties of $\hat{\varepsilon}_n$ change for different classification rules. This allows us to speak of a resubstitution error estimator $\hat{\varepsilon}_n^r \;=\; \Xi_n^r(\Psi_n, S_n)$ for each classification rule Ψ_n.

The performance of an error estimator can be assessed by the distribution of $\hat{\varepsilon}_n - \varepsilon_n$, called the *deviation distribution* [38]. For good performance, this distribution should be peaked (low-variance) and centered near zero (low-bias). See Figure 1.1 for an illustration.

The *bias* is defined as the first moment of the deviation distribution:

$$\mathrm{Bias}(\hat{\varepsilon}_n) \;=\; E[\hat{\varepsilon}_n - \varepsilon_n] \;=\; E[\hat{\varepsilon}_n] - E[\varepsilon_n]\,. \tag{1.3}$$

The error estimator $\hat{\varepsilon}_n$ is said to be *optimistically biased* if $\mathrm{Bias}(\hat{\varepsilon}_n) < 0$ and *pessimistically biased* if $\mathrm{Bias}(\hat{\varepsilon}_n) > 0$. It is *unbiased* if $\mathrm{Bias}(\hat{\varepsilon}_n) = 0$. The resubstitution error estimator is

usually optimistically biased.

The *deviation variance* is the variance of the deviation distribution:

$$\mathrm{Var}_{\mathrm{dev}}(\hat{\varepsilon}_n) \;=\; \mathrm{Var}(\hat{\varepsilon}_n - \varepsilon_n) \;=\; \mathrm{Var}(\hat{\varepsilon}_n) + \mathrm{Var}(\varepsilon_n) - 2\mathrm{Cov}(\varepsilon_n, \hat{\varepsilon}_n)\,. \tag{1.4}$$

Unlike in classical statistics, where estimators for fixed parameters are sought, here the quantity being estimated, namely ε_n, is random and thus a "moving target." This is why it is appropriate to consider the variance of the difference, $\mathrm{Var}(\hat{\varepsilon}_n - \varepsilon_n)$. However, if the classification rule is not overfitting, then $\mathrm{Var}(\varepsilon_n) \approx 0$ — in fact, overfitting could be defined as present if $\mathrm{Var}(\varepsilon_n)$ is large, since in that case the classification rule is learning the changing data and not the fixed underlying feature-label distribution. It follows, from the Cauchy-Schwartz Inequality that $\mathrm{Cov}(\varepsilon_n, \hat{\varepsilon}_n) \leq \sqrt{\mathrm{Var}(\varepsilon_n)\mathrm{Var}(\hat{\varepsilon}_n)} \approx 0$, and thus, from (1.4), $\mathrm{Var}(\hat{\varepsilon}_n - \varepsilon_n) \approx \mathrm{Var}(\hat{\varepsilon}_n)$. If an estimator is randomized, then it has additional *internal variance* $V_{\mathrm{int}} = \mathrm{Var}(\hat{\varepsilon}_n|S_n)$, which measures the variability due only to the internal random factors, while the full variance $\mathrm{Var}(\hat{\varepsilon}_n)$ measures the variability due to both the sample S_n and the internal random factors ξ. The following formula can be easily shown using the Conditional Variance Formula of probability theory:

$$\mathrm{Var}(\hat{\varepsilon}_n) \;=\; E[V_{\mathrm{int}}] + \mathrm{Var}(E[\hat{\varepsilon}_n|S_n])\,. \tag{1.5}$$

The first term on the right-hand side contains the contribution of the internal variance to the total variance. For nonrandomized $\hat{\varepsilon}_n$, $V_{\mathrm{int}} = 0$; for randomized $\hat{\varepsilon}_n$, $E[V_{\mathrm{int}}] > 0$.

The *root mean-square error* is the square root of the second moment of the deviation distribution:

$$\mathrm{RMS}(\hat{\varepsilon}_n) \;=\; \sqrt{E[(\hat{\varepsilon}_n - \varepsilon_n)^2]} \;=\; \sqrt{\mathrm{Bias}(\hat{\varepsilon}_n)^2 + \mathrm{Var}_{\mathrm{dev}}(\hat{\varepsilon}_n)} \tag{1.6}$$

The RMS is generally considered the most important error estimation performance metric. The other performance metrics appear within the computation of the RMS; indeed, all of the five basic moments — the expectations $E[\varepsilon_n]$ and $E[\hat{\varepsilon}_n]$, the variances $\mathrm{Var}(\varepsilon_n)$ and $\mathrm{Var}(\hat{\varepsilon}_n)$, and the covari-

ance $\mathrm{Cov}(\varepsilon_n, \hat{\varepsilon}_n)$ — appear within the RMS. Good error estimation performance requires that the bias, deviation variance, and RMS be as close as possible to zero.

This dissertation proposes and studies generalized resubstitution error estimators, which are defined in terms of arbitrary empirical measures. In addition to plain resubstitution, this family includes well-known error estimators, such as posterior-probability [39], Gaussian-process [40], bolstered resubstitution [22], and Bayesian [33, 34] error estimators. The empirical measures used in generalized resubstitution often contain hyperparameters that can be tuned to reduce the bias and variance of the estimator with respect to plain resubstitution.

The problem of practical error estimation is approached, in this dissertation, from the point of view of choosing an empirical measure $\hat{\nu}_n$ and applying the corresponding generalized resubstitution estimator, while tuning the hyperparameters of $\hat{\nu}_n$ to obtain low error estimation bias and variance. In addition to a general theoretical framework for generalized resubstitution, our contribution includes previously unavailable multi-class versions for some existing error estimators, and a new family of error estimators, called bolstered posterior-probability error estimation, which is an extension of the bolstered and posterior-probability estimators.

Chapter 2 includes an extensive empirical study using synthetic data and traditional classifiers, MNIST data and LeNet-5 convolutional neural network classifier, as well as a selection of UCI datasets[1] and fully connected neural network classifiers with various width and depth. In section 2.2.4.1, a generative model, consisting of multivariate Gaussian distributions for each of two classes, was used. Subsequently, performance of the generalized error estimators were analyzed for four classification rules were considered: linear support vector machine (SVM), nonlinear SVM with radial basis kernel function (RBF-SVM), classification tree (CART) with stopped splitting at 5 points per leaf node, and k-nearest neighbors (k-NN), with $k = 3, 5, 7$. Sections 2.2.4.2 presents empiricl results using the MNIST dataset and LeNet-5 convolutional neural network, whereas section 2.2.4.3 utilize real data sets from UCI repository and fully connected neural network classifiers with various depth and width, where depth is the number of deep layers and width is the number

[1] https://archive.ics.uci.edu/ml/datasets.php

of neurons in each layer.

1.2 Predicting Generalization in Deep Learning

Estimation of error and generalization in deep learning is undoubtedly among the most challenging topics in the field. Although true error and generalization gap may be well correlated, their fundamental basis is different. As described in section 1.1, the goal of error estimation is to measure the ability of a trained classifier in predicting the true label for future and unseen data. The generalization gap, on the other hand, explains how overfitted a model is on the training data. Generalization gap for a classifier is related to the deviation of standard resubstitution error estimate from the classifier's true error.

Formally, given a train dataset $\mathcal{D}_{train}$ and a test dataset $\mathcal{D}_{test}$, let $\psi_{\mathbf{w}}(\mathbf{x})$ be the network's prediction for an input $\mathbf{x}$, where $\mathbf{w}$ represents the parameters of the trained network. Then the generalization gap is defined as [41]:

$$g(\psi_{\mathbf{w}}; \mathcal{D}_{test}) = \frac{1}{|\mathcal{D}_{test}|} \sum_{(\mathbf{x},y)\in\mathcal{D}_{test}} \delta(\psi_{\mathbf{w}}(\mathbf{x}) = y) - \frac{1}{|\mathcal{D}_{train}|} \sum_{(\mathbf{x},y)\in\mathcal{D}_{train}} \delta(\psi_{\mathbf{w}}(\mathbf{x}) = y) \tag{1.7}$$

where$(\mathbf{x}, y)$ is pair of an input $\mathbf{x}$ and its true label.

Despite a substantial body of research, a thorough analysis of the underlying root factors that drive neural network generalization remains unanswered question [18, 42, 43, 44]. Through an extensive empirical study, [7] recently investigated several prominent complexity measures by controlling hyperparameters of the networks; this study suggested the insufficiency of traditional methods for evaluation of generalization in deep learning. Yet, empirical studies of generalization metrics can provide insights into why certain models generalize well [45], and therefore, help with development of better models.

Recent literature shows the increasing concern in deep learning community regarding the error estimation and predicting generalization of models; though the majority of generalization bounds are only tested on a small number of models [45, 46]. [47] recently provided a summary of generalization in deep neural networks. Among the proposed generalization measures, principled

complexity measures like PAC bounds are most attractive to statisticians due to their theoretical properties [48, 49, 50, 51]. On the other hand, practitioners have investigated several generalization measures through extensive empirical studies [7, 52], among which data augmentation appear to be continuously under active research.

Some of the recent approaches that use data augmentation include *Always Generalize* [53] which proposes a generalization measure that assesses a classifier's robustness to various types of data augmentation, *GAN-based* [54] which generates augmented data using GAN, *mixup* [55] which creates convex combinations of pairs of examples and their labels, *manifold-mixup* [56] which proposes using a regularizer that is based on interpolations of hidden representations. The latter leads to training flatter class-representations and avoids overconfident neural networks. Along the same direction, [57, 58] have practiced modifications of mixup and maifold-mixup methods to augment data towards predicting generalization in deep learning (PGDL).

Chapter 3 of this dissertation presents extensive empirical study on hundreds of convolutional neural networks, using the NeurIPS 2020 PGDL competition database [41]. The focus, in this chapter, is shifted from error estimation to predicting generalization gap by means of data augmentation. Bolstering, which provides systematic data augmentation, showed success in the majority of the experiments in sections 2.2.4 and 2.2.4.3. Yet, the bolstering approach that was employed requires a measure of distance between the training samples. For colored images, however, distance between samples is not straightforward and semantically meaningful [59].

For PGDL, section 3.2 proposes three novel approaches, for data augmentation based measures, from three viewpoints, and discuss the advantages and drawbacks of each method: 1- searching for an optimal set of parameters for augmentation factors, 2- generating multiple datasets using several ranges for the augmentation factor, and 3- generating a few data sets using distinct augmentation factors and multiplying each dataset by a weight according to the amount and strength of the corresponding augmentation factor and its type. The latter was inspired by the work in [53], the first runner up solution to the PGDL competition.

1.3 Hybrid Compartmental Model

Compartmental modeling is a popular method for estimating the kinetics of a naturally occurring substance (tracee) by tracking the kinetics of its isotope (tracer). It has been widely used for describing kinetics of drugs in pharmacokinetics [60, 61, 62] and protein turnover in metabolism assessments [1, 63, 64]. A compartmental model is characterized by its graphical structure and is identified by the values of its parameters. Each compartment is defined ideally as a well-mixed and kinetically homogeneous pool [63].

Currently, SAAM II [65, 66, 67] is the most common software for compartmental modeling. Examples of other tools for pharmacokinetic parameter estimation are SimuSolv [68][69] and Excel add-in program PKSolver [70]. Despite the success of conventional compartmental models (CCMs) in describing the metabolism of a single amino acid, quantifying the rate of conversion from one amino acid to another is not yet fully understood, particularly if the conversion occurs between non-accessible pools. This can be partially attributed to the graphical structures commonly used in CCMs.

CCMs require a modeler to specify the structure of the compartmental graph in every detail prior to fitting the experimental data. In particular, a modeler needs to specify the exact location of the fluxes between the pools. This demands extra effort to try and test multiple graphs until the best fit is achieved. A Hybrid Compartmental Model (HCM) structure, on the other hand, considers whole-body conversion rates, fits the data, and associates the conversion rate to one of the pools that leads to the best estimate of the conversion flux.

2. GENERALIZED RESUBSTITUTION ERROR ESTIMATORS

This chapter [1] explains the concept of generalized resubstitution error estimators and presents theorems as well as extensive empirical results illustrating the behaviour of generalized resubstitution error estimators.

2.1 Definitions

Working formally, given a feature vector $\mathbf{x} \in R^d$, a classifier ψ outputs a label $y = \psi(\mathbf{x}) \in \{0, 1, \ldots, c-1\}$. A classification rule takes sample data $S_n = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$ and produces a trained classifier ψ_n. The quantity of interest is the classification error probability:

$$\varepsilon_n \;=\; \nu(\{(\mathbf{x}, y) : \psi_n(\mathbf{x}) \neq y\})\,, \tag{2.1}$$

where the probability measure ν is supported on $R^d \times \{0, 1, \ldots, c-1\}$ and is the distribution of the pair of random variables $(\mathbf{X}, Y)$. A generalized resubstitution estimator $\hat{\varepsilon}_n$ is defined as:

$$\hat{\varepsilon}_n \;=\; \hat{\nu}_n(\{(\mathbf{x}, y) : \psi_n(\mathbf{x}) \neq y\})\,, \tag{2.2}$$

where $\hat{\nu}_n$ is a *generalized empirical probability measure*, i.e., a random probability measure supported on $R^d \times \{0, 1, \ldots, c-1\}$ that is a function of the sample data. If $\hat{\nu}_n$ is sufficiently close to ν, in a suitable sense, then $\hat{\varepsilon}_n$ is a good estimator of ε_n.

The basic example is provided by the standard empirical measure ν_n putting mass $1/n$ on each training point $(\mathbf{X}_i, Y_i)$, which yields the plain resubstitution error estimator:

$$\hat{\varepsilon}_n^{\,r} \;=\; \nu_n(\{(\mathbf{x}, y) : \psi_n(\mathbf{x}) \neq y\}) \;=\; \frac{1}{n}\sum_{i=1}^{n} I(\psi_n(X_i) \neq Y_i)\,, \tag{2.3}$$

where $I(\cdot)$ is an indicator variable. Notice that ν_n has no hyperparameters that allow estimator

[1] Part of this chapter is reprinted with permission [71].

bias and variance to be tuned. In the two-class case, the Vapnik-Chervonenkis (VC) theorem [37] guarantees that the empirical measure converges uniformly almost surely to the true measure, i.e., $\sup_{A\in\mathcal{A}} |\nu_n(A) - \nu(A)| \to 0$ as $n \to \infty$, with probability 1 and regardless of ν, provided that the family of sets $\mathcal{A}$ is small in a precise sense. If $\mathcal{A}$ is the family of sets $\{(\mathbf{x}, y) : \psi_n(\mathbf{x}) \neq y\}$ over all possible classifiers ψ_n, and $\mathcal{A}$ is small in the sense that the associated classification algorithm has a finite *VC dimension*, then the VC Theorem implies that $\hat{\varepsilon}_n^r$ becomes arbitrarily close to ε_n as $n \to \infty$ with probability 1, in a distribution-free manner. It is a simple corollary that if, in turn, the generalized empirical measure converges uniformly almost surely to the empirical measure, i.e., $\sup_{A\in\mathcal{A}} |\hat{\nu}_n(A) - \nu_n(A)| \to 0$ as $n \to \infty$, with probability 1 and regardless of ν, then $\hat{\varepsilon}_n$ also becomes arbitrarily close to ε_n as $n \to \infty$, in a distribution-free manner. In other words, as sample size increases, the generalized resubstitution error estimator should look more and more like the plain resubstitution estimator.

Let the feature vector $\mathbf{X} \in R^d$ and the label $Y \in R$ be jointly distributed with corresponding probability measure ν, such that $\nu(R^d \times \{0, 1, \ldots, c-1\}) = 1$. An event is a Borel set $A \subseteq R^d \times \{0, 1, \ldots, c-1\}$. A *classifier* ψ is a Borel-measurable function from R^d to $\{0, 1, \ldots, c-1\}$. In practice, one collects i.i.d. *training data* $S_n = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$, where each pair $(\mathbf{X}_i, Y_i)$ is distributed as $(\mathbf{X}, Y)$, and designs a classifier $\psi_n = \Psi_n(S_n)$, by means of a classification rule Ψ_n. The classification error ε_n is the probability of the misclassification event $A = \{(\mathbf{x}, y) : \psi_n(\mathbf{x}) \neq y\}$:

$$\varepsilon_n \;=\; \nu(\{(\mathbf{x}, y) : \psi_n(\mathbf{x}) \neq y\})\,. \tag{2.4}$$

We define a *generalized empirical measure* $\hat{\nu}_n$ to be a random probability measure supported on $R^d \times \{0, 1, \ldots, c-1\}$ almost surely that is a function of the data S_n. This definition includes the standard empirical measure ν_n that puts discrete mass $1/n$ on each data point:

$$\nu_n \;=\; \frac{1}{n} \sum_{i=1}^{n} \delta_{\mathbf{X}_i, Y_i}\,, \tag{2.5}$$

where $\delta_{\mathbf{X}_i,Y_i}$ is the (random) point measure located at $(\mathbf{X}_i, Y_i)$, defined by

$$\delta_{\mathbf{X}_i,Y_i}(A) \;=\; I((\mathbf{X}_i, Y_i) \in A)\,, \tag{2.6}$$

for each event A. Hence, $\nu_n(A)$ is simply the fraction of points in S_n that are contained in A.

Plugging in the standard empirical measure ν_n for ν in (2.4) yields the fraction of errors committed by ψ_n on S_n, i.e., the standard resubstitution error estimator:

$$\hat{\varepsilon}_n^r \;=\; \nu_n(\{(\mathbf{x}, y) : \psi_n(\mathbf{x}) \neq y\}) \;=\; \frac{1}{n}\sum_{i=1}^{n} I(\psi_n(\mathbf{X}_i) \neq Y_i)\,. \tag{2.7}$$

By analogy, plugging in a generalized empirical measure $\hat{\nu}_n$ for ν in (2.4) results in a *generalized resubstitution error estimator*:

$$\hat{\varepsilon}_n \;=\; \hat{\nu}_n(\{(\mathbf{x}, y) : \psi_n(\mathbf{x}) \neq y\})\,. \tag{2.8}$$

As we will see later, many generalized empirical measures have tunable hyperparameters, which can be adjusted in order to reduce error estimation bias and variance.

We consider below several examples of generalized resubstitution estimators, which are based on a broad family of generalized empirical measures of the form:

$$\hat{\nu}_n \;=\; \frac{1}{n}\sum_{i=1}^{n} \beta_{n,\mathbf{X}_i,Y_i}\,. \tag{2.9}$$

where $\beta_{n,\mathbf{X}_i,Y_i}$ is a random probability measure depending on the training point $\mathbf{X}_i, Y_i$. Comparing this to (2.5), we realize that the empirical probability measure in (2.9) can be seen as smoothed version of the standard empirical probability measure, where $\beta_{n,\mathbf{X}_i,Y_i}$ provides a smoothed version of the point measure $\delta_{\mathbf{X}_i,Y_i}$. This is not the only case of useful empirical probability measure for generalized resubstitution error estimation; Section 2.3 gives examples that are not of the form in (2.9).

Notice that a sufficient condition for (2.14) in Theorem 1 is the uniform convergence of the smoothed measure $\beta_{n,\mathbf{X}_i,Y_i}$ to the point measure $\delta_{\mathbf{X}_i,Y_i}$ for any training point $(\mathbf{X}_i, Y_i)$:

$$\sup_{A\in\mathcal{A}} |\beta_{n,\mathbf{X}_i,Y_i}(A) - \delta_{\mathbf{X}_i,Y_i}(A)| \;\to\; 0 \quad \text{a.s.} \tag{2.10}$$

Next, we consider the natural large-sample question of whether a generalized resubstitution error estimator approaches the true classification error as the training sample size increases to infinity. In particular, we are interested in the questions of consistency, i.e., whether $\hat{\varepsilon}_n \to \varepsilon_n$ a.s. as well as asymptotic unbiasedness, i.e., whether $E[\hat{\varepsilon}_n] \to E[\varepsilon_n]$, as $n \to \infty$. It turns out that the basic tool to address these questions is provided by the Vapnik-Chervonenkis Theorem [72, 37].

Note that for any fixed event $A \subseteq R^d \times \{0, 1, \ldots, c-1\}$, the standard empirical measure satisfies

$$\nu_n(A) \;=\; \frac{1}{n}\sum_{i=1}^{n} I((\mathbf{X}_i, Y_i) \in A) \;\to\; \nu(A) \quad \text{a.s.}, \tag{2.11}$$

by the Strong Law of Large Numbers (SLLN). Hence, for a fixed classifier ψ, we can plug in the fixed set $A = \{(\mathbf{x}, y) : \psi(\mathbf{x}) \neq y\}$ in the previous equation and conclude that the empirical classification error converges to the true error with probability 1. But this is not enough to obtain results concerning classifiers ψ_n designed from the data S_n, since these concern events $A_n = \{(\mathbf{x}, y) : \psi_n(\mathbf{x}) \neq y\}$, which are not fixed. What is needed instead is a *uniform* SLLN:

$$\sup_{A\in\mathcal{A}} |\nu_n(A) - \nu(A)| \;\to\; 0 \quad \text{a.s.}, \tag{2.12}$$

where $\mathcal{A}$ is a family of sets that must contain all events $A_n = \{(\mathbf{x}, y) : \psi_n(\mathbf{x}) \neq y\}$ that can be produced by the classification rule. In the case $c = 2$, it is known that (2.12) holds if $\mathcal{A}$ is small enough, in the sense that its *VC dimension* $V_{\mathcal{A}}$ is finite. The VC dimension is a nonnegative integer that measures the size of $\mathcal{A}$; a smaller VC dimension implies that the classification rule is more constrained and less sensitive to the data S_n, i.e., it is less prone to overfitting at a fixed sample size. For example, a linear classification rule in R^d has VC dimension $d + 1$, which is finite, and

small in low-dimensional spaces. If $V_{\mathcal{A}} < \infty$, the *Vapnik-Chervonenkis Theorem* [37, 73] yields the inequality:

$$P\left(\sup_{A\in\mathcal{A}} |\nu_n(A) - \nu(A)| > \tau\right) \leq 8(n+1)^{V_{\mathcal{A}}} e^{-n\tau^2/32}, \quad \text{for all } \tau > 0\,. \tag{2.13}$$

The term $e^{-n\tau^2/32}$ dominates, and the bound decreases exponentially fast as $n \to \infty$. It then follows from the First Borel-Cantelli Lemma that $\sup_{A\in\mathcal{A}} |\nu_n(A) - \nu(A)| \to 0$ a.s. [73, Thm A.8]. (Strictly speaking, it is necessary to assume that events of the kind $\sup_{A\in\mathcal{A}} |\nu_n(A) - \nu(A)| > \tau$ are measurable. General conditions to ensure that are discussed in [74]; such conditions are tacitly assumed throughout this work.)

Theorem 1. *In the case $c = 2$, if the family $\mathcal{A}$ of all events $A_n = \{(\mathbf{x}, y) : \psi_n(\mathbf{x}) \neq y\}$ that can be produced by a classification rule has finite VC dimension, and the generalized empirical measure converges uniformly to the standard empirical measure as sample size increases, i.e.,*

$$\sup_{A\in\mathcal{A}} |\hat{\nu}_n(A) - \nu_n(A)| \to 0 \quad \textit{a.s.}, \tag{2.14}$$

then the generalized resubstitution error estimator is consistent, $\hat{\varepsilon}_n \to \varepsilon_n$ a.s., as well as asymptotically unbiased, $E[\hat{\varepsilon}_n] \to E[\varepsilon_n]$, as $n \to \infty$, regardless of the feature-label distribution.

Proof. . From

$$\begin{aligned} |\hat{\nu}_n(A) - \nu(A)| &= |\hat{\nu}_n(A) - \nu_n(A) + \nu_n(A) - \nu(A)| \\ &\leq |\hat{\nu}_n(A) - \nu_n(A)| + |\nu_n(A) - \nu(A)|\,, \end{aligned} \tag{2.15}$$

it follows that

$$\sup_{A\in\mathcal{A}} |\hat{\nu}_n(A) - \nu_n(A)| \leq \sup_{A\in\mathcal{A}} |\hat{\nu}_n(A) - \nu_n(A)| + \sup_{A\in\mathcal{A}} |\nu_n(A) - \nu(A)|\,. \tag{2.16}$$

The first term on the right converges to zero a.s. by hypothesis, while the second term does so by

virtue of the VC Theorem. Hence, the left-hand side must also converge to zero a.s.,

$$\sup_{A \in \mathcal{A}} |\hat{\nu}_n(A) - \nu(A)| \;\to\; 0 \quad \text{a.s.} \tag{2.17}$$

Since

$$|\hat{\varepsilon}_n - \varepsilon_n| \;=\; |\hat{\nu}_n(A_n) - \nu(A_n)| \;\leq\; \sup_{A \in \mathcal{A}} |\hat{\nu}_n(A) - \nu(A)|, \tag{2.18}$$

it follows that $|\hat{\varepsilon}_n - \varepsilon_n| \to 0$ a.s. and the generalized resubstitution estimator is consistent.

Furthermore, since all random variables are uniformly bounded, the Dominated Convergence Theorem implies [73, Thm A.7] that

$$|E[\hat{\varepsilon}_n - \varepsilon_n]| \;\leq\; E[|\hat{\varepsilon}_n - \varepsilon_n|] \;\to\; 0\,, \tag{2.19}$$

i.e., the generalized resubstitution error estimator is asymptotically unbiased. All of these results are distribution-free, holding for any feature-label distribution ν. □

2.2 Generalized Resubstitution based on Smoothing the Error Count

In this section, several error estimators based on the smoothing error count are described.

