Bayesian Inference for Supervised Machine Learning: Algorithms and Applications

Thesis submitted in accordance with the requirements of the
Univeristy of Liverpool for the degree of
Doctor of Philosophy
by

Paul Byrnes

August 2019

Le mo cuisle Laura agus mo thuismitheoirì ...

Declaration

I, Paul Byrnes, declare that this thesis titled *Bayesian Inference for Supervised Machine Learning: Algorithms and Applications*, and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always give. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where this thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I contributed myself.

August 2019

Acknowledgements

Firstly, I would like to thank my supervisor Dr. Alejandro Diaz De la O. Without his help and guidance none of this work would have been possible. His ability to explain complex topics in a concise manner has resulted in new lines of research while his constant encouragement has provided motivation throughout. For this I am extremely grateful.

To my colleagues at the Risk Institute, I would like to express my gratitude for the help and advice received during my studies. I also want to gratefully acknowledge the support of the Engineering and Physical Science Research Council (EPSRC) and the Economic Social Research Council (ESRC) for funding my PhD studies.

Many thanks go to Dr. Maria Gabrani of IBM Research Zürich who kindly allowed me to undertake placement as part of her research group. This experience has been fundamental for both the progress of this research and my personal development. I would also like to thank Dr. Roberto Perdisci for hosting me during my research stay at the University of Georgia.

To my parents Kathleen and Paddy, the support received from day one has allowed me to proposer and achieve beyond what I thought was possible. Thank you for everything! And to my siblings Gemma, Caroline and Patrick for providing valuable advice and more importantly quality entertainment.

Finally, the work undertaken over the past years would not have been possible without the love and support of my dearest Laura. Her ability to make me smile and laugh has made all the work worthwhile.

Abstract

Advances made in computer development along with the curiosity regarding the use of data in the world around us has resulted in machine learning becoming an area of much interest in recent decades. Its capabilities in automating processes such as face recognition at airport security or self-driving vehicles has highlighted the potential positive influence it could have on society. Behind many of these processes are statistical models which identify patterns in data sets to allow for a decision making process to be formed. However, such models require the computation of unknown parameters which directly impact their predictive capabilities. This dissertation explores the development and application of Bayesian inference frameworks suitable for parameter identification for supervised machine learning methods. A recent analogy has opened up the possibility of interpreting Bayesian inference as rare event simulation. Bayesian Updating with Structural reliability methods (BUS), exploits the low acceptance rate in rejection based sampling, allowing for techniques from reliability analysis to solve the Bayesian updating task. A key principle for the BUS framework in terms of sample quality and sampler efficiency is the question of the termination of simulation. Currently, this is done through the use of a computationally expensive automatic stopping condition. To improve computational efficiency, two new stopping criteria are introduced. Aside from this reduction in cost, the proposed approaches not only simplify the implementation of the framework for the practitioner in terms of coding and theoretical understanding but also offer statistical guarantees of sampling from the correct distribution. With the emergence of large data sets has come the need for scalable algorithms which offer efficient solutions. To improve the suitability of BUS to such tasks, Support Vector Machines (SVM) are integrated into the BUS approach to allow for a reduction in total model evaluations in the presence of a large number of data observations. Additionally, the capabilities of the methods developed during this dissertation are illustrated on two real life breast cancer classification tasks. The first concerning the identification of cancerous tissue in

biopsy samples and the second the identification of relapse rates from patient molecular data. Aside from the suitability of the Bayesian inference frameworks to such problems, the potential of supervised machine learning in improving the diagnosis process for cancer patients is also discussed.

Table of contents

List of figures

List of tables

List of Acronyms

$2SMART$	Subset simulation by Support vector Margin Algorithm for Reliability esTimation
$aBUS$	Adaptive BUS
AIS	Annealed Importance Sampling
aCS	Adaptive Conditional Sampling
AUC	Area Under the Curve
BUS	Bayesian Updating using Structural reliability methods
BUS_{SVM}	BUS with Support Vector Machines
BUS_{+}	BUS with Post Processor
$c.o.v$	Coefficient of Variation
CDF	Cumulative Density function
DMC	Direct Monte Carlo
$DDMH$	Delayed Modified Metropolis Hastings
FN	False Negative
FP	False Positive
HMC	Hybrid Monte Carlo
LA	Laplace Approximation
LHS	Latin Hypercube Sampling
$nBUS$	Nested BUS
$nBUS_D$	Nested BUS Direct
MAP	Maximum A Posteriori
$MCMC$	Markov Chain Monte Carlo

MH	Metropolis Hastings
MMH	Modified Metropolis Hastings
MLE	Maximum Likelihood Estimation
PDF	Probability Density Function
ROC	Receiver Operating Characteristic
SuS	Subset Simulation
$SuS - Inf$ . .	Subset Simulation Infinity
SVM	Support Vector Machine
$TMCMC$ . . .	Transitional Markov Chain Monte Carlo
TME	Total Model Evaluation
TN	True Negative
TP	True Positive
VI	Variational Inference
WSI	Whole Slide Image

Chapter 1

Introduction

1.1 Motivation

The generation of massive amounts of data combined with advances in computational power has resulted in accelerated research around machine's ability to learn. One of the early milestones of research in computerising a learning process was a study using the game of checkers [193], where it was hypothesised that a computer could be programmed to learn how to play a better game of checkers than the person who wrote the program. While the potential of devising a learning scheme which can outperform a humans actions was highlighted, given the era (1950's) of this work, the learning framework was not yet feasible in economic terms to be applied to real life problems. Modern society has evolved into an environment enriched with automation from personal assistance applications on smart phones recognizing voice commands to streaming services learning to suggest films [27, 147]. The field of pattern recognition is concerned with the automatic discovery of regularities in data [34]. With these regularities being used to take actions such as classifying the data into different categories. Behind many of these processes are numerical models that identify data patterns to allow for a decision making process to be formed. On a basic level, the model learns a function which maps an input to a desired output. With the accuracy of the functional approximation of the relationship between the model's input and output determined by the degree of correctness of the decision taken.

Built upon assumptions compensating for limited knowledge and imprecise understanding of the world around us, the model is uncertain. A considered model could be a finite element analysis [137] applied within a computational

fluid dynamics model [224] or a model for simulating biochemical pathways for biochemical processes in systems biology [91]. In practice, regardless of the models accuracy or complexity, different sources of uncertainty appear during its formation. As only a finite data set has been observed, limited information about the data generation process' are available. Given the models assumptions, approximation error contributes to uncertainty between the true underlying process and the model. Establishing a model with outputs consistent with the real world outcomes requires unknown inputs referred to as *parameters* to be estimated. The inability to describe the inherent possible variability of these parameters introduces additional uncertainty. For a further discussion on different sources of uncertainty the reader is referred to [121].

Evidently, a numerical model is susceptible to uncertainties while explicitly accounting for these significantly increases the required computational budget. *Uncertainty quantification* is the general term for a group of methods that are used to analyse and make inferences about the output of a numerical model. Uncertainty quantification in numerical modelling may be categorised into two different types of problems [205]. *Forward uncertainty quantification* concerns the task of propagating the uncertainty in the model input through the model to predict the overall uncertainty in the model's output. The second problem is referred to as *inverse uncertainty quantification*. Solving an inverse problem involves making inferences about a physical system from data generated by that system [197]. Consider the task of classifying whether or not a patient partaking in a clinical trial has received the trial drug or a placebo replacement. In this setting, an example of an inverse problem would be the identification of the classification model's parameters based on observed clinical data. Once estimated, the parameter values may be used for forecasting drug allocation on future patients. The work presented during this dissertation primarily focuses on the latter type of uncertainty problem and in particular model parameter identification.

Bayesian inference [81, 83, 143] allows for unknown model parameters to be identified and updated based on some observed data. Even though Bayes' theorem [24] stems from the axioms of probability [126], the use of Bayes requires the specification of a prior distribution along with a particular view of uncertainty aswel as the observed data. As such, a prior distribution is specified representing current beliefs about the unknown parameters. Through the use of Bayes' theorem, these prior beliefs are updated with this updated knowledge being encapsulated in

a probability distribution referred to as the *posterior*. The posterior is a conditional probability distribution that allows for prior knowledge and information contained in observed data to be combined in order to infer the values of the unknown parameters. Due to the mathematical expression of the posterior, it is often the case that an analytical solution is unattainable and therefore cannot be computed. Aside from inferring parameter values, the expression of the posterior by Bayes' theorem allows for competing models to be compared against one another through a quantity referred to as the *model evidence*. The model evidence may be viewed as the plausibility of a model given in the presence of the observed data. Evidently, the posterior is a powerful probability distribution that in most cases is required to be computed using numerical methods. Such methods for Bayesian inference ideally not only estimate unknown model parameters but also compute the model evidence as a by product of simulation. Methods should also be suitable to a wide range of problems in terms of different data properties (e.g. dimensionality) and complexity of the posterior distribution (e.g. multi-modal distributions). While statistical guarantees of convergence to the correct posterior distribution with a reasonable computational cost for a given problem should also be given.

A proportion of numerical approaches for Bayesian inference view the task of computing the posterior distribution as an optimization problem. The idea is to approximate the potentially complex posterior distribution using the best distribution from a family of distributions (e.g. Gaussian). In this case the best distribution may refer to the distribution which minimizes the Kullback–Leibler (KL) divergence [35] with the posterior. As an approximation to the posterior is identified, these methods produce approximate samples. While often lauded for their computational efficiency, their inability to efficiently deal with multi-modal distributions (e.g. Laplace approximation [215]), the lack of guarantee of convergence (e.g. expectation propagation [151]) and the sensitivity of the sample quality to initial distribution choice (e.g. variational inference [35]) highlight some of the outstanding issues with these approaches.

An alternative to the above is to produce draws from the posterior. Rejection sampling offers a simplistic solution for generating samples directly from a probability distribution but quickly becomes inefficient in the presence of a large number of uncertain parameters. Rendering it unsuitable for many practical problems. Prior to drawing samples from the posterior, rejection sampling requires a prudent choice of an input parameter. This parameter directly influences the distribution

of the samples while also being responsible for the samplers efficiency. Choosing this parameter however has in general remained an open question [67].

Markov Chain Monte Carlo (MCMC) [79, 86] approaches consist of a widely implemented group of methods that produce draws from the posterior distribution. Initializing a Markov chain with the posterior as its stationary distribution allows for posterior samples to be generated as a sequence of the chain. The Metropolis Hastings (MH) [105] algorithm may be viewed as a pillar of MCMC [81]. In its standard form however, many obstacles may appear for the MH sampler. The first being the requirement of a sample *burn in* phase to ensure that the stationary distribution of the Markov chain is indeed the posterior. While there exists diagnostics for determining the length of the burn in period [79], in general it is a non-trivial task. Another issue with MH, is the inability to efficiently cope with problems with a large number of uncertain parameters. This is in contrast to Gibbs sampling [84] which explores the posterior in a component wise manner. However, due to its sampling updating process, the Gibbs sampler can become inefficient if the parameters of the posterior are highly correlated. Some specialized MCMC algorithms [106, 70] can cope with such high dimensions, however they require the posterior gradient to be computed for every generated sample. The choice of proposal distribution in MH directly impacts the samplers ability to populate parameter spaces that contain multiple regions of high probability. A number of frameworks have been proposed [85, 54, 124] to populate multi-modal distributions but with them have come new challenges. For example, the careful selection of additional parameters which influence algorithmic efficiency [150, 3]. As with MH, the majority of these samplers are primarily focused on sampling from the posterior distribution and are unable to offer estimates of the model evidence. This has lead to the development of MCMC based samplers with the estimation of the model evidence being the primary quantity of interest [204, 95, 22].

Recent work in [208] has addressed each of the above issues of MH in a single sampling algorithm. Through the interpretation of Bayesian inference as rare event simulation, Bayesian Updating with Structural reliability methods (BUS) enable frameworks from reliability engineering to solve the Bayesian inference problem. An advantage of BUS is that rare event simulation techniques already exist in the reliability literature and can be readily applied to Bayesian inference problems. One such technique, Subset Simulation (SuS) [11] is an advanced MCMC algorithm which is suitable to high dimensional problems and avoids the

requirement of a sample burn in period through its nested architecture. Aside from addressing sample burn in and dimensionality issues, BUS with SuS can efficiently populate multiple modes of complex distributions by using multiple Markov chains for exploration of the parameter space. Additionally, in contrast to many of the above referenced MCMC methods, BUS also offers an estimate of the model evidence. In its original form BUS, like rejection sampling requires choosing a parameter prior to simulation that dictates the distribution of the generated samples. If incorrectly chosen, a new choice of parameter must be made and the sampler is re-run. As expected, in the presence of a computationally expensive model this is not feasible. This parameter not only directly influences the distribution of samples but also the estimation of the model evidence. Like any algorithm, correct termination of simulation is crucial as it ensures the quantity of interest has been accurately computed. In terms of a stopping condition, posterior samples are conditional on a quantity assumed to be deterministic at termination but in truth is stochastic. This results in the samples not being truly conditional on the event that ensures they have been generated form the posterior distribution.

A reformulation named nested BUS ($nBUS$) [67], has been proposed which automatically learns the input parameter required under rejection sampling during simulation. This avoids any potentially unnecessary additional sampler runs while allowing for the model evidence to be computed. Given BUS is MCMC based, the samples are not independent. To protect against a potentially deteriorating sample quality and to ensure the samples are truly distributed according to the posterior, an automatic stopping condition is implemented. This takes the form of a nested loop which gives name to the reformulated framework. The stopping condition ensures that the correct probability distribution is being sampled from while also offering statistical assurances of any potential sampling errors. Performing correct stopping however comes at a cost in terms of computational efficiency. The nested loop requires the execution of a computationally intensive function with a numerical cost being dependent on the size of the data set. Rendering $nBUS$ unsuitable to problems with large amounts of data.

1.2 Aims and Objectives

The aim of this dissertation is the development of efficient Bayesian inference methods for computing unknown parameters for statistical models used in the

area of supervised machine learning. Specifically, to maintain algorithmic stability and accuracy while reducing computational expense in the presence of large data sets. To satisfy this the following objectives are addressed in this dissertation.

1. The identification of the suitability of existing reliability inspired Bayesian inference frameworks to supervised machine learning problems. An analysis in terms of parameter and model evidence estimation is also included.

2. The development of alternative stopping conditions which reduce computational cost and improve sample quality of $nBUS$. Where statistical guarantees of sampling from the correct probability distribution are offered.

3. The development of methods which address the scalability issues of $nBUS$ for parameter identification in problems requiring large data sets. A supervised machine learning algorithm is used as an approximation to the underlying computational expensive function. Reducing the computational expense will improve the suitability of the developed methods to a wider range of problems.

4. The implementation of the developed methods to problems concerning breast cancer detection. Two different data types are addressed to establish the capabilities of the frameworks on real world problems in terms of parameter estimation, model evidence estimation and predictive prowess. The potential of supervised machine learning as a tool for increasing diagnosis efficiency and quantifying uncertainty between practitioner's recommended diagnosis is also addressed.

1.3 Thesis Outline

The structure of this dissertation is as follows. In Chapter 2, a brief overview of machine learning is provided with a particular emphasis placed on three classification methods used during this thesis. The impact of accurate parameter estimation on predictive capabilities is highlighted. Bayesian inference is introduced in Chapter 3, where numerous existing methods for estimating unknown model parameters are reviewed. A reliability analysis based approach named BUS which is suitable for high dimensional and multi-modal problems is discussed in Chapter 4. To reduce the computational cost of the framework while maintaining sampling accuracy,

two new stopping conditions are presented in Chapter 5. The first which takes advantage of aspects of rejection sampling and the second involving properties of the model evidence. In Chapter 6, Support Vector Machines (SVM) are integrated into the *BUS* approach to ensure scalability to problems involving a large number of data observations. Through the use of SVM, the number of required calls to a computationally expensive function is reduced during a model run. To analyse the suitability of each of the developed frameworks to problems involving different data types, two real life problems from breast cancer detection are investigated in Chapter 7. The first concerning the identification of cancerous tissue in biopsy samples and the second the identification of relapse rates from patient molecular data. Different aspects of the potential of supervised machine learning for disease identification are also considered. Finally, Chapter 8 provides conclusions and outlines a number of future directions of research.

Chapter 2

Machine Learning

Advances made in computer development along with the curiosity regarding the use of data in the world around us has resulted in machine learning becoming an area of much interest in recent decades. Its capabilities in automating processes such as face recognition at airport security or self driving vehicles has highlighted the potential positive influence it could have on society. Behind many of these processes are statistical models which identify patterns in data sets to allow for a decision making process to be formed. This chapter broadly introduces machine learning with a particular emphasis placed on supervised learning approaches. Additionally, three classification models implemented during this thesis are discussed in detail.

2.1 Introduction

The possibility of programming computers to learn and enabling them to improve their decision making over time has highlighted their potential in becoming ubiquitous in every day life and a vital component in industry. Developing a successful understanding of how to best allow computers to learn would allow increased efficiency in many everyday tasks. Though the extent to which their learning compares to that of humans has not been identified, the potential is clear. Learning is required in cases where a computer program to solve a given problem cannot be directly written but instead requires examples of past experiences or data [2, 200]. The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience [153]. Originally stemming from the area of computer science, machine learning offers a

group of algorithms which analyse data in order to make decisions. In some cases, the use of an algorithm in isolation however, is not enough.

Consider an example of the problem of disease identification in tissue samples. The desired algorithmic output is the indication of whether or not the sample is contaminated with traces of the disease. We are unable to transform the input (tissue sample) to the desired diagnosis in pure algorithmic terms as the disease identification markers vary from specimen to specimen. With the introduction of data containing instances of both contaminated and uncontaminated tissue samples, what we lack in knowledge can be compensated for in terms of learning what constitutes the presence of the disease.

The introduction of data allows machine learning algorithms to make informed decisions based on the information available. Unknowingly, we are surrounded by examples of machine learning based technology. Personal assistance applications on smart-phones learn to recognize voice commands [147], allowing hands-free usage of a mobile phone in the event of an emergency. Streaming services learn to suggest films [27] to users based on their film choice history. In airport security, automatic passport control learns to detect faces [194]. This reduces potential passenger congestion whilst also increasing the capabilities of the airport security forces. In terms of autonomous vehicles, machine learning has enabled them to recognize objects [131]. This is paramount to human safety in the event that autonomous vehicles become a permanent fixture in society.

While in each application, there has been relative success, the effectiveness of a machine learning algorithm commonly depends on the quality of data used [155]. In truth, data driven models are only as good as the data which they were built with. As such, before the implementation of an algorithm, some data pre-processing steps may be required [182, 62]. Machine learning can typically be expressed in terms of three categories: unsupervised learning, supervised learning and reinforcement learning.

Unsupervised learning, also referred to as knowledge discovery [157], applies when the data consists of input instances only. The goal is to uncover interesting patterns or trends in the data. Given the practitioner is unaware of what pattern to look for along with no error metric being available, it is somewhat of a non-trivial task. Consider the case when the data set consists of tissue samples taken from a group of patients. This group consists of both healthy and infected subjects. However, no information regarding which specimens are infected is provided. For

this binary case, unsupervised learning groups the respective specimens into two categories based on their similarities and dissimilarities. This particular approach is commonly referred to as *clustering* [4, 119, 225] .

In contrast, *supervised learning* applies when both the algorithmic input and output are readily available. Consider the case when the data set consists of tissue samples taken from a group of patients. This group consists of both healthy and infected subjects. Each specimen is labelled as being either healthy or infected. In this case, a mapping is learned from the input (specimen) to the output (diagnosis) given a labelled dataset. As the output is readily available, supervised algorithms offer a diagnosis based on the observed tissue samples. This concept is discussed in more detail in Section 2.2.

A branch of machine learning which exists at the intersection of both unsupervised and supervised learning is referred to as *semi-supervised learning* [232]. Semi-supervised learning concerns both labelled and unlabelled instances and may be best used in problems where labelled data is expensive to obtain. In this case expensive may refer to the monetary costs or time costs associated with attaining class labels for data. An example would be identifying Edwards syndrome [213] in unborn children. This involves removing and testing cells from amniotic fluid by the insertion of a needle through the abdominal wall, with a major risk being the possibility of miscarriage. In this scenario, theoretically semi-supervised learning could use past cases to identify whether an unborn child has Edwards syndrome without requiring the testing procedure. In this setting, the expense being avoided is the potential danger to the unborn child.

The final category of machine learning is *reinforcement learning* [209]. This is concerned with machines ability to automatically determine the ideal behaviour for a specific task in order to maximize its performance. In this case of Edwards syndrome, a reinforcement learning system could be introduced in the form of surgical robots which would assist the practitioner during the procedure. The remainder of this dissertation is primarily concerned with supervised learning. For further information on unsupervised, semi-supervised or reinforcement learning, the reader is referred to the aforementioned references. Figure 2.1 offers a broad overview of the three categories along with their respective sub categories. The schematic breakdown of each category into subcategories can be much more detailed, but for this dissertation the highlighted subcategories are of great importance.

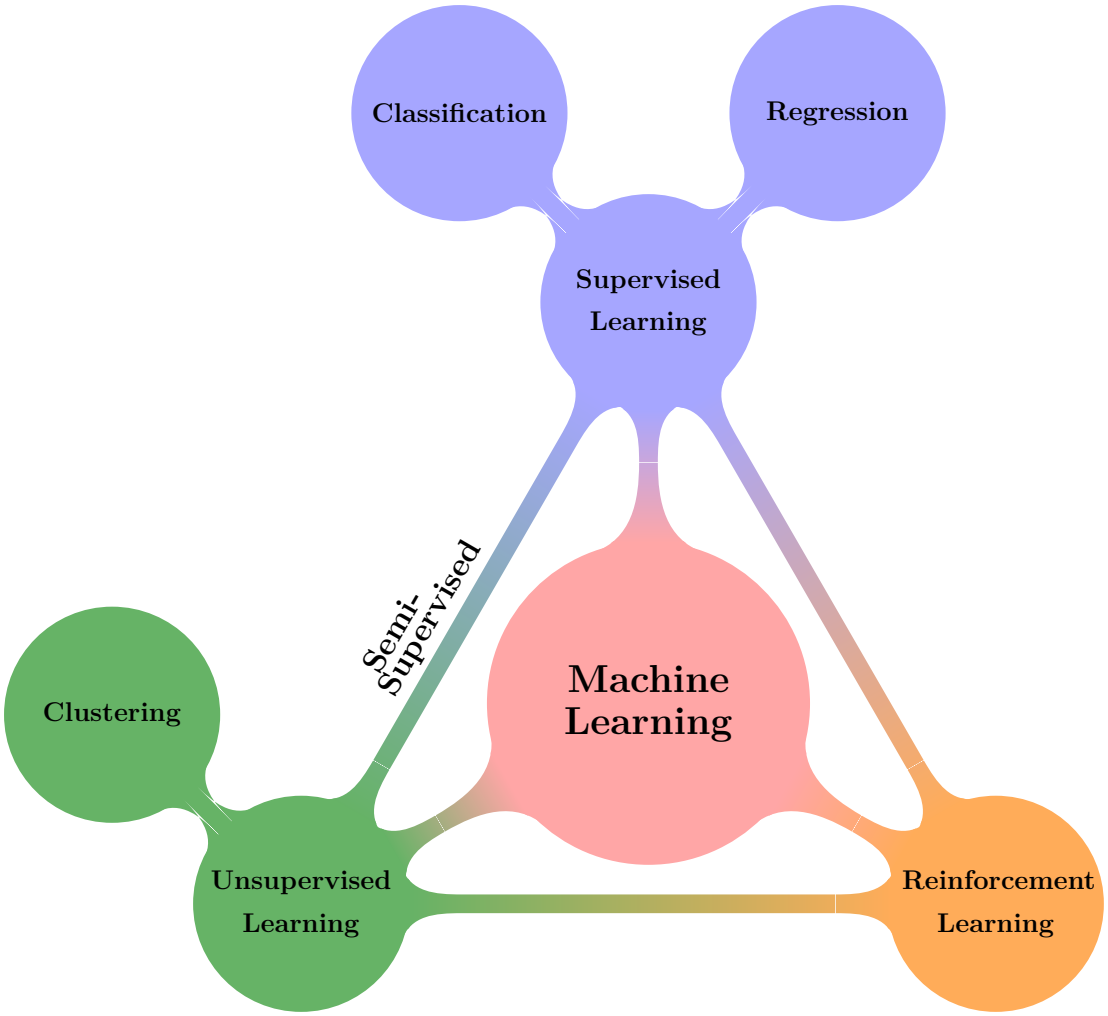


Figure 2.1: A schematic overview of the three main categories of machine learning along with their subcategories.

2.2 Supervised Learning

This section presents an initial introduction to supervised learning and its two sub-areas in regression and classification. Given this dissertation is concerned with the development of Bayesian inference methods for the estimation of unknown parameters of classification models, three classification methods implemented in later chapters are discussed in detail. This is followed by a brief overview of other existing approaches.

Let $\boldsymbol{X} \in \mathbb{R}^{n \times d}$ be a data matrix with n observations and d features. A feature is an individual descriptive characteristic of a phenomenon being observed. In

the case of determining the presence of a tumour in a tissue sample, the feature set could contain the size and colour of the specimen. Let $\boldsymbol{Y} \in \mathbb{R}^{n\times 1}$ be a vector representing the target output and $D = \{(x_i, y_i)\}_{i=1}^{n}$ the corresponding dataset. Supervised learning is concerned with modelling a functional relationship by inferring a mapping from the data input to the output. Initially, D is partitioned into two subsets for training and testing. To avoid model over-fitting in the sense of being biased towards the training set, numerous techniques exist ensuring a generalized model performance can be fairly evaluated. Widely applied methods include the hold-out method [123] and k-fold cross validation [211]. The former simply entails dividing the data set into a training and testing set. With the percentage of observations in each set specified by the practitioner. Commonly used ratios are 80% for training and 20% for testing [123]. The cross validation approach randomly splits the data set into k subsets and trains the model on the $k-1$ folds before testing on the remaining fold. This process is repeated k times with the average error computed using the individual error on each held out fold.

The model parameters are estimated during the training phase for the target functional mapping to be learned. Additionally, competing models varying in parameter choice may be compared using the training data in terms of some specified criterion. Once trained, the corresponding model is implemented on the previously unseen test data for predictive purposes. In turn, allowing for the quality of the mapping to be verified. As was outlined in Figure 2.1, supervised learning models may be further expressed in terms of two subcategories, namely regression and classification. While both utilize information from training data to learn an approximated function, the nature of the output differs.

2.2.1 Regression

Regression analysis is concerned with learning a real valued function. The learning task is to approximate $y = f(x)$, where y in this case is a continuous quantity. The target function f is assumed to be unknown and is estimated using the observed training data. In reality, many real data sets contain individual observations which are corrupted with random noise. This noise may stem from a stochastic process or due to sources of variability, which themselves are unobserved [34]. As such, the learning task may be defined as

$$y = f(x) + \epsilon \tag{2.1}$$

The noise term ϵ, is introduced to represent the deviation between the true function f and approximated function $\hat{f}$. This ϵ noise term for example could be modelled as a zero mean Gaussian random variable. Consider a data instance x_i with the respective observational output response given by

$$y_i = \theta_0 + \theta_1 x_{i,1} + \cdots + \theta_d x_{i,d} + \epsilon_i = x_i^T \boldsymbol{\theta} + \epsilon_i \tag{2.2}$$

with $\boldsymbol{\theta}$ denoting a vector of unknown model parameters. This is a linear regression [199] which assumes that the output is linear in the parameters $\boldsymbol{\theta}$. To construct an approximation of the target function as in Eq. 2.2, appropriate values of $\boldsymbol{\theta}$ are learned using data observations. An in depth discussion of how such parameters may be computed is presented in Chapter 3. Intuitively, $\boldsymbol{\theta}$ should be estimated to best fit the training data or to ensure a chosen error criterion is minimized.

Consider a one dimensional example for which a simple model $y = \sin(x)$ as in Figure 2.2 (a) is defined with the input assumed to be uniformly distributed on $[0, 4\pi]$. If $y(x)$ is considered to be computationally expensive to evaluate, one is more likely to be presented with a finite collection of data observations such as those in Figure 2.2 (b) rather than a continuous curve. It is clear that a linear regression would not be suitable for this particular set of data observations (black circles). A polynomial regression requires the order m of the polynomial to be specified such that

$$y_i = \theta_0 + \theta_1 x_{i,1} + \cdots + \theta_d x_{i,d}^m + \epsilon_i = x_i^{mT} \boldsymbol{\theta} + \epsilon_i \tag{2.3}$$

Note that even though y_i is a non-linear function of x_i, it is a linear function of the parameter vector $\boldsymbol{\theta}$. The influence of selecting m is evident from Figure 2.2 (c), where polynomials with $m = 1$, $m = 3$ and $m = 6$ approximate the output of $\sin(x)$. In Figure 2.2 (d), the true function (blue) is approximated by the output of the 6^{th} order polynomial regression model (red) fitted using the *error sum of squares* (SSE).

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2.4}$$

In practical terms, a regression problem could be the prediction of a patients age based on their blood pressure and cardiac cycle timing. A cardiac cycle refers to the time taken for the heart to complete a relaxation and contraction of both

the atria and ventricles. Here, f corresponds to the doctor's knowledge and ϵ for example whether or not they have made a mistake. For a comprehensive overview of existing regression models and their applications, the reader is referred to [199, 50, 68, 104, 18].

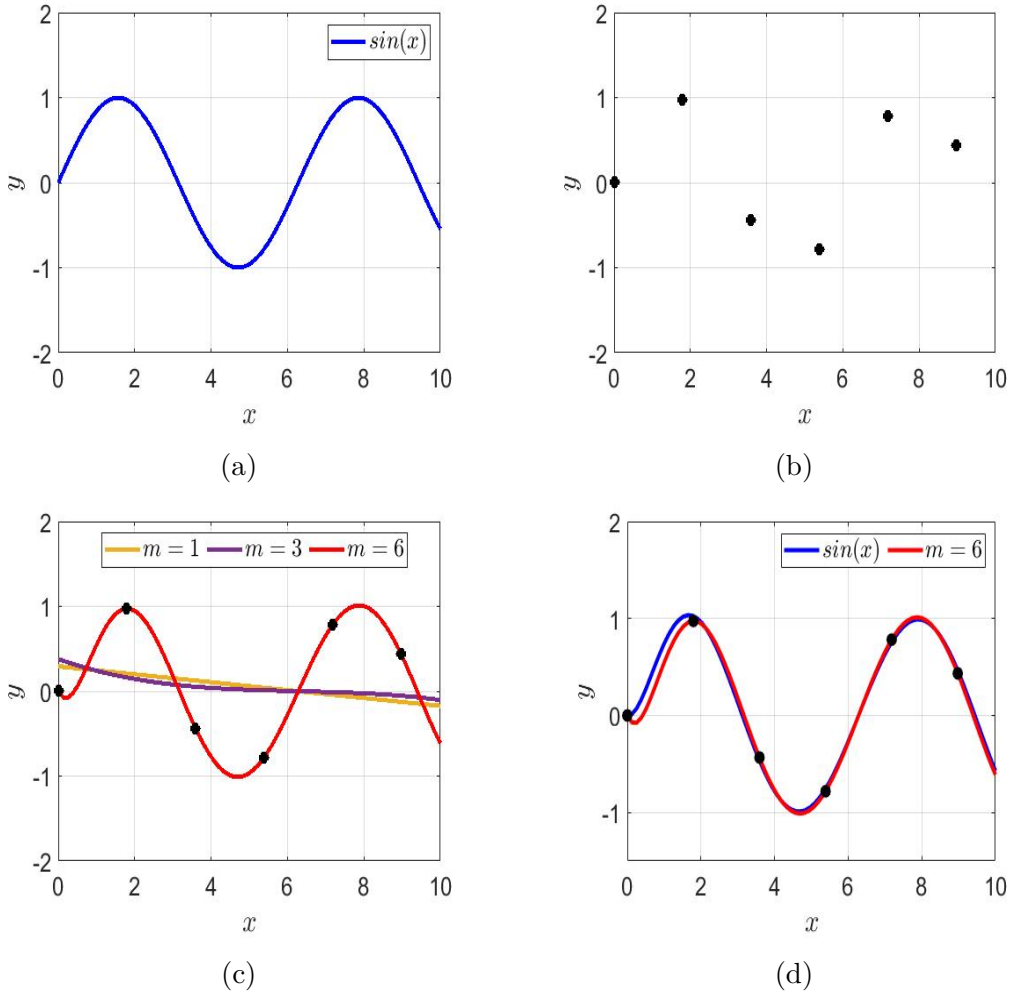


Figure 2.2: Panel (a) presents the underlying function with the available data observations given in panel (b). Three different orders of polynomial with $m = 1$, $m = 3$ and $m = 6$ are shown in panel (c). Panel (d) illustrates the comparison of $m = 6$ and the underlying function.

2.2.2 Classification

In line with regression analysis, classification aims to infer a function mapping based on training data. However, in this case the desired output is discrete. In classical terms, classification is a pattern recognition problem which assigns each

$\{x_i\}_{i=1}^n$ to one of $y \in \{1, ..., c\}$ classes, where c denotes the number of classes. The data set $D = \{(x_i, y_i)\}_{i=1}^n$ now consists of observations and class labels. During the training phase, the classifier learns a decision rule from the observations and labels. The subsequent process extracts information from this data to establish characteristics of each class membership. Allowing for future test data to be allocated a class label based on this information. The case of $c = 2$, is referred to as *binary classification* in which $y \in \{0, 1\}$ while $c > 2$ is referred to as *multi-class classification*. For the purpose of this dissertation we are concerned purely with problems which are mutually exclusive in the output i.e. each input can only be assigned one class label. The scenario of the classes not being mutually exclusive is referred to as *multi-label classification* [157].

Given the suitability of supervised learning to tasks from different domains, a number of classification approaches varying in their theoretical underpinnings have been developed. The remainder of this section provides a detailed overview of three methods used during this thesis as classification models. The discussed methods are Logistic Regression, Gaussian Processes (GP) and Support Vector Machines (SVM). Additionally, a brief overview of alternative approaches is also provided.

Logistic Regression

In Section 2.2.1, the concept of linear regression aimed at predicting continuous quantities was discussed. Logistic regression [107, 148] offers a method for modelling the conditional distribution of the output variable when the response is discrete. A relationship is formed between the observed data and the subsequent model through the introduction of a conditional likelihood function.

$$L(D|\boldsymbol{\theta}) = \prod_{i=1}^{n} p_i^{y_i}(1 - p_i)^{1-y_i} \tag{2.5}$$

which assumes independence between all data observations. The relationship between $\boldsymbol{\theta}$ and the data is represented as a linear combination with the probability of class membership denoted by

$$p_i = p(y_i = 1|x_i, \boldsymbol{\theta}) = \frac{1}{1 + e^{-x_i^T \boldsymbol{\theta}}} \tag{2.6}$$

The likelihood function is a crucial ingredient of statistics and its role is discussed in detail in Chapter 3. Formally, Eq. 2.6 is commonly referred to as the *logit* function and is widely applied in different areas of machine learning for mapping the output of regression functions onto the unit interval [183, 176, 207]. As is evident from Eq. 2.5, the logistic regression model is a binary classifier. Along with its simplicity, an advantage of this approach over competing models is its ability to quantify its degree of certainty of class membership through the use of probabilities. The purpose of the training set is for the best possible approximated function to be learned to allow for accurate future predictions to be confidently made. A crucial aspect of this is the estimation of the unknown model parameters. The unknown parameter vector $\boldsymbol{\theta}$, is learned from the training data. In essence, the resultant parameters help guide the model to make decisions once new previously unseen data or test data becomes available.

Typically the method of *maximum likelihood estimation* (MLE), which may viewed as the engine of classical statistics, has been implemented to estimate the subsequent parameter values [43, 61, 57, 160]. MLE seeks the value of the parameters for which, given the observed data, the probability of having observed that data is maximized. Finding the values of $\boldsymbol{\theta}$ for maximizing the likelihood, corresponds to establishing the critical points of a function when the first derivative equals 0. In the case of the second derivative evaluated at a given point being less than 0, the maximum point has been identified. Thus, finding the maximum likelihood estimate requires computing the first and second derivatives of the likelihood function. Direct differentiation of Eq. 2.5 can be a difficult task due to the multiplicative terms. Fortunately, a log transformation allows for this task to be simplified. Given that the logarithm is a monotonic function, any maximum of the likelihood function will also be a maximum of the log likelihood function and vice versa. Taking the log of Eq. 2.5 yields

$$\begin{aligned} \log(L(D|\boldsymbol{\theta})) &= \sum_{i=1}^{n} y_i \log(p_i) + (1 - y_i)\log(1 - p_i) \\ &= \sum_{i=1}^{n} [-\log(1 + e^{x_i^T \boldsymbol{\theta}}) + y_i \cdot x_i^T \boldsymbol{\theta}] \end{aligned} \tag{2.7}$$

The first derivative of the log-likelihood with respect to $\boldsymbol{\theta}$ is given by

$$\begin{aligned}\frac{\partial \log(L(D|\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}} &= \frac{\partial}{\partial \boldsymbol{\theta}} \sum_{i=1}^{n} [-\log(1 + e^{x_i^T \boldsymbol{\theta}}) + y_i \cdot x_i^T \boldsymbol{\theta}] \\ &= \sum_{i=1}^{n} [y_i - p_i] x_i \end{aligned} \tag{2.8}$$

In terms of the second derivative with respect to $\boldsymbol{\theta}$ and Eq. 2.8, the Hessian matrix is given by

$$\frac{\partial^2 \log(L(D|\boldsymbol{\theta}))}{\partial^2 \boldsymbol{\theta}} = -\sum_{i=1}^{n} x_i^T x_i p_i [1 - p_i] \tag{2.9}$$

Full derivations are included in Appendix A. Solving the maximization problem requires approximate numerical techniques as a direct numerical solution is not feasible. A number of frameworks which efficiently solve the optimization problem have been implemented [116]. MLE seeks to find

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \quad L(\boldsymbol{\theta}, D) \tag{2.10}$$

Certain limitations in the MLE estimate are prevalent. Firstly, the maximum likelihood estimator may not exist or may not be unique [136, 234]. The likelihood function may contain several local maxima, whereby a maximum found by a numerical method may not be the global maximum. This means that additional steps are required to verify if this is the case. In terms of uncertainty quantification, identifying a point estimate results in no additional information regarding the choice of parameter being available. As such, indications of the quality of the estimate, nor how well one can make predictions based on the estimate are not provided.

To illustrate the importance of accurate parameter estimation, Figure 2.3 presents a toy example. The problem concerns a synthetic binary response data set generated from two different Gaussian distributions namely $\mathcal{N}(-2, 0.9)$ and $\mathcal{N}(3, 1.5)$. The data set consists of 80 observations with each class being equally represented in the training data. Class 0 (blue) is the target predictive class. Three scenarios are simulated whereby model parameters are firstly poorly estimated (Figure 2.3 (a)), then the case of parameters which help identify a suitable function approximation for the classifier (Figure 2.3 (b)) through a point estimate and finally the case of generating a probability distribution under the Bayesian

paradigm for the estimated parameters (Figure 2.3 (c)). Bayesian inference which is introduced in Chapter 3, allows for the quantification of the uncertainty in the parameter estimates through the extraction of information from observed data. From Figure 2.3 (a), poorly estimated parameters have resulted in a classifier of random guessing with the model assigning a probability of 0.5 across the entire domain. By using point estimation an improvement can be seen in Figure 2.3 (b). A functional mapping has been successfully learned which identifies areas of the domain that signify potential class membership. Close to class 0, the model has correctly assigned a probability of 1 while the same can be said for class 1 with a probability of 0. However, the point estimate has resulted in straight lines as the predictive contours. These result in the model incorrectly assigning a high probability to class 0 for example in regions far away from the class observations. The advantage of generating a probability distribution over the unknowns is clear from Figure 2.3 (c). Similar to the point estimates, the model has correctly identified the target class with high probability. More importantly, the regions between classes and further away from the observations are now assigned probabilities closer to 0.5. The classifier stemming from the point estimates may be interpreted as being over confident while the introduction of a probability distribution has allowed for uncertain regions of the domain to be assigned appropriate values. Efficient frameworks for incorporating parameter uncertainty for classification tasks are presented in Chapter 5 and 6. For further information on the influence of parameter estimation on classifier quality please refer to [34, 151, 44, 165, 132].

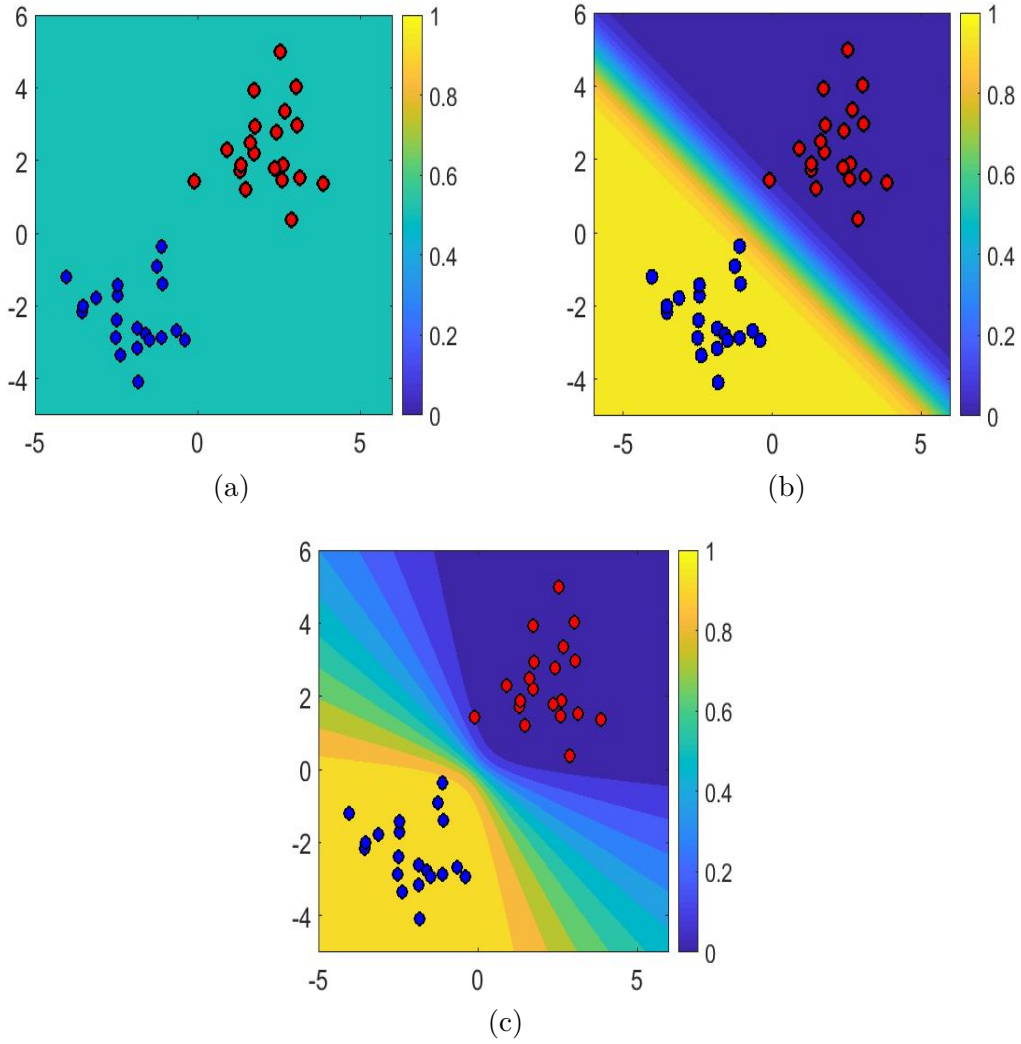


Figure 2.3: Influence of parameter identification on predictive performance. The predictive abilities of the logistic regression model differ greatly in the case of poorly estimated parameters (a), point estimates (b) and the incorporation of parameter uncertainty (c). The advantage of quantifying the uncertainty in the estimated values can be seen in panel (c) where appropriate probability values have been assigned to regions far away from the data observations.