2.2.1 Bolstered Resubstitution

Given an event $A \subseteq R^d \times \{0, 1, \ldots, c-1\}$, define its slices by

$$A_y \;=\; \{\mathbf{x} \in R^d \mid (\mathbf{x}, y) \in A\}\,, \quad y = 0, 1, \ldots, c-1\,. \tag{2.20}$$

It is clear that A_y is an event (i.e., a Borel set) in R^d for each y. Note that $\delta_{\mathbf{X}_i, Y_i}(A) = \delta_{\mathbf{X}_i}(A_{Y_i})$, where $\delta_{\mathbf{X}_i}$ is a point measure in R^d. Similarly, let $\beta_{n,\mathbf{X}_i,Y_i}(A) \;=\; \mu_{n,\mathbf{X}_i,Y_i}(A_{Y_i})$, where $\mu_{n,\mathbf{X}_i,Y_i}$ is an empirical measure on R^d. Though discrete bolstering is possible, in practice the *bolstering measure* $\mu_{n,\mathbf{X}_i,Y_i}$ is assumed to be absolutely continuous, with density function $p_{n,\mathbf{X}_i,Y_i}(\mathbf{x})$, so that

$$\beta_{n,\mathbf{X}_i,Y_i}(A) \;=\; \int_{A_{Y_i}} p_{n,\mathbf{X}_i,Y_i}(\mathbf{x})\, d\mathbf{x}\,. \tag{2.21}$$

The probability densities $p_{n,\mathbf{X}_i,Y_i}$ are called *bolstering kernels*. Plugging $\beta_{n,\mathbf{X}_i,Y_i}(A)$ in (2.9), and then in (2.8), yields the *bolstered resubstitution error estimator* proposed in [38] (here extended to the multi-class case). Note that the misclassification event $\{(\mathbf{x}, y) : \psi_n(\mathbf{x}) \neq y\}$ has slices $A_y = \{\mathbf{x} : \psi_n(\mathbf{x}) \neq y\}$ (If $c = 2$, these reduce to the complementary decision regions $\{\mathbf{x} : \psi_n(\mathbf{x}) = 0\}$ and $\{\mathbf{x} : \psi_n(\mathbf{x}) = 1\}$.) The bolstered resubstitution error estimator can be thus written as:

$$\hat{\varepsilon}_n^{br} \;=\; \frac{1}{n}\sum_{i=1}^{n}\int_{\{\mathbf{x}:\psi_n(\mathbf{x})\neq Y_i\}} p_{n,\mathbf{X}_i,Y_i}(\mathbf{x})\,d\mathbf{x}\,. \tag{2.22}$$

The integral in (2.22) gives the error contribution made by training point $(\mathbf{X}_i, Y_i)$; these are real-valued numbers between 0 and 1, unlike plain resubstitution, in which contributions are 0 or 1. Notice that this allows counting partial errors, including errors for correctly classified points that are near the decision boundary. The error contribution made by each point is the area of the shaded region divided by the area of the entire disk. The bolstered resubstitution error is the sum of all contributions divided by the number of points. Reproduced from [22].

In some cases (see an example below), it is possible to solve the integrals in (2.22) analytically, and the estimator is fast and low-variance. Otherwise, one has to apply approximations. For example, simple Monte-Carlo integration yields:

$$\hat{\varepsilon}_n^{br} \;\approx\; \frac{1}{nM}\sum_{i=1}^{n}\sum_{j=1}^{M} I(\psi_n(\mathbf{X}_{ij}^{\mathrm{MC}}) \neq Y_i)\,, \tag{2.23}$$

where $\{\mathbf{X}_{ij}^{\mathrm{MC}}; j = 1, \ldots, M\}$ are random points drawn from the density $p_{n,i}$, for $i = 1, \ldots, n$. In this case, the estimation procedure is randomized due to MC sampling.

The most common choice for bolstering kernels are multivariate Gaussian densities with mean $\mathbf{X}_i$ and covariance matrix $K_{n,\mathbf{X}_i,Y_i}$:

$$p_{n,\mathbf{X}_i,Y_i}(\mathbf{x}) \;=\; \frac{1}{\sqrt{(2\pi)^d\,\det(K_{n,\mathbf{X}_i,Y_i})}}\exp\left(-\frac{1}{2}\,(\mathbf{x}-\mathbf{X}_i)^T K_{n,\mathbf{X}_i,Y_i}^{-1}(\mathbf{x}-\mathbf{X}_i)\right)\,, \tag{2.24}$$

If the matrices $K_{n,\mathbf{X}_i,Y_i}$ are diagonal, with freely adjustable diagonal elements, then the procedure

is known as *Naive-Bayes bolstering* [75].

It can be shown that if $c = 2$ and $\psi_n(\mathbf{x}) = I(\mathbf{a}_n^T\mathbf{x} + b_n > 0)$ is a linear classifier, then the Gaussian-bolstered resubstitution error estimator can be computed efficiently as

$$\hat{\varepsilon}_n^{\mathrm{GS}} = \frac{1}{n}\sum_{i=1}^{n}\Phi\left(\frac{(-1)^{1-Y_i}(\mathbf{a}_n^T\mathbf{X}_i + b_n)}{\sqrt{\mathbf{a}_n^T K_{n,\mathbf{X}_i,Y_i}\mathbf{a}_n}}\right), \tag{2.25}$$

where $\Phi(x)$ is the cumulative distribution function of a standard $N(0,1)$ Gaussian random variable. In more general cases, one needs to employ approximations, such as Monte-Carlo sampling.

We consider below in detail the multivariate spherical Gaussian case with $K_{n,\mathbf{X}_i,Y_i} = \sigma_{n,Y_i}^2 I_d$. The hyperparameters here are the c standard deviations $\sigma_{n,j}$ for each class (here, kernel variance is not a function of $\mathbf{X}_i$). This demands much less effort than the Naive-Bayes case, which requires in general nd hyperparameters. (Nevertheless, the analysis below could be extended to the Naive-Bayes case with more effort.) In [22] (which considers only the case $c = 2$), the hyperparameters $\sigma^{n,j}$ are estimated by making the median distance of a point sampled from the corresponding kernel to the origin match the mean minimum distance $\hat{d}_{n,j}$ among training points in class j:

$$\hat{d}_{n,j} = \frac{1}{n_j}\sum_{i=1}^{n_j}||\mathbf{X}_{ij} - \mathbf{X}'_{ij}||, \quad j = 0, 1, \ldots, c-1, \tag{2.26}$$

where n_j is the number of points from class j ($n_j \geq 2$ is assumed), $\mathbf{X}_{ij}$ is a point in class j, and $\mathbf{X}'_{ij}$ is its nearest neighbor in class j.

Now, let R be the random variable corresponding to the distance to the origin of a point randomly selected from a unit-variance spherically-symmetric density with cumulative distribution function $F_R(x)$. The median distance of such a point to the origin is $\alpha_d = F_R^{-1}(1/2)$, where the subscript d indicates explicitly that α_d depends on the dimensionality. If the density has variance σ^2, all distances get multiplied by σ. Hence, $\sigma_{n,j}$ is the solution of the equation $\sigma_{n,j}\alpha_d = \hat{d}_{n,j}$, i.e.,

$$\sigma_{n,j} = \frac{\hat{d}_{n,j}}{\alpha_d}, \quad j = 0, 1, \ldots, c-1 \tag{2.27}$$

As shown in the proof of Theorem 2, under the assumption that ν is absolutely continuous, σ_n^j decreases to zero as sample size increases, and the bolstering kernel tends to a point mass at $\mathbf{X}_j$. The constant α_d can be interpreted as a "dimensionality correction," which adjusts the value of the estimated mean distance to account for the feature space dimensionality. Indeed, this approach to selecting the hyperparameters is applicable to any spherically-symmetric kernel. In the case of spherical Gaussian densities, R is distributed as a *chi* random variable with d degrees of freedom, and the median $\alpha_d = F_R^{-1}(1/2)$ can be easily computed numerically. For example, the values up to five dimensions are $\alpha_1 = 0.674$, $\alpha_2 = 1.177$, $\alpha_3 = 1.538$, $\alpha_4 = 1.832$, $\alpha_5 = 2.086$.

Next, we consider the asymptotic properties of the bolstered resubstitution estimator with spherical Gaussian kernels in the case $c = 2$. First, we define a classification rule to be *regular* if it produces "thin" decision boundaries. In the general case $c \geq 2$, the decision boundary D of classifier ψ_n is

$$D = \bigcup_{y=0}^{c-1} \partial A_y \tag{2.28}$$

where $A_y = \{\mathbf{x} : \psi_n(\mathbf{x}) \neq y\}$ are the misclassification event slices, as defined previously, and a point is in ∂A_y if it does not belong to the interior of either A_y or A_y^c. A classification rule Ψ_n is regular if D has Lebesgue measure zero for all its classifiers ψ_n. If the distribution of $\mathbf{X}$ is absolutely continuous (with respect to Lebesgue measure), i.e., if $\mathbf{X}$ is a continuous feature vector in the usual sense, then the probability that a training point $\mathbf{X}_i$ sits on the decision boundary is zero. The vast majority, if not all, classification rules encountered in practice are regular.

Theorem 2. *In the case $c = 2$, if Ψ_n is a regular classification rule with finite VC dimension and the distribution of $\mathbf{X}$ is absolutely continuous, then the bolstered resubstitution estimator with spherical Gaussian kernels, with hyperparameters $\sigma_{n,j}$ selected as in* (2.27)*, is consistent and asymptotically unbiased.*

Proof. By virtue of Theorem 1 and (2.9), it suffices to show that (2.10) holds, which in the present

case reduces to proving that

$$\sup_{A\in\mathcal{A}} |\mu_{n,\mathbf{X}_i,Y_i}(A_{Y_i}) - \delta_{\mathbf{X}_i}(A_{Y_i})| \;\to\; 0 \quad \text{a.s.}\,, \tag{2.29}$$

where $\mathcal{A}$ s the family of all events $\{(\mathbf{x}, y) : \psi_n(\mathbf{x}) \neq y\}$ that can be produced by the classification rule. Notice that, for any given $\tau > 0$, whenever $\sup_{A\in\mathcal{A}} |\mu_{n,\mathbf{X}_i,Y_i}(A_{Y_i}) - \delta_{\mathbf{X}_i}(A_{Y_i})| > \tau$, there is an $A^* \in \mathcal{A}$, which is a function of the data, such that $|\mu_{n,\mathbf{X}_i,Y_i}(A^*_{Y_i}) - \delta_{\mathbf{X}_i}(A^*_{Y_i})| > \tau$, with probability 1. In other words,

$$P\left(|\mu_{n,\mathbf{X}_i,Y_i}(A^*_{Y_i}) - \delta_{\mathbf{X}_i}(A^*_{Y_i})| > \tau \;\middle|\; \sup_{A\in\mathcal{A}} |\mu_{n,\mathbf{X}_i,Y_i}(A_{Y_i}) - \delta_{\mathbf{X}_i}(A^*_{Y_i})| > \tau\right) \;=\; 1\,, \tag{2.30}$$

which in turn implies that

$$P\left(\sup_{A\in\mathcal{A}} |\mu_{n,\mathbf{X}_i}(A_{Y_i}) - \delta_{\mathbf{X}_i}(A_{Y_i})| > \tau\right) \;\leq\; P\left(|\mu_{n,\mathbf{X}_i}(A^*_{Y_i}) - \delta_{\mathbf{X}_i}(A^*_{Y_i})| > \tau\right)\,. \tag{2.31}$$

By regularity of the classification rule, $\mathbf{X}_i$ belongs to the interior of $A^*_{Y_i}$ or $(A^*_{Y_i})^c$ with probability 1. Hence, we can find an open ball $B(\mathbf{X}_i, \rho)$ centered on $\mathbf{X}_i$ that is entirely contained in A_{Y_i} or $A^c_{Y_i}$. If the variance $\sigma^2_{n,i}$ tends to zero as at least $O(n)$, the Gaussian measure will concentrate exponentially fast inside such a ball, such that $P\left(|\mu_{n,\mathbf{X}_i}(A^*_{Y_i}) - \delta_{\mathbf{X}_i}(A^*_{Y_i})| > \tau\right) \to 0$ exponentially fast, for any $\tau > 0$, and the Theorem is proved, via (2.31) and the First Borel-Cantelli Lemma.

From (2.26) and (2.27), it suffices to show that the nearest neighbor $\mathbf{X}'_{ij}$ to $\mathbf{X}_{ij}$ converges to $\mathbf{X}_{ij}$ exponentially fast as $n \to \infty$. Note that, for any $\tau > 0$,[2]

$$P(||\mathbf{X}'_{ij} - \mathbf{X}_{ij}|| > \tau) = P(||\mathbf{X}_{kj} - \mathbf{X}_{ij}|| > \tau;\ \text{for all } k \neq i) = (1 - P(||\mathbf{X}_{lj} - \mathbf{X}_{ij}|| < \tau))^{n_j}, \tag{2.32}$$

for some $l \neq i$. Notice that, since $P(Y = j) > 0$, $n_j \to \infty$ as $O(n)$ a.s. as $n \to \infty$. If we can show that $P(||\mathbf{X}_{lj} - \mathbf{X}_{ij}|| < \tau) > 0$, then it follows from (2.32) that $P(||\mathbf{X}_{lj} - \mathbf{X}|| > \tau) \to 0$

[2]Equation (2.32) appears in a similar context in the proof of the Cover-Hart Theorem for nearest-neighbor classification [76]. The rest of the argument is distinct.

exponentially fast a.s. and the claim is proved. To ease notation, let $\mathbf{Z}' = \mathbf{X}'_{ij}$ and $\mathbf{Z} = \mathbf{X}_{ij}$. Since $\mathbf{Z}'$ and $\mathbf{Z}$ are independent and identically distributed with density $p_{\mathbf{X}}$, $\mathbf{Z}' - \mathbf{Z}$ has a density $p_{\mathbf{Z}'-\mathbf{Z}}$, given by the classical convolution formula:

$$p_{\mathbf{Z}'-\mathbf{Z}}(\mathbf{w}) \;=\; \int p_{\mathbf{X}}(\mathbf{w}+\mathbf{u})\, p_{\mathbf{X}}(\mathbf{w})\, d\mathbf{u}\,. \tag{2.33}$$

From this, we have $p_{\mathbf{Z}'-\mathbf{Z}}(\mathbf{0}) = \int p^2_{\mathbf{X}}(\mathbf{u})\, d\mathbf{u} > 0$. It follows, by continuity of the integral, that $p_{\mathbf{Z}'-\mathbf{Z}}$ must be nonzero in a neighborhood of $\mathbf{0}$, i.e., $P(||\mathbf{Z}'-\mathbf{Z}|| < \tau) > 0$, as was to be shown. $\square$

2.2.2 Posterior-Probability Generalized Resubstitution

The bolstered empirical measure relies on measures $\mu_{n,\mathbf{X}_i}$ on R^d, which provide smoothing in the $\mathbf{X}$ direction. If one performs smoothing in the Y direction, the so-called posterior-probability empirical measure results.

Given an event $A \subseteq R^d \times \{0, 1, \ldots, c-1\}$, define the slices

$$A_{\mathbf{x}} \;=\; \{y \in \{0, 1, \ldots, c-1\} \mid (\mathbf{x}, y) \in A\}\,, \quad \mathbf{x} \in R^d\,. \tag{2.34}$$

(Compare to the slices in (2.20).) Note that $\delta_{\mathbf{X}_i,Y_i}(A) = \delta_{Y_i}(A_{\mathbf{X}_i})$, where δ_{Y_i} is a point measure on $\{0, 1, \ldots, c-1\}$. Similarly, let $\beta_{n,\mathbf{X}_i,Y_i}(A) = \eta_{n,\mathbf{X}_i,Y_i}(A_{\mathbf{X}_i})$, where $\eta_{n,\mathbf{X}_i,Y_i}$ is an empirical measure on $\{0, 1 \ldots, c-1\}$. This is called a *posterior-probability measure* as $\eta_{n,\mathbf{X}_i,Y_i}(A_{\mathbf{X}_i})$ is to be interpreted as a "posterior-probability" estimate $\widehat{P}_n(Y_i \in A_{\mathbf{X}_i} \mid \mathbf{X} = \mathbf{X}_i)$. Plugging $\beta_{n,\mathbf{X}_i,Y_i}(A)$ in (2.9), and then in (2.8), yields the *posterior-probability resubstitution error estimator* (e.g., see [39], here extended to the multi-class case).

If $A = \{(\mathbf{x}, y) : \psi_n(\mathbf{x}) \neq y\}$ is the misclassification event, then $A_{\mathbf{x}} = \{\psi_n(\mathbf{x})\}^c$. Using the $\widehat{P}_n$ notation, it is easy to see that the posterior-probability resubstitution error estimator can be written as:

$$\hat{\varepsilon}_n^{\,\mathrm{ppr}} \;=\; \frac{1}{n}\sum_{i=1}^{n} \widehat{P}_n(\psi_n(\mathbf{X}_i) \neq Y_i \mid \mathbf{X} = \mathbf{X}_i)\,. \tag{2.35}$$

Here, $\widehat{P}_n(\psi_n(\mathbf{X}_i) \neq Y_i \mid \mathbf{X} = \mathbf{X}_i)$ is the error contribution made by training point $(\mathbf{X}_i, Y_i)$, rather

than 0 or 1 as in plain resubstitution. The idea is that if one is more confident that the classifier disagrees with the training label, this error should count more, and the reverse is true if one is not. This smoothes the error count of plain resubstitution and reduces variance.

The simplest concrete example is afforded by k-nearest neighbor (kNN) posterior probability estimation. Let $\{y^1(\mathbf{x}), \ldots, y^k(\mathbf{x})\}$ denote the labels of the k nearest training points to $\mathbf{x}$, for $k = 1, \ldots, n$. The k-nearest-neighbor (kNN) posterior probability measure is defined by

$$\widehat{P}_n(Y = y \mid \mathbf{X} = \mathbf{x}) \;=\; \frac{1}{k} \sum_{j=1}^{k} I(y^j(\mathbf{x}) = y)\,, \tag{2.36}$$

for $\mathbf{x} \in R^d$. This makes sense since the more labels y there are in the neighborhood of $\mathbf{x}$, the more likely it should be that its label is y. Plugging (2.36) into (2.35) leads to the kNN posterior-probability error estimator:

$$\hat{\varepsilon}_n^{\mathrm{kNN}} \;=\; \frac{1}{nk} \sum_{i=1}^{n} \sum_{j=1}^{k} I(\psi_n(\mathbf{X}_i) \neq y^j(\mathbf{X}_i))\,, \tag{2.37}$$

Clearly, the case $k = 1$ reduces to plain resubstitution.

2.2.3 Bolstered Posterior-Probability Generalized Resubstitution

A novel class of generalized resubstitution estimator results if one performs smoothing in both the $\mathbf{X}$ and Y directions. Notice that $\delta_{\mathbf{X}_i,Y_i}(A) = \delta_{\mathbf{X}_i}(A_{Y_i})\delta_{Y_i}(A_{\mathbf{X}_i})$, where the slices A_y and $A_{\mathbf{x}}$ are defined in (2.20) and (2.34), respectively. Let $\beta_{n,\mathbf{X}_i,Y_i}(A) = \mu_{n,\mathbf{X}_i,Y_i}(A_{Y_i})\eta_{n,\mathbf{X}_i,Y_i}(A_{\mathbf{X}_i})$, where $\mu_{n,\mathbf{X}_i,Y_i}$ and $\eta_{n,\mathbf{X}_i,Y_i}$ are respectively the bolstered and posterior-probability empirical measures defined previously. Plugging $\beta_{n,\mathbf{X}_i,Y_i}(A)$ in (2.9), and then in (2.8), yields the *bolstered posterior-probability resubstitution error estimator*, a new estimator that combines features of bolstered and posterior-probability resubstitution. Using the $\widehat{P}_n$ notation, it is easy to see that the bolstered posterior-probability resubstitution error estimator can be written as:

$$\hat{\varepsilon}_n^{\mathrm{bppr}} \;=\; \frac{1}{n} \sum_{i=1}^{n} \left(\int_{\{\mathbf{x}:\psi_n(\mathbf{x}) \neq Y_i\}} p_{n,\mathbf{X}_i,Y_i}(\mathbf{x})\, d\mathbf{x} \right) \widehat{P}_n(\psi_n(\mathbf{X}_i) \neq Y_i \mid \mathbf{X} = \mathbf{X}_i)\,. \tag{2.38}$$

This estimator seeks to combine the bias-reducing properties of the bolstered estimator with the variance-reducing properties of the posterior-probability estimator.

For example, with $c = 2$, and the Gaussian bolstering and k-nearest neighbor empirical measures, the bolstered posterior-probability error estimator for a linear classifier $\psi_n(\mathbf{x}) = I(\mathbf{a}_n^T\mathbf{x} + b_n > 0)$ can be computed efficiently as

$$\hat{\varepsilon}_n^{\mathrm{GS-kNN}} = \frac{1}{nk}\sum_{i=1}^{n}\left[\Phi\left(\frac{(-1)^{1-Y_i}(\mathbf{a}_n^T\mathbf{X}_i + b_n)}{\sqrt{\mathbf{a}_n^T K_{n,\mathbf{X}_i,Y_i}\mathbf{a}_n}}\right)\left(\sum_{j=1}^{k} I((-1)^{y^j(\mathbf{X}_i)}(\mathbf{a}_n^T\mathbf{x} + b_n) > 0)\right)\right]. \quad (2.39)$$

2.2.4 Empirical Results

In this section, the performance of several of the generalized resubstitution error estimators are evaluated empirically.

2.2.4.1 Synthetic Data Experiments

In this section, synthetic data was employed to investigate the performance of the plain resubstitution ("resub"), bolstered resubstitution with spherical Gaussian kernels, with hyperparameter estimated as in (2.27) ("bolster"), a variant of bolstering that applies the kernels only to correctly-classified training points ("semi-bolster"), the k-NN posterior-probability estimator, with $k = 3$ ("3NNpp"), and the bolstered k-NN posterior probability estimator with spherical Gaussian kernels, with hyperparameters as in the previous two cases ("bolster 3NNpp"). For comparison with resampling error estimators, we also include the 10-fold cross-validation estimator ("cross valid") [6] and the zero bootstrap estimator ("boot") [13] in the experiments.

The generative model consists of multivariate Gaussian distributions for each of two classes, containing d_n noisy features and $d-d_n$ informative features, for each sample size. The values of the noisy features are sampled independently from a zero-mean, unit-variance Gaussian distribution across both classes. For the informative features, the class mean vectors are $(-\delta, \ldots, -\delta)$ and $(\delta, \ldots, \delta)$, where the parameter $\delta > 0$ is adjusted to obtain a desired level of classification difficulty.

The covariance matrices for both classes are block matrices

$$\Sigma_{d\times d} = \sigma^2 \times \begin{bmatrix} \Sigma_{l_1\times l_1} & & & 0 \\ & \Sigma_{l_2\times l_2} & & \\ 0 & & \ddots & \\ & & & I_{d_n\times d_n} \end{bmatrix},$$

where σ^2 is a variance parameter and Σ_{l_i} is an $l_i \times l_i$ matrix,

$$\Sigma_{l_i\times l_i} = \begin{bmatrix} 1 & \rho & \cdots & \rho \\ \rho & 1 & \cdots & \rho \\ \vdots & \vdots & \ddots & \vdots \\ \rho & \rho & \cdots & 1 \end{bmatrix},$$

representing l_i correlated features, with correlation coefficient $-1 < \rho < 1$, such that $\sum_i l_i = d - d_n$.

In the experiments below, we considered $d = 10$ features, consisting of $d_n = 4$ noisy and $d - d_n = 6$ informative features. The latter are correlated in pairs, i.e., $l_1 = l_2 = l_3 = 2$, with correlation coefficient $\rho = 0.2$. Four classification rules were considered: linear support vector machine (SVM), nonlinear SVM with radial basis kernel function (RBF-SVM), classification tree (CART) with stopped splitting at 5 points per leaf node, and k-nearest neighbors (k-NN), with $k = 3, 5, 7$. We adjusted δ to produce moderate classification difficulty over a range of sample sizes $n = 20, 40, 60, 80$, and 100; the corresponding average classification errors and 100; see Table 2.1.