Gaussian Processes

Unlike logistic regression, Gaussian processes are applicable to problems where the target variable is continuous or discrete. The focus of this section is solely on the discrete case and in particular binary classification. The Gaussian process is a

stochastic process such that

$$f(\cdot) \sim GP(m(\cdot), k(\cdot, \cdot)) \tag{2.11}$$

where $f(\cdot)$ is a function drawn from the multivariate Gaussian distribution with mean $m(\cdot)$ and covariance $k(\cdot, \cdot)$. This means that, the Gaussian process is totally summarised by its mean and covariance functions. The reference to stochastic process in this case refers to the set of random variables being indexed by the set of all possible inputs as opposed to time. A Gaussian process may be viewed as a collection of random variables, any finite number of which have a joint Gaussian distribution [183]. The covariance function encodes the assumptions about the function which the practitioner seeks to learn. It characterises the dependencies between function values for any pair of inputs. A multitude of possible covariance functions exist with the majority requiring a prudent choice of hyperparameters denoted by the vector $\boldsymbol{\phi}$ [183]. A common choice of covariance function is the squared exponential [165, 34, 183]:

$$k(x, x^{'}) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}||x - x^{'}||^2 \right) \tag{2.12}$$

Here σ_f denotes the signal variance and l the length scale parameter of the covariance. The length scale is a measure of how smooth f is, with small values representing a function that changes quickly and large values a function that changes slowly. The signal variance represents the variation of f with respect to its mean with small values characterizing a function which stays close to its mean.

In a regression setting, the use of Gaussian likelihood function ensures the computations of predictions is straightforward as the relevant integrals are Gaussian and as such are computed analytically. For classification however, non-Gaussian likelihoods present the issue of having to approximate or sample from the relevant conditional probability distribution referred to as the posterior. A detailed overview of how to draw realizations from this distribution is offered in the following chapter. Given f, the class labels y are independent Bernoulli variables allowing for the joint likelihood function to be factorized as:

$$L(y|f) = \prod_{i=1}^{n} p(y_i|f_i) \tag{2.13}$$

This likelihood function depends on f only through its value at the corresponding observed inputs. For example, by using the cumulative distribution function (CDF) of the Gaussian distribution $\Phi(\cdot)$ the individual likelihood terms become $p(y_i|f_i) = \Phi(y_i f_i)$. Often in a binary setting since neither of the class labels is more probable, the mean of the prior over f is set to zero [165]. The combination of the zero mean function and the covariance function defines the Gaussian process. Using Bayes' rule, the posterior distribution over f for a given vector of hyperparameters $\boldsymbol{\phi}$ becomes

$$P(f|D, \boldsymbol{\phi}) = \frac{\mathcal{N}(f|0, k)}{P(D|\boldsymbol{\phi})} \prod_{i=1}^{n} \Phi(y_i f_i) \tag{2.14}$$

which is non-Gaussian. In order to predict the class label y^* for the unseen test inputs x^*, the distribution of the latent function value can be marginalised such that:

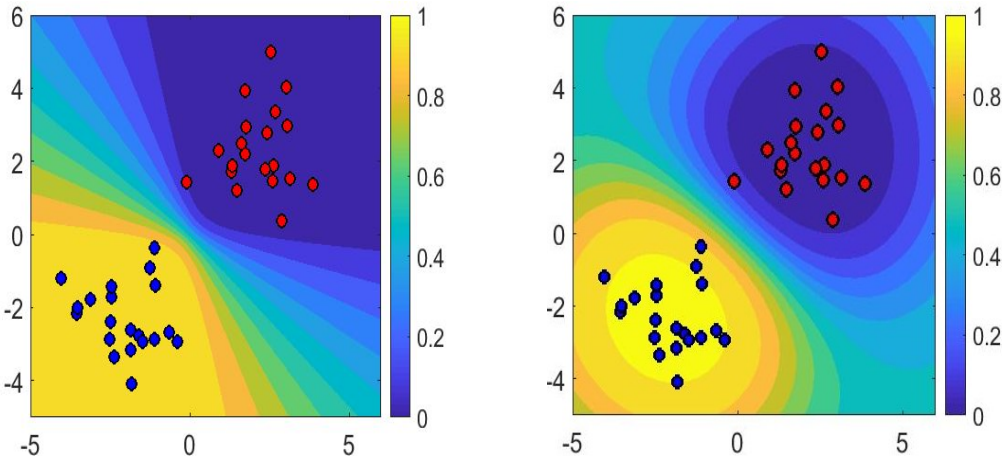
$$P(f^*|D, \boldsymbol{\phi}, x^*) = \int P(f^*|f, D, \boldsymbol{\phi}, x^*) P(f|D, \boldsymbol{\phi}) df \tag{2.15}$$

where f^* denotes a test value of the latent function. To obtain the predictive distribution the expectation of the above may be taken resulting in

$$P(y^*|D, \boldsymbol{\phi}, x^*) = \int P(y^*|f^*) P(f^*|D, \boldsymbol{\phi}, x^*) df^* \tag{2.16}$$

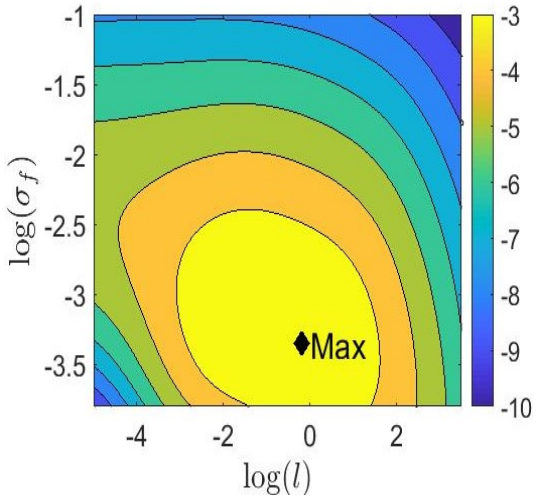
Unfortunately, neither the denominator in Eq. 2.14 or the predictive distribution of Eq. 2.16 may be computed analytically. Unlike logistic regression, Gaussian processes are non-parametric methods that offer great flexibility in terms of capturing non-linear decision boundaries between classes. To illustrate these properties, consider again the binary classification example concerning data observations drawn from two Gaussian distributions previously discussed for the logistic regression classifier. Figure 2.4 provides a visual comparison between the logistic regression (a) and Gaussian process classifier (b) decision boundaries. The ability of the Gaussian process classifier in producing a more appropriate class decision boundary is evident. As with logistic regression, the Gaussian process assigns a probability of approximately 0.5 in regions where class membership is uncertain. In terms of hyperparameter selection for the covariance function, a grid search was implemented whereby the negative log likelihood was maximized for each combination of the squared exponential hyperparameters considered. Figure

2.4 (c) illustrates the distribution of the log likelihood with the hyperparameters producing the maximum negative log likelihood identified. The resulting hyperparameter values were used for the classification task in Figure 2.4 (b).



(a) Logistic Regression Predictions.

(b) Gaussian Process Predictions.



(c) Gaussian Process Log Likelihood.

Figure 2.4: The predictive capabilities of the logistic regression model (a) and a Gaussian process classifier (b) differ greatly in terms of the treatment of the decision boundary between classes. The advantage of the Gaussian process is evident from its ability to capture a highly non-linear decision boundary through the use of the chosen covariance function. Panel (c) illustrates the distribution of the log likelihood with the hyperparameter combination producing the maximum negative log likelihood identified.

The non-parametric nature of the Gaussian process results in potential disadvantages in terms of computational expense in the presence of a large number of observations when computing the covariance function. The computation involves inverting a matrix as large as the number of training points and as the number of these grow, the problem becomes more and more expensive. Sparse extensions of the Gaussian process have been developed to allow for this issue to be solved, however they are not addressed in this study [183, 135]. For more information on the Gaussian process classification model the reader is referred to [132, 165].

Support Vector Machines

Similar to Gaussian processes, Support Vector Machines (SVM) [58, 219] are a non-parametric method but differ in that they are deterministic in their output. Assume that the labelled classes are perfectly separated by a $(d-1)$ dimensional hyperplane. SVM focuses on the task of identifying the optimal separating hyperplane between the two classes. Consider the decision function to be a separating hyperplane

$$f(x) = \langle \boldsymbol{\theta}, x \rangle - a = 0 \tag{2.17}$$

where both $\boldsymbol{\theta}$ and bias term $a \in \mathbb{R}$ are unknown parameters of the SVM model and $\langle \cdot, \cdot \rangle$ denotes the vector dot product. SVM proposes to identify the largest distance between the two observation classes. In essence, the goal is to maximize this distance to allow for a greater geometric separation between class observations. Consider the vector of unknown $\boldsymbol{\theta}$ to be perpendicular to Eq. 2.17. Conceptually the dot product enables the projection of x onto $\boldsymbol{\theta}$, allowing for the identification of which side of the hyperplane the observation is on i.e. the identification of class membership. Many possible values of $\boldsymbol{\theta}$ exist, therefore constraints are required to ensure the correct choice. The model takes the form of an optimization problem the aim of which is to maximize the distance in the margin subject to the constraints

$$\langle \boldsymbol{\theta}, x \rangle - a \quad \geq 1, \qquad \text{if} \quad y_i = +1 \tag{2.18}$$

$$\langle \boldsymbol{\theta}, x \rangle - a \quad \leq -1, \qquad \text{if} \quad y_i = -1 \tag{2.19}$$

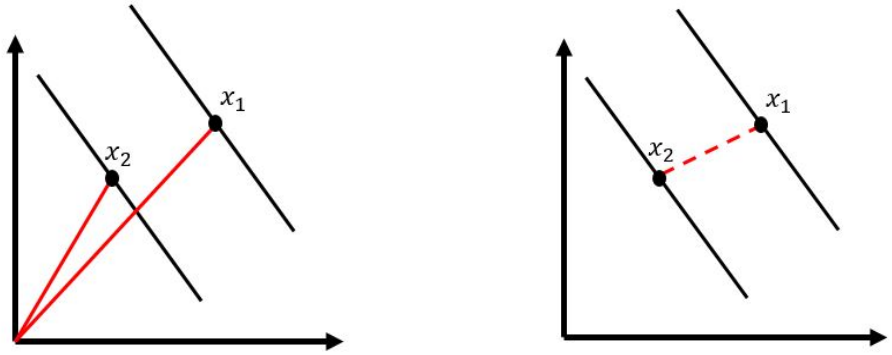
which can be compactly put in the form

$$y_i(\langle \boldsymbol{\theta}, x_i \rangle - a) \geq 1 \tag{2.20}$$

The solution to this classification problem is uniquely defined by selecting the SVM classifier which is furthest from the training points. A rescaling of the problem is applied such that the closest points to the separating hyperplane satisfy the equation

$$y_i(\langle \boldsymbol{\theta}, x_i \rangle - a) - 1 = 0 \tag{2.21}$$

Allow for the largest distance between the two observation classes to be referred to as the *margin*. That is, any x satisfying Eq. 2.21 is located on the margin and is referred to as a *support vector*. To formulate an expression for the desired width of the margin, consider two training points x_1 and x_2 as illustrated in Figure 2.5. Where x_1 is located on the support vector of the positive class and x_2 the negative.



(a) (b)

Figure 2.5: Two training points located on the support vectors of the positive class (x_1) and the negative class (x_2) as given in the left panel. Taking the dot product between the difference of x_1 and x_2 and a normalised vector allows for an expression for the width between the two support vectors to be derived. This distance is referred to as the margin.

By normalising $\boldsymbol{\theta}$ and considering its perpendicular properties to the hyperplane, the margin width in terms of x_1 and x_2 is denoted by

$$w = \langle (x_1 - x_2), \frac{\boldsymbol{\theta}}{||\boldsymbol{\theta}||} \rangle \tag{2.22}$$

Which is simply the dot product between the two vectors and the normalised parameter vector. Using Eq. 2.21 and substituting in the expressions for x_1 and x_2 result in the margin width being given by

$$w = \frac{\boldsymbol{\theta} x_1 - \boldsymbol{\theta} x_2}{||\boldsymbol{\theta}||} = \frac{1 + a - a + 1}{||\boldsymbol{\theta}||} = \frac{2}{||\boldsymbol{\theta}||} \tag{2.23}$$

The optimal linear classifier is obtained by maximizing the margin under the constraint of correct classification of the training observations. Reformulating and defining the margin in quadratic programming form produces the following optimization task.

$$\min_{\boldsymbol{\theta},a} \quad \frac{||\boldsymbol{\theta}||^2}{2} \quad \text{s.t} \quad y_i(\langle \boldsymbol{\theta}, x_i \rangle - a) \geq 1 \quad \text{for} \quad i = 1, ..., n \tag{2.24}$$

A convenient approach to solve for n linear constraints is the use of Lagrange multipliers

$$L(\boldsymbol{\theta}, a, \alpha) = \frac{||\boldsymbol{\theta}||^2}{2} - \sum_{i=1}^{n} \alpha_i [y_i(\langle \boldsymbol{\theta}, x_i \rangle - a) - 1] \tag{2.25}$$

where $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_n)^T$ contains the n Lagrange multipliers with $\alpha_i \geq 0$ for $i = 1, ..., n$. Hence, the problem is to minimize the loss function with respect to the hyperplane parameters while maximizing it with respect to the Lagrange multipliers. The saddle point solution to Eq. 2.25 implies obtaining the partial derivatives of both unknown parameters resulting in

$$\frac{\partial L(\boldsymbol{\theta}, a, \alpha)}{\partial a} = \sum_{i=1}^{n} \alpha_i y_i \tag{2.26}$$

$$\frac{\partial L(\boldsymbol{\theta}, a, \alpha)}{\partial \boldsymbol{\theta}} = \boldsymbol{\theta} - \sum_{i=1}^{n} \alpha_i y_i x_i \tag{2.27}$$

The partial derivative with respect to $\boldsymbol{\theta}$ is defined as a linear sum of the training observations whose α are strictly positive. Such points are referred to as the support vectors and may be viewed as the points which are most difficult to classify. This allows for the unknown parameters of the hyperplane to be given by

$$\boldsymbol{\theta} = \sum_{j=1}^{S} \alpha_j y_j x_j \tag{2.28}$$

where S is the number of support vectors. Intuitively, this is indicating that the equation of the optimal hyperplane can be defined in terms of training observations which are close to it. Essentially the decision function is defined purely in terms of its support vectors. This property makes SVM a viable model for problems with limited amount of data. Utilizing this functional form and substituting into Eq. 2.25 allows for the optimization problem in terms of the Lagrange multipliers

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \tag{2.29}$$

The final classification function may be expressed as

$$y(x) = sgn(f(x)) = sgn\left(\sum_{i=1}^{n} \alpha_i y_i \langle x_i, x \rangle - a \right) \tag{2.30}$$

The main concepts of SVM and derivation details have stemmed from the assumption that the classes are linearly separable. By using kernel transformations SVM can easily be extended to non-linear scenarios. Based on the so-called *kernel trick*, the input data is mapped into some other possibility infinite dimensional space sometimes called a feature space [201]. A non-linear projector ϕ, acts as the feature map with $k(\cdot)$ denoting the kernel. Under this transformation, the classification problem only changes in the relationship between the training observations. With the updated expression given by

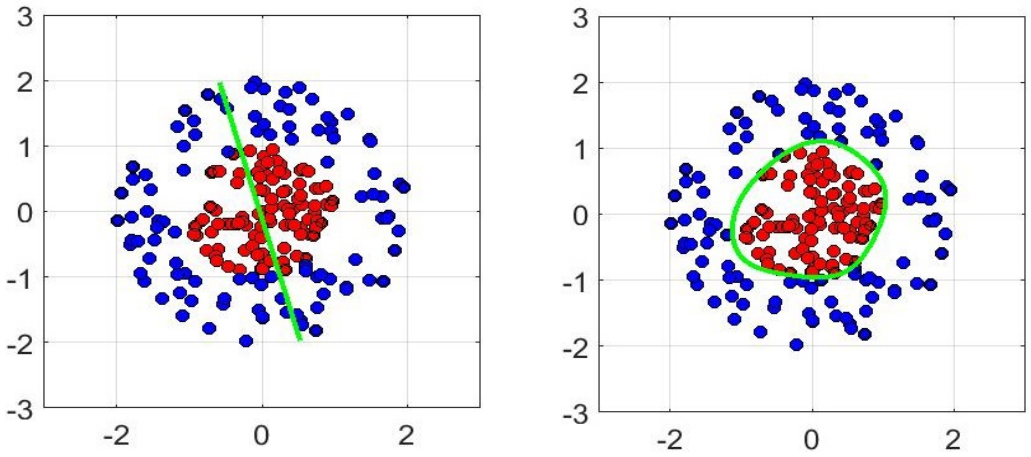
$$y(x) = sgn(f(x)) = sgn\left(\sum_{i=1}^{n} \alpha_i y_i (k\langle x_i, x \rangle) - a \right) \tag{2.31}$$

Figure 2.6 presents a toy example for the comparison between the linear and kernel SVM. The data set consists of 200 observations with each class containing 100 instances. Class 1 (red) is generated uniformly on a unit disk with radius $r_1 = \sqrt{u}$ where $u \sim [0, 1]$. The angle t is uniformly distributed on $[0, 2\pi]$ resulting in each data observation with coordinates $(r \cdot cos(t), r \cdot sin(t))$. Class -1 differs only in the radius given by $r_2 = \sqrt{3 \cdot u + 1}$. Figure 2.6 (a) illustrates the shortcomings of linear SVM when dealing with non-linearly separable data sets. Figure 2.6 (b) capitalizes on the advantages of introducing a non-linear operator to the classifiers decision making process. For this example the chosen kernel is the Gaussian radial

basis function (RBF).

$$k(x_1, x_2) = \exp\left(-\frac{1}{2\sigma^2}||x_1 - x_2||^2 \right) \tag{2.32}$$

where σ denotes the hyperparameter required to be carefully chosen by the practitioner. The flexibility of SVM is evident with the introduction of a kernel function allowing for highly non-linear datasets to be accurately classified. For further details on choosing kernel functions and there subsequent hyperparameter values please refer to [31, 49, 155, 183].



(a) (b)

Figure 2.6: Comparison of linear SVM (a) and kernel SVM (b).

It is worth noting that their exists a Bayesian interpretation of SVM referred to as Relevance Vector Machines (RVM) [216]. The RVM may be viewed as a special case of a Gaussian process with a sparse inducing prior. This choice of prior make it suitable to data sets with a large number of observations.

Alternative Machine Learning Approaches

Aside from the aforementioned methods used in this thesis, a number of alternative machine learning methods are worth mentioning. Note that some of the methods are applicable to both classification and regression tasks. However given the scope of this thesis, the focus is on classification variants. Neural networks [93] are at present one of the most well known learning models in machine learning applications. The term neural network has its origins in attempts to find

mathematical representations of information processing in biological systems [218]. This method takes inspiration from the interaction of neurons in the brain. The general idea is that the n input values are combined to create a response. The simplest form of neural network can be defined by

$$f(x) = \sum_{i=1}^{n} \alpha_i \psi(\boldsymbol{\theta}_i^T x + b_i) \tag{2.33}$$

where α_i is the coefficient associated with neuron i and $\psi(\cdot)$ a sigmoidal function with parameters $\boldsymbol{\theta}$ and b_i. If this response exceeds a threshold b_i the neuron emits a signal with the final prediction comprising of a linear combination of the signals from the collection of neurons. Layers of neurons between the input and output layer may be sequentially added to the neural network architecture. Thus, the first layer receives the inputs $\{x_i\}_{i=1}^n$ while the final layer emits a linear combination of the signals received from the second to last layer to make predictions. For classification tasks, the final layer of the neural network is a mapping onto the unit interval. This for example could be done using the sigmoid logit function specified in logistic regression. Although neural networks have seen success in computer vision [56], reinforcement learning [202] and speech recognition [26] aswel as other areas, disadvantages arise with their lack of interpretability as the number of layers increases along with the task of pre-specifying the network architecture. Note that their exists a theoretical connection between neural networks and Gaussian processes whereby [161] showed that a neural network with a single hidden layer converges to a Gaussian process as the number of neurons increases.