The bias, variance, and RMS of each error estimator were estimated as sample-average approximations of (2.48), (2.49) and (1.6), respectively, by training the classifier 200 times using independently generated data sets and approximating the true classification error using a large test data set of size 5000. The bolstered resubstitution and bolstered posterior-probability error estimators for nonlinear classifiers used $M = 100$ Monte-Carlo points in their computation (the

Sample Size	Linear SVM	RBF SVM	CART	3NN	5NN	7NN
$n = 20$	0.311	0.311	0.377	0.329	0.315	0.315
$n = 40$	0.268	0.255	0.350	0.297	0.286	0.273
$n = 60$	0.244	0.234	0.341	0.288	0.267	0.263
$n = 80$	0.233	0.232	0.332	0.283	0.264	0.251
$n = 100$	0.224	0.225	0.330	0.277	0.261	0.249

Table 2.1: Classification errors in the synthetic data experiment.

estimators are computed exactly for the linear SVM). The results are displayed in Tables 2.2, 2.3, and 2.4; boldface indicates the best error estimator for each combination of classification rule and sample size. We can see in Table 2.2 that in almost all cases, the best bias value is obtained by one of the three bolstered error estimators. For a very small sample size $n = 20$, bootstrap is the best estimator in four out of the six classification rules; however, as will be seen later, bootstrap is much more computationally intensive than all other error estimators. Plain resubstitution is heavily negatively-biased at small sample sizes, as expected. All three bolstered estimators are able to significantly reduce this bias, except in the case of nearest neighbor classification, with small numbers of neighbors. What happens in this case is that the classification boundary is very complex, which affects negatively the bias-reducing properties of bolstering (incidentally, nearest-neighbor classification rules have infinite VC dimension, and Theorem 2 does not apply). As already indicated in [77], this can be corrected by using the semi-bolstered resubstitution error estimator, which only applies bolstering kernels to correctly-classified points. Indeed, as can be seen in Table 2.2, semi-bolstered resubstitution produces the best bias values in nearly all cases in the 3NN and 5NN experiments. One can also see in the table that, as the number of neighbors increases from 3 to 7, the original bolstered error estimator becomes less biased; this occurs because the classification boundary becomes less rough. At $k = 7$ neighbors, the best results are obtained by the bolstered-3NNpp estimator, a member of the family of bolstered posterior probability resubstitution estimators proposed in this work. The plain 3NNpp estimator has good bias properties only in the linear SVM case (indeed, [39] warned that the resubstitution-like posterior-probability

Classification Rule	Sample Size	resub	bolster	semi-bolster	3NNpp	bolster 3NNpp	cross valid	boot
Linear SVM	$n = 20$	-0.292	-0.076	-0.066	-0.070	-0.025	0.011	**0.010**
	$n = 40$	-0.169	-0.030	0.008	-0.039	**0.002**	0.021	0.027
	$n = 60$	-0.107	**-0.004**	0.043	-0.009	0.021	0.028	0.029
	$n = 80$	-0.079	**-0.001**	0.048	-0.004	0.021	0.029	0.025
	$n = 100$	-0.063	**0.008**	0.058	0.006	0.028	0.030	0.019
RBF SVM	$n = 20$	-0.277	-0.107	-0.100	-0.086	-0.066	0.070	**0.042**
	$n = 40$	-0.204	-0.085	-0.069	-0.035	**-0.018**	0.067	0.033
	$n = 60$	-0.168	-0.078	-0.058	-0.032	**-0.016**	0.072	0.036
	$n = 80$	-0.157	-0.056	-0.033	-0.015	**0.001**	0.053	0.023
	$n = 100$	-0.143	-0.059	-0.033	-0.021	**-0.006**	0.054	0.023
CART	$n = 20$	-0.354	-0.060	-0.054	-0.124	-0.026	0.017	**0.014**
	$n = 40$	-0.323	-0.040	-0.033	-0.119	**-0.013**	0.034	0.017
	$n = 60$	-0.314	-0.028	-0.020	-0.116	**-0.004**	0.031	0.013
	$n = 80$	-0.305	-0.022	-0.013	-0.111	**0.001**	0.036	0.014
	$n = 100$	-0.304	-0.016	-0.008	-0.108	**0.005**	0.030	0.011
3NN	$n = 20$	-0.175	-0.112	-0.034	-0.132	-0.124	0.031	**0.032**
	$n = 40$	-0.150	-0.105	**-0.028**	-0.122	-0.114	0.048	0.040
	$n = 60$	-0.144	-0.097	**-0.023**	-0.112	-0.105	0.046	0.037
	$n = 80$	-0.143	-0.096	**-0.021**	-0.113	-0.106	0.046	0.035
	$n = 100$	-0.136	-0.096	**-0.022**	-0.114	-0.107	0.049	0.036
5NN	$n = 20$	-0.124	-0.091	**0.023**	-0.086	-0.076	0.047	0.045
	$n = 40$	-0.110	-0.074	**0.018**	-0.071	-0.062	0.051	0.040
	$n = 60$	-0.099	-0.059	**0.023**	-0.061	-0.05	0.057	0.044
	$n = 80$	-0.095	-0.057	**0.023**	-0.060	-0.051	0.056	0.041
	$n = 100$	-0.091	-0.055	**0.022**	-0.059	-0.050	0.057	0.040
7NN	$n = 20$	-0.107	-0.073	**0.044**	-0.066	-0.056	0.055	0.045
	$n = 40$	-0.083	-0.047	0.046	-0.044	**-0.033**	0.062	0.044
	$n = 60$	-0.081	-0.042	0.043	-0.042	**-0.030**	0.059	0.038
	$n = 80$	-0.070	-0.032	0.048	-0.034	**-0.022**	0.064	0.043
	$n = 100$	-0.072	-0.034	0.044	-0.035	**-0.022**	0.062	0.039

Table 2.2: Bias results in the synthetic data experiment. The best bias value in each row is printed in bold.

estimator was expected to be significantly biased). The cross-validation and bootstrap estimators are positively biased, which is expected since both estimators employ classifiers trained on data sets of smaller effective sample size than the original one [6].

On the other hand, Table 2.3 shows that the bolstered estimators, but not semi-bolstering, achieve the smallest variance in nearly all cases. Indeed, semi-bolstering trades off less bias for more variance, since its bolstered empirical measure is more sparse than in full bolstering, as was

also noted in [77]. Notice that both cross-validation and bootstrap display large variable at small sample sizes, which is expected, since they are resampling estimators; resubtitution-like estimators avoid resampling and should be less variable [38].

Finally, in Table 2.4, we can see that the RMS (which combines bias and variance in a single metric) reveals a clear superiority of the bolstered estimators, and in particular the new bolstered-3NNpp estimator, over plain resubstitution, cross-validation, and bootstrap, except in the case of nearest-neighbor classification with a small number of neighbors, due to the aforementioned bias issue. In this case, semi-bolstering produces the best compromise between bias and variance. Nevertheless, with $k = 7$, we can see that the new bolstered-3NNpp estimator achieves the best RMS values, except at the very small sample size $n = 20$.

In order to examine the results further, Figures 2.1 and 2.2 display the box plots. These confirm the observations made previously about the bias and variance of the different error estimators. We can see, additionally, that at small sample size $n = 20$, all error error estimators, except for cross-validation, tend to be skewed towards optimistic biases, an effect that which disappears as sample size increases. Cross-validation is always skewed towards pessimistic bias, at all sample sizes.

In addition to the statistical issues discussed above, a very important issue is the computational complexity of the various error estimators, particularly in cases where thousands (or more) error estimates must be computed, as in wrapper feature selection. Table 2.5 displays the average computation time obtained by the error estimators in the experiment. The results confirm that plain resubstitution is lightning fast; its drawback is its large negative bias, as already mentioned. We can see that the plain posterior-probability error estimator, despite some bias issues, is also very fast. Combined with its small variance, this makes this estimator attractive for computationally-expensive classification tasks. Cross-validation (at 10 folds and no repetition) is the next fastest error estimator. Its poor variance properties under small sample sizes — and, in the case of wrapper feature selection, the issue of selection bias [78] — makes it unattractive. The bolstered resubstitution estimators are less fast but still much faster than the bootstrap. The latter is tens of times slower than the other estimators, in most cases, a fact that was already noted in [38].

Classification Rule	Sample Size	resub	bolster	semi-bolster	3NNpp	bolster 3NNpp	cross valid	boot
Linear SVM	$n = 20$	0.0038	0.0035	0.0049	0.0056	**0.0033**	0.0168	0.0101
	$n = 40$	0.0053	**0.0016**	0.0033	0.0038	0.0018	0.0091	0.0059
	$n = 60$	0.0033	**0.0013**	0.0023	0.0027	**0.0013**	0.0053	0.0034
	$n = 80$	0.0018	**0.0010**	0.0016	0.0019	**0.0010**	0.0033	0.002
	$n = 100$	0.0017	**0.0008**	0.0013	0.0015	**0.0008**	0.0028	0.0017
RBF SVM	$n = 20$	**0.0043**	**0.0043**	0.0044	0.0092	0.0067	0.0262	0.0141
	$n = 40$	0.0017	**0.0013**	0.0016	0.0031	0.0018	0.0113	0.0055
	$n = 60$	0.0012	0.0008	0.0011	**0.002**	0.0011	0.007	0.0036
	$n = 80$	0.0008	0.0006	0.0009	0.0015	**0.0007**	0.0045	0.0023
	$n = 100$	0.0007	**0.0005**	0.0007	0.0011	0.0006	0.0035	0.0019
CART	$n = 20$	0.0028	**0.0030**	0.0032	0.0070	0.0036	0.0228	0.0082
	$n = 40$	0.0018	**0.0012**	0.0024	0.0032	0.0014	0.0108	0.0037
	$n = 60$	0.0012	**0.0008**	0.0009	0.0020	0.0009	0.0066	0.0022
	$n = 80$	0.0009	**0.0007**	0.008	0.0015	**0.0007**	0.0048	0.0017
	$n = 100$	0.0008	**0.0005**	0.006	0.0012	**0.0005**	0.0039	0.0014
3NN	$n = 20$	0.0064	**0.0022**	0.0059	0.0035	0.0027	0.0154	0.0081
	$n = 40$	0.0033	**0.0010**	0.0030	0.0015	**0.0010**	0.0074	0.0036
	$n = 60$	0.0023	**0.0006**	0.0017	0.0009	**0.0006**	0.0051	0.0026
	$n = 80$	0.0017	0.0005	0.0013	0.0007	**0.0004**	0.0036	0.0017
	$n = 100$	0.0013	**0.0004**	0.0011	0.0005	0.0003	0.0030	0.0014
5NN	$n = 20$	0.0082	**0.0042**	0.0064	0.0068	0.0047	0.018	0.0098
	$n = 40$	0.0033	**0.0013**	0.0029	0.0022	**0.0013**	0.0075	0.0035
	$n = 60$	0.0023	0.0009	0.0020	0.0014	**0.0008**	0.0054	0.0026
	$n = 80$	0.0019	0.0007	0.0016	0.0011	**0.0006**	0.0039	0.0019
	$n = 100$	0.0014	0.0005	0.0013	0.0009	**0.0004**	0.0032	0.0015
7NN	$n = 20$	0.0091	**0.0052**	0.0066	0.0090	0.0064	0.0192	0.0113
	$n = 40$	0.0037	**0.0016**	0.0032	0.0030	**0.0016**	0.0076	0.0037
	$n = 60$	0.0025	**0.0010**	0.0021	0.0020	**0.0010**	0.0051	0.0026
	$n = 80$	0.0019	0.0007	0.0016	0.0012	**0.0006**	0.0039	0.0019
	$n = 100$	0.0014	**0.0005**	0.0012	0.0010	**0.0005**	0.0030	0.0015

Table 2.3: Variance results in the synthetic data experiment. The smallest variance in each row is printed in bold.

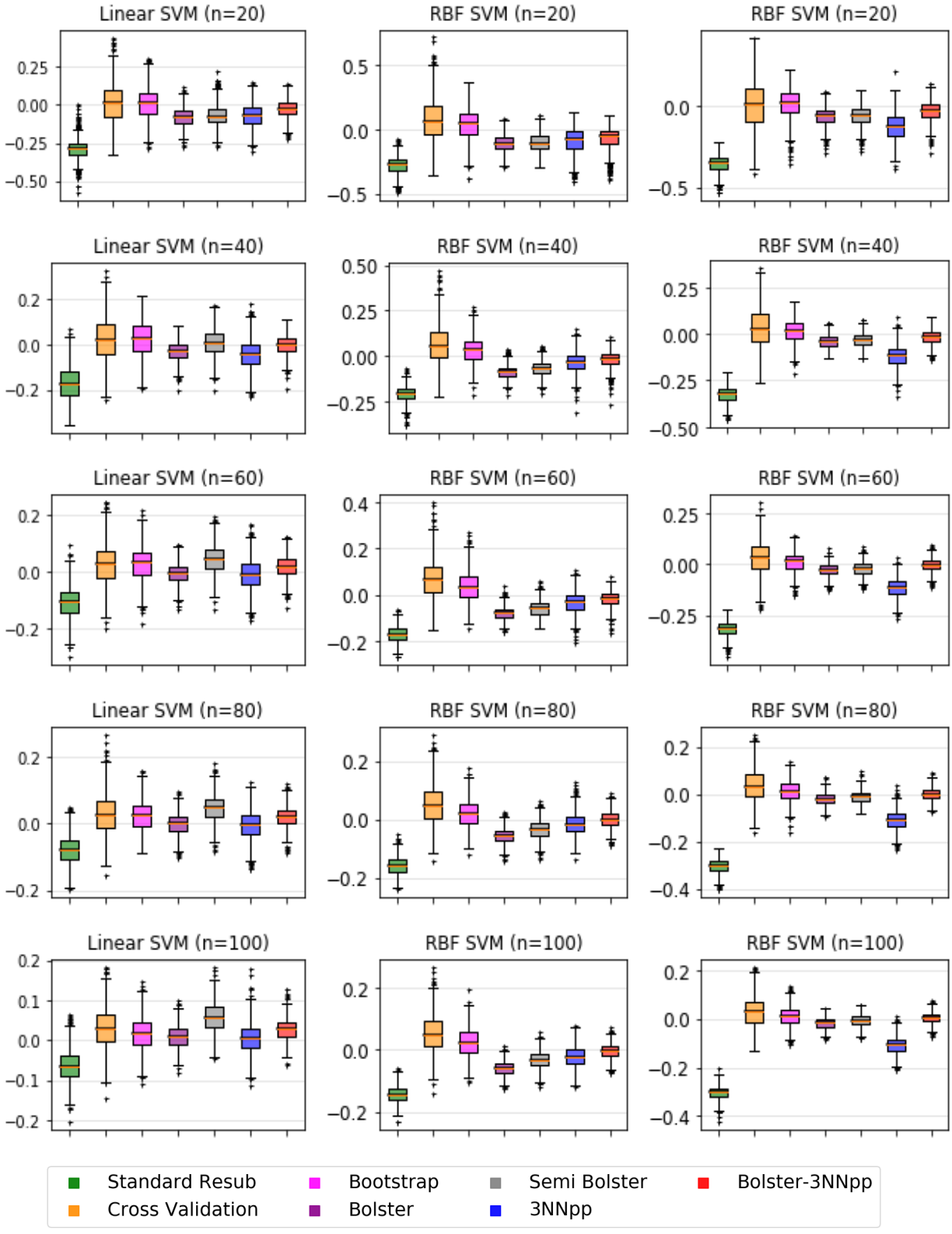


Figure 2.1: Boxplots for SVM and CART classification rules in the synthetic data experiment.

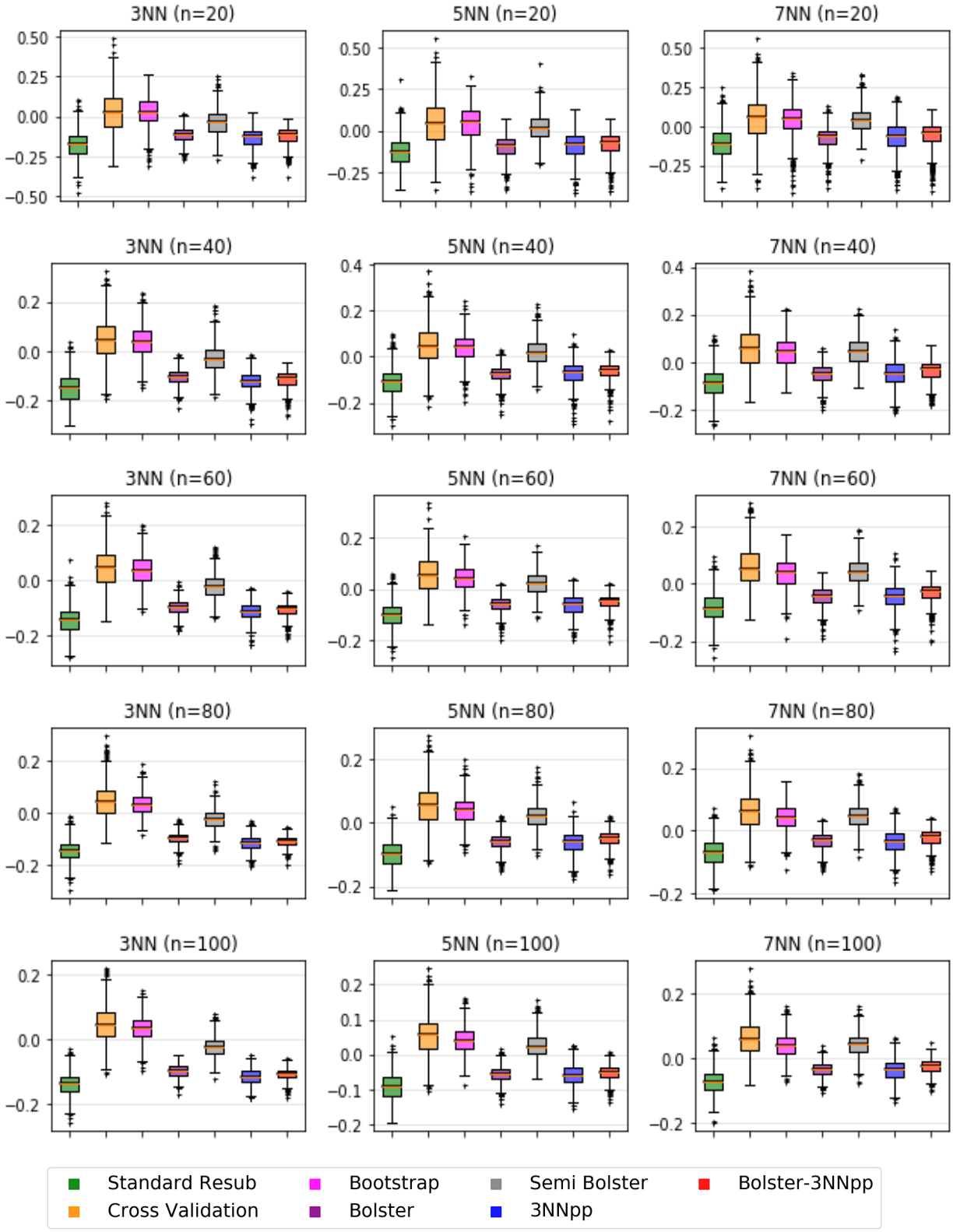


Figure 2.2: Boxplots for nearest-neighbor classification rules in the synthetic data experiment.

Classification Rule	Sample Size	resub	bolster	semi-bolster	3NNpp	bolster 3NNpp	cross valid	boot
Linear SVM	$n = 20$	0.2981	0.0963	0.0963	0.1029	**0.0626**	0.1302	0.1010
	$n = 40$	0.1836	0.0506	0.0579	0.073	**0.0422**	0.0976	0.0811
	$n = 60$	0.1213	**0.0362**	0.0650	0.0532	0.0421	0.0783	0.0651
	$n = 80$	0.0896	**0.0309**	0.0623	0.0440	0.0378	0.0641	0.0508
	$n = 100$	0.0751	**0.0289**	0.0680	0.0396	0.0391	0.0611	0.0453
RBF SVM	$n = 20$	0.2844	0.1258	0.1196	0.1289	**0.1050**	0.1765	0.1259
	$n = 40$	0.2086	0.0921	0.0798	0.0652	**0.0455**	0.1255	0.0816
	$n = 60$	0.1718	0.0831	0.0666	0.0545	**0.0364**	0.1101	0.0702
	$n = 80$	0.1596	0.0614	0.0441	0.0412	**0.0271**	0.0852	0.0535
	$n = 100$	0.1455	0.0632	0.0432	0.0388	**0.0242**	0.0806	0.0495
CART	$n = 20$	0.3581	0.0812	0.0785	0.1494	**0.0650**	0.1520	0.0914
	$n = 40$	0.326	0.053	0.0492	0.1314	**0.0401**	0.1093	0.0631
	$n = 60$	0.3164	0.0401	0.0363	0.1242	**0.0306**	0.0867	0.0489
	$n = 80$	0.3067	0.0334	0.0301	0.1175	**0.0265**	0.0783	0.0431
	$n = 100$	0.3049	0.0277	0.0251	0.1134	**0.0239**	0.0692	0.039
3NN	$n = 20$	0.1924	0.1227	**0.0869**	0.1450	0.1337	0.1239	0.0955
	$n = 40$	0.1616	0.1092	**0.0573**	0.1284	0.1179	0.1020	0.0721
	$n = 60$	0.1524	0.1015	**0.0461**	0.1159	0.1069	0.0838	0.0622
	$n = 80$	0.1485	0.0981	**0.0452**	0.1169	0.1079	0.0756	0.0532
	$n = 100$	0.1418	0.0981	**0.0378**	0.1157	0.1089	0.0775	0.0538
5NN	$n = 20$	0.1532	0.1090	**0.0832**	0.1175	0.1033	0.1382	0.1097
	$n = 40$	0.1253	0.0841	**0.0531**	0.0868	0.0738	0.1034	0.0721
	$n = 60$	0.1109	0.0662	**0.0550**	0.0729	0.0583	0.0903	0.0666
	$n = 80$	0.1031	0.0644	**0.0461**	0.0671	0.0548	0.0821	0.0573
	$n = 100$	0.0994	0.0585	**0.0477**	0.0662	0.0539	0.0828	0.0566
7NN	$n = 20$	0.1465	0.1011	**0.0913**	0.1116	0.0977	0.1504	0.1188
	$n = 40$	0.1024	0.0617	0.0756	0.0666	**0.0519**	0.1093	0.0744
	$n = 60$	0.0952	0.0516	0.0659	0.0580	**0.0424**	0.0915	0.0628
	$n = 80$	0.0806	0.0439	0.0625	0.0525	**0.0297**	0.0877	0.0587
	$n = 100$	0.0824	0.0394	0.0595	0.0461	**0.0312**	0.0796	0.0559

Table 2.4: RMS results in the synthetic data experiment. The best RMS value in each row is printed in bold.

2.2.4.2 MNIST Data Experiments

In this section we present results of a simple experiment that indicate the potential of generalized resubstitution estimators in image classification with convolutional neural networks (CNN). The experiment uses the well-known MNIST data set and the LeNet-5 CNN architecture.

The MNIST training data set contains $60,000$ 28×28 grayscale images of handwritten digits

Classification Rule	Sample Size	resub	bolster	semi-bolster	3NNpp	bolster 3NNpp	cross valid	boot
Linear SVM	$n = 20$	0.00012	7.72	7.17	2.18	6.94	3.98	111.75
	$n = 40$	0.00013	13.23	13.01	1.84	13.28	5.17	111.83
	$n = 60$	0.00013	19.95	19.27	1.73	18.84	6.23	136.64
	$n = 80$	0.00013	28.06	26.59	1.76	28.15	9.41	194.78
	$n = 100$	0.00014	35.67	33.98	1.95	36.04	14.51	259.07
RBF SVM	$n = 20$	0.00021	9.37	7.81	1.98	9.34	5.76	141.81
	$n = 40$	0.00029	17.53	15.23	1.74	17.96	7.11	166.50
	$n = 60$	0.00040	27.99	28.09	1.65	30.94	9.90	222.74
	$n = 80$	0.00053	37.45	36.41	1.67	45.82	12.60	272.67
	$n = 100$	0.00070	53.53	50.92	1.87	68.38	18.73	367.78
CART	$n = 20$	0.00014	5.99	4.26	1.87	7.33	3.15	91.54
	$n = 40$	0.00014	12.15	11.37	1.97	16.45	4.79	99.37
	$n = 60$	0.00015	18.06	15.79	1.96	22.57	4.89	112.91
	$n = 80$	0.00015	24.08	21.96	1.51	27.54	4.94	113.20
	$n = 100$	0.00016	34.82	31.87	1.98	36.51	6.00	133.62
3NN	$n = 20$	0.0012	63.17	55.19	3.30	121.20	10.54	263.76
	$n = 40$	0.0017	125.55	120.16	3.91	239.93	11.28	300.8
	$n = 60$	0.0028	188.34	185.14	4.50	360.82	11.96	346.19
	$n = 80$	0.0029	254.18	252.59	5.21	397.00	12.82	390.31
	$n = 100$	0.0034	319.55	319.53	5.90	445.04	13.56	517.62
5NN	$n = 20$	0.0011	61.95	51.02	3.24	118.45	10.4	261.14
	$n = 40$	0.0016	123.18	113.18	3.84	235.02	11.12	297.05
	$n = 60$	0.0021	185.08	176.05	4.43	253.90	11.79	346.69
	$n = 80$	0.0027	249.10	238.91	5.11	391.04	12.61	417.84
	$n = 100$	0.0034	310.63	300.12	5.76	436.26	13.35	499.96
7NN	$n = 20$	0.0011	53.07	41.08	2.76	101.20	9.01	230.98
	$n = 40$	0.0016	116.56	104.69	3.67	221.75	10.65	287.32
	$n = 60$	0.0022	175.79	163.67	4.23	329.95	11.23	335.52
	$n = 80$	0.0027	237.74	224.36	4.89	377.69	12.03	416.78
	$n = 100$	0.0034	299.76	286.09	5.53	422.69	12.72	477.60

Table 2.5: Average computation time (in milliseconds) in the synthetic data experiment.

between 0 and 9 (hence, 10 classes). It is well-known that LeNet-5 can achieve accuracies of upwards of 99% on this data set; e.g., see [79]. Problems with small classification error tend to be easier in terms of error estimation performance [6]. To make the problem more challenging, we train the LeNet-5 classifier on random subsets of $n = 200, 400, 600$ and 800 images from the original data set. The remaining data are used to obtain accurate test-set estimates of the true classification error, in order to compute estimates of the bias, variance, and RMS of each error

estimator, using 200 independently drawn training data sets for each sample size. The LeNet-5 network was trained using 200 epochs of stochastic gradient descent, with batch size 32, employing 10% of the training data in each case as a validation data set to stop training early if the validation loss was not reduced for 10 consecutive epochs.

We investigate the performance of bolstered resubstitution with diagonal Gaussian kernels, which leads to a "Naive-Bayes" bolstering resubstitution estimator, as explained in Section 2.2.1. This is done since spherical kernels tend to perform poorly in very high-dimensional spaces [80]. Likewise, k-nearest neighbor posterior-probability estimators led to too much bias in this high-dimensional space, and are not considered further.

If X_{ijk} denotes pixel k in image i of class j, the mean minimum distance $d_{n,jk}$ among pixels k in class j is:

$$\hat{d}_{n,jk} = \frac{1}{n_j} \sum_{i=1}^{n_j} |X_{ijk} - X'_{ijk}|\,, \quad j = 0, 1, \ldots, c-1\,,\ k = 1, \ldots, 28 \times 28\,,$$

where n_j is the number of images in class j ($n_j \geq 2$ is assumed) and X'_{ijk} is the nearest pixel (in value) in position k to X_{ijk} among images in class j. The bolstering kernel standard deviations are then given by:

$$\sigma_{n,jk} = \frac{\hat{d}_{n,jk}}{\alpha_1}\,, \quad j = 0, 1, \ldots, c-1\,,\ k = 1, \ldots, 28 \times 28\,, \tag{2.40}$$

where $\alpha_1 = 0.674$, as seen in Section 2.2.1.