Classification and Regression Trees (CART) [42] are decision tree based methods which may be utilized for different predictive tasks depending on the nature of the target variable. These methods aim to predict the value of a target variable by learning simple decision rules inferred from the features of the training data. A tree structure is created whereby every node of the tree represents a feature, each branch represents a decision rule and each leaf represents an outcome. Figure 2.7 illustrates an example of a binary classification tree for the problem of cancer diagnosis. At the initial node, it is checked whether or not a lump has been detected in the initial examination of the patient. If so, the lump is further examined to establish whether or not abnormalities are present. If irregularities in the lump are found, a tissue sample through a biopsy is taken for further examination. The resulting examination will determine if the cells in the tissue

sample are benign (non-cancerous) or malignant (cancerous). For more complex decision tree structures, metrics such as the *information gain* [142] are utilized to determine the progression of the tree at each node.

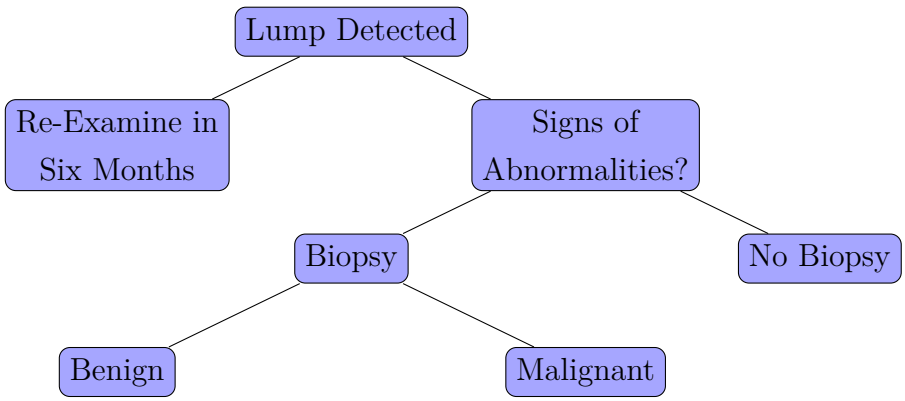


Figure 2.7: Example of a binary classification decision tree for cancer diagnosis. At each node, the tree is split depending on a binary choice. This is carried out sequentially until the resulting class nodes have been reached.

CART models may be easy to interpret and scale well to large data sets, however, they generalize poorly due to the greedy nature of the decision tree construction. One way to reduce the variance of predictions is to average together many predictions. There exists a group of *ensemble methods* [34] which aggregate multiple outputs made by a diverse set of individual decision trees. One such method, namely a random forest [138] is made up of many decision trees. This approach uses two key concepts that gives it the name random. Firstly, during training each tree in a random forest learns from a random sample of the training data. The idea is that by training each tree on different samples, although each tree might have high variance with respect to a particular set of the training data, overall, the entire forest will have lower variance but not at the cost of increasing the bias. Secondly, only a subset of the entire feature set is considered for splitting each node of each decision tree. The reason for doing this is the correlation of the trees in an ordinary bootstrap sample. If one or a number of features are very strong predictors for the target variable, these features will be selected in many trees, causing them to become correlated. Therefore, this step may be viewed as a de-correlation process. Overall, the random forest fits an individual decision tree to each of these smaller data sets and aggregates the predictions of the set of decision trees. This approach is particularly useful for problems with unbalanced data sets [138].

An alternative to the random forest is the Bayesian Additive Regression Trees (BART) approach proposed in [55]. As opposed to viewing the data as random as is the case with the random forest, BART treats the observed data as being fixed and the parameters characterizing the ensemble of trees as being random. BART can be distinguished from other ensemble-of-trees models due to its underlying probability model. As a Bayesian model, BART consists of a set of priors for the tree structure and the leaf parameters and a likelihood for the data in the nodes. The aim of the priors is to provide regularization, preventing any single tree from dominating the total fit while a tailored MCMC scheme is utilized to sample from the resulting posterior distribution of the generated trees. For more information on these methods, the reader is referred to the aforementioned references.

Aside from the above another notable classification model is the naive Bayes [34] classifier. Naive Bayes is a probabilistic classifier which assumes conditional independence between every pair of features given the class variable. Using Bayes' rule discussed in Chapter 3, the conditional probability of class membership can be decomposed as

$$P(y|x_1, ..., x_n) \propto Q(y) \prod_{i=1}^{n} p(x_i|y) \tag{2.34}$$

where $Q(y)$ is the prior probability of membership of class y and $p(x_i|y)$ the likelihood of the class label having generated the input features. The assumptions of independence allows for the likelihood function to be expressed as a product of individual likelihood terms. The resulting classification rule is given by:

$$y = \arg\max_y \quad Q(y) \prod_{i=1}^{n} p(x_i|y) \tag{2.35}$$

In spite of its apparently over-simplified assumptions, the naive Bayes classifier has worked quite well in many real-world situations, famously document classification and spam filtering [195]. The reader is referred to the aforementioned references for further information on the above methods.

2.2.3 Performance Evaluation

As previously mentioned, the classifier approximates a function which to the best of the model's knowledge appropriately assigns unseen data to their respective classes. In order to evaluate the quality of the function approximation, this

section introduces widely applied performance metrics for classification tasks. A *confusion matrix*, also referred to as a contingency table, is a visualization of the performance of a supervised algorithm. A given problem with c classes, produces a $c \times c$ confusion matrix with the rows representing the actual class and the columns the predicted class. It allows for a visualization of the model performance based on a set of test data for which the true class labels are known. In a binary setting, allow for the two classes to be denoted as the positive and negative class. In a confusion matrix, TP is the number of positive observations correctly identified as being positive, TN is the number of negative observations correctly identified as being negative, FP is the number of negative observations incorrectly identified as being positive and FN is the number of positive observations incorrectly identified as being negative. An example of a confusion matrix can be seen in Table 2.1.

	P_+	P_-	
A_+	True Positive	False Negative	m_1
A_-	False Positive	True Negative	m_0
	n_0	n_1	

Table 2.1: Confusion matrix where A_+ represents the actual positive class, A_- the actual negative class, P_+ the predicted positive class and P_- the predicted negative class. Define m_0 as the sum of the number of false positives and true negatives, m_1 as the sum of the number of true positives and false negatives, n_0 as the sum of the number of true positives and false positives and n_1 as the sum of the number of false negatives and true negatives.

The target objective of a classification task varies from problem to problem. In cancer diagnosis for example, ones goal would be the minimization of the false negative rate. The case of informing a potentially ill patient that they are not infected, could result in much graver consequences than that of the opposite scenario. For such tasks, the confusion matrix provides a degree of clarity

surrounding a models performance. This summary of the model output allows for the following terms to be derived

- Sensitivity $= P(A_+|P_+) = \frac{TP}{TP+FN} = \frac{TP}{m_1}$
- Specificity $= P(A_-|P_-) = \frac{TN}{TN+FP} = \frac{TN}{m_0}$
- FPR $= P(A_+|P_-) = \frac{FP}{TN+FP} = \frac{FP}{m_0}$
- FNR $= P(A_-|P_+) = \frac{FN}{TP+FN} = \frac{FP}{m_1}$

Aside from the ease of interpretation, the confusion matrix allows for both the accuracy and error rates to be computed.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.36}$$

Due to its simplicity and ease of interpretation, accuracy has been a widely adopted measure of performance. In practice however, accuracy can produce misleading results in the case of highly imbalanced data sets [130, 222]. Consequently, it is important to implement a number of difference performance metrics in order to establish a fair indication of the models predictive capabilities. The *Area Under the Curve* (AUC) from the Receiver Operating Characteristic (ROC) curve analysis, is a widely used metric for interpreting predictive performance in two class problems. ROC analysis was first developed during World War 2 for the analysis of radio signals. The AUC is achieved by plotting the True Positive Rate (TPR), also known as the sensitivity, versus the False Positive Rate (FPR). For classifiers which output a probability of class membership, the misclassification threshold chosen to separate the classes is $t \in [0, 1]$ [47]. Considering all possible values of t allows the ROC curve to be plotted. For a given event G and chosen threshold t let

$$TPR(t) = P(G \geq t|P_+) \tag{2.37}$$

$$FPR(t) = P(G \geq t|P_-) \tag{2.38}$$

with the corresponding ROC curve defined as

$$ROC(\cdot) = \{(g, f(g)), \qquad g \in [0, 1]\} \tag{2.39}$$

Here, $f(\cdot)$ denotes the ROC function. The resulting area under the curve is given by

$$AUC = \int_0^1 f(g)dg \tag{2.40}$$

As a demonstration, ROC curves for three different models are displayed in Figure 2.8. The optimal classifier produces a curve close to the top left corner, with a small false positive rate and large true positive rate having a theoretical maximum of 1. Whereas a curve along the diagonal reflects a classifier purely random in its predictions with a AUC of 0.5.

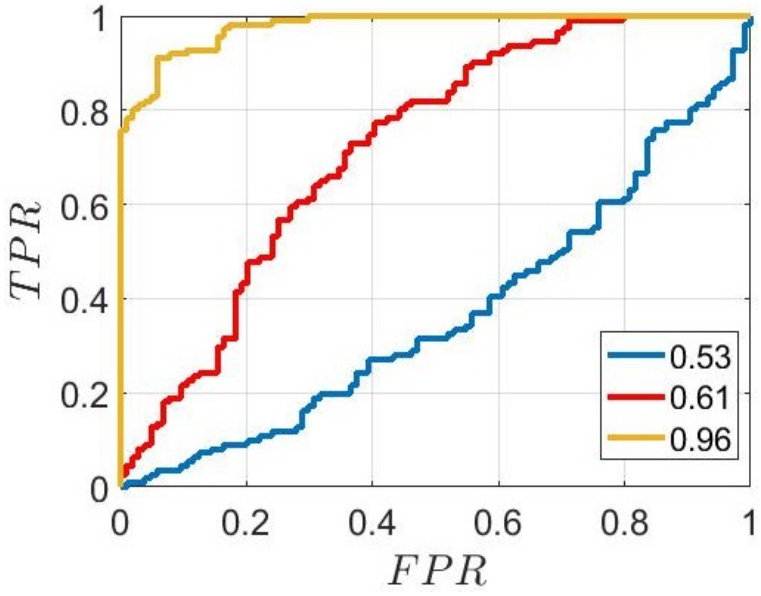


Figure 2.8: ROC curve comparison of three competing classification models.

Given Eq. 2.40 is a point estimate, [102] define an expression for the standard error of the empirical AUC by

$$SE(\hat{AUC}) = \sqrt{\frac{\hat{A}(1-\hat{A}) + (y_1 - 1) \cdot (Q1 - \hat{A}^2) + (y_0 - 1) \cdot (Q2 - \hat{A}^2)}{y_0 \cdot y_1}}$$

$$Q1 = \frac{\hat{A}}{2 - \hat{A}}, \qquad Q2 = \frac{2 \cdot \hat{A}^2}{1 + \hat{A}} \tag{2.41}$$

with $\hat{A}$ denoting the AUC, y_0 the number of negative class labels and y_1 the number of positive class labels. The $\hat{AUC}$ can be thought of as an estimate for AUC. With the true AUC quantity obtained with an infinite sample size. For a

comprehensive overview of ROC curve analysis and its theoretical underpinnings, the reader is referred to [13, 103, 94].

2.3 Chapter Summary

Motivated by the research presented in this dissertation, this chapter provides a broad overview of machine learning with a particular emphasis placed on supervised learning. Section 2.2.2 discusses classification along with the introduction of three classification methods applied in this dissertation. A proportion of classification models, are probabilistic and require the identification of unknown model parameters. The importance of which for predictive performance was illustrated using logistic regression [107, 148]. One approach for computing these parameters, namely maximum likelihood estimation (MLE), was derived. However, the need for a numerical optimization framework combined with the absence of uncertainty quantification estimates, highlight the requirement for a robust framework which offers statistical assurances regarding the estimated parameters. In the following chapter, a statistical inference approach named Bayesian inference is introduced. Bayesian inference allows for the quantification of parameter uncertainties and for predictions to be made.

Unlike logistic regression, the ability of Gaussian processes to address classification problems containing non-linear decision boundaries is discussed. A margin based model aiming to identify an optimal separating hyperplane between two classes is also introduced. Support Vector Machines (SVM) interpret this as an optimization task, allowing for an observation to belong to a given class once the side of the hyperplane which it belongs has been identified. However while SVM are suitable to large data sets due to the sparse inducing dual representation of the optimization problem, unlike Gaussian processes they are deterministic in their output. Aside from the three presented approaches, a brief overview of alternative existing classification methods is also provided. Additionally, different metrics for analysing classifiers predictive performance were also discussed along with the importance of quantifying variation in metric point estimates highlighted.

Chapter 3

Bayesian Inference

This chapter introduces the concept of Bayesian inference. As discussed in Chapter 2, accurate identification of model parameter values directly influences the predictive capabilities of a classification model. Bayesian statistics allows for the identification of model parameters along with the uncertainty in the subsequent estimations to be quantified. In order to do so, a complex probability distribution referred to as the *posterior*, is required to be computed. The mathematical formulation of which however, in most cases does not have an analytical solution. Aside from parameter identification, the posterior distribution allows for the relative evidence for a number of competing models to be assessed along with model predictions to be made. Techniques for sampling from the posterior which provide the basis for frameworks developed in this dissertation, are discussed along with their corresponding shortcomings in their current form highlighted.

3.1 Introduction

Extracting information from data through computer modelling has become common in science and engineering. In terms of data, the world around us is being captured in everyday tasks, such as medical appointments and banking transactions. As discussed in Chapter 1, constructing a model which is a fair representation of reality however, often results in different sources of uncertainties appearing. By adopting probability as the language of uncertainty, statistical inference provides a framework for measuring these uncertainties, allowing the practitioner to reliably evaluate possible outcomes in terms of probability statements [156]. Probability theory is said to have been inspired by games of chance in 17^{th} century France

and initiated by the Fermat-Pascal correspondence [167]. In this sense, it has been a relative latecomer in relation to the historical development of other areas of mathematics.

Numerous schools of thought co-exist in terms of what probability actually means. The *frequentist* interpretation of probability associates the probability of an event with the frequency of the event out of a number of simulated trials. An issue of this interpretation highlighted by [19], lies in the finite set of trials. Therefore, in practice the exact frequency of the event of interest cannot be observed. Motivated by problems arising in quantum mechanics, *propensity* probability [101] interprets probability as physical disposition. It can be thought of as a tendency of a given type of physical property to yield an outcome of a certain kind [174, 178]. In this work, uncertainties are represented under a *Bayesian* interpretation of probability where probabilities represent a degree of belief in a proposition. The most apparent difference between the Bayesian and frequentist schools of thought is the attribution of randomness in a statistical model. From a Bayesian point of view, probabilities may represent uncertainties about quantities which are not directly observable e.g. parameters of a model. On the other hand, advocates of the frequentist approach affirm that only the data is random. Subsequently, values of the parameters are fixed and therefore should not be considered random variables. For an in depth discussion on the interpretations of probability theory, the reader is referred to [101].

In terms of general uncertainty quantification, Bayesian inference encapsulates a solid mathematical foundation for probability models to be updated in the presence of data. Consequently, such models represent a degree of belief in the data, resulting in the appearance of conditional probabilities. The term Bayesian was first coined in honour of a British mathematician, Reverend Thomas Bayes [24]. Bayes wrote a manuscript detailing a framework, which was later generalized by Laplace [133], to be known as *Bayes'* theorem. The use of Bayes requires the specification of a prior distribution along with a particular view of uncertainty. Typically, the Bayesian interpretation of probability can be expressed in terms of two subgroups: *objectivity* and *subjectivity*, which differ in their understanding of probability. The subjective school of thought regard probability as a personal belief, with the degree of belief regarding an event stemming directly from the decision maker. In contrast, the objective school of thought interpret probability purely based on the knowledge of the observed event. In theory, in the event that

the same information is available to various decision makers, they would all reach the same conclusions [180]. However, no general consensus exists regarding the true definition or aim of objective Bayesian analysis [29].

3.2 The Posterior Distribution

Let $\boldsymbol{\theta} \in \mathbb{R}^d$ represent a set of unknown model parameters which we seek to make inference about based on an observed dataset D. Here $\boldsymbol{\theta}$ is a vector of unknown parameters with the values of $\boldsymbol{\theta}$ being a subset of the sample space $\boldsymbol{\Theta}$. Prior knowledge of $\boldsymbol{\theta}$ may be characterised by a probability distribution $Q(\boldsymbol{\theta})$. Through the construction of a probabilistic model referred to as the likelihood function $L(D|\boldsymbol{\theta})$, the observed data is embedded in the Bayesian framework. The likelihood function represents the plausibility of observing D given a certain $\boldsymbol{\theta}$. Bayes' theorem allows for the prior beliefs about $\boldsymbol{\theta}$ to be updated by

$$P(\boldsymbol{\theta}|D) = \frac{Q(\boldsymbol{\theta})L(D|\boldsymbol{\theta})}{P_D} \tag{3.1}$$

where $P(\boldsymbol{\theta}|D)$ denotes the posterior distribution and P_D the evidence of the data. The posterior distribution combines $Q(\boldsymbol{\theta})$, $L(D|\boldsymbol{\theta})$ and D to characterise the updated beliefs of the unknown $\boldsymbol{\theta}$. If $\boldsymbol{\theta}$ is continuous the *evidence* or *marginal likelihood* is given by

$$P_D = \int Q(\boldsymbol{\theta})L(D|\boldsymbol{\theta})d\boldsymbol{\theta} \tag{3.2}$$

Eq. 3.2 acts as a normalizing constant ensuring $P(\boldsymbol{\theta}|D)$ is a valid Probability Density Function (PDF). The synonym marginal likelihood refers to the interpretation that P_D is obtained by marginalising over the unknown model parameters. The computation of P_D generally proves challenging, due to the potentially high dimensional integral being analytically infeasible. In many applications it is possible to compute $L(D|\boldsymbol{\theta})$ point wise but its dependence on $\boldsymbol{\theta}$ is intractable. This results in an analytical solution of $P(\boldsymbol{\theta}|D)$ being unattainable. A simple one dimensional illustration of the Bayesian updating process is shown in Figure 3.1.

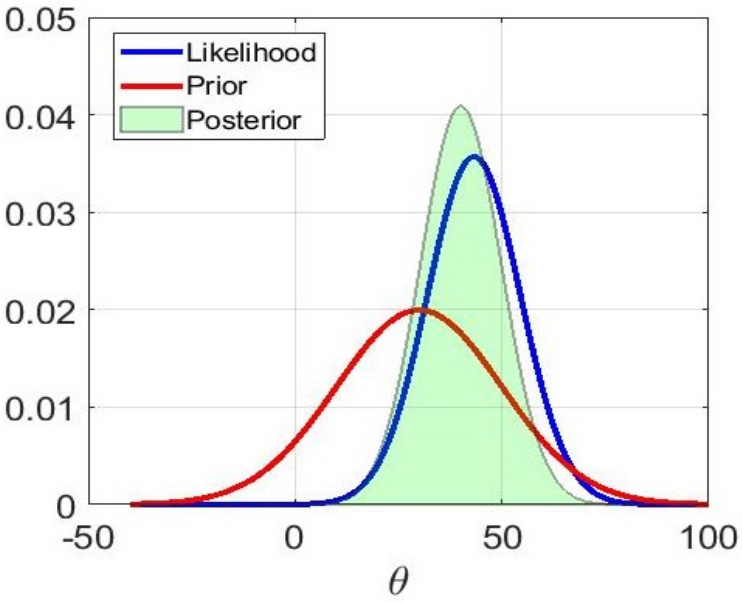


Figure 3.1: One dimensional example of $L(D|\boldsymbol{\theta})$ and $Q(\boldsymbol{\theta})$ being combined to produce $P(\boldsymbol{\theta}|D)$.

The choice of prior distribution has remained a key research question in Bayesian statistics. The prior allows for the practitioner to incorporate expert information (if available) into the modelling process by reflecting pre-data understanding of $\boldsymbol{\theta}$. In reality, prior beliefs or knowledge of an event may vary from problem to problem. In the case of expressing ignorance towards $\boldsymbol{\theta}$, a *noninformative* prior can be chosen. In general, such a choice results in an expression for the posterior which is dominated by the likelihood function by 'letting the data speak for itself' [228]. Both Bayes and Laplace, performed Bayesian analysis using a constant prior distribution for $\boldsymbol{\theta}$, although the motivations of each in doing so were considerably more sophisticated than simply stating each possible value of $\boldsymbol{\theta}$ should receive equal prior weighting. Common choices include the uniform distribution [133] and *Jeffrey's* prior [114]. There exists the possibility for the practitioner to express one's personal probability that $\boldsymbol{\theta}$ lies in a given subset of $\boldsymbol{\Theta}$ through the use of a *subjective* prior [28]. For these expert beliefs surrounding $\boldsymbol{\theta}$ to be quantified, questions of the practitioner must be asked. In practice, given the complexity of statistical models, it has been noted [226] that a strict methodology for the elicitation of subjective prior probabilities can be extremely difficult. Frameworks for allowing an expert to quantity ones beliefs are given in [80]. Yuen et al. [230] argue that the use of a noninformative prior may not significantly influence the parameter identification process but can greatly

influence the model evidence used for model class selection as discussed in Section 3.2.1.

The likelihood function is a key ingredient of any Bayesian framework by creating a link between the model (M) and the data D. Intuitively, the name likelihood function stems from the unknown parameter for which $L(D|\boldsymbol{\theta})$ is large is more 'likely' to have generated D than a $\boldsymbol{\theta}$ for which $L(D|\boldsymbol{\theta})$ is small. The likelihood function forms a key concept of the *likelihood principle*, which makes explicit the natural conditional idea that only the actual observed D should be relevant to conclusions or evidence about $\boldsymbol{\theta}$. Unlike a probability however, it is not clear if $L(D|\boldsymbol{\theta})$ has any particular meaning in isolation [28]. Berger et al. [30] attain that the likelihood principle does not inform the practitioner of how to approach interpretation unless viewed through the use of a 'Bayesian filter'. Interpreting probability as the language of uncertainty, allows for uncertainty reflected in $L(D|\boldsymbol{\theta})$ to be expressed probabilistically. Furthermore, the use of $L(D|\boldsymbol{\theta})$ in this setting allows for subsets of $\boldsymbol{\Theta}$ to be compared in terms of some method of averaging over $L(D|\boldsymbol{\theta})$. This is in contrast to the so called 'adhockeries' introduced for non Bayesian approaches to averaging the likelihood function [92]. In the event of $\boldsymbol{\theta}$ being extremely large, a criticism of the likelihood principle may stem from the failure of $L(D|\boldsymbol{\theta})$ to provide clearly interpretable information. Instead, [30] views this as an indication that prior information must be used in such situations. Assuming the observations x, in the observed data are independent, the resultant likelihood function may be expressed as

$$L(D|\boldsymbol{\theta}) = \prod_{i=1}^{n} p(x_i|\boldsymbol{\theta}) \tag{3.3}$$

For the remainder of this dissertation, it is assumed unless otherwise stated the likelihood function is of this form. In this case, the posterior distribution may be viewed as a weighted likelihood function with the weighting being imposed by the chosen prior distribution.

3.2.1 Model Class Selection

Aside from inferring the values of model parameters, the posterior may be used for rating competing models against one another. In this section allow for the models to refer to physical models governed for example by partial differential equations [224]. This process termed, Bayesian model class selection, is essentially

Bayesian updating at the model class level for comparing competing models [142] by choosing the model which provides the most evidence in line with the data. The issue of model selection occurs when one must choose from a variety of competing model structures. This is complicated by the fact that models with more parameters will likely be able to better replicate some training data than models with less parameters. However, this may result in a model which is over complex and is referred as being *overfitted*. In the event that different model structures are available, one may use Bayes' theorem to express the probability that a model M_i from a finite set of models $\{M_i, i = 1, ...n\}$ is suitable given the data D.

$$P(M_i|D) = \frac{P(D|M_i)Q(M_i)}{P(D)} \tag{3.4}$$

where $Q(M_i)$ represents ones prior beliefs and $P(D|M_i)$ the likelihood of M_i generating the observed data. Bayesian model class selection is not to be confused with model averaging which involves estimating the same quantity with a variety of models before averaging the estimates to establish how likely each model is. A trade-off exists in statistical modelling whereby, a slight increase in model performance due to a great increase in model complexity can be detrimental to predictive performance due to an over reliance on the observed data [115]. Therefore, in model class selection it is important to penalize complicated models [21]. Pioneering work on Bayesian methods by [115], highlighted the need for a quantification of *Ockhams razor* which has roughly been translated by [203] to *'It is vain to do with more what can be done with fewer'*. Ockham's razor states that an explanation of facts should be no more complicated than necessary. In terms of model class selection, this philosophy highlights that the chosen model should fit the observed data well yet be as simple as possible.

Edwards [73] states that *'within the framework of a statistical model, all of the information which the data provided concerning the relative merits of two hypotheses is contained in the likelihood ratio of those hypotheses'*. In other words, evidence between two competing hypotheses exists if the probability of D under one hypothesis is greater than the probability of D under another. Note that in this context the evidence referred to is the evidence which exists within a particular dataset. The Bayesian approach to hypothesis testing was developed by [115] allowing for competing models to be ranked against one another while

incorporating subsequent uncertainties. The so called *Bayes' factor* utilizes the model evidence between two competing models to compare how well each predicts the data.

For a correct interpretation of Bayesian model class selection, it is vital to acknowledge that if all models contained in the class of models $\{M_i, i = 1, ...n\}$ are inadequate or of poor quality, the best model among these models will be chosen. By inspecting the posterior plausibility of the individual classes this will become adherent. To aid analysis, [118] suggests the inclusion of a relative simple model among the class of individual models. By checking whether the posterior plausibility of the simple model are relatively small in comparison to the largest posterior plausibilities, it may be inferred that the overall performance of the considered class of models could be poor.

3.2.2 Predictive Posterior

Aside from inferring the unknown parameters of a model and allowing for competing models to be compared in an intuitive manner, the posterior distribution can also be used in a predictive setting. In terms of machine learning discussed in Chapter 2, consider again the training data set $D = \{(x_i, y_i)\}_{i=1}^n$, where x are the given data observations and y the corresponding class labels. Let x^* denote an unseen data observation for which class membership y^* is required to be determined. For the desired predictive quantity to be computed, the posterior distribution is marginalised in terms of $\boldsymbol{\theta}$ as:

$$P(y^*|D) = \int P(y^*|D, \boldsymbol{\theta}) P(\boldsymbol{\theta}|D) d\boldsymbol{\theta} \tag{3.5}$$

This distribution is commonly referred to as the *predictive posterior*. The predictive distribution is weighted by how plausible the parameters are based on the training dataset. In the case of the logistic regression model discussed in Chapter 2, this predictive quantity may be expressed by

$$P(y^*|D) \approx \int \sigma(x^{*T}\boldsymbol{\theta}) P(\boldsymbol{\theta}|D) d\boldsymbol{\theta} \tag{3.6}$$

where $\sigma(\cdot)$ denotes the aforementioned logit function. Due to the requirement of taking a product of sigmoid functions, this expression is analytically intractable.

In summary, Bayesian inference offers a robust framework to quantify the uncertainty in estimating model parameters, model selection and making predictions. Due to the presence of a high dimensional integral, the posterior cannot be computed by conventional numerical means. There exists a family of approaches [151, 215] which approximate $P(\boldsymbol{\theta}|D)$ by a Gaussian distribution. Commonly, converging to a solution through a pre-specified number of iterations or by comparison of the difference in the generated moments with a chosen threshold. Although the practice of approximating $P(\boldsymbol{\theta}|D)$, using a family of standard distributions ensures computational efficiency, in the event that the target density is highly non-Gaussian or of a complex topology, problems may occur. Under Laplace Approximation (LA) for example, sample accuracy has an inverse relationship with the number of uncertain parameters, whilst the convergence of Expectation Propagation (EP) is not always guaranteed [34, 151]. For the purpose of this dissertation, focus is placed on methods which sample from the posterior as opposed to approximating the posterior.

3.3 Methods for Bayesian Inference

Having discussed the importance of parameter estimation under the Bayesian paradigm, this section presents a number of approaches for identifying their unknown values. A Bayesian framework for point estimation is firstly discussed before three widely applied approaches which produce independent and dependent samples from the posterior distribution are introduced. The chosen sampling methods are reviewed as they will be of particular importance in subsequent chapters.

3.3.1 Maximum A Posteriori (MAP) Estimation

The maximum a posteriori (MAP) is a point estimate of the mode of the posterior and may be given by:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \quad P(\boldsymbol{\theta}|D) \tag{3.7}$$

This selects the parameter that produces the highest posterior density (the mode). Observe that since $P(\boldsymbol{\theta}|D) \propto Q(\boldsymbol{\theta})L(D|\boldsymbol{\theta})$, then the MAP estimate coincides with the MLE in Chapter 2 when the prior distribution is uniform. While the

advantage of the MAP is its ability to incorporate prior information into the estimated parameter, it is by no means a full Bayesian treatment of the posterior distribution. As Bayesian methods are traditionally characterised in terms of entire distributions which summarize the data and draw inferences, a single point estimate may not fairly represent all of the information contained in the distribution. As the MAP value does not consider the uncertainty regarding $\hat{\boldsymbol{\theta}}$ into the estimate, in some cases it may be sufficient to specify a *credible interval*. For which it may be deduced that the true parameter values lies within a given range $[a, b]$ with a specified probability p. Aside from the lack of information regarding the uncertainty in $\hat{\boldsymbol{\theta}}$, like MLE the MAP may identify a local maximum as opposed to a global while the issue of taking the derivative of a potentially high dimensional complex distribution must also be addressed. As an alternative, two methods which allow for samples to be generated from the entire posterior distribution are discussed in the following sections.

3.3.2 Rejection Sampling

Rejection sampling is a resampling method which allows the generation of independent draws of any distribution that is defined in terms of its density [79]. Resampling methods may be described as generation techniques which require draws of random variables in more than one step. This section discusses a specific form of rejection sampling algorithm which utilizes a scaled version of the prior distribution for the proposal of samples. This type of rejection sampler provides the basis for the Bayesian inference methods discussed in this thesis and as such is of great importance. For an overview of more general forms of rejection samplers, the reader is referred to [79, 81].

Let the potentially unnormalized distribution $\pi(\boldsymbol{\theta}|D)$ be the target posterior, with the relationship between the normalized and unnormalized posterior be given by $P(\boldsymbol{\theta}|D) \propto \pi(\boldsymbol{\theta}|D)$. Let $Q(\boldsymbol{\theta})$ represent the normalized prior distribution from which independent samples are readily available. Rejection sampling proposes selecting a positive constant $k \in \mathbb{R}$ such that

$$\pi(\boldsymbol{\theta}|D) \leq kQ(\boldsymbol{\theta}) \tag{3.8}$$

ensures $\boldsymbol{\theta} \sim \pi(\boldsymbol{\theta}|D)$. Independent samples are generated as outlined in Algorithm 1.

Algorithm 1 Rejection Sampling

1: Define $Q(\boldsymbol{\theta})$ and k
2: Generate u uniformly with support $[0, 1]$ independently of $\boldsymbol{\theta}$
3: Generate $\boldsymbol{\theta} \sim Q(\boldsymbol{\theta})$
4: $r = \frac{\pi(\boldsymbol{\theta}|D)}{kQ(\boldsymbol{\theta})}$
5: **if** $u < r$ **then**
6: Accept $\boldsymbol{\theta}$
7: **else**
8: Reject $\boldsymbol{\theta}$
9: **end if**

Let u be a uniform random variable on $[0, 1]$, r the sample acceptance ratio and p_A the probability of a sample being accepted. Algorithm 1 is an iterative procedure in that $Q(\boldsymbol{\theta})$ is repeatedly sampled from until $\boldsymbol{\theta}$ is accepted with respect to r and u. To prove that the rejection algorithm produces samples from $\pi(\boldsymbol{\theta}|D)$, simply requires to show that the conditional density $P(\boldsymbol{\theta}|u < r)$ is $\pi(\boldsymbol{\theta}|D)$.

$$P(\boldsymbol{\theta}|u < r) = \int_0^1 P(\boldsymbol{\theta}, u|u < r)du = \frac{\int_0^1 P(\boldsymbol{\theta}, u)P(u < r|\boldsymbol{\theta})du}{p_A}$$

By the independence between $\boldsymbol{\theta}$ and u, $P(\boldsymbol{\theta}, u) = Q(\boldsymbol{\theta})P(u)$. Allowing for the conditional distribution to be

$$P(\boldsymbol{\theta}|u < r) = \frac{Q(\boldsymbol{\theta})}{p_A} \int_0^1 P(u)I[u < r]du$$

where $I[\cdot]$ is referred to as the indicator function equalled to 1 if the argument is true and 0 otherwise. For $u \in [0, 1]$, $P(u) = 1$. While the likelihood of $\boldsymbol{\theta}$ resulting in sample acceptance is the ratio r given in Algorithm 1.

$$P(\boldsymbol{\theta}|u < r) = \frac{Q(\boldsymbol{\theta})}{p_A} \cdot \frac{\pi(\boldsymbol{\theta}|D)}{kQ(\boldsymbol{\theta})} = \frac{\pi(\boldsymbol{\theta}|D)}{kp_A} \tag{3.9}$$

The efficiency of rejection sampling is determined by the probability of sample acceptance

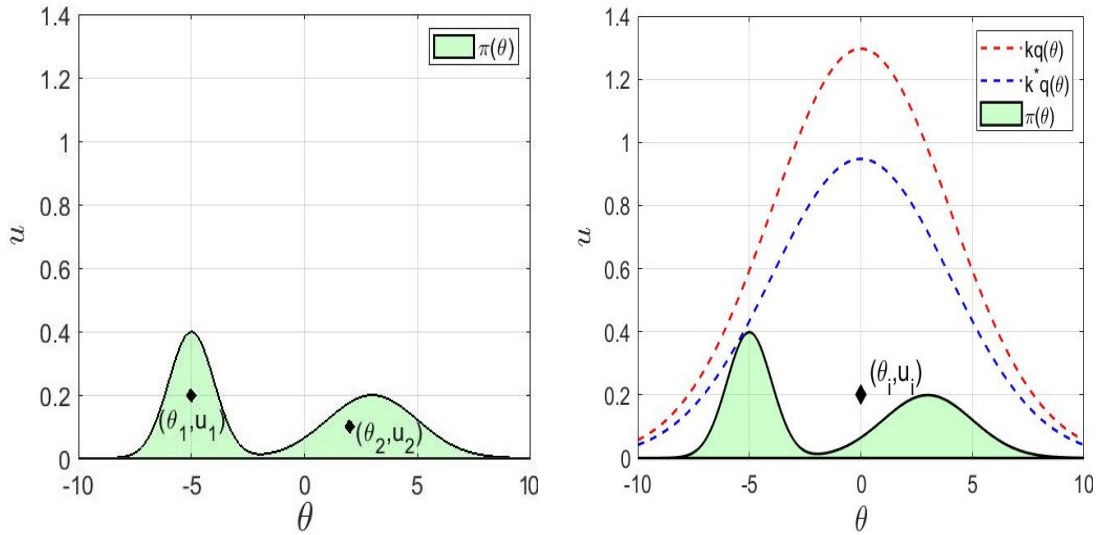
$$p_A = \int \int_0^1 q(\boldsymbol{\theta})P(u)I[u < r]dud\boldsymbol{\theta} = \int Q(\boldsymbol{\theta})\frac{\pi(\boldsymbol{\theta}|D)}{kQ(\boldsymbol{\theta})}d\boldsymbol{\theta} = \frac{1}{k}\int \pi(\boldsymbol{\theta}|D)d\boldsymbol{\theta} \tag{3.10}$$

Substituting Eq. 3.10 into Eq. 3.9 yields

$$P(\boldsymbol{\theta}|u < r) = \frac{\pi(\boldsymbol{\theta}|D)}{\int \pi(\boldsymbol{\theta}|D)d\boldsymbol{\theta}} \tag{3.11}$$

Therefore on average k samples of $\boldsymbol{\theta}$ are required in order to have one accepted sample $\boldsymbol{\theta} \sim \pi(\boldsymbol{\theta}|D)$. Eq. 3.10 reveals that for a large k the computational costs to generate independent samples from the target distribution will increase. Great care must be taken when choosing k, especially in the case of higher dimensional distributions. Its value should be chosen small enough to ensure $\boldsymbol{\theta} \sim \pi(\boldsymbol{\theta}|D)$ but large enough to allow for efficient sampling.

Figure 3.2 presents a one dimensional example of direct sampling in the case of a tractable posterior distribution versus rejection sampling for the intractable case. In the case of rejection sampling, samples are drawn uniformly under $kq(\boldsymbol{\theta})$ as in Algorithm 1, while samples above $\pi(\boldsymbol{\theta}|D)$ such as (θ_i, u_i) are rejected. Figure 3.2 (b) highlights the influence of choosing k appropriately with the optimal value denoted by k^*. By choosing a value too large, samples can potentially be proposed in the regions further away from $\pi(\boldsymbol{\theta}|D)$ in comparison to selecting k^*. While the more closely $Q(\boldsymbol{\theta})$ matches $\pi(\boldsymbol{\theta}|D)$, the lower the rejection rates can be made.



(a) (b)

Figure 3.2: Rejection sampling one-dimensional example. In the case of possible direct sampling (a), sample locations can potentially be taken as points drawn uniformly under the target distribution. For rejection sampling (b), samples are drawn uniformly under $kQ(\boldsymbol{\theta})$ as in Algorithm 1. With samples above $\pi(\boldsymbol{\theta}|D)$ such as (θ_i, u_i) being rejected.

To alleviate the issue of algorithmic inefficiency, adaptive rejection samplers that increase the acceptance rates by taking advantage of particular properties of the posterior density have been proposed [89, 146]. However, such techniques require guarantees of the posterior being log-concave or unimodal. In a different approach, [75] aim to reduce sample rejection rates through the introduction of kernel methods to construct a problem dependent proposal distribution. The authors however, acknowledge issues in problems beyond a moderate number of dimensions along with the case that the majority of the mass of the distribution is based around the mode.

Eq. 3.8 may be reformulated to allow for the *rejection principle* to be derived. Let the relationship between the normalized and possibly unnormalized posterior be given by $P(\boldsymbol{\theta}|D) \propto \pi(\boldsymbol{\theta}|D)$.

$$\begin{aligned} & \pi(\boldsymbol{\theta}|D) \leq kQ(\boldsymbol{\theta}) \\ \implies & \frac{\pi(\boldsymbol{\theta}|D)}{kQ(\boldsymbol{\theta})} \leq 1 \\ \implies & \frac{Q(\boldsymbol{\theta})L(D|\boldsymbol{\theta})}{kQ(\boldsymbol{\theta})} \leq 1 \\ \implies & cL(D|\boldsymbol{\theta}) \leq 1 \end{aligned} \tag{3.12}$$

with $c = k^{-1}$ being referred to as the *likelihood multiplier* and for any $\boldsymbol{\theta}$ which satisfies Eq. 3.12, $\boldsymbol{\theta} \sim P(\boldsymbol{\theta}|D)$. The issue of selecting c along with rejection samplings low sample acceptance rates forms the basis of a Bayesian inference framework discussed in Chapter 4.

3.3.3 Markov Chain Monte Carlo

In the previous section, the low sample acceptance rates along with the task of carefully choosing the likelihood multiplier highlighted potential difficulties encountered in the rejection sampling algorithm. As an alternative, Markov Chain Monte Carlo (MCMC) methods can be used to sample from probability distributions that are complex and have unknown normalization. This is achieved by relaxing the requirement that the samples should be independent.

Let $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_n$ be a sequence where, $\boldsymbol{\theta}_i \in \mathbb{R}^d$ for all $i \in \{1, 2, ..., n\}$ and d the dimension of the vectors. This sequence is called a *Markov chain* if the following

rule holds for the joint PDF of any $\boldsymbol{\theta}_i$

$$P(\boldsymbol{\theta}_i|\boldsymbol{\theta}_{i-1}, ..., \boldsymbol{\theta}_1) = T(\boldsymbol{\theta}_i|\boldsymbol{\theta}_{i-1}) \tag{3.13}$$

That is, a Markov chain is a stochastic process having the property that, given the present state, the future states are conditionally independent of the past states. The Markov chain generates a correlated sequence of states with each step in the sequence being drawn from a transition operator $T(\boldsymbol{\theta}'|\boldsymbol{\theta})$ giving the probability of moving from state $\boldsymbol{\theta}$ to state $\boldsymbol{\theta}'$. According to the *Markov property*, the transition probabilities depend only on the current state $\boldsymbol{\theta}$. A basic requirement for $T(\boldsymbol{\theta}'|\boldsymbol{\theta})$ is the property of stationary such that given $\boldsymbol{\theta} \sim \pi(\boldsymbol{\theta})$

$$\pi(\boldsymbol{\theta}') = \int T(\boldsymbol{\theta}'|\boldsymbol{\theta})\pi(\boldsymbol{\theta})d\boldsymbol{\theta} \tag{3.14}$$

If Eq. 3.14 holds, $\pi(\cdot)$ is referred to as the stationary PDF with transition PDF $T(\boldsymbol{\theta}'|\boldsymbol{\theta})$. Ensuring that any Markov chain beginning with a sample from the stationary distribution will return a sample from the same stationary distribution. Such a Markov chain holds the properties of irreducibility and aperiodicity [214]. An initial *burn in* period allows for the sequence of samples to be distributed according to the desired stationary distribution. Identifying the length of this burn in period and general chain convergence is a non-trivial task with various strategies being discussed in [86, 117, 185]. A sufficient condition to show that $\pi(\cdot)$ is the stationary distribution of the Markov chain is to prove that the detailed balance equation exists.

$$T(\boldsymbol{\theta}'|\boldsymbol{\theta})\pi(\boldsymbol{\theta}) = T(\boldsymbol{\theta}|\boldsymbol{\theta}')\pi(\boldsymbol{\theta}') \tag{3.15}$$

Which states that a step starting at equilibrium and transitioning under $T(\boldsymbol{\theta}'|\boldsymbol{\theta})\pi(\boldsymbol{\theta})$ has the same probability 'forwards' $\boldsymbol{\theta} \rightarrow \boldsymbol{\theta}'$ and 'backwards' $\boldsymbol{\theta}' \rightarrow \boldsymbol{\theta}$.