In order to further reduce the bias of the Naive-Bayes bolstered resubstitution estimator in this high-dimensional space, we employ a data-driven calibration procedure: we multiply the kernel standard deviation by a constant $\kappa > 0$, which is adjusted so as to minimize the estimated bias of the estimator. The bias is roughly estimated by training the classifier on a random sample of 80% of the images form the available training data, and testing it on remaining 20%. Depending on the computational cost of training the classifiers, this process can be repeated a number of times r and the results averaged. The point is that the bias does not need to be accurately estimated in

order to find a useful value for κ. The calibration process consists of starting at $\kappa = 1$, computing the corrected Naive-Bayes bolstered resubstitution estimate, and increasing κ by a fixed step-size (here, 0.1) until the magnitude of the roughly estimated bias does not decrease for two consecutive iterations. A similar model-based calibration method was proposed in [80]; however, the procedure proposed here is entirely data-driven and makes no modeling assumptions.

Finally, the naive-Bayes bolstered resubstitution estimator is computed by Monte-Carlo, as in (2.23), where $\{\mathbf{X}_{ij}^{\mathrm{MC}}; j = 1, \ldots, M\}$ are random images generated by drawing each pixel from a Gaussian distribution with mean equal to the original pixel value and standard deviation in (2.40). This generates "noisy images" where the intensity of the noise in each pixel is correlated with the variability of pixel values at that position across the training data (for that digit class). Here we employed $M = 100$ Monte-Carlo images for each training image. A few of these Monte-Carlo images can be seen in Figure 2.3.

The results of the experiment are displayed in Tables 2.6, 2.7 and 2.8. We can see that, while Naive-Bayes bolstered resubstitution is able to improve somewhat the optimistic bias of resubstitution, calibration succeeded into reducing the bias to nearly zero, especially as sample size increases. The generalized resubstitution estimators were not able to match the low variance of plain resubstitution. Calibration increased the variance of the plain Naive-Bayes estimator, as might be expected (though much less in the case $r = 5$ than in the case $r = 1$). When bias and variance are combined in the RMS metric, the clear winners are the calibrated estimators, particularly at $r = 5$. Even at $r = 1$, (i.e., just one additional step of classifier training), there is a substantial improvement. If more than $r = 5$ repetitions are used, it is expected that results will improve further, though at a higher computational cost. Notice that the bias, variance, and RMS of all estimators decrease monotonically with increasing sample size.

Finally, Table 2.9 displays the average computation time for the error estimators in the experiment. The results again confirm that plain resubstitution is very fast. The superior performance of the calibrated naive-Bayes bolstered resubstitution estimators come at a computational price; this is due to the fact that additional classifiers need to be trained in order to perform the calibration

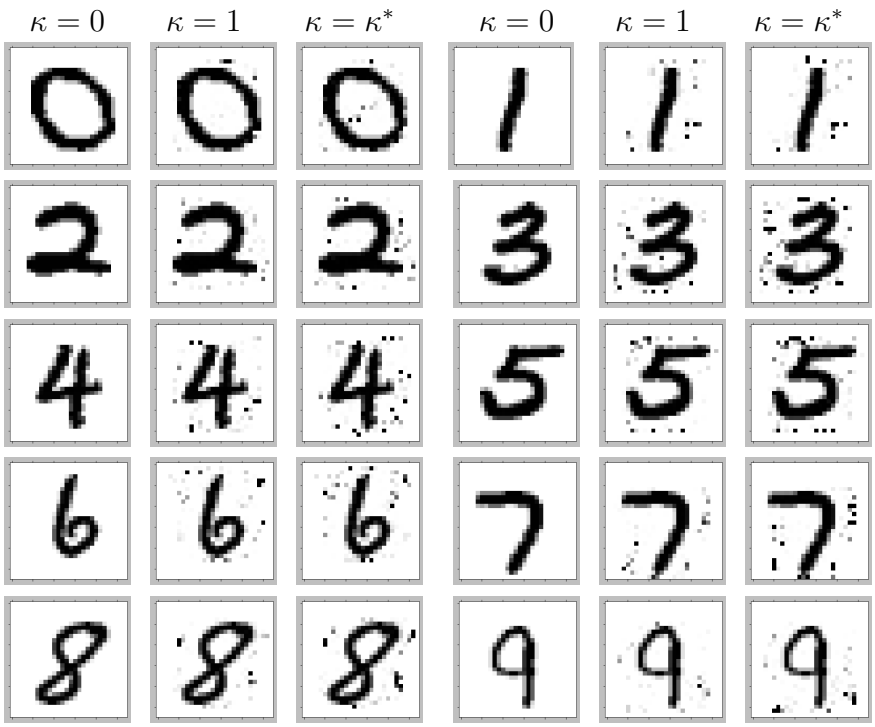


Figure 2.3: Examples of Monte-Carlo images used in the Naive-Bayes bolstered resubstitution error estimator, with $n = 600$. The parameter κ^* is the optimal correction factor for the calibrated naive Bayes bolstered error estimator (r=1). The cases $\kappa = 0$ and $\kappa = 1$ refer to the original image and the uncorrected Naive-Bayes bolstered image, respectively.

Sample Size	resub	nBbolster	calibrated nBbolster (r=1)	calibrated nBbolster (r=5)
$n = 200$	-0.131	-0.082	0.019	**0.014**
$n = 400$	-0.100	-0.064	**0.007**	0.012
$n = 600$	-0.080	-0.052	**0.005**	**0.005**
$n = 800$	-0.072	-0.051	0.003	**0.001**

Table 2.6: Bias results in the MNIST data experiment. The best bias value in each row is printed in bold.

Sample Size	resub	nBbolster	calibrated nBbolster (r=1)	calibrated nBbolster (r=5)
$n = 200$	**0.0006**	0.0013	0.0064	0.0027
$n = 400$	**0.0002**	0.0008	0.0029	0.0013
$n = 600$	**0.0002**	0.0006	0.0018	0.0006
$n = 800$	**0.0001**	0.0003	0.0011	0.0006

Table 2.7: Variance results in the MNIST data experiment. The best variance value in each row is printed in bold.

Sample Size	resub	nBbolster	calibrated nBbolster (r=1)	calibrated nBbolster (r=5)
$n = 200$	0.1336	0.0897	0.0822	**0.0519**
$n = 400$	0.1010	0.0702	0.0505	**0.0418**
$n = 600$	0.0812	0.0576	0.0403	**0.3040**
$n = 800$	0.0728	0.0539	0.0301	**0.0200**

Table 2.8: RMS results in the MNIST data experiment. The best RMS value in each row is printed in bold.

Sample Size	resub	nBbolster	calibrated nBbolster (r=1)	calibrated nBbolster (r=5)
$n = 200$	0.00043	7.24	65.91	298.51
$n = 400$	0.00085	15.42	127.78	597.24
$n = 600$	0.00131	25.54	216.99	928.61
$n = 800$	0.00175	35.59	296.86	1342.19

Table 2.9: Average computation time (in seconds) in the MNIST data experiment.

procedure.

2.2.4.3 UCI Data Experiments

This section presents a set of empirical study to examine the performance of generalized resubstitution error estimators using fully connected neural network classifiers and a selection of UCI datasets [3]. The networks were created from a permutation of various hyperparameter values. The varying hyperparameters were depth and width, whereas hyperparameters batch size and dropout rate were kept fixed. Depth and width were selected from sets of 5 and 8 values, respectively; depth $\in \{1, 2, 3, 4, 5\}$ and width $\in \{10, 20, 30, 40, 2d, 4d, 6d, 8d\}$, where d is the dimension of the input data. Batch size was set to the closes integer to $\frac{n}{10}$, for n being the training sample size, and dropout rate was set to 5%.

The networks were trained for maximum of 800 epochs or until the training accuracy reaches 95%, whichever occurs first, using ADAM optimizer with initial learning rate of 0.001 and a

[3] https://archive.ics.uci.edu/ml/datasets.php

scheduled learning rate decay of 0.5 at every 100 iterations. The training dataset was selected from the UCI database [81]. The choice of datasets was based on the size of training samples. Inspired by [82, 83], I selected the datasets with less than 5000 total data points as the datasets with small sample size, which included total of 90 datasets. Hence, this empirical study includes training $5 \times 8 = 40$ networks for each datasets, generating total of 3600 networks for all 90 datasets.

Subsequently, to assure the consistency of the trained networks, the networks that reached 800 training epochs with a training accuracy below 95% were discarded. Among the 3066 remaining networks, 172 of them had a true error of larger than 0.5, and therefore were not included in the plots. The vertical and horizontal axis in Figures 2.4, 2.5 and 2.6 show the $|Deviation|$ and True Error, respectively, where $|.|$ takes the absolute value and $Deviation = estimated\ error - true\ error$. Good candidates of error estimators are the one with deviations as small as possible. Each point in the figures refers to the error estimate by one of the error estimators per network per dataset. The error estimators, in this set of experiments, include standard resubstitution, spherical bolstered, posterior probability, and spherical bolstered-posterior probability error estimators.

Figure 2.4 includes subplots for networks with fixed depth (D) and width (W). Each column of subplots show the result for a fixed width but various depths. Each row of subplots, on the other hand, include the result for a fixed depth but various widths. This figure illustrates that wider networks include more points for each error estimator, which suggests that the wider networks had more capacity to reach 95% accuracy in under 800 epochs. Additionally, the performance of the error estimators seem not to be affected much by the depth or width of the network. Yet, one can observe that the performance of some error estimators, including the bolstered ones show a pattern with respect to the true error.

Figures 2.5 and 2.6 displays the absolute value of deviation from the true error versus the true error for all 2894 networks. Figure 2.6 maps all subplots from Figure 2.5 in a single plot for easier comparison of error estimators. Bolstered posterior probability error estimator presented the lowest deviation when the true error was not very small. Otherwise, when the true error was small, the deviation of bolstered resubstitution varies from very small (~ 0) to large (~ 0.4), indicating

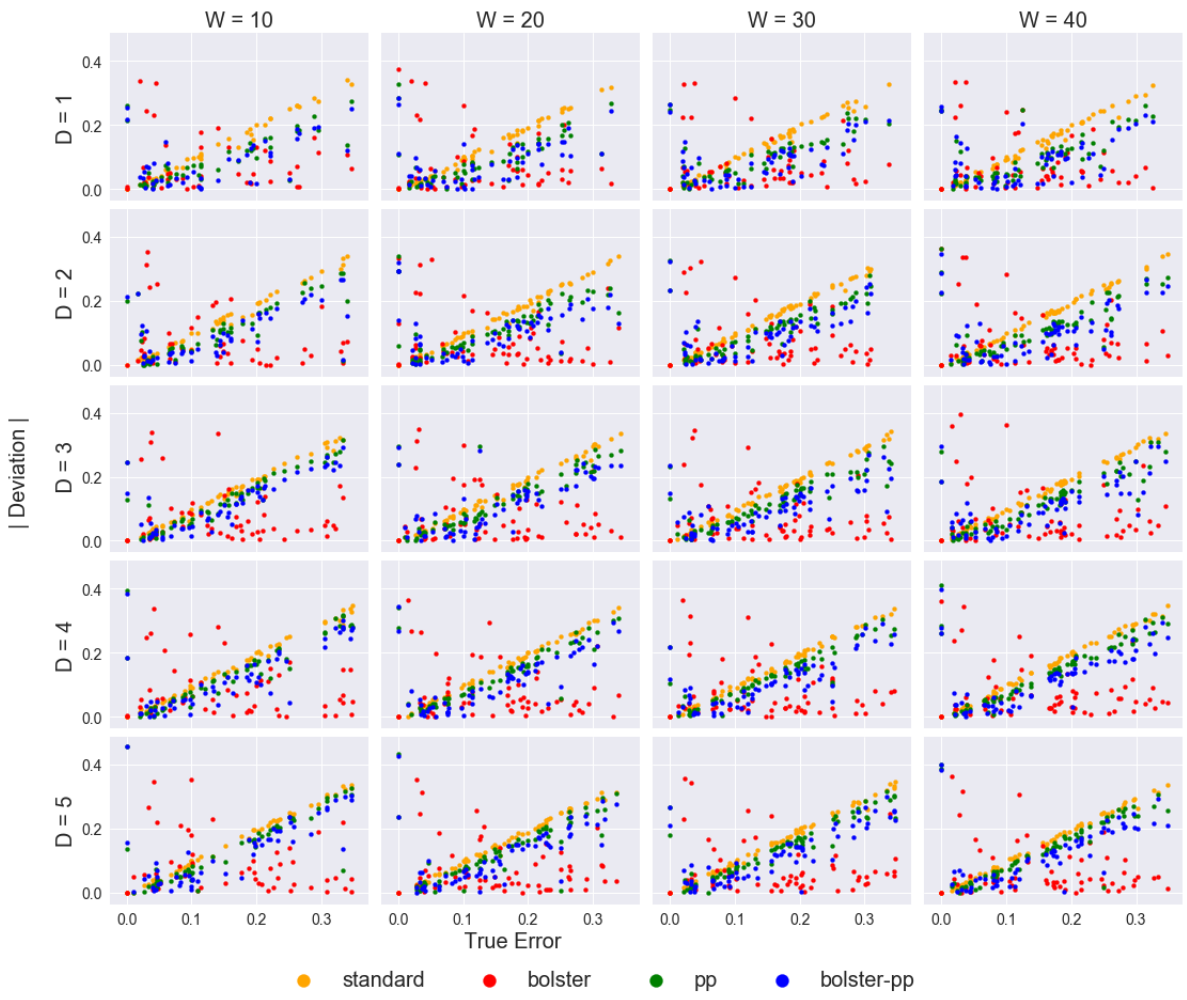


Figure 2.4: Magnitude of deviation of error estimators from the true error for selected networks with fixed depth (D) and width (W).

that bolstering parameter was not optimal for some datasets and networks that generate both small standard resubstitution and small true error.

Since the networks are trained for a high training accuracy, small true error indicates that the trained classifier has learned the important features through training dataset. In other words, the training data represented all important features of the underlying distribution, from which the test set was also generated, and meanwhile, the classifier had enough capacity to learn all those features. In such case, no bolstering is needed and the standard resubstituion error estimator is a good representative of the true error due to its small deviation from the true error.

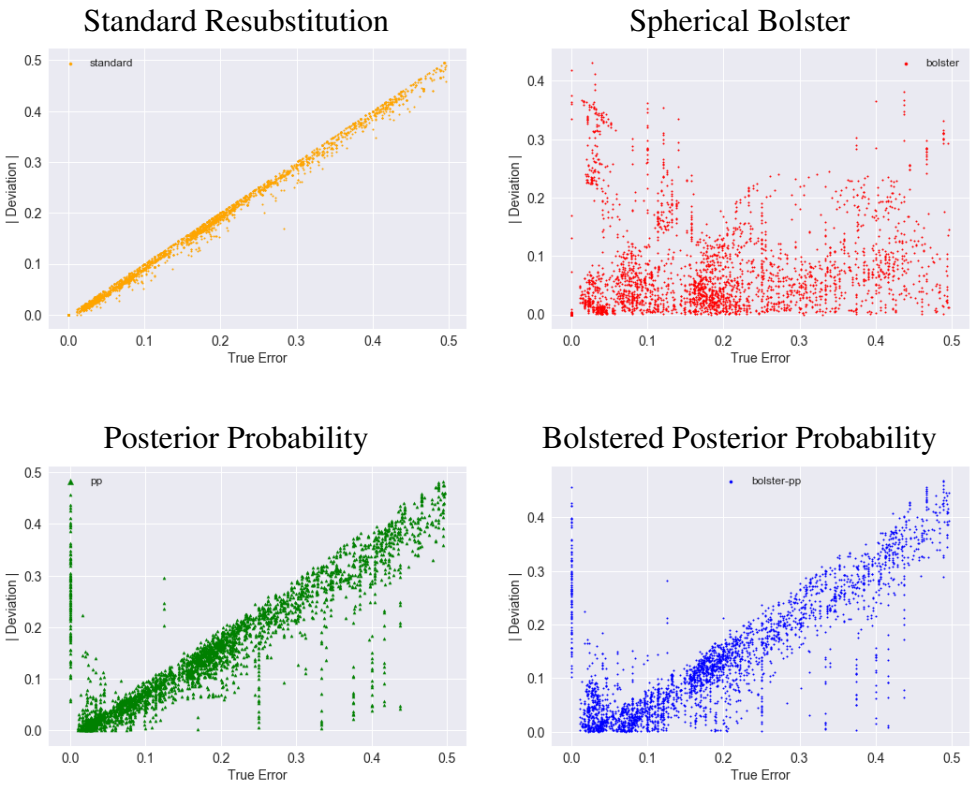


Figure 2.5: Magnitude of deviation of each error estimator from the true error versus true error for all netowrks.

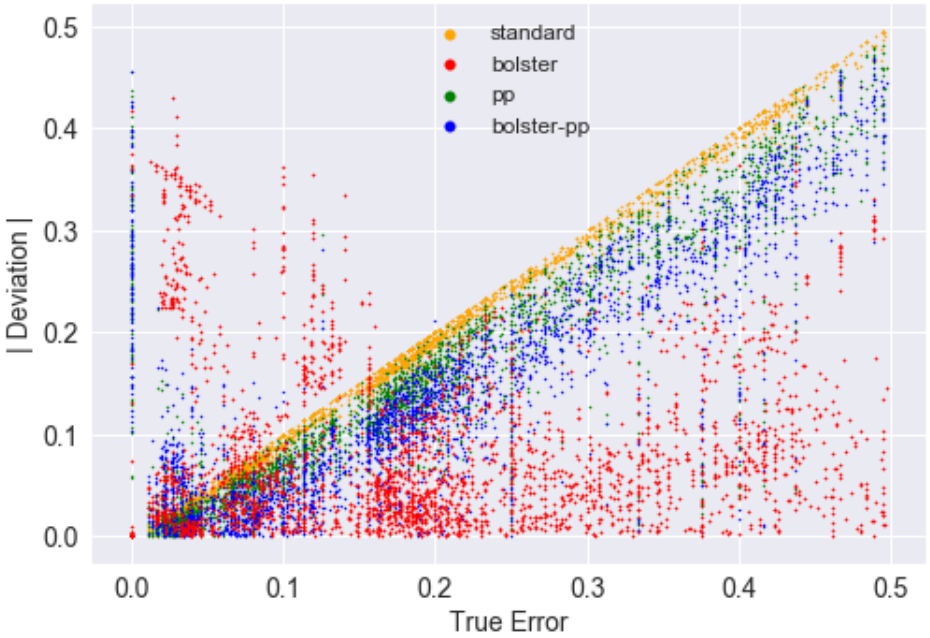


Figure 2.6: Magnitude of deviation of all error estimators from the true error versus true error for all networks.

2.3 Bayesian Generalized Resubstitution for Neural Networks

All previous examples of generalized resubstitution were based on smoothing the error count. This section gives an example that shows that the family of generalized resubstitution estimators is more general than that. For the flow of this subsection, many notations are defined again. For consistency of terminologies with deep learning literature, in this section, "accuracy" is used instead of "error" in some parts of this section. Classification accuracy and error estimates are related through Acc $= 1-\hat{\varepsilon}$. In the following, notations and formulations for Bayesian generalized resubstitution are described in section 2.3.1. Subsequently, section 2.3.2 presents empirical results.

2.3.1 Definitions and Methods

Let $\mathbf{X} \in R^d$ be a random *feature vector*, $Y \in \{0, 1, \ldots, c-1\}$ be the corresponding random *label*, and P be the joint probability that determines the relationship between $\mathbf{X}$ and Y. A classifier ψ is a (Borel-measurable) function from R^d to $\{0, 1, \ldots, c-1\}$, designed such that the classification accuracy Acc $= P(\psi(\mathbf{X}) = Y)$ is maximized. It can be shown that an optimal maximum-accuracy classifier satisfies

$$\psi^*(\mathbf{x}) = \arg\max_{y=0,1,\ldots,c-1} P(Y = y \mid \mathbf{X} = x)\,. \tag{2.41}$$

which is equivalent minimum-error classifier; $\psi^*(\mathbf{x}) = \arg\min_y P(Y \neq y \mid \mathbf{X} = x)$, for $y = 0, 1, \ldots, c-1$. In practice, P is unknown, and one designs a classifier ψ_n that is accurate as possible based on the information contained in training data $S_n = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$, where each pair $(\mathbf{X}_i, Y_i)$ is distributed as $(\mathbf{X}, Y)$. In the discriminative approach to classification, estimates $\widehat{P}_n(Y = y \mid \mathbf{X} = x)$ of the unknown conditional probability are derived using the training data S_n and plugged in (2.41) to obtain a classifier

$$\psi_n(\mathbf{x}) = \arg\max_{y=0,1,\ldots,c-1} \widehat{P}_n(Y = y \mid \mathbf{X} = \mathbf{x})\,, \tag{2.42}$$

with the justification being that the closer the estimates are to the true conditional probability, the closer the obtained classifier will be to the optimal classifier. The classical and Bayesian

approaches to discriminative classification derive distinct estimates $\widehat{P}_n(Y = y \mid \mathbf{X} = \mathbf{x})$ based on a parametric model $\{P_\theta(Y = y \mid \mathbf{X} = \mathbf{x}); \theta \in \Theta\}$. In the classical approach, it is assumed that the true conditional probability is a member of the parametric family, i.e., there is a parameter θ^* such that $P(Y = y \mid \mathbf{X} = \mathbf{x}) = P_{\theta^*}(Y = y \mid \mathbf{X} = \mathbf{x})$. A point estimator θ_n is formed from the training data, typically by optimizing a data-fit score, and one chooses

$$\widehat{P}_n^{\,cl}(Y = y \mid \mathbf{X} = \mathbf{x}) = P_{\theta_n}(Y = y \mid \mathbf{X} = \mathbf{x})\,. \tag{2.43}$$

By contrast, in the Bayesian approach, the parameter is a random variable, with a prior density $p(\theta)$ and one sets

$$\widehat{P}_n^{\,bay}(Y = y \mid \mathbf{X} = \mathbf{x}) \;=\; \int_\Theta \nu_\theta(Y = y \mid \mathbf{X} = \mathbf{x})\, p(\theta \,|\, S_n)\, d\theta\,, \tag{2.44}$$

where $p(\theta \mid S_n)$ is the parameter posterior density. Hence, the difference between the classical and Bayesian approaches is that the former is based on optimization, while the latter is based on marginalization. As a result, the Bayesian approach may overcome overfitting issues that result from optimization based on a small amount of data.

A neural network with L layers and n_k neurons in the k-th layer is defined recursively by

$$\begin{aligned} \alpha_i^k(\mathbf{x}) \;&=\; \sum_{j=0}^{n_k-1} w_{ji}^k \beta_j^{k-1}(\mathbf{x}) + w_{0i}^k\,, \quad i = 0, 1, \ldots, n_k - 1\,, \\ \beta_i^{k-1}(\mathbf{x}) \;&=\; \sigma(\alpha_i^{k-1}(\mathbf{x}))\,, \quad i = 0, 1, \ldots, n_k - 1\,, \end{aligned} \tag{2.45}$$

for $k = 1, \ldots, L$, where $\beta_i^0(\mathbf{x}) \;=\; x_i$ for $i = 1, \ldots, d$ are the inputs to the network and σ is a component-wise nonlinearity. The outputs of the network are

$$f_i(\mathbf{x}, \theta) \;=\; \alpha_i^L(\mathbf{x})\,, \quad i = 0, 1, \ldots, n_L = c - 1\,, \tag{2.46}$$

where $\theta = \{w_{ji}^k\}$ is the vector of all parameters in the neural network. The softmax model

$$P_\theta(Y = y \mid \mathbf{X} = \mathbf{x}) \;=\; \frac{\exp\left(f_y(\mathbf{x}, \theta)\right)}{\sum_{i=0}^{c-1} \exp\left(f_i(\mathbf{x}, \theta)\right)}\,, \quad y = 0, 1, \ldots, c-1\,, \tag{2.47}$$

provides a parametric discriminative model for both classical and Bayesian neural network classification.