3.3.4 Gibbs Sampling

One of the most widely used MCMC methods is Gibbs sampling [84]. The intuition behind Gibbs sampling is to represent the vector $\boldsymbol{\theta} = \{\theta_1, ..., \theta_d\}$ in terms of its components and to iteratively update each component of $\boldsymbol{\theta}$ whilst holding all other components constant. Each step of the Gibbs sampling procedure involves replacing the value of one of the d parameters by a value drawn from the

distribution of the chosen component conditioned on the values of the remaining components. This procedure is repeated either by cycling through the components in a particular order or by choosing the components to be next updated at random. A generic Gibbs sampling algorithm is given in Algorithm 2.

Algorithm 2 Gibbs Sampling

1: Initialize $\boldsymbol{\theta} = \{\theta_1^0, ..., \theta_d^0\}$
2: **for** $i = 1, ..., t$ **do**
3: $\theta_1^i \sim \pi(\theta_1^i | \theta_2^{i-1}, ..., \theta_d^{i-1})$
$\theta_2^i \sim \pi(\theta_2^i | \theta_1^i, ..., \theta_d^{i-1})$
$\vdots$
$\theta_d^i \sim \pi(\theta_d^i | \theta_1^i, \theta_2^i, ..., \theta_{d-1}^i)$
4: **end for**

At each of the t iterations of the Gibbs sampler, their are d steps. At each t, an ordering of the vector components is chosen and the conditional densities are sampled from. The availability of Just Another Gibbs Sampler (JAGS) [177] and Bayesian inference Using Gibbs Sampling (BUGS) [141] software packages has helped increase the usage of Gibbs sampling across the wider scientific community. To illustrate the capabilities of the Gibbs sampler, consider the following example concerning a bivariate Gaussian distribution. The target distribution is given by $\pi(\boldsymbol{\theta}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with $\boldsymbol{\theta} = \{\theta_1, \theta_2\}$ and

$$\boldsymbol{\mu} = [\mu_1, \mu_2]$$

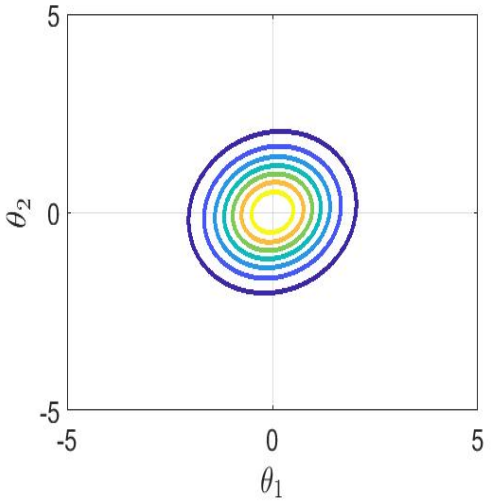
$$\boldsymbol{\Sigma} = \begin{bmatrix} 1 & \rho_{1,2} \\ \rho_{2,1} & 1 \end{bmatrix} \tag{3.16}$$

The vector $\boldsymbol{\mu}$ denotes the mean, $\boldsymbol{\Sigma}$ the covariance matrix and ρ the correlation between parameters. Before implementing the Gibbs sampler, the conditional distributions for θ_1 and θ_2 must be available. Let the respective conditional

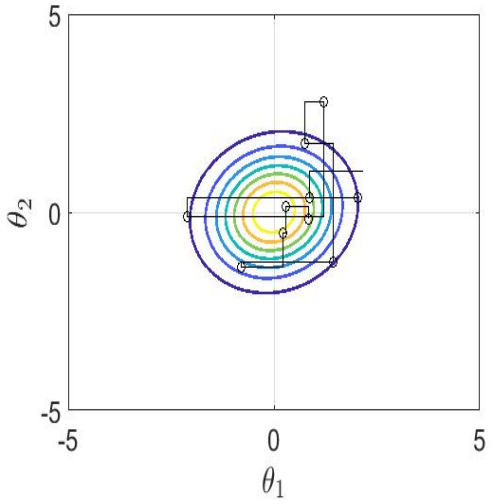
distributions be expressed as

$$\pi(\theta_1^t|\theta_2^{t-1}) = \mathcal{N}(\mu_1 + \rho_{2,1}(\theta_2^{t-1} - \mu_2), \sqrt{1-\rho_{2,1}^2})$$
$$\pi(\theta_2^t|\theta_1^t) = \mathcal{N}(\mu_2 + \rho_{1,2}(\theta_1^t - \mu_1), \sqrt{1-\rho_{1,2}^2}) \tag{3.17}$$

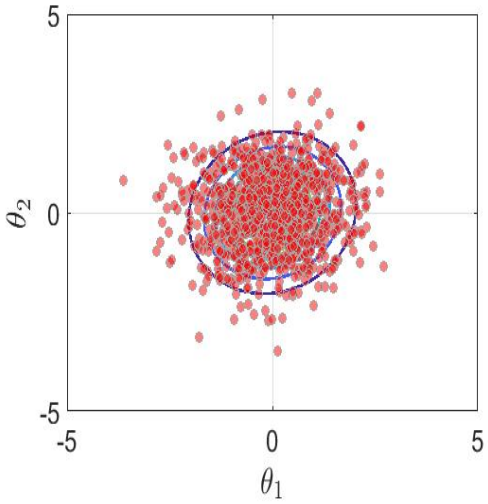
Let t denote the current state of the parameters as outlined in Algorithm 2. Both conditional distributions are univariate Gaussians with means dependent on the most recent state of the conditioning parameter and a variance dependent on the target distributions covariance matrix. Figure 3.3 presents an example of how the Gibbs sampler progresses. For this problem, $\boldsymbol{\mu} = [0, 0]$ and $\rho_{1,2} = \rho_{2,1} = 0.1$. The sequential update of samples at each iteration is evident from Figure 3.3 panel (b). For this two dimensional problem, a horizontal and vertical movement is required in the parameter space for every sample drawn. The resulting posterior samples are illustrated in Figure 3.3 (c). Due to the representation of $\boldsymbol{\theta}$ in terms of its d components, Gibbs sampling is a viable approach for high dimensional tasks. However, the nature of the sample updates as highlighted in Figure 3.3 (b) renders the algorithm inefficient when the parameter components are highly correlated. To illustrate this, consider again the above example but with $\rho_{1,2} = \rho_{2,1} = 0.9$. The increased correlation between parameters will ensure that the target distribution is more skewed. Figure 3.4 (b) reveals the difficulties of the Gibbs sampler when dealing with highly correlated parameters. The sampler is less effective in comparison to the lowly correlated parameter problem in Figure 3.3 (b) as the sequential updates in terms of moments along the x and y axes for every sample drawn results in an inefficient exploration of the target distribution. This drawback can result in the Gibbs sampler becoming very inefficient when drawing samples from the target distribution.



(a) Target Distribution.

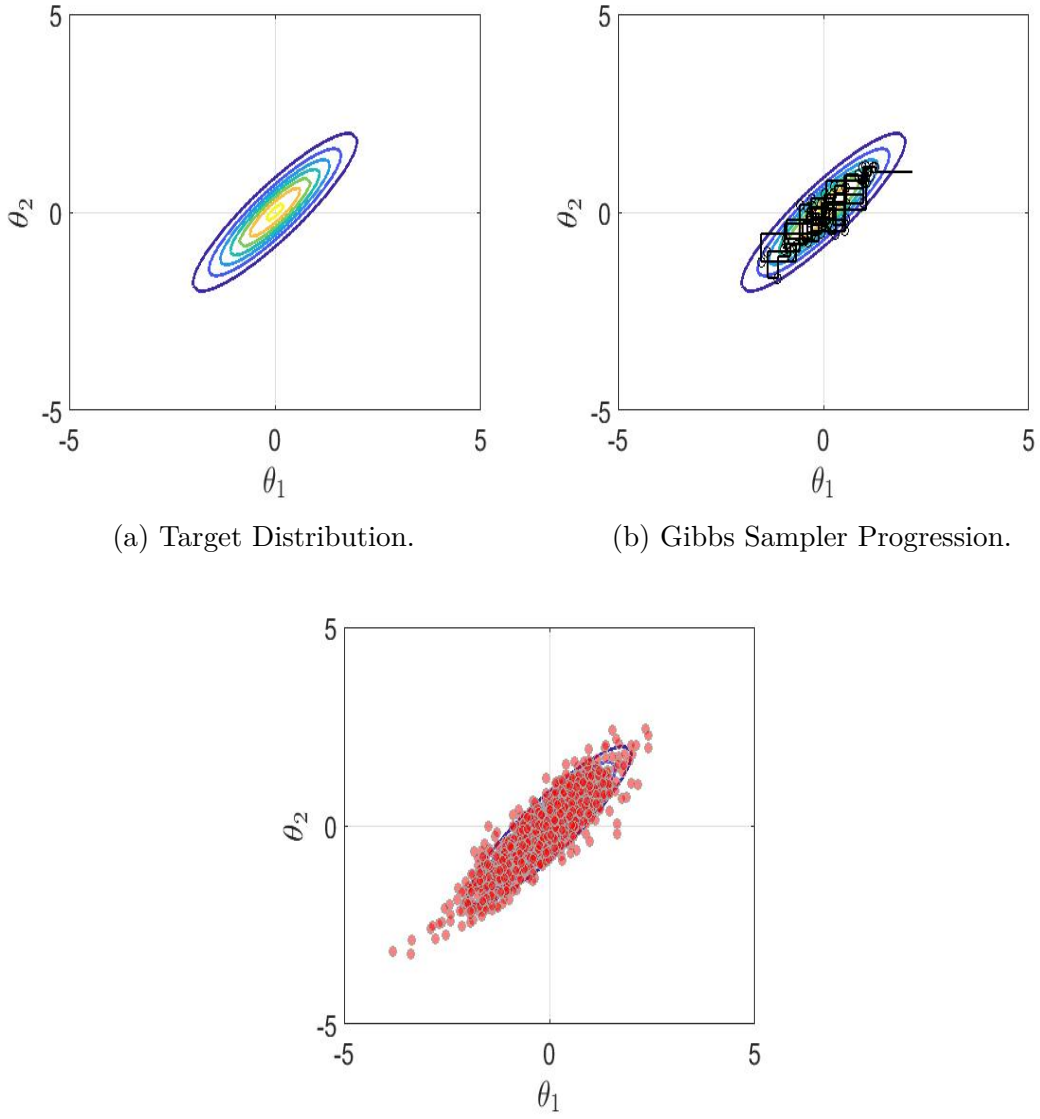


(b) Gibbs Sampler Progression.



(c) Posterior Samples.

Figure 3.3: Illustration of the sampling capabilities of the Gibbs sampler. Panel (a) presents the target distribution while the progression of the Gibbs sampler is highlighted in panel (b). Panel (c) presents the resulting posterior samples.



(a) Target Distribution.

(b) Gibbs Sampler Progression.

(c) Posterior Samples.

Figure 3.4: Illustration of the sampling capabilities of the Gibbs sampler on a problem with highly correlated parameters. Panel (a) presents the target distribution while the progression of the Gibbs sampler is highlighted in panel (b). Panel (c) presents the resulting posterior samples.

3.3.5 Metropolis Hastings

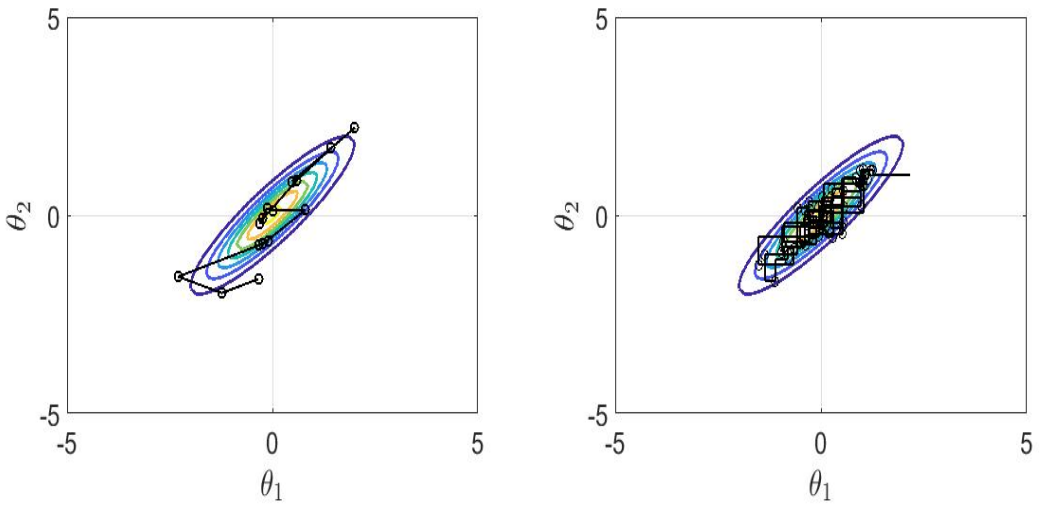
Unlike the Gibbs sampler, the Metropolis Hastings (MH) [105] algorithm allows for more flexibility with respect to the behaviour of the sample update in the parameter space. By simulating a Markov chain with a specified target density as

the stationary distribution, MH generates state $\boldsymbol{\theta}'$ from the previous state $\boldsymbol{\theta}$ as in Algorithm 3 with the introduction of an acceptance/rejection step.

Algorithm 3 Metropolis-Hastings

1: Draw a sample $\boldsymbol{v}$ from a proposal distribution $S(\cdot|\boldsymbol{\theta})$
2: Generate u uniformly with support $[0, 1]$
3: $r(\boldsymbol{v}) = \frac{\pi(\boldsymbol{v})}{\pi(\boldsymbol{\theta})} \cdot \frac{S(\boldsymbol{\theta}|\boldsymbol{v})}{S(\boldsymbol{v}|\boldsymbol{\theta})}$
4: **if** $u > r(\boldsymbol{v})$ **then**
5: $\quad \boldsymbol{\theta}' = \boldsymbol{v}$
6: **else**
7: $\quad \boldsymbol{\theta}' = \boldsymbol{\theta}$
8: **end if**

Consider $\boldsymbol{v}$ to be a sample generated from the proposal distribution $S(\cdot|\boldsymbol{\theta})$. MH either accepts or rejects $\boldsymbol{v}$ to be the updated state $\boldsymbol{\theta}'$ of the Markov chain. The random walk nature of MH in comparison to the sequential updating of Gibbs sampling is apparent in Figure 3.5. For the highly correlated parameter example discussed in Section 3.3.4, the ability of MH to explore a greater proportion of the region of the target distribution in a more efficient manner is evident from the sample progressions. For a further discussion on this topic the reader is referred to [34].



(a) MH Sampler. (b) Gibbs Sampler.

Figure 3.5: Comparison of the sample progression of the Gibbs and MH sampler. Panel (a) presents the random walk nature of the MH algorithm and panel (b) the sequential update of the Gibbs sampler.

To show that MH fulfils detailed balance, it needs to be shown that Eq. 3.15 is satisfied. Consider again Eq. 3.15 and the two possible updates for $\boldsymbol{\theta}'$. If the proposed $\boldsymbol{v}$ is rejected and $\boldsymbol{\theta}' = \boldsymbol{\theta}$ then detailed balance is trivial. In the case of $\boldsymbol{\theta}' \neq \boldsymbol{\theta}$ consider the transition operator,

$$T(\boldsymbol{\theta}'|\boldsymbol{\theta}) = \min\left\{1, \frac{\pi(\boldsymbol{\theta}')}{\pi(\boldsymbol{\theta})} \cdot \frac{S(\boldsymbol{\theta}|\boldsymbol{\theta}')}{S(\boldsymbol{\theta}'|\boldsymbol{\theta})}\right\} \cdot S(\boldsymbol{\theta}'|\boldsymbol{\theta}) \tag{3.18}$$

Correspondingly,

$$T(\boldsymbol{\theta}|\boldsymbol{\theta}') = \min\left\{1, \frac{\pi(\boldsymbol{\theta})}{\pi(\boldsymbol{\theta}')} \cdot \frac{S(\boldsymbol{\theta}'|\boldsymbol{\theta})}{S(\boldsymbol{\theta}|\boldsymbol{\theta}')}\right\} \cdot S(\boldsymbol{\theta}|\boldsymbol{\theta}') \tag{3.19}$$

Using the relation $\min(1, \frac{a}{b})$=$\min(1, \frac{b}{a})\frac{a}{b}$ for all $a, b \in \mathbb{R}^+$

$$T(\boldsymbol{\theta}|\boldsymbol{\theta}') = \min\left\{1, \frac{\pi(\boldsymbol{\theta}')}{\pi(\boldsymbol{\theta})} \cdot \frac{S(\boldsymbol{\theta}|\boldsymbol{\theta}')}{S(\boldsymbol{\theta}'|\boldsymbol{\theta})}\right\} \cdot S(\boldsymbol{\theta}'|\boldsymbol{\theta}) \cdot \frac{\pi(\boldsymbol{\theta})}{\pi(\boldsymbol{\theta}')} \tag{3.20}$$

Rearranging allows for the relationship to be expressed as

$$\min\left\{1, \frac{\pi(\boldsymbol{\theta}')}{\pi(\boldsymbol{\theta})} \cdot \frac{S(\boldsymbol{\theta}|\boldsymbol{\theta}')}{S(\boldsymbol{\theta}'|\boldsymbol{\theta})}\right\} = \frac{T(\boldsymbol{\theta}|\boldsymbol{\theta}')}{S(\boldsymbol{\theta}'|\boldsymbol{\theta})} \cdot \frac{\pi(\boldsymbol{\theta}')}{\pi(\boldsymbol{\theta})}$$

$$T(\boldsymbol{\theta}'|\boldsymbol{\theta})\pi(\boldsymbol{\theta}) = T(\boldsymbol{\theta}|\boldsymbol{\theta}')\pi(\boldsymbol{\theta}') \tag{3.21}$$

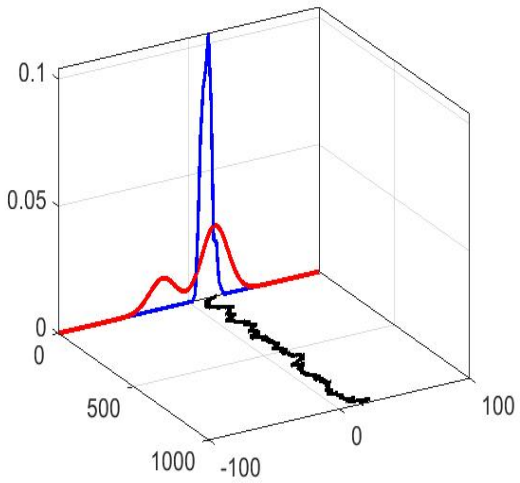
Therefore, any Markov chain generated by MH with $\pi(\cdot)$ as the stationary distribution produces samples from $\pi(\cdot)$. It is worth noting that in rejection sampling proposals are repeatedly made until one is accepted, whereas for MH only one proposal is made, if it is not accepted, the Markov chain of the standard MH algorithm stays in its current state. Let the sample acceptance rate p_A, in Algorithm 3 be defined as the average of $\min[r(\cdot), 1]$ for all proposed samples. In some respects, p_A represents the efficiency of MH sampling [79, 117].

The proposal governs the distribution of the sample and directly influences the transition of the Markov chain from one state to another. Great care must be taken when choosing $S(\cdot)$. MH is valid for any $S(\cdot)$ but appropriate choices would ensure the rapid exploration of $\pi(\cdot)$. One could choose a proposal based on some

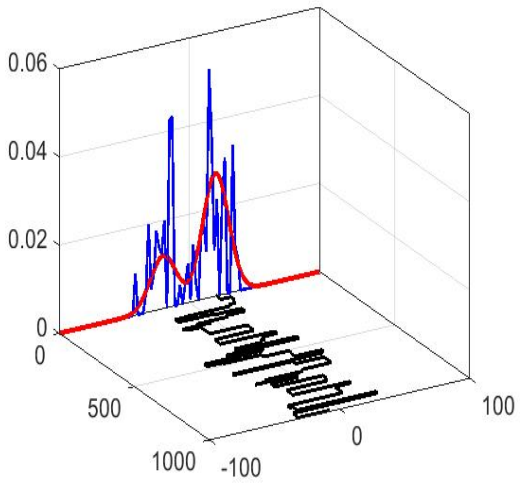
data D, but under MH, using past information for $S(\cdot)$ would result in a chain which was not Markov. One common choice is a Gaussian centred at the current state $\boldsymbol{\theta}$.

The question of sample dependence and how one would control the rate of correlation may also be linked to the proposal. The *spread* of $S(\cdot)$ refers to a measure of dispersion of $S(\cdot)$ e.g. standard deviation or variance. Sample dependence along with the rate of efficient exploration of the parameter space is influenced by the spread. Choosing a value too small will allow for a high p_A but an increased dependence due to the geometric location of the proposed sample with respect to the current one. While a value too large will diminish p_A, producing a greater number of repeated samples which again will result in an increased sample dependency. One strategy is to choose the spread of $S(\cdot)$ to be of the same order as the spread of $\pi(\cdot)$ if this quantity is accessible [12].