We seek an accuracy estimator $\widehat{\mathrm{Acc}}_n$ that reliably approximate $\mathrm{Acc}_n \;=\; P(\psi_n(\mathbf{X}) = Y \mid S_n)$ based on the training data S_n, in the sense that the *deviation* $\widehat{\mathrm{Acc}}_n - \mathrm{Acc}_n$ is minimized. The usual metrics to be minimized are moments of the deviation. The *bias* is defined as the first moment of the deviation:

$$\mathrm{Bias}(\widehat{\mathrm{Acc}}_n) \;=\; E[\widehat{\mathrm{Acc}}_n - \mathrm{Acc}_n] \;=\; E[\widehat{\mathrm{Acc}}_n] - E[\mathrm{Acc}_n]\,. \tag{2.48}$$

The accuracy estimator $\widehat{\mathrm{Acc}}_n$ is optimistic, unbiased, or pessimistic according to whether $\mathrm{Bias}(\widehat{\mathrm{Acc}}_n)$ is positive, zero, or negative, respectively. The *deviation variance* is

$$\mathrm{Var}_{\mathrm{dev}}(\widehat{\mathrm{Acc}}_n) \;=\; \mathrm{Var}(\widehat{\mathrm{Acc}}_n - \mathrm{Acc}_n) \;=\; \mathrm{Var}(\widehat{\mathrm{Acc}}_n) + \mathrm{Var}(\mathrm{Acc}_n) - 2\mathrm{Cov}(\mathrm{Acc}_n, \widehat{\mathrm{Acc}}_n)\,. \tag{2.49}$$

Previous subsections defined baseline and bolstered posterior-probability classification error estimators as examples of *generalized resubstitution* estimators. The baseline posterior-probability estimator (here formulated as an accuracy estimator) was given by:

$$\widehat{\mathrm{Acc}}_n^{\mathrm{pp}} \;=\; \frac{1}{n} \sum_{i=1}^{n} \widehat{P}_n(Y = \psi(\mathbf{X}_i) \mid \mathbf{X} = \mathbf{X}_i)\,, \tag{2.50}$$

where $\widehat{P}_n(Y = \psi(\mathbf{X}_i) \mid \mathbf{X} = \mathbf{X}_i)$ is computed via the parametric estimates (2.43) or (2.44) for classical and Bayesian neural networks, respectively. This estimator is based on the simple and reasonable idea of averaging the estimated posterior probabilities of correct classification for each training point. If all training points are classified with high confidence in their respective classes, the estimated posterior probabilities, and thus the estimated accuracy, will be large, and the converse will be true, otherwise. However, this may produce an optimistically-biased estimator,

if the classification algorithm is overfitting for the given sample size and complexity. An alternative method is provided by the *bolstered posterior-probability* accuracy estimator:

$$\widehat{\mathrm{Acc}}_n^{\mathrm{bpp}} \;=\; \frac{1}{n}\sum_{i=1}^{n} \alpha_{n,i}\, \widehat{P}_n(Y = \psi_n(\mathbf{X}_i) \mid \mathbf{X} = \mathbf{X}_i)\,. \tag{2.51}$$

with coefficients

$$\alpha_{n,i} \;=\; \int_{A_{Y_i}} p_{n,\mathbf{X}_i,Y_i}(\mathbf{x};\varphi)\, d\mathbf{x}\,, \tag{2.52}$$

where $A_j = \{\mathbf{x} : \psi_n(\mathbf{x}) = j\}$, for $j = 0, 1, \ldots, c-1$, are the decision regions corresponding to the classifier ψ_n, and the probability densities $p_{n,\mathbf{X}_i,Y_i}$ are called *bolstering kernels*, which depend on hyperparameters φ_i, for $i = 1, \ldots, n$. Notice that $0 \leq \alpha_{n,i} \leq 1$, so that $\widehat{\mathrm{Acc}}_n^{\mathrm{bpp}} \leq \widehat{\mathrm{Acc}}_n^{\mathrm{pp}}$; this has the effect of reducing optimistic bias in the baseline posterior-probability estimator $\widehat{\mathrm{Acc}}_n^{\mathrm{pp}}$. The most common choice for bolstering kernels are multivariate Gaussian densities with mean $\mathbf{X}_i$ and covariance matrix $K_{n,i}(\varphi_i)$:

$$p_{n,\mathbf{X}_i,Y_i}(\mathbf{x};\varphi_i) \;=\; \frac{1}{\sqrt{(2\pi)^d\,\det(K_{n,i}(\varphi_i))}} \exp\left(-\frac{1}{2}\,(\mathbf{x}-\mathbf{X}_i)^T K_{n,i}(\varphi_i)^{-1}(\mathbf{x}-\mathbf{X}_i)\right)\,,$$

Here, we only consider the case where the matrices $K_{n,i}(\varphi_i)$ are diagonal, with freely adjustable diagonal elements (variances), $K_{n,i}(\varphi_i) = \mathrm{diag}(\varphi_{i1}, \ldots, \varphi_{id})$; this is known as *Naive-Bayes bolstering* [75]. In the limiting case where all variances φ_{ik} tend to zero, all coefficients $\alpha_{n,i}$ tend to 1, and the bolstered estimator reduces to the baseline posterior-probability estimator, which is typically optimistic. On the other hand, the large the variances are, the more mass "escapes" from each decision region, and the bolstered estimator eventually becomes pessimistic. Hence, the variances can in principle be optimized to produce an exactly unbiased estimator. The integrals required to compute the coefficients $\alpha_{n,i}$ can be computed analytically in very few cases (e.g., when $c = 2$, the decision boundary is linear, and Gaussian bolstering kernels are used). In practice,

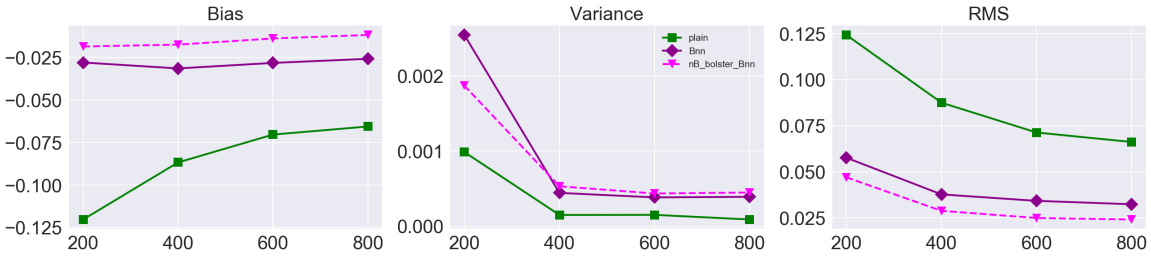


Figure 2.7: Bias, variance, and RMS of error estimators for Lenet-5 classifier and MNIST dataset.

Monte-Carlo estimates are used:

$$\alpha_{n,i} \approx \frac{1}{M}\sum_{j=1}^{M} I(\mathbf{X}_{ij}^{\mathrm{MC}} \in A_{Y_i}), \tag{2.53}$$

where $\{\mathbf{X}_{ij}^{\mathrm{MC}}; j = 1, \ldots, M\}$ are random points drawn from the density $p_{n,\mathbf{X}_i,Y_i}$, for $i = 1, \ldots, n$.

2.3.2 Empirical Results

This section presents the empirical results of the Bayesian generalized resubstitution on Bayesian neural networks that were trained on synthetic and MNIST datasets. The networks include a Lenet-5 CNN (on MNIST data) and a fully connected network with two hidden layers and 512 nodes in each layer (on MNIST data). Additionally, using synthetic data and a one-hidden layer networks was used to study the effect of width of the network on classification error (or accuracy). The results of experiments with MNIST dataset are displayed in Figures 2.7 and 2.8, whereas Figure 2.9 presents error estimates from the experiments with synthetic data. The synthetic data was generated in the same way as explained in section 2.2.4.1.

Figures 2.7 and 2.8 compare the bias, variance and RMS values of the standard resubstitution error estimator with the Bayesian posterior probability and bolstered Bayesian posterior probability. Bias, variance, and RMS of the error estimators were computed over 50 repeats of the experiments. Increasing the number of experiments leads to smoother diagram for variance in figure 2.8 and would better show the decrease in variance as the sample size increases. However, the

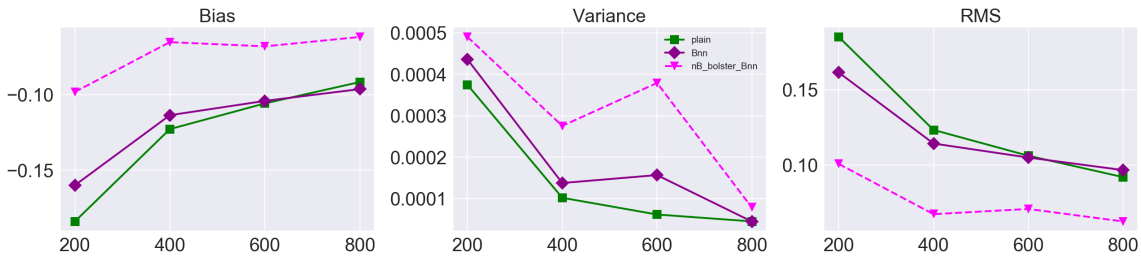


Figure 2.8: Bias, variance, and RMS of error estimators for fully connected network (depth = 2, width = 512) classifier and MNIST dataset.

Bayesian approach is computationally very expensive, and therefore we did not repeat the experiment for more than 50 times.

MNIST data can be regarded as high dimensional data with 28×28 dimensions, which contributes to the high cost of running experiments on Bayesian networks. Also, bolstering and Bayesian network together lead to high computation and time complexity. The experiments with shallow networks and synthetic data, as illustrated in Figure 2.8, suggest that more than 50 repeats of experiments may be required to draw a clear conclusion for behaviour of the bolstered posterior probability error estimator. However, I used the synthetic data to investigate the behaviour of the Bayesian posterior probability error estimator as the network gets wider (i.e. number of nodes in each layer increases). To do so, I ran the experiments for a shallow network (with only 1 hidden layer) with 10, 30, and 100 neurons wide.

Figure 2.9 illustrates that as the network's width increased, the standard resubstitution error was also increased, which indicates more overfitting on the training data. This is consistent with what one expects because a wider network has more capacity to fit the features that are present in the training dataset. The Bayesian posterior probability error estimate consistently presented less bias than the standard resubstitution. Although the Bayesian posterior probability error estimator was unbiased when the network was not very wide (W=10), its variance was large. Hence it did not consistently beat the standard resubstitution. As the network became wider, the variance decreased notably and therefore the RMS value followed the behavior of the bias. Since the bias of

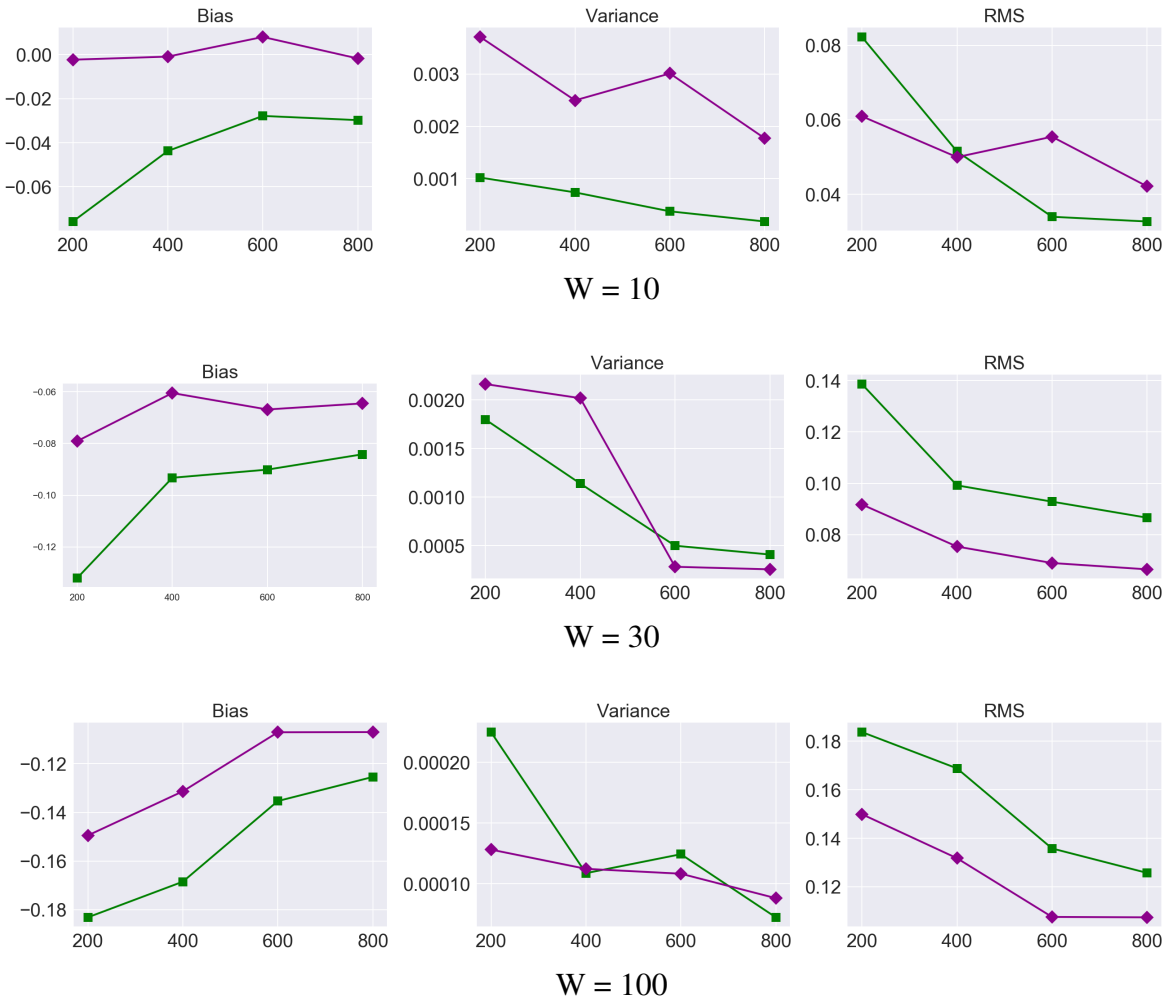


Figure 2.9: Bias, variance, and RMS of error estimators for fully connected network classifiers with one hidden layer and various width (W) using binary-class synthetic data

standard resubstitution was consistently lowered by Bayesian posterior probability error estimator, the latter was a better error estimator in terms of RMS values for wider networks, as a result of lowered variance compared to the networks with smaller width.

3. PREDICTING GENERALIZATION IN DEEP LEARNING

In this chapter, generalized resubstitution error estimators were employed to empirically estimate the generalization of deep neural network models. Section 3.2 presents three data augmentation approaches towards predicting generalization gap in deep learning, where I used a large set of pre-trained deep models, provided by the NeurIPS 2020 competition [41], which was the first competition on PGDL.

3.1 Definitions

This section defines the notations and evaluation metrics that are used in the rest of this chapter. For consistency with deep learning literature, new notations are used for some of the previously defined variables. For example, here instead of S_n, the training data is denoted by $\mathcal{D}_{train}$.

3.1.1 Notations

Denote the i^{th} hyperparameter of a network by $\theta_i \in \Theta_i$, for $i = 1, 2, \ldots, H$, where H is the total number of types of varying hyperparameters. A network can be created by a vector of selected hyperparameters $\boldsymbol{\theta} \triangleq (\theta_1, \theta_2, \theta_3, \ldots, \theta_H)$, where $\boldsymbol{\theta}$ is a member of the hyperparameter space $\Theta \triangleq \Theta_1 \times \Theta_2 \times \Theta_3 \times \cdots \times \Theta_H$. For the experiments in this chapter, all members of Θ are used to create networks. Hence, given the set of hyperparameters, the total number of created networks is equal to $|\Theta|$. For example, if θ_1 and θ_2 are a network's width and depth, with possible values of $\{2, 3, 4\}$ and $\{10, 20\}$, respectively, then a total of 6 networks will be created.

As defined by equation 2.46, let the output of a network be $f_i(\mathbf{x}, \theta)$, where $\mathbf{x}$ is a data point and θ is one set of selected hyperparameters. Let $\mathcal{D}_{train} = \{(\mathbf{x}_i, y_i)\}$ and $\mathcal{D}_{val} = \{(\mathbf{x}_j, y_j)\}$ be the training and validation datasets, respectively, where $i = 1, 2, N_t$ and $j = 1, 2, N_v$. Given a vector θ and a data sample $\mathbf{x}$, a trained network produces wights $\mathbf{w}$ and outputs the prediction $f_{\mathbf{w}}(\mathbf{x})$. The

true accuracy of the model is defined as

$$Acc = \frac{1}{|\mathcal{D}_{val}|} \sum_{(\mathbf{x},y)\in\mathcal{D}_{val}} \delta(f_{\mathbf{w}}(\mathbf{x}) = y) \tag{3.1}$$

The goal is to find a measure $\mu : (f_{\mathbf{w}}, \mathcal{D}) \rightarrow \mathbb{R}$ such that $[\mu(\mathbf{w}, \mathcal{D}_{train}) - Acc]$ is minimized. Alternatively, a good measure μ can order the models in from highest to lowest generalization ability.

3.1.2 Conditional Mutual Information

To be consistent with the NeurIPS 2020 competitions, I used Conditional Mutual Information (CMI) [1] to evaluate the proposed generalization predictors for the image data experiments. Inspired by the algorithms for Inductive Causation (IC) [84], CMI evaluates the existence and strength of causal relationship between the measure μ and observed generalization of a network in a causal probabilistic graph [7]. The main goal is to determine whether or not there is an edge between μ and generalization by a conditional independence test. To do so, given the observed hyperparameters conditional mutual information between μ and generalization should be computed. Conditioned on the set of hyperparameters $\theta_{\mathbf{i}}$, the mutual information between μ and generalization gap measures the ability of μ in predicting the difference of generalization between pairs of models.

Borrowing the notations from [41], let $V_\phi(\theta, \theta') \triangleq \operatorname{sgn}(\phi(\theta_1) - \phi(\theta'))$ for any function $\phi : \Theta \rightarrow \mathbb{R}$ and any pair of $(\theta, \theta') \in \Theta \times \Theta$. Also, denote the generalization gap by g and the set of observed hyperparameters by $\mathcal{O}$. Then, the total CMI metric is defined as the average of $\mathcal{J} \triangleq \min_{\mathcal{O}} \hat{\mathcal{I}}(V_g, V_\mu|\mathcal{O})$ across all datasets:

$$CMI(\mu) = \sum_{\mathcal{D}} \mathcal{J}(\mu; \mathcal{D}) \tag{3.2}$$

where $\hat{\mathcal{I}}(V_g, V_\mu|\mathcal{O}) = \frac{\sum_{\mathcal{O}_k} p_c \, \mathcal{I}(V_g, V_\mu|\mathcal{O})}{\mathcal{H}(V_g|\mathcal{O})}$, $p_c = \frac{1}{\prod_{i=1}^{N} |\Theta_i|}$ because all $\mathcal{O}_k$ are equally probable. $\mathcal{H}$ is a normalization factor defined by the conditional entropy and $\mathcal{I}$ is the mutual information between

[1] CMI derivation and equations are brought from the original references [7, 41]

V_g and V_μ, defined as:

$$\mathcal{H}(V_g|\mathcal{O}) = \sum_{\mathcal{O}_k} p_c \sum_{V_g} p(V_g|\mathcal{O}_k) \log\left(p(V_g|\mathcal{O}_k)\right)$$

and

$$\mathcal{I}(V_g, V_\mu|\mathcal{O}_k) = \sum_{V_g} \sum_{V_\mu} p(V_g, V_\mu|\mathcal{O}_k) \log\left(\frac{p(V_g, V_\mu|\mathcal{O}_k)}{p(V_\mu|\mathcal{O}_k)p(V_g|\mathcal{O}_k)}\right)$$

For the detailed derivation of the MCI metric, see [7].

3.2 Image Data Experiments

This section presents the simple, yet effective, augmented posterior probability generalization predictors, which are inspired by the bolstered posterior probability error estimators that was described in section 2. I employed the package provided by the PGDL competition [41] to run experiments for evaluation of several variations of augmented posterior probability generalization predictors. The PGDL package includes trained networks and their configurations, utilities, scoring code, as well as datasets for all tasks in phase 1 and 2. The package is open-sourced and is available under the Apache 2.0 license [2].

In PGDL competition, participants were required to build a function which takes as inputs a trained neural network and training data, and returns as output the generalization ability of the trained network for an unseen test data. The competition consisted of two phases: 1- development phase, when participants develop functions given network configurations and datasets, and 2- the evaluation phase, in which the submitted functions were tested on new datasets which were not initially available to the participants.

In this dissertation, the experiments were performed on three variations of the proposed generalization predictor: 1- when augmentation parameters were found via greedy search in the space of possible parameters' values, 2- when the amount of augmentation was computed using a random sample from normal distribution with mean 0 and standard deviation σ, 3- Weighted augmented

[2] https://sites.google.com/view/pgdl2020/

Task #	Architecture	# of Models	Dataset	# of samples
1	VGG-like	96	Cifar10 [85]	60,000
2	Network in Network	54	SVHN [86]	99,000
6	Network in Network	96	Oxford Flowers [87]	8,189
7	Network in Network	48	Oxford Pets [88]	7,400
8	VGG-like	64	Fashion MNIST [89]	70,000
9	Network in Network	32	Cifar10 [85]	60,000

Table 3.1: Network architecture and datasets that were used for PGDL tasks.

- semi posterior probability with weights on augmented samples, when augmented samples were generated in the same way as the second variation. The latter approach was inspired by the first runner-up solution in PGDL competition [53] as well as the semi-bolstered approach for the k-NN classifiers in section 2. The rest of this section describes each of the above approaches in more details.

3.2.1 Networks Configurations and Datasets

Phase 1 consisted of two tasks (tasks 1 and 2), whereas phase 2 was performed on four tasks (tasks 6-9). Table 3.1 presents more details of the tasks[3].

3.2.2 Augmentation Parameter Search

Note: this approach is referred to by "Par-Search" throughout the rest of this chapter.

Generalization predictors that are based on simple data augmentations are prone to overfit the training data. Similarly, error estimates based on simple and general augmentations of the training data does not notably improve the standard resubstitutin error and are typically largely biased. In practice, the general framework is to set the augmentation factor equal to a random sample from a uniform distribution over a *wide range* of possible values for the corresponding factor.

In addition to the increased variability, the aforementioned data augmentation reduces the ability of the image generator to create meaningful images for the error estimation task. An image is meaningful if it is generated with a large enough augmentation factor and yet be realistic. For

[3]https://github.com/google-research/google-research/tree/master/pgdl

example, a horse with 80 degrees rotation is not realistic but an airplane with the same rotation can exist. Hence, providing a narrow but meaningful range for augmentation factors can improve the error estimate, and perhaps the generalization prediction.

Par-Search was initially inspired by the calibration approach that was introduced in section 2.2.4 for the MNIST data experiments. The procedure is to first define sets of possible and *narrow* ranges for each augmentation factor, and then search for the best combination of ranges that results in the lowest estimate of the bias of the error estimator. The bias is roughly estimated by training the classifier on a random sample of 80% of the images form the available training data, and testing it on the remaining 20%.

An exhaustive search would consider all combinations of augmentation parameters' ranges, but would require many steps to test all possibilities. Alternatively, a greedy search explores the parameters' space in a shorter time, but the final set does not guarantee the lowest bias estimate. Indeed, the main drawback of this approach is that the model should to be trained and tested on many combinations of the parameter ranges, which can be cumbersome. For deep neural network models, Par-Search is infeasible due to the large requirements for memory over a long period of time. Next subsection describes an alternative approach that increases the amount of augmented data but requires only one training step for the model.

3.2.3 Multiple Augmented Sets

Note: this approach is referred to by "Multi-Augment" throughout the rest of this chapter.

Training multiple models in Par-Search approach was impractical for deep neural networks. Alternatively, I generated multiple augmented datasets with distinct parameter ranges and used them for trained model error estimation. Inspired by the bolstering approach in chapter 2, the choice of parameters' range in Multi-Augment method is determined by a sample from a normal distribution. The procedure is to, for each input image $\mathbf{x}$, generate M augmented images and average the outcomes. Each of the M images is generated by adding an augmentation to "x" with a distinct augmentation factor. The augmentation factor for each of the M images is a random sample drawn from a normal distribution $\mathcal{N}(0, \sigma)$, where σ is the standard deviation equal to one-

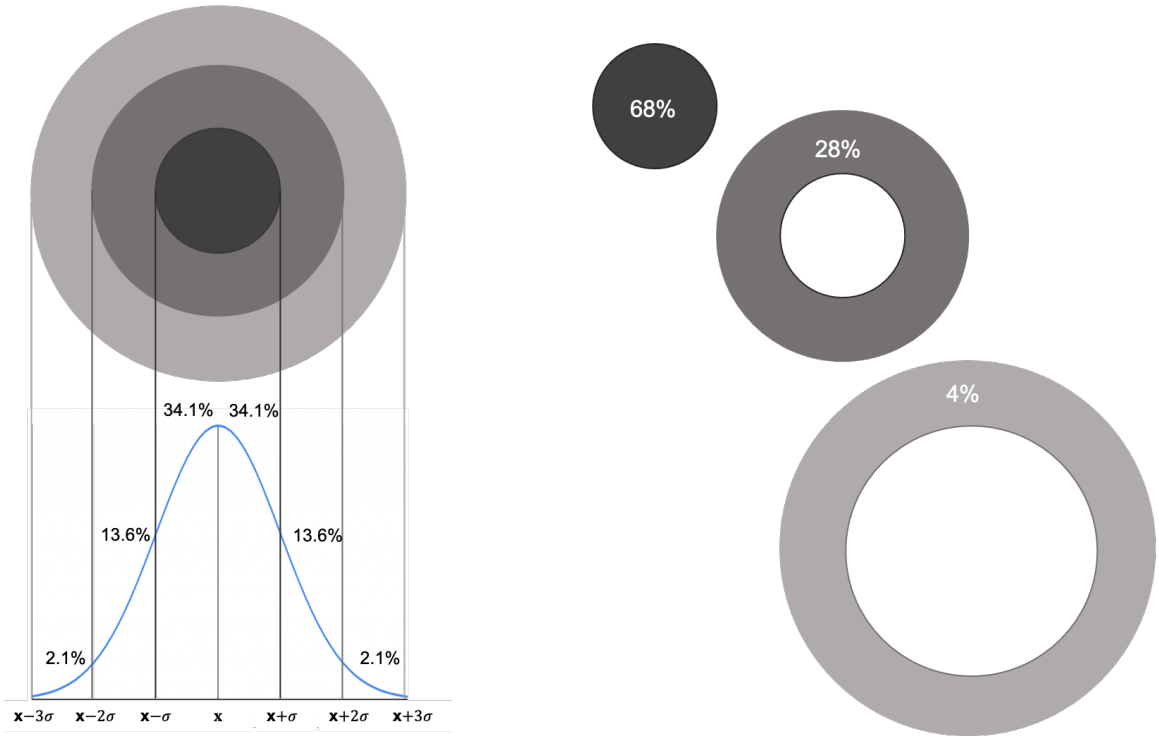


Figure 3.1: Rings that were used as ranges for augmentation factor in multiple augmented datasets generation.

third of the maximum value, in magnitude, that would make sense for that type of augmentation. For example, if the rotation must be between -90 and 90 degrees, then the rotation augmentation factor would be a sample from $\mathcal{N}(0, 30)$, and therefore, %99 of the randomly generated factors are between -90 and 90 degrees.

In the following experiments, I resembled a normal distribution by manually specifying the augmentation range. To do so, I defined a meaningful maximum value for each factor according to the corresponding augmentation type. Then assuming that, in a normal distribution, 99% confidence interval is within three standard deviation of the mean, I defined three rings around each data point $\mathbf{x}$. The first ring covers $[-\sigma, \sigma]$ and includes 68% of the augmented data. The second ring covers $([-2\sigma, -\sigma), (\sigma, 2\sigma])$ and contains 27% of the augmented data points. The third ring covers $([-3\sigma, -2\sigma), (2\sigma, 3\sigma])$ and includes 4% of the augmented data points. (See Figure 3.1 for illustration.)