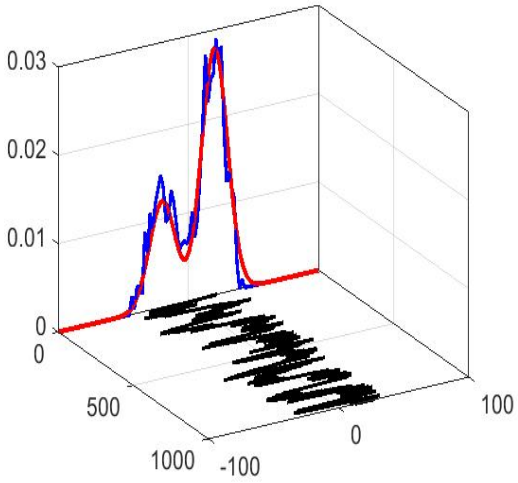
To illustrate the importance of appropriate spread choice, consider a simple one dimensional example for which MH generates samples from a distribution which is a mixture of two Gaussians, each with a standard deviation of 100 and means of -40 and 0 respectively. Investigating the influence of the proposals standard deviation, three different values were chosen, $\sigma = \{1, 500, 8\}$. Figure 3.6 presents each scenario with the sample progressions (black), target distribution (red) and the MH approximations (blue). For the case of $\sigma = 1$ (Figure 3.6 (a)), only one of the modes is explored. This may suggest that as previously stated, the proposed value is too small. The case of $\sigma = 500$ (Figure 3.6 (b)), reveals that samples are remaining in the same state for many successive iterations, which increases the sample dependence. Lastly, in the case of $\sigma = 8$ (Figure 3.6 (c)), an appropriate balance between exploration and repeated samples appears in the sample progression. This is highlighted in the approximation to the posterior density which accurately captures the multi model nature of the distribution. As such, for this example an appropriate choice of standard deviation for the proposal distribution is $\sigma = 8$.



(a) $\sigma = 1$



(b) $\sigma = 500$



(c) $\sigma = 8$

Figure 3.6: Influence of the proposal distributions standard deviation on the MH sampling scheme. The target distribution is outlined in red, distribution of samples in blue and sample progression in black. In the case of $\sigma = 1$ (a), the sampler becomes trapped at one of the modes while for $\sigma = 500$ (b) many repeated samples are generated during the iterations. Using a spread of $\sigma = 8$ (c) results in an accurate approximation to the posterior distribution while also striking a balance between exploration and sample efficiency. The plots in this example were achieved using a MATLAB® toolbox given in [157].

When the proposal distribution is symmetric, $S(\boldsymbol{\theta}'|\boldsymbol{v}) = S(\boldsymbol{v}|\boldsymbol{\theta}')$, $r(\boldsymbol{v})$ in Algorithm 3 is simplified to $r(\boldsymbol{v}) = \pi(\boldsymbol{v})/\pi(\boldsymbol{\theta}')$. This is the Metropolis algorithm of which MH is a generalization [166].

3.4 Advanced MCMC Methods

The MCMC algorithms described in Section 3.3.3 are designed to generate samples from the posterior distribution by letting it equal the stationary distribution of the Markov chain. As previously discussed, such MCMC methods in there standard form run into a series of issues such as: potential inefficient exploration of the parameter space, sampling inefficiency in high dimensions (e.g. MH), becoming trapped at local modes in a multi-modal distribution and the lack of an estimate of the model evidence. As large body of literature exists to address each of these drawbacks. this section provides an overview of various advanced MCMC methods. In line with the notation used in previous sections allow for the unnormalized density to be given by $\pi(\boldsymbol{\theta}) = Q(\boldsymbol{\theta})L(D|\boldsymbol{\theta})$.

3.4.1 Adaptive MCMC

As outlined in Section 3.3.5, the parameters of the MCMC proposal distribution directly influence the efficiency of the sampler. Tuning of associated parameters such as proposal variances is crucial to achieve efficient mixing, but can also be very difficult. As the Markov chain progresses, adaptive MCMC algorithms attempt to deal with this problem by automatically learning better proposal distribution parameters. In a pioneering paper by [99] an Adaptive Metropolis (AM) algorithm is introduced. AM proposes to define a Gaussian proposal distribution with a covariance matrix calibrated using the sample path of the MCMC chain. A crucial point of AM is how the covariance of the proposal depends on the history of the chain. After an initial non-adaption period, the Gaussian proposal is centred at the current sample $\boldsymbol{\theta}^i$ of the Markov chain with a covariance $\boldsymbol{\Sigma}$ of:

$$\boldsymbol{\Sigma} = s_d Cov(\boldsymbol{\theta}^0, ..., \boldsymbol{\theta}^{i-1}) + s_d \epsilon I_d \tag{3.22}$$

where s_d is a parameter that depends on the dimensionality d, ϵ a constant chosen very small, I_d the d-dimensional identity matrix and Cov denoting the sample covariance. The role of ϵ is to ensure that Eq. 3.22 does not become singular while

setting the scaling parameter as $s_d = \frac{2.4^2}{d}$ has been shown to optimize chain mixing when the target distribution is Gaussian [82]. As Eq. 3.22 may initially be a poor approximation for the optimal proposal covariance, the chain may be require additional states [191] before adapting the covariance structure. As such, an initial non-adaption period t is defined. Let $\mathbf{\Sigma}_i$ denote the updated covariance structure of the i^{th} sample and $\mathbf{\Sigma}_0$ a covariance matrix chosen prior to the Markov chain being initialized. At the i^{th} state of the Markov chain, the proposal covariance $\mathbf{\Sigma}$ is chosen as follows:

$$\mathbf{\Sigma} = \begin{cases} \Sigma_0, & i \leq t \\ \Sigma_i, & i > t \end{cases} \tag{3.23}$$

Therefore, the covariance is only updated once the number of generated states of the Markov chain is greater than t. A similar Adaptive Proposal (AP) approach has been proposed in [98] using the same concept of updating the covariance structure based on the progression of the Markov chain. AM was later extended to include component wise moves suitable for high dimensional problems [100] while also being combined with delayed rejection to continuously alternate between larger and smaller steps in the Markov chain [97]. However, as highlighted in [220], the covariance adaption strategy utilized in AM and AP works well in relatively simple problems but can become inefficient and unreliable when confronted with posteriors with heavy tails or complex topologies. Along another line of thought, the seminal papers of Roberts et al. [187, 188] have resulted in a spate of literature that uses information from the sample acceptance rate to tune the spread of the proposal distribution. In these papers, a theoretical and exhaustive investigation is presented concerning the estimation of the correlation structure of the target distribution and incorporation of this information when choosing the proposal. By defining optimal scaling as a function of the sample acceptance rate, appropriate scaling that minimizes the first order autocorrelation of the sampler achieves an optimal acceptance rate proved to be 0.234. This coincides with the rate first suggested by [82]. It is worth noting that this was proved under the assumption that $d \to \infty$. For one dimensional increments, numerical studies in [188] reveal that the optimal acceptance rate is approximately 0.44. This optimal reference value has been used for adaptive scaling of the proposal for component wise MH methods [172]. Using the findings of [187, 188], Atchade and Rosenthal [9] adopt the stochastic approximation algorithm by Robbins and Monro [184] to

automatically find the spread of the proposal. The resulting sampler, named Adaptive Random Walk Metropolis (ARWM) interprets the task of finding an optimal spread as a root finding problem. ARWM iteratively updates the spread such that the asymptotic rate of acceptance is approximately 0.234. The Robbins and Monro method is a well known recursive algorithm of the form

$$\boldsymbol{\theta}^{i+1} = \boldsymbol{\theta}^{i} + \gamma^{i}(h(\boldsymbol{\theta}^{i}) + \epsilon^{i+1}) \tag{3.24}$$

where γ is a sequence of positive step sizes subject to constraints highlighted in [184]. This approach is typically used to solve the equation $h(\boldsymbol{\theta}) = 0$ when the function h is unknown or difficult to compute but can be estimated with a noise ϵ. For ARWM, let the covariance of the proposal distribution be defined by $\boldsymbol{\Sigma} = \sigma^2 I_d$, where like before, I_d is the d-dimension identity matrix. ARWM interprets the task of updating σ as follows:

$$\sigma^{i+1} = \sigma^{i} + \gamma^{i}(r(\boldsymbol{v}^{i+1}) - 0.234) \tag{3.25}$$

with $r(\cdot)$ denoting the sample acceptance ratio from MH in Algorithm 3 and $\boldsymbol{v}$ the proposed value for the parameter update of the Markov chain in Algorithm 3. In order to approximate the optimal spread value, it is assumed that its true value lives in the set $A_\sigma = \{\sigma : \epsilon \leq \sigma \leq a\}$. This assumption functions on the premise that ϵ is chosen to be very small and a sufficiently large. The generation of samples in ARWM coincides with those in MH with σ updated by Eq. 3.25 after the proposed sample of the Markov chain has either been accepted or rejected. In the event that $\sigma^{i+1} < \epsilon$, σ^{i+1} is assigned a default value of ϵ and accordingly if $\sigma^{i+1} > a$, σ^{i+1} is assigned a default value of a. The purpose of keeping σ inside the set A_σ is to avoid degeneracy of the algorithm. However, the method will be unable to find the optimal value of σ if the set is misspecified. Similar stochastic approximation approaches have been utilized to improve the scalability of MCMC methods to large data sets [223, 52]. Alternative adaptive approaches in the literature include the use of multiple chains to explore the parameter space for appropriate proposal choices [51, 87] and regeneration algorithms that seek to approximate the proposal using a mixture of components [88, 159, 192]. In practice, these methods either require significant memory storage or complicated fittings of high dimensional mixture distributions. Although much focus has been placed on improving the efficiency of MCMC through adaptively tuning the

proposal distribution, [6, 74] highlight that adaptation needs to be done carefully to ensure that sampling is from the correct ergodic distribution. Further discussion on this can be found in [5, 186].

3.4.2 Auxiliary MCMC

Auxiliary MCMC methods concern the introduction of an additional variable which helps improve the efficiency of the exploration of the sample space. It is often easer to sample from an augmented probability distribution than the original distribution of the parameter of interest. This section discusses two such frameworks which do not require a proposal distribution to be specified. Hamiltonian Monte Carlo (HMC) [70] is a MCMC sampler that borrows theories from physics to incorporate information about the gradient of the target distribution to improve mixing. Hamiltonian dynamics is one way that physicists describe how objects move throughout a system by describing an object's motion in terms of its location and momentum at some time. HMC introduces an additional random vector $\boldsymbol{v} \in \mathbb{R}^d$ referred to as a momentum variable. The joint distribution by HMC is defined by means of the energy function:

$$\begin{aligned} H(\boldsymbol{\theta}, \boldsymbol{v}) &= -\log\pi(\boldsymbol{\theta}, \boldsymbol{v}) \\ &= -\log\pi(\boldsymbol{v}|\boldsymbol{\theta}) - \log\pi(\boldsymbol{\theta}) \\ &= K(\boldsymbol{\theta}, \boldsymbol{v}) + V(\boldsymbol{\theta}) \end{aligned} \tag{3.26}$$

where $K(\cdot)$ and $V(\cdot)$ are interpreted as the kinetic and potential energies of a Hamiltonian system. HMC alternates between generating random momentum variables $\boldsymbol{v}$ and simulating the Hamiltonian dynamics of the system by solving

$$\begin{aligned} \frac{d\boldsymbol{\theta}}{dt} &= \frac{\partial H}{\partial \boldsymbol{v}} \\ \frac{d\boldsymbol{v}}{dt} &= -\frac{\partial H}{\partial \boldsymbol{\theta}} \end{aligned} \tag{3.27}$$

where t represents some point in time of a continuous time process. Conceptually, HMC utilizes Hamiltonian dynamics as a proposal function for a Markov chain in order to explore the target distribution. At each time step, the momentum and parameter variables are updated with a Metropolis ratio computed using the

proposed values. This process is continued iteratively until termination. In general Eq. 3.27 does not usually have an analytical expression and as such there exists no general methodology for producing this continuous time process. However, there does exist a discretization technique that produces a Markov chain and is well suited to Hamiltonian equations as it preserves the stationary distribution [52]. The *leap-frog* method updates the parameters at different time steps with the length of the time steps being a crucial ingredient for the success of the sampler. Suitable time step sizes are required for accurate simulation of the Hamiltonian dynamics [70]. A very small stepsize leads to random walk behaviour. This causes the Markov chain to evolve slowly, thus having highly correlated samples which do not explore the support efficiently. On the other hand, a poorly chosen stepsize leads to inaccurate simulation of the Hamiltonian dynamics, thus introducing numerical errors in the proposal for the Markov chain. Appropriate choices for the step size has been studied in the literature with the state of the art being the No-U-Turn-Sampler (NUTS) variant [106]. Additionally, the requirement of the calculation of the target distributions derivative hinders its general applicability. As computing this derivative is a non-trivial task, stochastic approximation methods have been suggested to address this issue [52]. Implementations of both HMC and NUTS are readily available in the probabilistic programming language STAN [48]. Aside from HMC, slice sampling [163] is an alternative auxiliary MCMC sampler. The basic idea of slice sampling is to augment the sample space with the introduction of an auxiliary variable $v \in \boldsymbol{V} \subseteq \mathbb{R}$. Allow for the joint probability distribution $P(v, \boldsymbol{\theta})$ to be expressed as follows:

$$P(v, \boldsymbol{\theta}) = \begin{cases} \frac{1}{Z}, & \text{if} \quad 0 < v < \pi(\boldsymbol{\theta}) \\ 0, & \text{Otherwise} \end{cases} \tag{3.28}$$

where $Z = \int \pi(\boldsymbol{\theta}) d\boldsymbol{\theta}$ and $\pi(\boldsymbol{\theta})$ is the target distribution of $\boldsymbol{\theta}$. Slice sampling firstly generates v from the conditional distribution specified as a uniform on the interval $(0, \pi(\boldsymbol{\theta}))$.

$$P(v|\boldsymbol{\theta}) = \boldsymbol{V}_{[0,\pi(\boldsymbol{\theta})]}(v) \tag{3.29}$$

Following this, the conditional distribution with support $A_v = \{\boldsymbol{\theta} : v < \pi(\boldsymbol{\theta})\}$ is addressed as:

$$P(\boldsymbol{\theta}|v) = \boldsymbol{V}_{A_v}(\boldsymbol{\theta}) \tag{3.30}$$

Since the marginal satisfies $\int P(v, \boldsymbol{\theta})dv = \pi(\boldsymbol{\theta})$, samples from the target distribution can be recovered by disregarding the auxiliary component of the joint samples. Therefore, to sample for $\boldsymbol{\theta}$ one can jointly sample for $(v, \boldsymbol{\theta})$ and marginalize v. In practice, generating independent draws uniformly over A may be a difficult task and as such an iterative Gibbs sampler may be used to draw pairs of samples from the joint density. An advantage of slice sampling over Gibbs however, is that it does not require the specification of the full conditionals [158].

3.4.3 Annealing Methods

The influence of the proposal spread on the MH algorithms ability to populate multiple modes was discussed in Section 3.3.5. Originally proposed as an optimization method, simulated annealing [124] avoids becoming trapped at local modes of the parameter space by expressing the target distribution as a sequence of intermediate distributions. When specifically sampling from the posterior, simulated annealing involves targeting

$$\pi_{\beta_j}(\boldsymbol{\theta}) = L(D|\boldsymbol{\theta})^{\beta_j} Q(\boldsymbol{\theta}) \tag{3.31}$$

where β_j is referred to as a cooling schedule such that $0 = \beta_0 < \beta_1 < ... < \beta_J = 1$, $L(D|\boldsymbol{\theta})$ the likelihood function and $Q(\boldsymbol{\theta})$ the prior distribution. For $\beta_j = 0$, the sampling process is akin to drawing from the prior distribution. For an increase in j, the influence of the likelihood function on the target distribution becomes greater and greater before the distribution being sampled is the posterior for $\beta_j = 1$. The sequence of cooling values directly influences the sampling process. A cooling schedule which increases too quickly may result in the Markov chain becoming trapped in local modes while a slow cooling schedule will incur unnecessarily large computational costs. The selection of the cooling schedule has resulted in many algorithmic developments based on the simulated annealing process. One such sampler, Transitional Markov Chain Monte Carlo (TMCMC) [54] proposes to select β_{j+1} based on β_j. This is done by ensuring the change

between two adjacent intermediate PDF's is made small such that it is possible to smoothly transition the samples. Given N samples $\{\boldsymbol{\theta}_j^i\}_{i=1}^N \sim \pi_{\beta_j}(\boldsymbol{\theta})$, the sample set $\{\boldsymbol{\theta}_{j+1}^i\}_{i=1}^N$ is obtained by resampling $\{\boldsymbol{\theta}_j^i\}_{i=1}^N$. This resampling is done by a sequence of importance weights.

$$w_j^i = L(D|\boldsymbol{\theta}_j^i)^{\beta_{j+1}-\beta_j} \qquad i = 1, ..., N \tag{3.32}$$

The resampling can be easily done by setting the resampled sample to be $\boldsymbol{\theta}_j^i$ with probability

$$\hat{w}_j^i = \frac{w_j^i}{\sum_{i=1}^N w_j^i} \tag{3.33}$$

with $\hat{w}_j^i$ being the normalized importance weight. To avoid the degeneracy problem [63] of few highly weighted samples dominating the set of samples, a Markov chain uses each resampled $\boldsymbol{\theta}_j^i$ as a seed. TMCMC utilizes the MH algorithm with each chain having $\pi_{\beta_{j+1}(\boldsymbol{\theta})}$ as the target distribution. To ensure a smooth transition between the intermediate PDFs, [54] suggest choosing β_j adaptively such that the coefficient of variation (c.o.v) of the importance weights at a given intermediate PDF is equal to 100%. Additionally, TMCMC simulates an estimate for the model evidence through the importance weights by:

$$P_D \approx \prod_{j=0}^{J} (\frac{1}{N} \sum_{i=1}^{N} w_j^i) \tag{3.34}$$

Although TMCMC has proved popular amongst the engineering community [96], it has been argued that TMCMC requires a burn in period as it does not ensure that the initial seeds of the Markov chains equal the target stationary distribution [53]. Another approach to combining MCMC and annealing is to run multiple chains in parallel at different temperatures. Metropolis-Coupled MCMC (MC^3) [210, 85] also referred to as replica exchange or parallel tempering [71], simulates $J+1$ independent Markov chains with each chain having a different temperature. MC^3 proceeds by running one chain at each of the $J+1$ values of β. Again a sequence of tempered distributions $\pi_{\beta_j}(\boldsymbol{\theta})$ with $0 \leq \beta_J < \beta_{J-1} < ... < \beta_0 = 1$ for $j = 0, ..., J$ is defined. The product of the resulting $J+1$ sampled distributions

may be given by:

$$\pi(\boldsymbol{\theta}) = \prod_{j=0}^{J} \pi_{\beta_j}(\boldsymbol{\theta}^j) \tag{3.35}$$

The β_0 chain has the stationary distribution $\pi_{\beta_0}(\boldsymbol{\theta}^0) = \pi(\boldsymbol{\theta})$. Therefore, after a long run of samples, $\boldsymbol{\theta}^0$ should approximately be drawn from the target distribution. The idea is that the chains corresponding to smaller β and have higher temperatures can mix more easily. This mixing information can then be 'transferred' to the β_0 chain. This allows for an increase in the speed of convergence of $\boldsymbol{\theta}^0$ to the distribution $\pi(\boldsymbol{\theta})$. MC^3 exhibits two different types of dynamics during chain progression. On some iterations, a *within temperature move* is attempted by updating each $\boldsymbol{\theta}^j$ by some type of MCMC update. This update for example could be in the form of MH with $\pi_{\beta_j}(\boldsymbol{\theta}^j)$ being the stationary distribution. On other iterations a *temperature swap* is attempted. This involves choosing two different β values, say β_k and β_l and then proposing to swap their respective chain values, i.e. to interchange the current values of $\boldsymbol{\theta}^k$ and $\boldsymbol{\theta}^l$. This proposed swap is accepted according to a typical Metropolis step with an acceptance ratio of:

$$\min\left(1, \frac{\pi_{\beta_l}(\boldsymbol{\theta}^k)\pi_{\beta_k}(\boldsymbol{\theta}^l)}{\pi_{\beta_l}(\boldsymbol{\theta}^l)\pi_{\beta_k}(\boldsymbol{\theta}^k)}\right) \tag{3.36}$$

Otherwise it is rejected and the values of $\boldsymbol{\theta}^k$ and $\boldsymbol{\theta}^l$ remain unchanged. Successfully swapping states allows a chain that is otherwise stuck on one mode of the parameter space to explore other modes. In general the rejected values are discarded, however some works have attempted to make additional use of them [64]. Though MC^3 allows good mixing with multi-modal target distributions, the mixing properties of the sampler depend strongly on the temperature schedule [150]. Both [125] and [127] link the mean acceptance rate of temperature swaps to the choice of temperature schedule. Similarly, coinciding with the work discussed in Section 3.4.1, [8] later proposed to space the temperatures such that the proportion of temperature swaps is approximately 0.234. This is extended further in [150] by adapting the temperature continuously during the simulation. Other suggestions for setting the temperature schedule can be found in [71]. Aside from the choice of temperature schedule, MC^3 can suffer in terms of execution time if the number of chains bing simulated is very large. A parallel MC^3 algorithm [3] retains the

ability of MC^3 to populate multiple modes while maintaining a fast execution time. The framework makes use of two widely applied parallel programming models in message passing and shared memory. Empirical findings in [3] claim to indicate a linear speed improvement over MC^3. Alternative parallelization of MCMC in non-annealing based methods include the Multiple-try MH sampler [140] and the generalized MH sampler [46]. It is worth noting that a coherent way of dealing with annealing in MCMC simulations was independently discovered under the name simulated tempering [145]. Simulated tempering interprets the temperature schedule itself as a random variable while unlike simulated annealing allowing its value to either increase or decrease. A Metropolis step to compute the temperature schedule is introduced during simulation by augmenting the parameter space. However, the extra computational costs of the algorithm stemming from dynamically selecting the temperature value is dependent on the fraction of time spent at each temperature level of the intermediate PDF's. For more information the reader is referred to [145].