I let $M = 25$, and generated 17, 7, and 1 samples by using the first, second, and third rings, respectively, as the range for the augmentation factor. The samples in each ring were generated from a uniform distribution defined by the minimum and maximum of the ring. The second and third rings contain two intervals, one with negative and one positive. For the factors that accept both positive and negative values, I generated, respectively, 7 and 1 random boolean digits (b). If $b = 0$, then the corresponding factor's range was set to the negative interval. Otherwise, it was set to the positive interval. For the augmentation types that only accept positive values, I modified the rings to $(0, \sigma]$, $(\sigma, 2\sigma]$, and $(2\sigma, 3\sigma]$ for the first, second, and third rings, respectively.

3.2.4 Weighted Augmented - Semi Posterior Probability

Note: this approach is referred to by "Weighted Augment - Semi PP" throughout the rest of this chapter.

In this approach, a penalty is calculated for each augmented data depending on whether or not it is correctly classified by the trained network. Given an original input $(\mathbf{x}, y)$, denote the augmented image by $\mathbf{x}^*$. A trained model $\psi_n(.)$ classifies $\mathbf{x}^*$ in the correct class with posterior probability $\hat{P}_n\left(\psi_n(\mathbf{x}^*) = y|\mathbf{X} = \mathbf{x}^*\right)$. This posterior probability estimate is used to define the penalty for correctly classified augmented images, whereas for the incorrectly classified $\mathbf{x}^*$, the penalty is computed as a function of the strength and amount of augmentation.

As suggested by [53], reasonable augmentation of input data does not change the labels predicted by a trained models, if the trained model is generalizable. In other words, if the model has learned the right features, then it should be robust to augmented data that is from the underlying data distribution. The strength of augmentation was defined according to the change in an image's texture after. The larger difference between the texture of an input image and its augmented image the stronger the augmentation is.

Although the strength of augmentation may depend on the present, given the datasets for all tasks of PGDL, I defined categories of strong, moderate, and weak augmentation types. Each category is assigned a strength rank from 1 to 3, which will be used later for computation of misclassification penalties. For each augmentation type, let r be the rank and f_i be the amount

Augmentation	Category	Range
Cut Height	Strong	$[0, 0.5 \times \text{image height}]$
Cut Width	Strong	$[0, 0.5 \times \text{image width}]$
Height Shift	Strong	$[0, 0.5 \times \text{image height}]$
width Shift	Strong	$[0, 0.5 \times \text{image width}]$
Zoom out or in (Horizontal)	Strong	$[50\%, 150\%]$
Zoom out or in (Vertical)	Strong	$[50\%, 150\%]$
Sobel Filter	Moderate	boolean $(0, 1)$
Darkness or Brightness	Weak	$[50\%, 150\%]$
Rotation	Weak	$[-\pi/4, \pi/4]$
Horizontal Flip	Weak	boolean $(0, 1)$

Table 3.2: Image augmentations types and categories with their range set.

of augmentation that was applied to the image. Also, let superscripts S, M, and W represent the strong, moderate and weak augmentation types. The strong category, with $r^S = 1$, includes random cutout, horizontal shift, vertical shift, and zoom in. Sobel filter was assumed to be a moderate augmentation type with $r^M = 2$. The weak category, with $r^W = 3$, includes brightness, rotation, and horizontal flip. Table 3.2 presents the more details about the augmentation types. Figures 3.2, 3.3, and 3.4 display some random examples of generated images using augmentation types form each category.

For each augmentation type i, a factor f_i was randomly selected from the range that is specified in table 3.2. For zoom augmentation, if $f_{\text{zoom}} \in [50\%, 100\%]$, the image is zoomed in (i.e. the center of the image becomes larger and the sides of the image are removed). Otherwise, the image is zoomed out (i.e. the image becomes smaller and the edge pixels are repeated to fill the empty pixels on the sides). Similarly, for brightness, if $f_{\text{brightness}} \in [50\%, 100\%]$ the image becomes darker and if $f_{\text{brightness}} \in [100\%, 150\%]$ the image becomes brighter. Subsequently, a normalized factor f_i' was calculated as following, where $|.|$ takes the absolute value:

- $f'_{cut} = 4 \times \frac{\text{area of cutout}}{\text{area of image}}$, where subscript cut refers to the random cutout.
- $f'_{hs} = 2 \times f_{\text{height shift}}$, where subscript hs refers to random shift.
- $f'_{ws} = 2 \times f_{\text{width shift}}$, where subscript ws refers to the height shift.

- $f'_{zh} = 2 \times |\frac{100 - f_{\text{zoom}}}{100}|$, where subscript zh refers to the horizontal zoom.
- $f'_{zv} = 2 \times |\frac{100 - f_{\text{zoom}}}{100}|$, where subscript zv refers to the vertical zoom.
- $f'_{b} = 2 \times |\frac{100 - f_{\text{brightness}}}{100}|$, where subscript b refers to the brightness.
- $f'_{r} = 4 \times |\frac{f_{rotation}}{\pi}|$, where subscript r refers to the rotation.
- $f'_{f} = f_{flip} \in \{0, 1\}$ and $f'_{s} = f_{\text{sobel filter}} \in \{0, 1\}$.

Let $F'^{S} = \{f'_{cut}, f'_{hs}, f'_{ws}, f'_{zh}, f'_{zv}\}$ and $F'^{W} = \{f'_{b}, f'_{r}\}$ be the set of normalized factors for strong and weak augmentations, respectively. Then, the penalty for an incorrectly classified augmented image $\mathbf{x}^*$ is:

$$\lambda'_{\mathbf{x}^*} = \left(1 - \max_{f' \in F'^{S}} f'\right) r^{S} + \left(1 - \max_{f' \in F'^{W}} f'\right) r^{W} + (1 - f'_{f}) r^{W} + (1 - f'_{s})\, r^{M}, \tag{3.3}$$

where $0 \leq \lambda'_{\mathbf{x}^*} \leq 9$, and $\mathbf{x}^*$ is the augmented image generated from the original input image $\mathbf{x}$. Figures 3.6 and 3.7 display two examples of calculated λ^* for two images with various augmentations. The penalty associated to each $(\mathbf{x}, y) \in \mathcal{D})$ is:

$$\lambda_{\mathbf{x}} = \begin{cases} 1 - \hat{P}_n(\psi_{n,\theta}(\mathbf{x}^*) = y | \mathbf{X} = \mathbf{x}^*) & \text{if } \psi_{n,\theta}(\mathbf{x}^*) = y \\ \max(1, \lambda'_{\mathbf{x}^*}) & \text{if } \psi_{n,\theta}(\mathbf{x}^*) \neq y \end{cases} \tag{3.4}$$

and the generalization measure μ is computed as negative of the sum of all penalties.

$$\mu(\psi_{n,\theta}, \mathcal{D}) = - \sum_{(\mathbf{x},y) \in \mathcal{D}} \lambda_{\mathbf{x}} \tag{3.5}$$

Therefore μ returns a larger penalty, in magnitude, for models with poor generalization.

3.2.5 Discussion

The experimental results suggest that the *Weighted Augment - Semi PP* method performed better than the other two methods of this chapter. Table 3.3 compares the CMI score of the methods

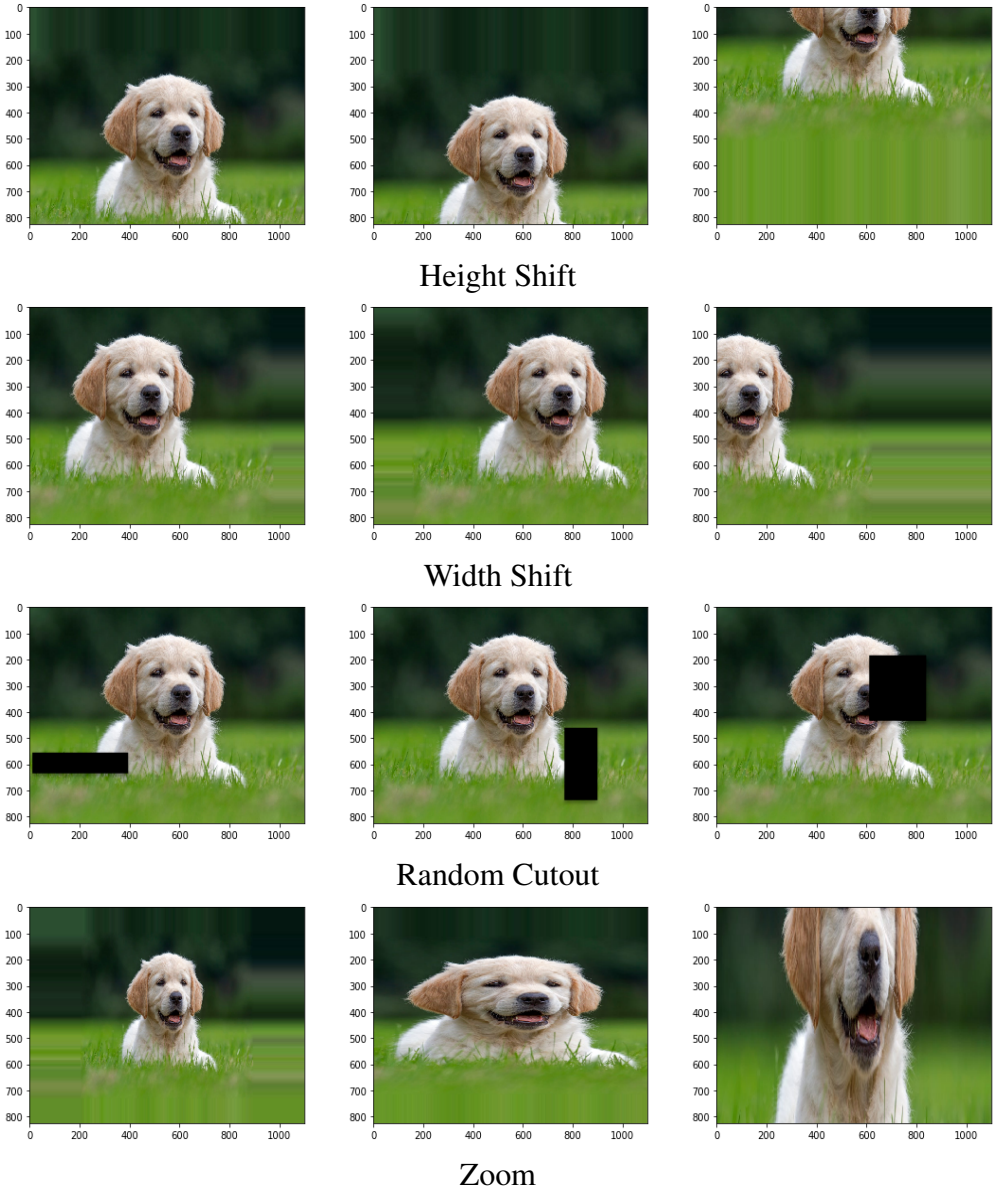


Figure 3.2: Strong augmentations; the augmentation factor becomes larger from left to right.

of this chapter with the top winners of the PGDL competition. The competition ranks were based on the average of the CMI score on phase 2 tasks (tasks 6-9). Beside the score, the percentage change in performance between phase 1 and 2 is notably an important factor. A significant drop in phase 2 scores (large negative percent change) for a particular method indicates that the method overfits the data in phase 1. A large improvement in the phase 2 scores (large positive percent

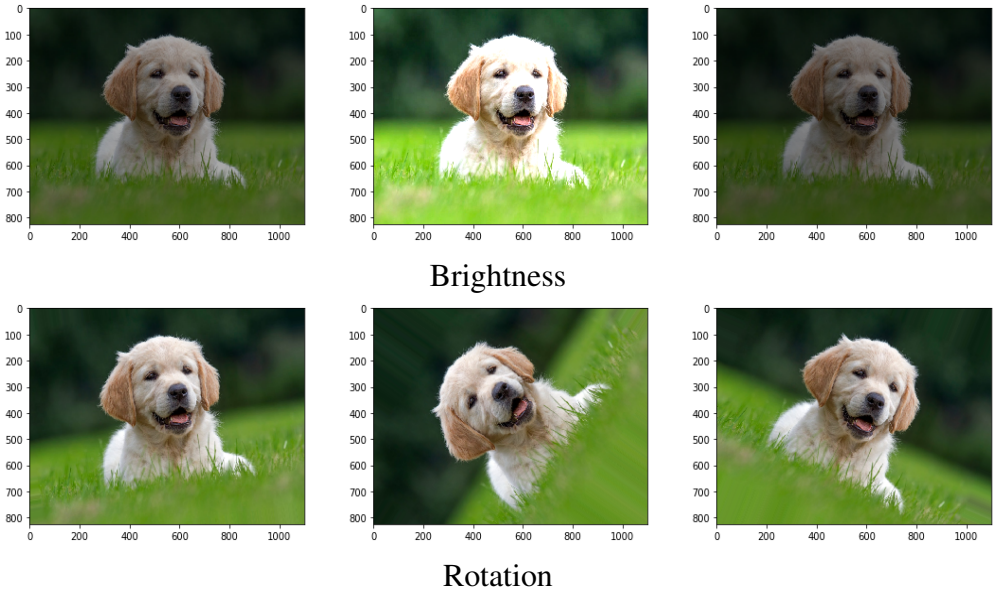


Figure 3.3: Weak augmentations; the augmentation factor becomes larger from left to right.

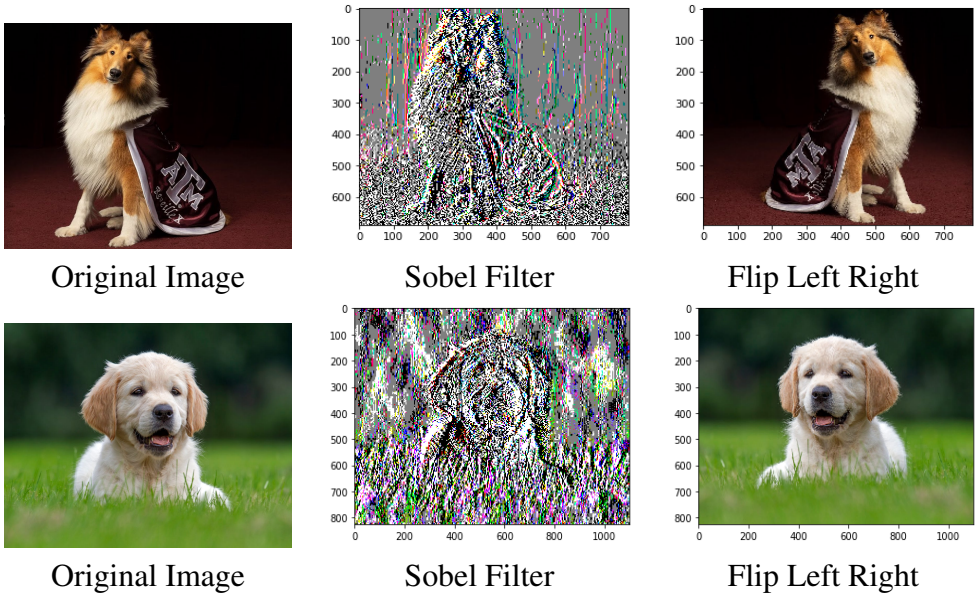


Figure 3.4: Examples of the two fixed augmentation types: Sobel filter and left right flip. These two augmentation types may affect various images differently depending on how symmetric and contrasted they are.

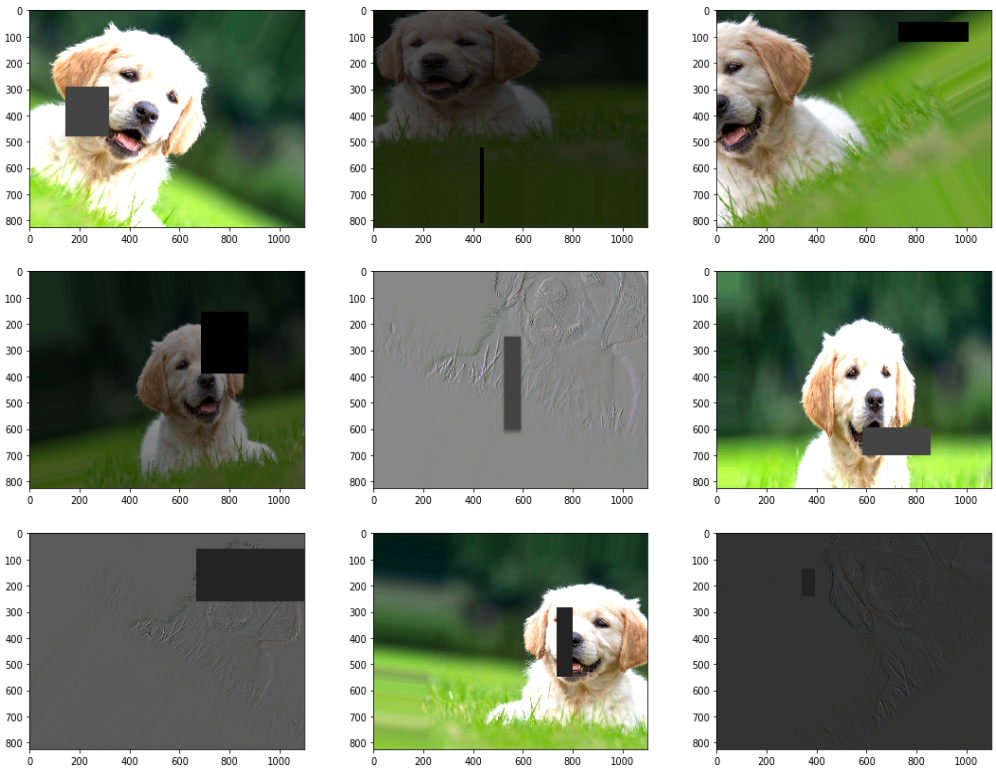
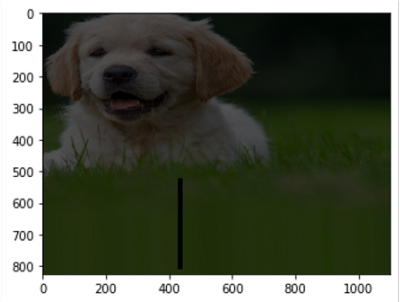


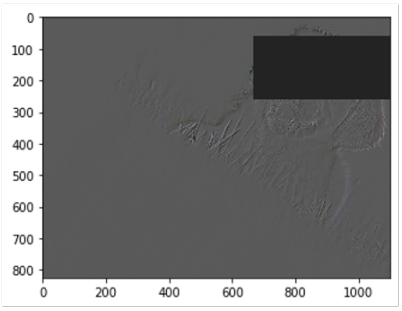
Figure 3.5: Examples of images generated with combination of all eight augmentation types. Flip and sobel filter was randomly applied to images with probability of 0.5.



$f'_{cut} = 0.05$ $f'_{b} = 0.56$
$f'_{hs} = 0.25$ $f'_{r} = 0.00$
$f'_{ws} = 0.34$
$f'_{zh} = 0.01$ $f'_{f} = 1$
$f'_{zv} = 0.00$ $f'_{s} = 0$

$$\lambda'_{x^*} = (1 - 0.34)\times 1 + (1 - 0.56)\times 3 + (1 - 1)\times 3 + (1 - 0)\times 2 = 3.98$$

Figure 3.6: Example of an augmented image with *large* misclassification penalty.



$f'_{cut} = 0.38$ $f'_{b} = 0.20$
$f'_{hs} = 0.05$ $f'_{r} = 0.93$
$f'_{ws} = 0.22$
$f'_{zh} = 0.09$ $f'_{f} = 1$
$f'_{zv} = 0.11$ $f'_{s} = 1$

$$\lambda'_{x^*} = (1 - 0.38)\times 1 + (1 - 0.93)\times 3 + (1 - 1)\times 3 + (1 - 1)\times 2 = 0.83$$

Figure 3.7: Example of an augmented image with *small* misclassification penalty.

change) suggests that the method underfit the data from phase 1.

Weighted Augment - Semi PP is fairly robust to the change of network architecture and training dataset. The percentage change in tasks 6 and 7 were 1.25% and -9.66%, respectively, which is considerably smaller than the top three winners, Interplex (13.58% and -67.49%), Always-Generalize (-80.88% and 18.45%), and BrAIn (49.95% and 18.45%). *Multiple-Augmentation* did not achieve a high score in either tasks, which can be due to the fact that the augmented data in this method is close to the original images and the increased number of augmented data did not help in generalization prediction. Additionally, the percent difference was large indicating that this approach may be vulnerable to overfitting or underfitting. The *Par-Search* method returns good score when the dataset is small. Otherwise, it did not finish the job in the allocated time and memory.

Name	phase 1 score	Task 6	Task 7	Task 8	Task 9
Weighted aug - semi pp	24.91	21.25	**21.60**	15.38	20.14
Multiple Aug	4.21	4.72	3.44	2.34	3.06
Par search	–	62.51	59.46	–	–
Interplex [57]	38.73	**43.99**	12.59	9.24	**22.92**
Always-Generalize [90]	**41.79**	7.99	7.85	12.41	10.12
BrAIn [91]	9.27	13.90	7.56	**16.23**	2.28

Table 3.3: Scores for the tree approaches of data augmentation presented in this section compared to the top three NeurIPS first PGDL competition. A higher score shows the better generalization prediction. The highest score is printed in bold.

Name	phase 1 score	Task 6	Task 7	Task 8	Task 9
Weighted aug - semi pp	24.91	**+1.25%**	**-9.66%**	**-38.25%**	**19.04%**
Multiple Aug	4.21	12.11%	-18.29%	44.41%	27.32%
Par search	–	–	–	–	–
Interplex	38.73	13.58%	-67.49%	-76.17%	-33.23%
Always-Generalize	41.79	-80.88%	-81.21%	-70.95%	-75.78%
BrAIn	9.27	49.95%	18.45%	+75.10%	-75.40%

Table 3.4: Percentage change of the scores in tasks 6-9 compared to the phase 1 score. The lower percentage indicates the more robustness of the corresponding method to changes in data and network configuration. The lowest percentage change is printed in bold.

The high requirements of the Par-Search approach makes it impractical for deep neural networks and large datasets. It also requires the network to be trained from scratch multiple time, which is extremely unfavorable among the deep learning communities. The Multi-Augment approach alleviates the requirements of Par-Search by removing all steps for the model training. Instead of searching for the best ranges of augmentation factors, it generates multiple augmented datasets using several possible ranges for augmentation factors and combine them to create a large augmented dataset. In this dissertation, I considered three possible ranges, and named them rings. The number augmented samples that are generated using the factors in each ring should depend on how well the ring can describe the optimal range for the factor. In this dissertation, I defined the rings based on the first, second, and third standard deviation of a Gaussian distribution around any input image. Subsequently, I defined the importance of each ring (i.e. the number of samples

generated using each ring) based on the confidence interval of the mean of Gaussian distributions.

Although Muti-Augment improved the time and memory efficiency, compared to Par-Search, it increases the training sample size by 25 fold and remains inefficient for the tasks that contain large training datasets. In fact, the majority of augmented images were close to the original images and did not capture the generalization gap of the models. Of course, this happened in the tasks that are presented in this dissertation, where almost all of the networks were trained up to more than 95% training accuracy. Most models were over-fitted on the training data and therefore the Multi-Augment approach in the present form does not correlate well with the true error and neither with the generalization gap. However, in future research, one may try the performance of Multi-Aug method in error estimation tasks where the models are not over-fitting the training data.

This chapter starts with definitions and equations of conventional compartmental models (CCM). Subsequently, hybrid compartmental models (HCM) are introduced and compared with the CCMs. Additionally, HCM was used to model the metabolism of Phenylalanine (Phe) and Tyrosine (Tyr). Details of model structure and parameter estimation procedure, for the Phe-Tyr system, are illustrated in the subsection for experimental results.

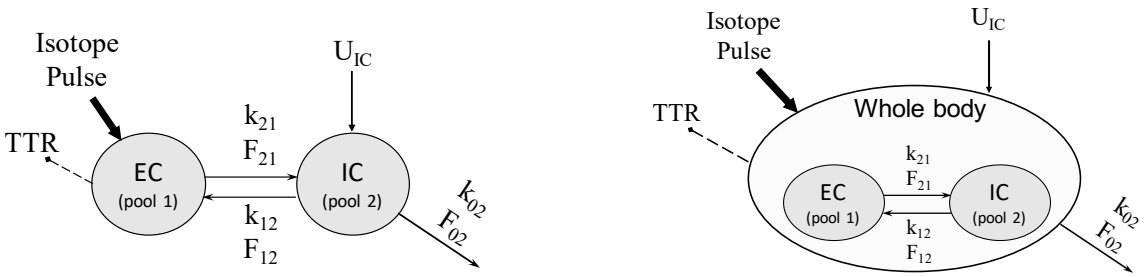
4.1 Conventional Compartmental Models (CCM)

Consider the two-compartment model with EC and IC pools depicted in Figure 4.1a. The EC pool represents the blood and fluids surrounding the cells, and its irreversible loss is assumed to be zero. The IC pool represents the tissue and fluid inside the cell, which is also the site of metabolism.

Sampling from the IC pool requires surgery or a biopsy from the tissue and therefore, IC pool remains mostly non-accessible in clinical studies. The model on figure 4.1a is defined by a second order differential equation, for which the time-domain solution is given by

$$q_1(t) = A_1 e^{\lambda_1 t} + A_2 e^{\lambda_2 t} \tag{4.1}$$

This time domain-equation is regressed on the TTR-time data. Once a good fit is achieved, the fractional rates (k_{ij}s) are calculated as $k_{11} = -(A_1\lambda_1 + A_2\lambda_2)/(A_1 + A_2)$ for the sum of all tracer out-flows from pool 1, $k_{22} = (A_2\lambda_1 + A_1\lambda_2)/(A_1 + A_2)$ for the sum of all tracer out-flows from pool 2, and $k_{12}k_{21} = (A_1A_2(\lambda_1 - \lambda_2)^2)/((A_1 + A_2)^2)$ as the product of tracer transfer rates between extra and intracellular pools. With the assumption that there is no loss from the EC pool (i.e. $k_{01} = 0$), all parameters k_{ij}s in the model in Figure 4.1a are uniquely identifiable: $k_{21} = -(k_{11})$, $k_{12} = (k_{12}k_{21})/k_{21})$, and $k_{02} = -k_{22} - k_{12}$. If the extracellular pool were mixed with another fast-mixing pool with some loss of substrate, then the parameters would not be uniquely identifiable, but could be interval-identifiable as explained by [1, 63]. Finally, the tracee fluxes are calculated as $F_{ij} = k_{ij}Q_j$, and the protein breakdown into a pool is computed as the accumulated out-flow



(a) CCM (b) HCM

Figure 4.1: Single-substrate two-compartment models

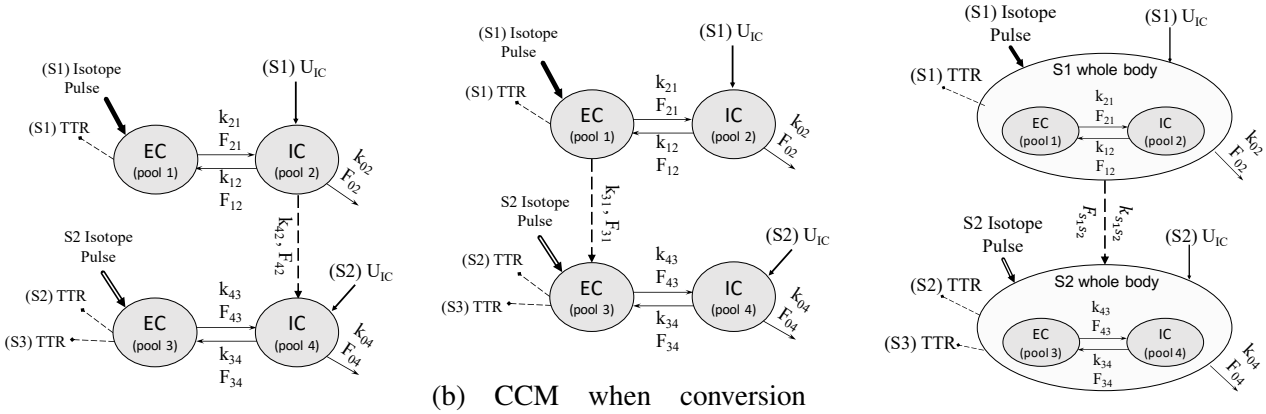
fluxes from the pool minus the in-flow fluxes to that pool.

4.2 Hybrid Compartmental Models (HCM)

An HCM is similar to CCM for single-substrate two-compartmental models, except for the sketch of the models (see Figure 4.1b). For a multi-substrate model, however, the two modeling approaches are different in both structure and parameters estimation. A multi-substrate model CCM often face challenges in finding the best graph representation that fits the sampled TTR. Unfortunately, the solution to this problem is not known prior to regressing the model on the TTR data. In such case, the structure is solely determined by physiological insights.

Our proposed HCM address this issue by considering multiple tracee alternatives when needed. Indeed, it assigns a Whole body (Wb) mode for the conversion rate and simultaneously remains in the Multi-compartment (Mc) mode the Mc-mode for the rest of the parameters. Figure 4.2 illustrates the HCM version of the two CCMs for a two-substrate four-compartment mode. The $k_{s_1 s_2}$ parameter in HCM is the tracer fractional conversion rate from substrate 1 to substrate 2 and corresponds to the CCM conversion parameters k_{42} and k_{31} in figures 4.2a and 4.2b, respectively. This HCM can easily be generalized to a two-substrate multi-compartments model.

Once the tracer HCM is identified, all possible tracee alternative models are produced and the tracee parameters are calculated by simple math operations. Note that three tracee models can be generated from the HCM depicted on figure 4.2c by moving the conversion flux: 1- between the intracellular pools, 2- between the extracellular pools, and 3- between two substrates pools, each



(a) CCM when conversion occurs between intracellular pools.

(b) CCM when conversion occurs between extracellular pools.

(c) HCM that associates the conversion to the whole body.

Figure 4.2: Models for a two-substrate four-compartment model.

consisting of the merged IC and EC pool of one substrate.

In the following, we present the mathematical derivations for HCMs. Since the isotopic tracers do not alter the biochemical function of the tracee, their behaviour is similar to the linear time invariant (LTI) systems. Let $h(t)$ be the impulse responses an LTI system. Then the response of the LTI system to an arbitrary input function $i(t)$ can be computed as:

$$f(t) = i(t) * h(t) = \int_{-\infty}^{+\infty} i(\tau)h(t-\tau)d\tau \tag{4.2}$$

Hence, the equations for the Phe-Tyr model in figure 4.3 can be defined as:

$$h_{phe6}(t) = A_1 e^{\lambda_1 t} + A_2 e^{\lambda_2 t} \tag{4.3a}$$

$$h_{tyr4}(t) = B_1 e^{\gamma_1 t} + B_2 e^{\gamma_2 t} \tag{4.3b}$$

$$f_{phe6}(t) = h_{phe6}(t) \tag{4.3c}$$

$$f_{tyr4}(t) = h_{tyr4}(t) \tag{4.3d}$$

$$f_{tyr6}(t) = k_{phe\to tyr}\left(f_{Phe6}(t) * f_{tyr4}(t)\right) \tag{4.3e}$$

$$= k_{phe\to tyr}\left(\sum_{i=1}^{2} \alpha_i e^{-\lambda_i t} + \sum_{j=1}^{2} \omega_j e^{-\gamma_j t}\right) \tag{4.3f}$$

where the third and fourth equalities are true because the input is a bolus of isotopic tracer, and therefore $i(t) = \delta(t)$. Also, $\alpha_1 = \frac{A_1 B_1}{\gamma_1 - \lambda_1} + \frac{A_1 B_2}{\gamma_2 - \lambda_1}$, $\alpha_2 = \frac{A_2 B_1}{\gamma_1 - \lambda_2} + \frac{A_2 B_2}{\gamma_2 - \lambda_2}$, $\omega_1 = -\left(\frac{A_1 B_1}{\gamma_1 - \lambda_1} + \frac{A_2 B_1}{\gamma_1 - \lambda_2}\right)$, and $\omega_2 = -\left(\frac{A_1 B_2}{\gamma_2 - \lambda_1} + \frac{A_2 B_2}{\gamma_2 - \lambda_2}\right)$.

Theorem 3. *For a two-substrate model with n and m compartments for the first and second substrate, respectively, when $h_{S_1}(t) = \sum_{i=1}^{n} A_i e^{\lambda_i t}$ and $h_{S_2}(t) = \sum_{j=1}^{m} B_j e^{\gamma_j t}$, and when one substrate is a natural product of the other one, the product substrate can be described by:*

$$f_{product}(t) = k_{S_1 \to S_2} \left(\sum_{i=1}^{n} \alpha_i e^{-\lambda_i t} + \sum_{j=1}^{m} \omega_j e^{-\gamma_j t} \right) \tag{4.4}$$

where $\alpha_i = \sum_{j=1}^{m} \frac{A_i B_j}{\gamma_j - \lambda_i}$ and $\omega_j = \sum_{i=1}^{n} \frac{A_i B_j}{\lambda_i - \gamma_j}$.

Proof. Let $\mathcal{L}\{.\}$ denote the Laplace transform. Then

$$\begin{aligned}
\mathcal{L}\{f_{tyr6}\} &= \mathcal{L}\{f_{Phe6}(t) * f_{tyr4}(t)\} \\
&= \mathcal{L}\{f_{Phe6}(t)\}\mathcal{L}\{f_{tyr4}(t)\} \\
&= \left(\sum_{i=1}^{n} \frac{A_i}{s + \lambda_i} \right) \left(\sum_{i=1}^{m} \frac{B_j}{s + \gamma_j} \right) \\
&= \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{A_i B_i}{(s + \lambda_i)(s + \gamma_j)}
\end{aligned}$$

which, by partial fraction expansion, can be simplified to:

$$\mathcal{L}\{f_{tyr6}\} = \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{A_i B_j}{\gamma_j - \lambda_i} \left(\frac{1}{s + \lambda_i} - \frac{1}{s + \gamma_j} \right)$$

and using the inverse Laplace transform, we have:

$$\begin{aligned}
&\mathcal{L}^{-1}\left\{\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{A_iB_j}{\gamma_j-\lambda_i}\left(\frac{1}{s+\lambda_i}-\frac{1}{s+\gamma_j}\right)\right\}\\
&=\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{A_iB_j}{\gamma_j-\lambda_i}\left(e^{-\lambda_i t}-e^{-\gamma_j t}\right)\\
&=\sum_{i=1}^{n}\sum_{j=1}^{m}\left(\frac{A_iB_j}{\gamma_j-\lambda_i}e^{-\lambda_i t}+\frac{-A_iB_j}{\gamma_j-\lambda_i}e^{-\gamma_j t}\right)\\
&=\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{A_iB_j}{\gamma_j-\lambda_i}e^{-\lambda_i t}+\sum_{j=1}^{m}\sum_{i=1}^{n}\frac{A_iB_j}{\lambda_i-\gamma_j}e^{-\gamma_j t}
\end{aligned}$$

By replacing $\alpha_i = \sum_{j=1}^{m}\frac{A_iB_j}{\gamma_j-\lambda_i}$ and $\omega_j = \sum_{i=1}^{n}\frac{A_iB_j}{\lambda_i-\gamma_j}$ and multiplying by the conversion rate $k_{S_1\to S_2}$, one obtains equation 4.4. □

Corollary 3.1. *The HCM described in theorem 3 has total of* $2(n+m)+1$ *parameters to be identified, including two parameters per compartment and one parameter for the conversion rate.*

Compartmental models significantly reduce the number of unknown parameters (a.k.a degrees of freedom) compared to the non-compartmental approach with $2(n+m)+2p$ unknown parameters for the model of theorem 3, where p is the number of exponential terms for the metabolite product.

4.3 HCM versus CCM for Multi-Substrate Models

HCM and CCM share many similarities. The pipeline to build each one of them involves four major steps: 1- specifying the model's graphical representation, 2- fitting the tracer model on the TTR-time data, 3- computing the tracer irreversible losses and fractional transfer rates (k_{ij}), and 4- calculating the tracee fluxes by $F_{ij} = k_{ij}Q_j$ as well as the protein break-down rates. In both methods, the tracer model was identified by non-linear iterative regressions, which involves with minimizing a cost function, meaning that a tracer parameter is reported as a variable with mean and standard deviations. Once a good fit on the tracer model is achieved, the exact tracee parameters are computed by basic mathematical operations, though the standard deviation of the tracer parameters are propagated during the tracee calculations.

In CCM, a modeler needs to specify the tracer model in every detail, fit it on the TTR-time data, and proceed only if the model can properly fit the data. Otherwise, the modeler needs to adjust the model's structure and repeat the process until a good fit is achieved. In contrast to the CCM, the HCM does not require the exact location of the conversion rate in the tracer model. Instead, it identifies the conversion rate in the Wb-mode and fits a two-mode tracer model on the TTR-time data. This reduces the required time and efforts for the tracer model construction and identification. Consequently, it generates alternative tracee models by considering all of the possibilities of the exact location of the conversion flux. For each one of those alternative models, HCM computes all of the tracee parameters by simple mathematical operations. Importantly, the modeler can choose the tracee alternative that makes more physiological sense.

As an example, consider a four-compartment, two-substrate model in Figure 4.2 for Phe and Tyr. The model in 4.2a is more physiologically sound than the one in 4.2b but the latter latter fitted our PHE-TYR TTR data whereas the former completely failed the fitting (see [1] for more details). This problem does not exist for the HCM approach, because it considers whole-body conversion, fits the data, and associates the conversion to specific pools by evaluating the tracee alternative. In other words, HCM takes a partial-step for determining the structure of the model, a full step for parameter identification, and then completes the structure at the end (i.e. it starts with a partially physiologically right model, fits the data, and picks the completely physiologically sound alternative at the end).

4.4 Experimental Results

The data was obtained in the Center for Translational Research in Aging and Longevity (CTRAL) at Texas A&M University. The study protocol and pulse method have previously been described in [92]. As a brief description, a peripheral line was placed in a vein of the lower arm and for blood sampling in a superficial dorsal vein of the contralateral hand, and the hand was placed in a thermostatically controlled hot box. After a venous blood sample was collected to measure baseline enrichment, an intravenous (IV) pulse, containing the stable tracers of the amino acids Phenylalanine (L-[ring-$^{13}C_6$]-Phenylalanine (Phe-6)) and Tyrosine (L-[ring-$^{2}H_4$]-Tyrosine (Tyr-4)), was admin-

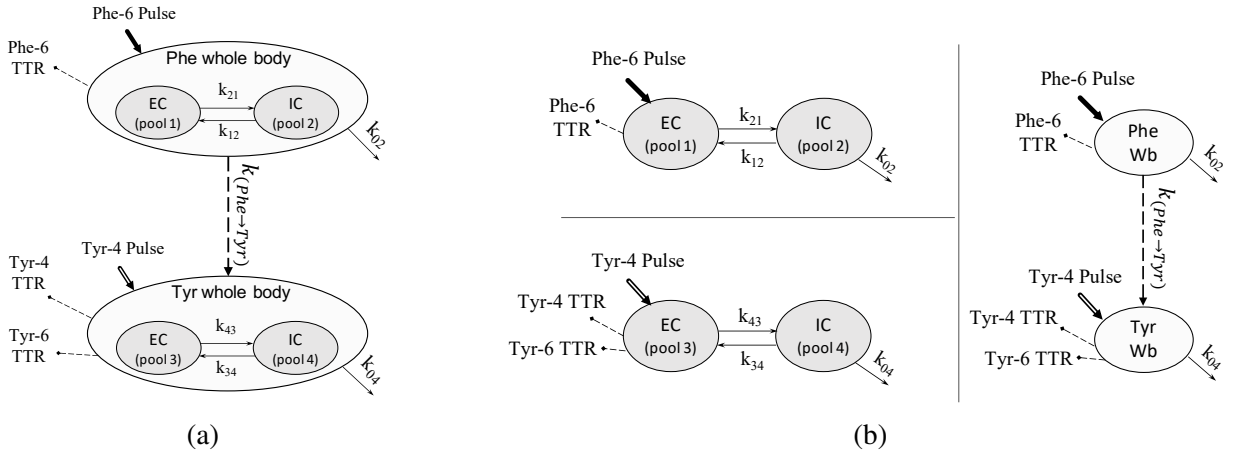


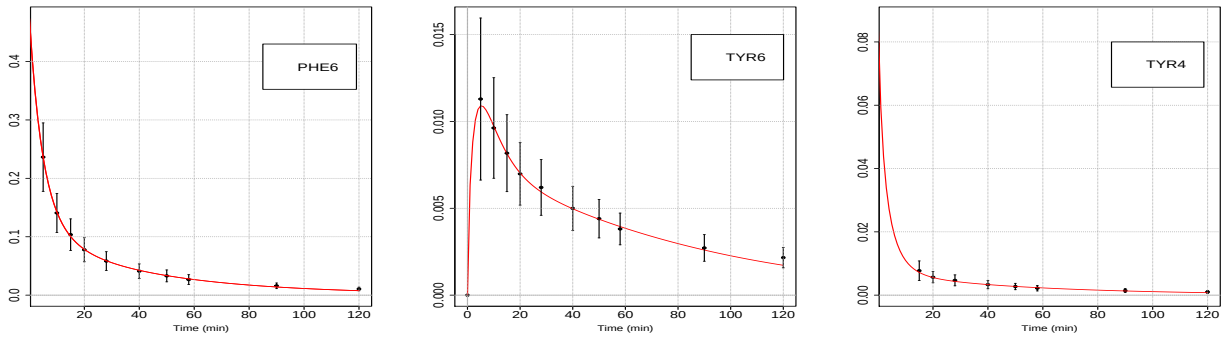
Figure 4.3: HCM tracer model for the Phe-Tyr four-compartmental model

istered. Enrichments of Phe-6, Tyr-4, and also Tyr-6 (metabolite product of $^{13}C_6$Phenylalanine) were measured in the plasma samples, in non-equally-spaced times. Measurements were started after 5 minutes and continued for two hours.

4.4.1 Tracer model

We used the HCM of Figure 4.3 to model Phe-Tyr metabolism. As pointed out by previous studies [93, 94, 1], tyrosine tracers may experience mixing issues. Hence, the early blood samples may extract some injected Tyr isotopes before they fully circulate in the system. We removed the first two samples (i.e samples taken at times 5 and 10 minutes) from Tyr-4 TTRs to avoid the overestimation of the tyrosine pool sizes. Yet, the initial points in the Tyr-6 TTR data set may carry important metabolic information as they are the metabolite product of Phe-6 isotopes, and therefore we did not remove points from the Tyr-6 data set.

The final tracer HCM of figure 4.3a was fitted on the three TTR data sets by simultaneously fitting the three sub-models in figure 4.3b. Figure 4.4 shows the final fits and their residuals. See table 4.1 for the calculated tracer fractional rates. All implementations were done in the open-source software R. We used R packages quadprog [95] for calculation of initial values, minpack.lm for non-linear regressions, and stats [96] for the rest of calculations.



(a) PHE6 (b) TYR6 (c) TYR4

Figure 4.4: HCM fits on the experimental TTR samples.

Variable	Value
k_{02}	0.026 ± 0.005
k_{04}	0.02 ± 0.001
k_{21}	0.131 ± 0.036
k_{12}	0.027 ± 0.007
$k_{phe \to tyr}$	0.033 ± 0.0082
k_{34}	0.023 ± 0.01
k_{43}	0.127 ± 0.013

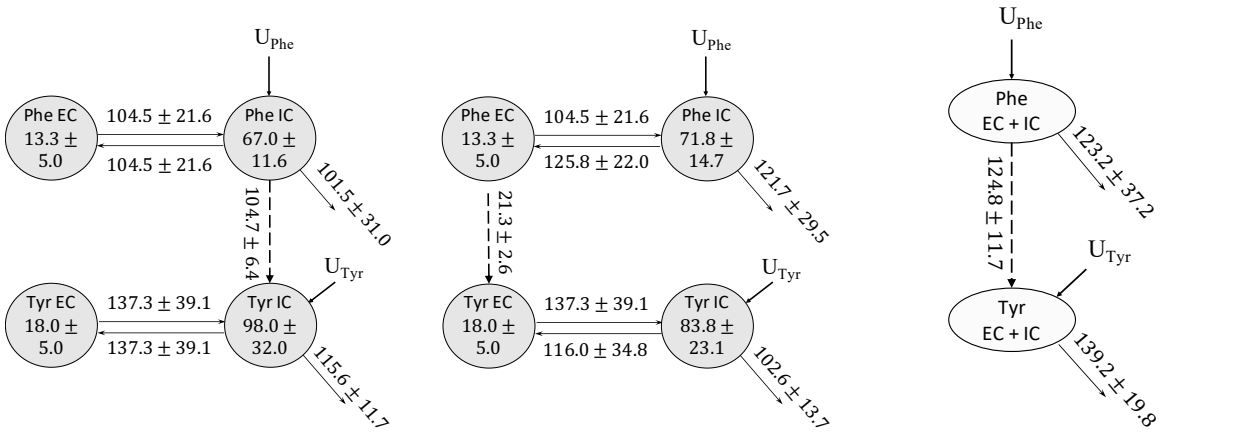
Table 4.1: $mean \pm SE$ for the tracer fractional rate parameters identified by HCM. The units are min^{-1}.

Note that prior to the HCM fitting, we first performed an outlier detection and initial value estimation for each of the three TTR data sets, i.e. Phe-6, Tyr-4, and Tyr-6, separately from the other two. Once outliers were removed and initial values were estimated, we fitted the HCM model on all three TTR data sets using a weighted least square regression approach. The weights were given to the residuals, where the residuals are defined as the difference between the model's prediction and the TTR data. For each TTR data set, the weight was fixed on the inverse of the TTR range for $t < 60min$ whereas the half of that value for $t > 60min$. For example, for Phe-4, we calculated $w = [\max(TTR_{Phe6}) - \min(TTR_{Phe6}]$ and set the residual weights to $\frac{1}{w}$ for $t < 60$ and $\frac{0.5}{w}$ for $t > 60$.

For the outlier detection, a sum of two exponential functions like equation 4.1 was fitted on the pulsed TTRs (Phe-6 and Tyr-4) whereas the Tyr-6 data set (which is the Phe-6 metabolite product) was fitted by a sum of three exponential functions (in the form of $TTR_{Tyr6} = -A_1 e^{\lambda_1 t} + A_2 e^{\lambda_2 t} + A_3 e^{\lambda_3 t}$). Finally, the points that were outside the 95% confidence interval of the estimated function were assumed to be outliers and removed. Note that the regressions in this step were only for the outlier detection and not for the final fit. Afterwards, the initial values were estimated by means of the curve peeling method.

4.4.2 Tracee model

The tracer HCM model was identified with the parameters in table 4.1. Consequently, three alternative tracee models (Figure 4.5) were created. The fluxes for each alternative model were calculated by tracee equations in the physiologically steady state, where the pool sizes stay constant and the sum of in-flow and out-flow fluxes is zero. The extracellular pool sizes and EC-to-IC fluxes are similar in all alternative models. More specifically, the extracellular pool sizes Q_1 for Phe and Q_3 for Tyr were calculated as the ratio of the isotopic tracer dose and the estimated TTR at $t = 0$, i.e. $dose_{tracer}/(A_1 + A_2)$. Also, the EC-to-IC Phe flux $F_{21} = k_{21}Q_1$ and the EC-to-IC Phe flux $F_{43} = k_{43}Q_3$ were similar in three tracee alternative models. The rest of fluxes are calculated differently for the alternative tracee models and are shown in figure 4.5. In 4.5c, the conversion flux was calculated as $(Q_{PheEC} + Q_{PheIC})k_{Phe \to Tyr}$. Units for pool sizes are $\mu mol.kgFFM^{-1}$,



(a) Alternative 1: Conversion occurs between IC pools. $U_{Phe} = 206.2 \pm 35.7$ and $U_{Tyr} = 10.9 \pm 5.1$.

(b) Alternative 2: Conversion occurs between EC pools. $U_{Phe} = 143.0 \pm 27.1$ and $U_{Tyr} = 81.3 \pm 30.6$.

(c) Alternative 3: Conversion occurs from Phe to Tyr in Whole body pools. $U_{Phe} = 248.0 \pm 41.3$ and $U_{Tyr} = 14.4 \pm 7.0$.

Figure 4.5: HCM tracee alternative models.

and for fluxes are $\mu mol.kgFFM^{-1}.h^{-1}$.

Alternative 1: The conversion is assumed to take place between intracellular pools (see Figure 4.5a). Since the pools are in steady state, $F_{12} = F_{21} = k_{21}Q_1$, and $Q_2 = F_{12}/k_{12}$. Similarly, $F_{34} = F_{43} = k_{43}Q_3$, and $Q_4 = F_{34}/k_{34}$. In addition, the protein breakdown to Phe is $U_{Phe} = F_{02} + F_{Phe\rightarrow Tyr}$, and to Tyr is $U_{Tyr} = F_{04} - F_{Phe\rightarrow Tyr}$.

Alternative 2: The conversion is assumed to take place between extracellular pools (see Figure 4.5b). Due to the Q_1 steady state, $F_{12} = F_{21} + F_{Phe\rightarrow Tyr}$. Subsequently, calculate $Q_2 = F_{12}/k_{12}$, $F_{02} = k_{02}Q_2$, and finally $U_{Phe} = F_{12} + F_{02} - F_{21}$. Similarly, due to the steady state of Q_3, $F_{34} = F43 - F_{Phe\rightarrow Tyr}$, $Q_4 = F_{34}/k_{34}$, $F_{04} = k_{04}Q_4$, and $U_{Tyr} = F_{34} + F_{04} - F_{43}$.

Alternative 3: The Wb pool size is assumed to be equal to the sum of EC and IC pool sizes. The EC-IC transfer fluxes are calculated similar to the alternative model 1, whereas the irreversible loss, conversion flux, and the protein break-down are calculated with the Wb pool sizes (see figure 4.5c). Therefore, the conversion flux is $F_{Phe\rightarrow Tyr} = k_{Phe\rightarrow Tyr}(Q_1 + Q_2)$, irreversible loss

for Phe is $F_{02} = k_{02}(Q_1 + Q_2)$, and for Tyr is $F_{04} = k_{04}(Q_3 + Q_4)$. The protein break-down to Phe is $U_{Phe} = F_{Phe \to Tyr} - F_{02}$, and to Tyr is $U_{Tyr} = F_{04} - F_{Phe \to Tyr}$.

4.4.3 CCM Equivalent

Although the present HCM modeling is different from the CCM approach in [1], the current HCM is more comparable to the sub-model B because we assumed no loss from the EC pool. In this subsection, some of the previously published CCM results on the same Phe and Tyr data are reprinted. See [1] for more details on the respective CCM model.

Figure 4.6a is unidentifiable, and therefore, is described by two uniquely identifiable models in 4.6b. Dashed lines denote sampling sites. TTR1, TTR2, and TTR3 are the tracer-to-tracee ratio samples for Phe-6, Tyr-4, and Tyr-6, respectively. Solid and open large arrows represent the pulsed isotopic tracer injection for Phe and Tyr, respectively. Compartments 1 and 3 are plasma and fast tissues whereas compartments 2 and 4 represent slow tissues. This model is not uniquely identifiable, and therefore, is described by interval identification strategy using the two uniquely identifiable models in 4.6b. Subscripts "max" and "min" refer to the lower and upper bounds for the intervals.

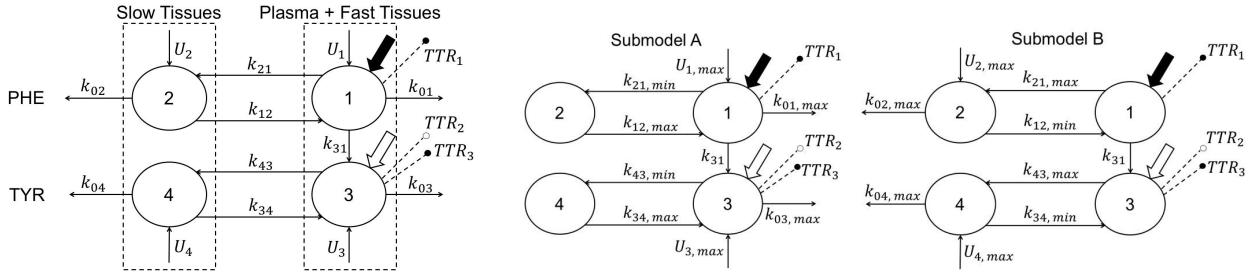
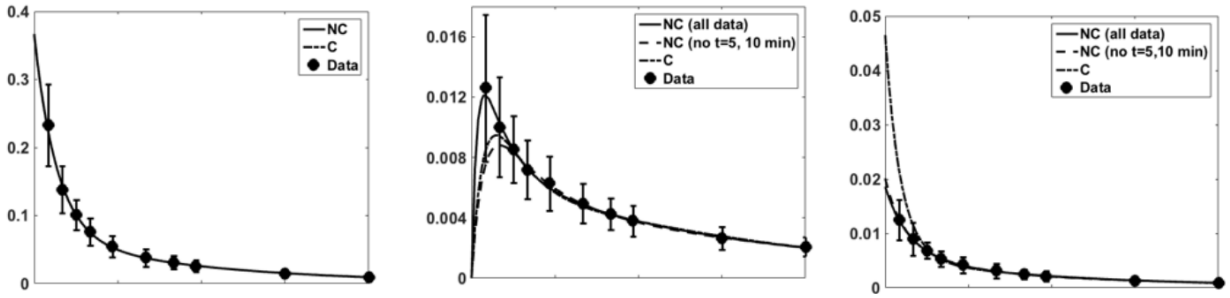


Figure 4.6: Compartmental model of [1].

A comparison of our results with the previously published literature suggest that the Tracee alternative 2 yields the most physiologically-sound values ($mean \pm SE$): $21.3 \pm 2.6 \mu mol/kg\ ffm/h$



(a) PHE6 (b) TYR6 (c) TYR4

Figure 4.7: Non-compartmental and compartmental fits in [1]. Solid dots are the TTR data points averaged on all subjects. The slashed line represents the multi-exponential fit for non-compartmental approach when samples at t=5 and 10 minutes of Tyr curves are not considered (NC (no t=5, 10 min)). Error bars represent SE.

for Phe to Tyr conversion flux, 143.0 ± 27.1 and 81.3 ± 30.6 for the protein breakdown into Phe and into Tyr, respectively. Previously published work focuses on non-compartmental analysis to compute the Wb rate of appearance (WbRa) and Phe to Tyr conversion flux ,a.k.a. the net protein breakdown (netPB) flux. Note that the non-compartmental approaches are likely to underestimate the fluxe [63, 1, 97]. Our alternative model 2 estimates all of the protein breakdown values slightly larger than those reported by the previously published non-compartmental approaches.

The Phe to Tyr conversion flux was calculated based on non-compartmental approaches applied to primed constant infusion studies as 5.2 ± 0.7 in 20 healthy individuals [98], 8.0 ± 0.8 in 10 young adults [92], and 7.1 ± 0.6 in 17 older adults [92]. In [99] higher estimates of 11.1 ± 5.8 for a fasted and 12.7 ± 7.7 for a fed state are reported. In [1], a non-compartmental analysis was performed (on the exact data that we used in this work), and the protein breakdown flux was reported as $5.7 \pm 1.5 \mu mol/kg\ ffm/h$.

The non-compartmental WbRa underestimates the protein breakdown because it does not consider the IC biochemical activities including the irreversible loss. The non-compartmental analysis of WbRa production ($mean \pm SE\ \mu mol/kg\ ffm/h$) is reported by [98] as 99.1 ± 5.1 for Phe and 65.6 ± 4.9 for Tyr. Furthermore,[92] reports WbRa 87.9 ± 2.3 and 53.8 ± 3.6 in young adults for Phe and Tyr, respectively. The same study also reports 93.5 ± 4.6 and 63.7 ± 4.8 for WbRa Phe and

Tyr, respectively, in old adults. The non-compartmental WbRa for both Phe and Tyr were reported in [1] as 50.4 ± 4.9 and 47.8 ± 9.0, respectively, which were a bit lower than those of previously mentioned publications.

Our HCM outperformed the respective CCM in terms of fitting the TTR data, particularly when comparing the Tyr-6 fits by the two modeling approaches (figure 4.4 vs. figure 4.7). In contrast to the CCM, the HCM tracer model reproduced well the early time points Tyr-6 data. This is an important feature of the model because experimental observations show that Tyr-6 is detectable in the blood samples only after it is produced as a result of the metabolic conversion of Phe-6, which suggest that the early time point samples may carry important metabolic information and should not be discarded.

When comparing the fractional rates in groups, one can bypass computation of fluxes, keep the conversion fractional rate and pool sizes separate, and replace the flux computations with a multivariate analysis. More specifically, instead of computing the fluxes using the steady-state formula ($F_{ij} = k_{ij}Q_j$) and comparing the groups' fluxes, one can use machine learning algorithms to distinguish the groups differences given the pool size and fractional rates. However, if the fluxes are the quantities of interest, the tracee alternative models should be generated.

5. SUMMARY

5.1 Classification Error Estimation

We introduced and investigated the family of generalized resubstitution error estimators. This is a broad family of classification error estimators who can all be computed in the same way by using different empirical measures. They do not require resampling and retraining of classifiers. As such, these are generally fast error estimators, which can be used in settings where computational complexity is an issue, such as in wrapper feature selection for large data sets. We showed empirically, by means of numerical experiments, that generalized resubstitution error estimators display excellent small-sample performance for traditional classification algorithms. Furthermore, generalized resubstitution estimators typically have hyper-parameters that can be tuned, which adds flexibility. In this dissertation, a data-driven calibration procedure was proposed for the bolstering hyper-parameter to further control the bias and variance. Furthermore, the application of generalized resubstitution error estimators was examined in deep fully connected neural network using a selection of datasets from the UCI database, where the spherical bolstered error estimator had the smallest deviation from the true error.

5.2 Predicting Generalization in Deep Learning

To extend the idea of bolstering to colored image data, three types of data augmentation approaches were proposed: Par-search, Multi-Augment, and Weighted Augment - posterior probability. The first approach searches for the best set of augmentation parameter to generate new datasets that help most with predicting the generalization of a trained model. Although this approach was successful in small datasets, it is not practical for average to large datasets due to its large time complexity. Alternatively, the Multi-Augment approach generates multiple datasets using various pre-defined sets of augmentation parameters and averages the outputs. This approach improved the time efficiency of the generalization prediction, compared to the Par-Search method. Yet, it did not result in a high CMI score in any of the tasks. The third attempt was to use weighted augmented

data, where weights are calculated based on the strength and amount of augmentation. This approach resulted in a relatively good CMI score, and more importantly did not overfit or underfit a specific dataset, as suggested by the small percentage differences between tasks.

5.3 Hybrid Compartmental Model

For the study of protein turn over in humans, we proposed hybrid compartmental model (HCM), which consists of one tracer model and several tracee alternative models. The tracer model structure is created prior to the TTR observations. Its parameters are identified by fitting the model on the TTR-time observations. Then, the tracee alternative models are generated and their computed parameters are analyzed according to physiological insights. This approach reduces the time and effort for finding a detailed conventional compartmental model that fits the TTR-time data points, and allows us to pick the most meaningful tracee model after all fractional rates and fluxes are identified.

We employed the HCM framework to evaluate the protein metabolism using a pulse of phenylalanine (Phe) and tyrosine (Tyr) stable isotopes. Our results show that the HCM tracer model outperformed the CCM in fitting the TTR data and estimated the Phe to Tyr conversion rate as $0.033 \pm 0.0082 min^{-1}$. The selected HCM tracee alternative model calculated the protein breakdown ($mean \pm SE\ \mu mol/kg\ ffm/h$) to Phe and Tyr as 143.0 ± 27.1 and 81.3 ± 30.6, respectively. It also yielded $21.3 \pm 2.6 \mu mol/kg ffm/h$ for the net whole-body protein breakdown. The proposed HCM could be particularly helpful in situations where the group metabolic difference is of interest. In that case, one could bypass the steady state flux equations, and instead, perform multivariate analysis or employ machine learning approaches that utilize the known pool sizes (Q_i) and the conversion fractional rates (k_{ij}).

REFERENCES

[1] A. Mason, M. P. Engelen, I. Ivanov, G. M. Toffolo, and N. E. Deutz, "A four-compartment compartmental model to assess net whole body protein breakdown using a pulse of phenylalanine and tyrosine stable isotopes in humans," *American Journal of Physiology-Endocrinology and Metabolism*, vol. 313, no. 1, pp. E63–E74, 2017.

[2] G. Toussaint, "Bibliography on estimation of misclassification," *IEEE Transactions on Information Theory*, vol. IT-20, no. 4, pp. 472–479, 1974.

[3] D. Hand, "Recent advances in error rate estimation," *Pattern Recognition Letters*, vol. 4, pp. 335–346, 1986.

[4] G. McLachlan, "Error rate estimation in discriminant analysis: recent advances," in *Advances in Multivariate Analysis* (A. Gupta, ed.), Dordrecht: D. Reidel, 1987.

[5] R. Schiavo and D. Hand, "Ten more years of error rate research," *International Statistical Review*, vol. 68, no. 3, pp. 295–310, 2000.

[6] U. Braga-Neto and E. Dougherty, *Error Estimation for Pattern Recognition*. New York: Wiley, 2015.

[7] Y. Jiang, B. Neyshabur, H. Mobahi, D. Krishnan, and S. Bengio, "Fantastic generalization measures and where to find them," *arXiv preprint arXiv:1912.02178*, 2019.

[8] P. Lachenbruch and M. Mickey, "Estimation of error rates in discriminant analysis," *Technometrics*, vol. 10, pp. 1–11, 1968.

[9] T. Cover, "Learning in pattern recognition," in *Methodologies of Pattern Recognition* (S. Watanabe, ed.), pp. 111–132, New York, NY: Academic Press, 1969.

[10] G. Toussaint and R. Donaldson, "Algorithms for recognizing contour-traced hand-printed characters," *IEEE Transactions on Computers*, vol. 19, pp. 541–546, 1970.

[11] M. Stone, "Cross-validatory choice and assessment of statistical predictions," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 36, pp. 111–147, 1974.

[12] B. Efron, "Bootstrap methods: Another look at the jacknife," *Annals of Statistics*, vol. 7, pp. 1–26, 1979.

[13] B. Efron, "Estimating the error rate of a prediction rule: Improvement on cross-validation," *Journal of the American Statistical Association*, vol. 78, no. 382, pp. 316–331, 1983.

[14] B. Efron and R. Tibshirani, "Improvements on cross-validation: The .632+ bootstrap method," *Journal of the American Statistical Association*, vol. 92, no. 438, pp. 548–560, 1997.

[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[16] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[17] M. R. Yousefi, J. Hua, and E. R. Dougherty, "Multiple-rule bias in the comparison of classification rules," *Bioinformatics*, vol. 27, no. 12, pp. 1675–1683, 2011.

[18] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, "Do imagenet classifiers generalize to imagenet?," *arXiv preprint arXiv:1902.10811*, 2019.

[19] C. Smith, "Some examples of discrimination," *Annals of Eugenics*, vol. 18, pp. 272–282, 1947.

[20] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *arXiv preprint arXiv:1609.04836*, 2016.

[21] P. Bartlett, S. Boucheron, and G. Lugosi, "Model selection and error estimation," *Machine Learning*, vol. 48, pp. 85–113, 2002.

[22] U. Braga-Neto and E. Dougherty, "Bolstered error estimation," *Pattern Recognition*, vol. 37, no. 6, pp. 1267–1281, 2004.

[23] C. Sima, U. Braga-Neto, and E. Dougherty, "Bolstered error estimation provides superior feature-set ranking for small samples," *Bioinformatics*, vol. 21, no. 7, pp. 1046–1054, 2005.

[24] C. Sima, S. Attoor, U. Braga-Neto, J. Lowey, E. Suh, and E. Dougherty, "Impact of error estimation on feature-selection algorithms," *Pattern Recognition*, vol. 38, no. 12, pp. 2472–2482, 2005.

[25] X. Zhou and K. Mao, "The ties problem resulting from counting-based error estimators and its impact on gene selection algorithms," *Bioinformatics*, vol. 22, pp. 2507–2515, 2006.

[26] Y. Xiao, J. Hua, and E. Dougherthy, "Quantification of the impact of feature selection on cross-validation error estimation precision," *EURASIP J. Bioinformatics and Systems Biology*, 2007.

[27] B. Hanczar, J. Hua, and E. Dougherty, "Decorrelation of the true and estimated classifier errors in high-dimensional settings," *EURASIP Journal on Bioinformatics and Systems Biology*, vol. 2007, 2007. Article ID 38473, 12 pages.

[28] M. Kaariainen and J. Langford, "A comparison of tight generalization bounds," in *Proceedings of the 22nd International Conference on Machine Learning*, Bonn, Germany, 2005.

[29] M. Kaariainen, "Generalization error bounds using unlabeled data," in *Proceedings of COLT'05*, 2005.

[30] Q. Xu, J. Hua, U. Braga-Neto, Z. Xiong, E. Suh, and E. Dougherty, "Confidence intervals for the true classification error conditioned on the estimated error," *Technology in Cancer Research and Treatment*, vol. 5, no. 6, pp. 579–590, 2006.

[31] A. Zollanvari, U. Braga-Neto, and E. Dougherty, "Analytic study of performance of error estimators for linear discriminant analysis," *IEEE Transactions on Signal Processing*, vol. 59, no. 9, pp. 1–18, 2011.

[32] A. Zollanvari, U. Braga-Neto, and E. Dougherty, "Exact representation of the second-order moments for resubstitution and leave-one-out error estimation for linear discriminant analysis in the univariate heteroskedastic gaussian model," *Pattern Recognition*, vol. 45, no. 2, pp. 908–917, 2012.

[33] L. Dalton and E. Dougherty, "Bayesian minimum mean-square error estimation for classification error – part I: Definition and the bayesian mmse error estimator for discrete classification," *IEEE Transactions on Signal Processing*, vol. 59, no. 1, pp. 115–129, 2011.

[34] L. Dalton and E. Dougherty, "Bayesian minimum mean-square error estimation for classification error – part II: Linear classification of gaussian models," *IEEE Transactions on Signal Processing*, vol. 59, no. 1, pp. 130–144, 2011.

[35] U. M. Braga-Neto and E. R. Dougherty, *Error estimation for pattern recognition*. John Wiley & Sons, 2015.

[36] G. McLachlan, *Discriminant Analysis and Statistical Pattern Recognition*. New York: Wiley, 1992.

[37] L. Devroye, L. Györfi, and G. Lugosi, *A probabilistic theory of pattern recognition*. Springer, 1996.

[38] U. Braga-Neto and E. Dougherty, "Is cross-validation valid for microarray classification?," *Bioinformatics*, vol. 20, no. 3, pp. 374–380, 2004.

[39] G. Lugosi and M. Pawlak, "On the posterior-probability estimate of the error rate of nonparametric classification rules," *IEEE Transactions on Information Theory*, vol. 40, no. 2, pp. 475–481, 1994.

[40] A. Hefny and A. F. Atiya, "A new monte carlo-based error rate estimator," in *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, pp. 37–47, Springer, 2010.

[41] Y. Jiang, P. Foret, S. Yak, D. M. Roy, H. Mobahi, G. K. Dziugaite, S. Bengio, S. Gunasekar, I. Guyon, and B. Neyshabur, "Neurips 2020 competition: Predicting generalization in deep learning," *arXiv preprint arXiv:2012.07976*, 2020.

[42] B. Neyshabur, Z. Li, S. Bhojanapalli, Y. LeCun, and N. Srebro, "Towards understanding the role of over-parametrization in generalization of neural networks," *arXiv preprint arXiv:1805.12076*, 2018.

[43] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning (still) requires rethinking generalization," *Communications of the ACM*, vol. 64, no. 3, pp. 107–115, 2021.

[44] C. A. Corneanu, S. Escalera, and A. M. Martinez, "Computing the testing error without a testing set," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2677–2685, 2020.

[45] Y. Jiang, P. Natekar, M. Sharma, S. K. Aithal, D. Kashyap, N. Subramanyam, C. Lassance, D. M. Roy, G. K. Dziugaite, S. Gunasekar, *et al.*, "Methods and analysis of the first competition in predicting generalization of deep learning," in *NeurIPS 2020 Competition and Demonstration Track*, pp. 170–190, PMLR, 2021.

[46] G. K. Dziugaite, A. Drouin, B. Neal, N. Rajkumar, E. Caballero, L. Wang, I. Mitliagkas, and D. M. Roy, "In search of robust measures of generalization," *Advances in Neural Information Processing Systems*, vol. 33, pp. 11723–11733, 2020.

[47] V. C. Madala, *A study of Generalization in Deep Neural Networks*. University of California, Santa Barbara, 2021.

[48] G. K. Dziugaite and D. M. Roy, "Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data," *arXiv preprint arXiv:1703.11008*, 2017.

[49] B. Neyshabur, S. Bhojanapalli, and N. Srebro, "A pac-bayesian approach to spectrally-normalized margin bounds for neural networks," *arXiv preprint arXiv:1707.09564*, 2017.

[50] P. L. Bartlett, D. J. Foster, and M. J. Telgarsky, "Spectrally-normalized margin bounds for neural networks," *Advances in neural information processing systems*, vol. 30, 2017.

[51] V. Nagarajan and J. Z. Kolter, "Deterministic pac-bayesian generalization bounds for deep networks via generalizing noise-resilience," *arXiv preprint arXiv:1905.13344*, 2019.

[52] Y. Jiang, D. Krishnan, H. Mobahi, and S. Bengio, "Predicting the generalization gap in deep networks with margin distributions," *arXiv preprint arXiv:1810.00113*, 2018.

[53] S. Aithal, D. Kashyap, and N. Subramanyam, "Robustness to augmentations as a generalization metric," *CoRR, abs/2101.06459*, 2021.

[54] Y. Zhang, A. Gupta, N. Saunshi, and S. Arora, "On predicting generalization using gans," *arXiv preprint arXiv:2111.14212*, 2021.

[55] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[56] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, D. Lopez-Paz, and Y. Bengio, "Manifold mixup: Better representations by interpolating hidden states," in *International Conference on Machine Learning*, pp. 6438–6447, PMLR, 2019.

[57] P. Natekar and M. Sharma, "Representation based complexity measures for predicting generalization in deep learning," *arXiv preprint arXiv:2012.02775*, 2020.

[58] Y. Schiff, B. Quanz, P. Das, and P.-Y. Chen, "Predicting deep neural network generalization with perturbation response curves," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[59] R. Rahaman *et al.*, "Uncertainty quantification and deep ensembles," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[60] J. Matis, T. Wehrly, and C. Metzler, "On some stochastic formulations and related statistical moments of pharmacokinetic models," *Journal of pharmacokinetics and biopharmaceutics*, vol. 11, no. 1, pp. 77–92, 1983.

[61] A. Rescigno, "Compartmental analysis and its manifold applications to pharmacokinetics," *The AAPS journal*, vol. 12, no. 1, pp. 61–72, 2010.

[62] M. Joly and J. M. Pinto, "A general framework for multi-compartmental analysis of drug chemotherapy dynamics in human immunodeficiency virus type-1 infected individuals," *Applied Mathematical Modelling*, vol. 36, no. 12, pp. 5830–5843, 2012.

[63] C. Cobelli, D. Foster, and G. Toffolo, *Tracer kinetics in biomedical research: from data to model*. Springer Science & Business Media, 2007.

[64] C. Cobelli, G. Toffolo, D. M. Bier, and R. Nosadini, "Models to interpret kinetic data in stable isotope tracer studies," *American Journal of Physiology - Endocrinology And Metabolism*, vol. 253, no. 5, pp. E551–E564, 1987.

[65] P. H. R. Barrett, B. M. Bell, C. Cobelli, H. Golde, A. Schumitzky, P. Vicini, and D. M. Foster, "Saam ii: simulation, analysis, and modeling software for tracer and pharmacokinetic studies," *Metabolism*, vol. 47, no. 4, pp. 484–492, 1998.

[66] M. Berman and M. F. Weiss, *Users Manual for SAAM (Simulation, Analysis and Modeling)*, vol. 78. Department of Health, Education, and Welfare, Public Health Service . . . , 1978.

[67] C. Cobelli and D. M. Foster, "Compartmental models: theory and practice using the saam ii software system," in *Mathematical modeling in experimental nutrition*, pp. 79–101, Springer, 1998.

[68] E. Steiner, T. Rey, and P. McCroskey, "Simusolv reference guide," *Dow chemical company*, 1990.

[69] W. Neely, G. Blau, and G. Agin, "The use of simusolv to analyze fish bioconcentration data," *Chemometrics and Intelligent Laboratory Systems*, vol. 1, no. 4, pp. 359–366, 1987.

[70] Y. Zhang, M. Huo, J. Zhou, and S. Xie, "Pksolver: An add-in program for pharmacokinetic and pharmacodynamic data analysis in microsoft excel," *Computer methods and programs in biomedicine*, vol. 99, no. 3, pp. 306–314, 2010.

[71] P. Ghane and U. Braga-Neto, "Generalized resubstitution for classification error estimation," *arXiv preprint arXiv:2110.12285*, 2021.

[72] V. Vapnik and A. Chervonenkis, "On the uniform convergence of relative frequencies of events to their probabilities," *Probability and its Applications*, vol. 16, pp. 264–280, 1971.

[73] U. M. Braga-Neto, *Fundamentals of Pattern Recognition and Machine Learning*. Springer, 2020.

[74] D. Pollard, *Convergence of Stochastic Processes*. New York: Springer, 1984.

[75] X. Jiang and U. Braga-Neto, "A naive-bayes approach to bolstered error estimation in high-dimensional spaces," 2014. Proceedings of the IEEE International Workshop on Genomic Signal Processing and Statistics (GENSIPS'2014), Atlanta, GA.

[76] T. Cover and P. Hart, "Nearest-neighbor pattern classification," *IEEE Trans. on Information Theory*, vol. 13, pp. 21–27, 1967.

[77] U. Braga-Neto and E. Dougherty, "Bolstered error estimation," *Pattern Recognition*, vol. 37, no. 6, pp. 1267–1281, 2004.

[78] C. Ambroise and G. McLachlan, "Selection bias in gene extraction on the basis of microarray gene expression data," *Proc. Natl. Acad. Sci.*, vol. 99, no. 10, pp. 6562–6566, 2002.

[79] S. Tabik, D. Peralta, A. Herrera-Poyatos, and F. Herrera, "A snapshot of image pre-processing for convolutional neural networks: case study of mnist," *International Journal of Computational Intelligence Systems*, vol. 10, no. 1, pp. 555–568, 2017.

[80] C. Sima, T. Vu, U. Braga-Neto, and E. Dougherty, "High-dimensional bolstered error estimation," *Bioinformatics*, vol. 27, no. 21, pp. 3056–3064, 2014.

[81] D. Dua and C. Graff, "UCI machine learning repository," 2017.

[82] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim, "Do we need hundreds of classifiers to solve real world classification problems?," *The journal of machine learning research*, vol. 15, no. 1, pp. 3133–3181, 2014.

[83] S. Arora, S. S. Du, Z. Li, R. Salakhutdinov, R. Wang, and D. Yu, "Harnessing the power of infinitely wide deep nets on small-data tasks," *arXiv preprint arXiv:1910.01663*, 2019.

[84] T. S. Verma and J. Pearl, "Equivalence and synthesis of causal models," in *Probabilistic and Causal Inference: The Works of Judea Pearl*, pp. 221–236, 2022.

[85] A. Krizhevsky, "Learning multiple layers of features from tiny images," tech. rep., 2009.

[86] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[87] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

[88] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar, "Cats and dogs," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[89] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *CoRR*, vol. abs/1708.07747, 2017.

[90] D. Kashyap, N. Subramanyam, *et al.*, "Robustness to augmentations as a generalization metric," *arXiv preprint arXiv:2101.06459*, 2021.

[91] C. Lassance, L. Béthune, M. Bontonou, M. Hamidouche, and V. Gripon, "Ranking deep learning generalization using label variation in latent geometry graphs," *arXiv preprint arXiv:2011.12737*, 2020.

[92] N. E. Deutz, J. J. Thaden, G. A. ten Have, D. K. Walker, and M. P. Engelen, "Metabolic phenotyping using kinetic measurements in young and older healthy adults," *Metabolism*, vol. 78, pp. 167–178, 2018.

[93] W. James, P. Garlick, P. Sender, and J. Waterlow, "Studies of amino acid and protein metabolism in normal man with l-[u-14c] tyrosine," *Clinical science and molecular medicine*, vol. 50, no. 6, pp. 525–532, 1976.

[94] L. L. Moldawer, I. Kawamura, B. R. Bistrian, and G. L. Blackburn, "The contribution of phenylalanine to tyrosine metabolism in vivo. studies in the post-absorptive and phenylalanine-loaded rat," *Biochemical Journal*, vol. 210, no. 3, pp. 811–817, 1983.

[95] B. A. Turlach and A. Weingessel, "quadprog: Functions to solve quadratic programming problems. r package version 1.5-5," 2013.

[96] R Core Team, *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013. ISBN 3-900051-07-0.

[97] G. A. Ten Have, M. P. Engelen, R. R. Wolfe, and N. E. Deutz, "Phenylalanine isotope pulse method to measure effect of sepsis on protein breakdown and membrane transport in the pig," *American Journal of Physiology-Endocrinology and Metabolism*, vol. 312, no. 6, pp. E519–E529, 2017.

[98] M. P. Engelen, R. Jonker, J. J. Thaden, G. A. Ten Have, M. S. Jeon, S. Dasarathy, and N. E. Deutz, "Comprehensive metabolic flux analysis to explain skeletal muscle weakness in copd," *Clinical Nutrition*, 2020.

[99] J. Marchini, L. Castillo, T. Chapman, J. Vogt, A. Ajami, and V. Young, "Phenylalanine conversion to tyrosine: comparative determination with l-[ring-2h5] phenylalanine and l-[1-13c] phenylalanine as tracers in man," *Metabolism*, vol. 42, no. 10, pp. 1316–1322, 1993.