3.4.4 Model Evidence Estimation

Aside from TMCMC and Simulated Tempering another suitable method for model evidence estimation is nested sampling [204]. Nested sampling is predominantly concerned with the estimation of the model evidence as in Eq. 3.2. As a by-product, the method produces a set of weighted samples from the posterior. In nested sampling, define X:

$$\begin{aligned} X(\lambda) &= p_{\boldsymbol{\Theta}}(L(D|\boldsymbol{\theta}) \geq \lambda) \\ &= \textstyle\int_{L(D|\boldsymbol{\theta})\geq\lambda} Q(\boldsymbol{\theta}) d\boldsymbol{\theta} \end{aligned} \tag{3.37}$$

as the cumulant prior mass covering all likelihood values greater that λ. As λ increases, the enclosed mass X decreases from 1 to 0. Let $\mathcal{L}(X)$ be the inverse function of $X(\lambda)$. It is assumed that there exists a function $\mathcal{L}(X)$ such that if one is given X, $\mathcal{L}(X(\lambda)) \equiv \lambda$. Adopting this expression for the inverse function allows for the model evidence to become a one dimensional integral over the unit range.

$$P_D = \int_0^1 \mathcal{L}(X) dX \tag{3.38}$$

The Nested sampling algorithm begins with N samples $\{\boldsymbol{\theta}^i\}_{i=1}^N$ from the prior $Q(\boldsymbol{\theta})$. The sample $\boldsymbol{\theta}^j$ which resulted in the smallest likelihood value $\lambda_{\min}$ is identified while the corresponding value of $X(\lambda_{\min})$ is estimated. The sample $\boldsymbol{\theta}^j$ is replaced by a sample drawn from $Q(\boldsymbol{\theta})$ subject to satisfying $L(D|\boldsymbol{\theta}) \geq \lambda_{\min}$. In practice, the target distribution conditional on $L(D|\boldsymbol{\theta}) \geq \lambda_{\min}$ may be sampled from via MCMC sampling. This process is repeated until a series of X values and the corresponding $\mathcal{L}(X)$ have been obtained. Similar to many MCMC approaches, there is no clear stopping condition for nested sampling. Skilling [204] defined a possible termination after completing a pre-specified number of steps or when the largest current likelihood value would not increase the estimate evidence by more than some small threshold ϵ. However this choice of ϵ will directly impact the level of bias within the estimated evidence. Aside from nested sampling, other notable methods capable of estimating the model evidence include Reversible Jump MCMC (RJMCMC) [95] and Asymptotically Independent Markov Sampling (AIMS) [22]. However, nested sampling is included in this overview as the Bayesian updating methods developed during this thesis are also dependent on nested architectures.

3.5 Chapter Summary

This chapter has introduced Bayesian inference. The theory behind Bayesian inference along with the importance it holds in numerical simulation has been discussed. Bayesian inference provides a solid mathematical framework for the updating of probability models in the evidence of data. The posterior distribution allows for the quantification of updated beliefs regarding model parameters, competing models in the presence of data to be compared against one another and for predictions to be made. The mathematical expression however, is rendered analytically intractable due to the presence of a high dimensional integral. A selection of methods which offer a solution by directly sampling from the posterior have been discussed.

Rejection sampling allows for the generation of independent and identically distributed samples from a posterior distribution. However, it is known to become highly inefficient in the presence of a large number of uncertain variables while an input parameter which directly influences the distribution of the samples is also required to be chosen. These shortcomings provide the basis of an alternative Bayesian updating framework presented in Chapter 4. Markov Chain Monte Carlo

(MCMC) allows for the samples generated as a sequence of a Markov chain to be distributed according to a specified probability distribution. Although widely considered a benchmark method in terms of sampling accuracy, difficulties lie in identifying when samples have reached their desired stationary distribution. The Metropolis Hastings (MH) algorithm may be viewed as a pillar of MCMC but as discussed in Section 3.3.5 becomes inefficient in high dimensions along with struggling to draw samples from distributions with multiple modes. Section 3.4 provided an overview of advanced MCMC methods which seek to address these issues.

A framework stemming from the area of reliability analysis is presented in Chapter 4 as an alternative solution for Bayesian inference problems. This MCMC based technique allows for the efficient generation of samples in the presence of many uncertain variables while guaranteeing the correct target posterior distribution has been reached. Additionally, the model evidence is automatically computed as a by product of simulation, allowing for model class selection to be implemented.

Chapter 4

Bayesian Updating with Structural reliability methods (BUS)

In the previous chapter, the importance of computing the posterior distribution along with the difficulty in doing so was discussed. Two methods for solving the Bayesian inference problem, in rejection sampling and Metropolis Hastings, were reviewed while their respective shortcomings highlighted. This chapter introduces an advanced MCMC based method which solves the inefficiency of rejection sampling while also addressing the issues of standard MCMC. Originally stemming from an area of engineering referred to as reliability analysis, the framework allows for problems with multi modal and high dimensional inferential tasks in Bayesian inference to be efficiently addressed through rare event simulation. Additionally, it also holds the capabilities of automatically computing the model evidence at no additional computational cost.

4.1 Reliability Analysis

Reliability analysis aims to evaluate the probability of failure of a system of interest. Traditionally, these are representations or abstractions of physical systems. In engineering, this could mean an aircraft (or some of its components), a structure such as a bridge amongst many other examples. Failure in a system occurs when a state of unacceptable performance has been reached, which often stems from the demand exceeding the systems capacity. The response of a system can be

represented by a limit state function $g(\boldsymbol{\theta})$. Let the limit state function represent the relationship between $\boldsymbol{\theta}$ and the output response. It contains all available information of a systems behaviour such as material properties, loads etc. The failure domain F is defined as

$$F = \{\boldsymbol{\theta} \in \mathbb{R}^d : g(\boldsymbol{\theta}) > b\} \tag{4.1}$$

The failure domain is populated by the values of $\boldsymbol{\theta}$ such that the output of $g(\boldsymbol{\theta})$ exceeds a critical threshold b. Let $\boldsymbol{\theta} = (\theta_1, ..., \theta_d)$ be a vector of uncertain parameters distributed as $\pi(\boldsymbol{\theta})$. The probability of failure p_F can be expressed by

$$p_F = p(\boldsymbol{\theta} \in F) = \int_F \pi(\boldsymbol{\theta}) d\boldsymbol{\theta} \tag{4.2}$$

The dimensionality of $\boldsymbol{\theta}$ renders the numerical treatment of the integral in Eq. 4.2 difficult. In order to understand why it is difficult to solve this, consider *Direct Monte Carlo* (DMC). Through the introduction of an indicator function, Eq. 4.2 is given by

$$p_F = \int I[g(\boldsymbol{\theta}) > b] \pi(\boldsymbol{\theta}) d\boldsymbol{\theta} \tag{4.3}$$

where $I[\cdot]$ denotes the indicator function defined as

$$I[\boldsymbol{\theta}] = \begin{cases} 1 & \text{if } g(\boldsymbol{\theta}) > b \\ 0 & \text{Otherwise} \end{cases} \tag{4.4}$$

The idea behind DMC is a straight forward application of the *law of large numbers* that states if the values of $\boldsymbol{\theta}$ are i.i.d samples from the PDF $\pi(\boldsymbol{\theta})$, then the empirical average converges to the true value of the target quantity as N goes to infinity. Using this property, DMC approximates the integral in Eq. 4.2 as a finite sum of N samples.

$$\hat{p}_F^{DMC} = \frac{1}{N} \sum_{i=1}^{N} I[\boldsymbol{\theta}_i] \tag{4.5}$$

This estimate of p_F is determined purely by the output of $I[\cdot]$ by identifying the number of failure occurrences out of the total number of samples. Given the stochastic nature of the DMC estimator, in reliability analysis a standard measure

of accuracy of the estimator is the *coefficient of variation* (c.o.v), $\delta(\hat{p}_F)$ which is defined as the ratio of the standard deviation to the expected value of $\hat{p}_F$. With the c.o.v of the DMC estimate given by

$$\delta(\hat{p}_F^{DMC}) = \sqrt{\frac{1 - \hat{p}_F^{DMC}}{N \cdot \hat{p}_F^{DMC}}} \tag{4.6}$$

Eq. 4.6 reveals that DMC depends solely on $\hat{p}_F^{DMC}$ and N, whilst being independent of the parameter input space. As such, the estimators accuracy does not suffer from the so called *curse of dimensionality*. This term was first introduced by Dr. Richard Bellman [25] to describe phenomena that arise when analysing data in high dimensional spaces that do not occur in low dimensional settings. In the case of small failure probabilities, DMC requires a large number of limit state function evaluations which typically rely on the computation of a cost intensive computer model such as a finite element analysis. Eq. 4.6 illustrates that in this case, the number of N samples required to achieve an acceptable level of accuracy is inversely proportional to p_F and therefore very large. As such, DMC is not a feasible solution in terms of computational demands for these types of problems. This deficiency of DMC has motivated research to develop advanced simulation algorithms capable of efficiently estimating small failure probabilities representing *rare events*. It is worth noting that, even though the computational costs associated with DMC are restrictive for many problems, it is a very robust method and is often used as a benchmark for other reliability methods [234].

4.1.1 Transformation of Input Variables

Many reliability methods choose to express Eq. 4.2 in terms of a vector $\boldsymbol{e} \in \mathbb{R}^d$ distributed according to a standard Normal distribution. The coefficients of $\boldsymbol{e}$ are assumed to be independent, in contrast to the possibly dependent parameter vector $\boldsymbol{\theta}$. This assumption however, is not a limitation, since in simulation one may start from independent parameters to generate dependent parameters [234]. There exists a transformation Γ^{-1} which maps the standard normal $\boldsymbol{e}$ to the physical parameter space $\boldsymbol{\theta}$. Consider the simple case of $\boldsymbol{\theta} \sim \mathcal{N}(\cdot|\mu_i, \sigma_i^2)$, with μ_i and σ_i^2 denoting the mean and variance of $\boldsymbol{\theta}_i$. The necessary pre processing step

is standardization

$$\boldsymbol{e}_i = \frac{\boldsymbol{\theta}_i - \mu_i}{\sigma_i^2} \tag{4.7}$$

In terms of reliability analysis the subsequent transformation of the limit state function $g(\boldsymbol{e}) = g(\Gamma^{-1}(\boldsymbol{\theta}))$ allows for the failure probability to be given by

$$p_F = \int I[g(\boldsymbol{e}) > b]\pi(\boldsymbol{e})d\boldsymbol{e} \tag{4.8}$$

where $\pi(\boldsymbol{e})$ denotes the standard normal PDF. In practice, a transformation can be performed in several ways depending on the available information. Two common approaches for parameter transformation are the *Rosenblatt* [190] and *Nataf* transformations [66].

4.2 Subset Simulation

In well designed and complex structures, the likelihood of a rare event occurring is extremely small, meaning rare event simulation techniques which alleviate the shortcomings of DMC are required. Subset Simulation (SuS) [11] offers an efficient solution to the engineering reliability problem by expressing F as a series of intermediate failure events. SuS approximates Eq. 4.2 by generating a sequence of random samples of $\boldsymbol{\theta}$ conditional on increasingly rare failure events $F = F_m \subseteq ... \subseteq F_2 \subseteq F_1$. Each intermediate failure event in the sequence is defined as

$$F_i = \{g(\boldsymbol{\theta}) > b_i\} \tag{4.9}$$

where $\{b_i\}_{i=1}^m$ is an increasing sequence of threshold values adaptively chosen during a simulation run, $g(\cdot)$ representative of the limit state function and m the number of intermediate failure events. In practical cases it is difficult to make a rational choice of the b_i values in advance, so the b_i are chosen adaptively such that $b_1 < ... < b_{m-1} < b_m = b$, with b representing the target critical threshold [233]. By the product rule, SuS approximates p_F by exploiting the conditional dependence on intermediate failure levels

$$p_F \approx \hat{p}_F = p(F_0) \prod_{i=1}^{m} p(F_i|F_{i-1}) \tag{4.10}$$

with F_0 being a rather frequent event and each F_i progressively more rare. Choosing $p(F_i|F_{i-1})$ in the correct manner, allows p_F to be calculated through simulation. Hence, the original rare event problem is broken down into a series of nested failure events. Instead of solving a single reliability problem, a series of intermediate reliability problems are solved and combined analytically to provide an estimate for p_F. The algorithm initializes simulation in the following manner.

Let $p_0 \in [0, 1]$ be the level probability which in essence governs how many intermediate failure thresholds are required to reach the failure domain F. A prudent choice of level probability is required with the optimal value shown to satisfy $p_0 \in [0.1, 0.3]$ [235]. Let $N \in \mathbb{N}$ be the total number of generated samples at each level. Beginning at $level_0$, the algorithm probes the input space generating N i.i.d samples $\{\boldsymbol{\theta}_0^i\}_{i=1}^N$ by DMC from the prior PDF. Next, the limit state function $g(\boldsymbol{\theta})$ is evaluated for all $\{\boldsymbol{\theta}_0^i\}_{i=1}^N$. The limit state function evaluations $\{g_0^i\}_{i=1}^N$ and $\{\boldsymbol{\theta}_0^i\}_{i=1}^N$ are placed in descending order with respect to $\{g_0^i\}_{i=1}^N$. The number of failure samples $n_F(0)$ is identified for $level_0$ as those for which $\{g_0^i\}_{i=1}^N > b$ is true. Here b represents the critical threshold corresponding to the final failure probability and is given as an input into SuS. If $n_F(0)/N > p_0$, simulation is terminated and SuS reduces to DMC. For this reason, it has been found that SuS is best suited to rare events with $p_F < 10^{-3}$ [12]. If $n_F(0)/N < p_0$, the algorithm proceeds to $level_1$. At $level_1$, an intermediate failure threshold b_1 is computed such that $nc = p_0 \cdot N$ values of $\{g_0^i\}_{i=1}^N$ exceed its value. The purpose of this is to identify the samples from $level_0$ to be used as seeds to populate F_1. The resulting sample set chosen based on $\{g_0^i\}_{i=1}^{nc}$ is $\{\boldsymbol{\theta}_0^i\}_{i=1}^{nc}$. Each sample in this set is used as a seed for an individual Markov chain to populate F_1 through a specialized component-wise MH algorithm as outlined in Algorithm 5. This sampler is discussed in detail in Section 4.2.1. Having the seeds inside the target region F_1, allows one to discard any burn in period usually required in MCMC simulations to grow a single Markov chain. Each of the nc independent Markov chains contains $ns = p_0^{-1}$ states including the initial seed values. This ensures that N samples $\{\boldsymbol{\theta}_1^i\}_{i=1}^N$ from $\pi(\boldsymbol{\theta}|F_1)$ are drawn. As each of the nc chain seeds are already located in F_1 by choice of b_1, in total $N - nc$ samples are generated at $level_1$. A detailed discussion of the component-wise MH algorithm and the role of MCMC in SuS is given in Section 4.2.1. Similar to $level_0$, the limit state function $g(\boldsymbol{\theta})$ is evaluated for all $\{\boldsymbol{\theta}_1^i\}_{i=1}^N$. The evaluations of $\{g_1^i\}_{i=1}^N$ and $\{\boldsymbol{\theta}_1^i\}_{i=1}^N$ are again placed in descending order with respect to $\{g_1^i\}_{i=1}^N$. The stopping

criterion is checked to establish whether $n_F(1)/N > p_0$. If so, simulation stops and $p_F \approx p_0^1(n_F(1)/N)$. If not, the algorithm continues to $level_2$ and proceeds in the same manner as outlined for $level_1$. *SuS* sequentially generates intermediate levels until the number of failure samples at any given level is greater than p_0. A pseudo code for *SuS* is provided in Algorithm 4.

Algorithm 4 Subset Simulation

Define N, p_0, $nc = p_0 N$, $ns = p_0^{-1}$ and b
Initialize $m = 0$, where m is the current simulation level
Generate N samples $\{\boldsymbol{\theta}_m^i\}_{i=1}^N$ from the input PDF $\pi(\cdot)$
For each $\{\boldsymbol{\theta}_m^i\}_{i=1}^N$ evaluate g_m^i
Sort the N values of g_m^i and the corresponding $\{\boldsymbol{\theta}_m^i\}_{i=1}^N$ in descending order with respect to g_m^i
Set $n_F(m) = \sum_{i=1}^N (g_m^i > b)$
while $p_0 > n_F(m)/N$ **do**
    $m = m + 1$
    Calculate $b_m = \frac{g_{m-1}^{nc} + g_{m-1}^{nc+1}}{2}$
    Store the first nc samples of the ordered set $\{\boldsymbol{\theta}_{m-1}^i\}_{i=1}^N$ as 'seeds'
    Using $\{\boldsymbol{\theta}_{m-1}^j\}_{j=1}^{nc}$ draw the remaining $N - nc$ samples from $\pi(\cdot|F_m)$ via MCMC as follows:
    for $j = 1, ..., nc$ **do**
        Starting with $\{\boldsymbol{\theta}_{m-1}^j\}$ as an initial seed, generate $\{\boldsymbol{\theta}_m^k\}_{k=1}^{ns-1} \sim \pi(\cdot|F_m)$ states of a Markov chain using the Modified Metropolis Hastings in Algorithm 5. In this algorithm, the sample acceptance criteria involves evaluating g_m with respect to b_m for each generated sample.
    end for
    Sort the N values of g_m^i and the corresponding $\{\boldsymbol{\theta}_m^i\}_{i=1}^N$ in descending order with respect to g_m^i
    Calculate $n_F(m) = \sum_{i=1}^N (g_m^i > b)$
end while
Return $p_F \approx p_0^m \frac{n_F(m)}{N}$

Figure 4.1 presents a 2-dimensional example taken from [38] illustrating the sampling process of *SuS*. This problem analyses a series system composed of four limit state functions each dependent on $\boldsymbol{\theta} = \{\theta_1, \theta_2\}$.

$$g(\boldsymbol{\theta}) = min \begin{cases} h + 0.1 \cdot (\theta_1 - \theta_2)^2 - \frac{(\theta_1+\theta_2)}{\sqrt{2}} \\ h + 0.1 \cdot (\theta_1 - \theta_2)^2 + \frac{(\theta_1+\theta_2)}{\sqrt{2}} \\ (\theta_1 - \theta_2) + \frac{j}{\sqrt{2}} \\ (\theta_2 - \theta_1) + \frac{j}{\sqrt{2}} \end{cases} \tag{4.11}$$

The parameters h and j are set equal to 3 and 7 respectively [59]. SuS sequentially generates samples until the number of failure samples (red) according to the stopping criterion of Algorithm 4 has been satisfied. It is evident that SuS identifies the multi-modality of the failure domain after $level_1$ before stopping simulation after $level_3$. With 23 samples belonging to the final intermediate failure domain, SuS estimates p_F by $\hat{p}_F = 2.3 \cdot 10^{-3}$ with the reference solution $p_F = 2.2 \cdot 10^{-3}$ [59].

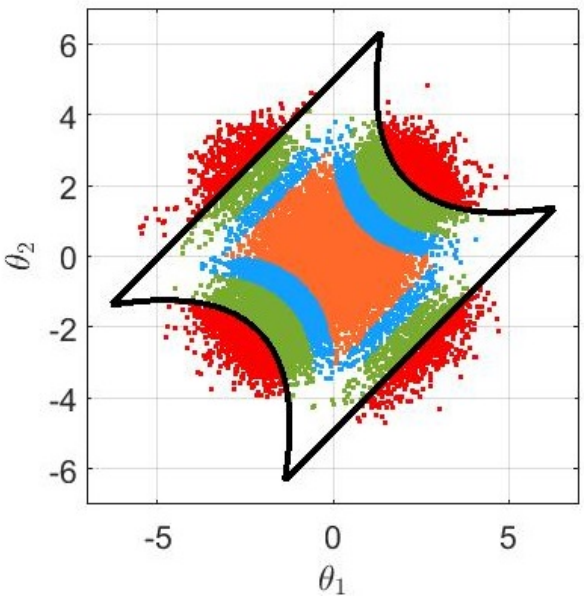


Figure 4.1: SuS performed with $N = 1000$ and $p_0 = 0.1$. Samples generated at $level_0$ (orange), $level_1$ (blue), $level_2$ (green) and $level_3$ (red) are shown. SuS iteratively generates samples which gradually approach the failure domain. The multi modality of the failure domain stemming from the limit state function expression is evident and is identified by SuS after $level_1$. The final critical threshold b_3 before termination is represented by a black line.

4.2.1 Subset Simulation: The Role of MCMC

The generation of samples at every intermediate level of SuS results in the efficiency of SuS relying heavily on the chosen MCMC sampling scheme. Conditioning the sample on the current intermediate failure domain, modifies the MCMC strategy discussed in previous chapters. Let $\boldsymbol{\theta}$ represent the current state of the chain and $\boldsymbol{\theta'}$ the proposed sample of the transition to the next state. Consider the stationary distribution $\pi(\boldsymbol{\theta}|F_i)$ and the transition operator $T(\boldsymbol{\theta}|\boldsymbol{\theta'})$ with the target distribution for a proposed candidate sample given by

$$\pi(\boldsymbol{\theta'}|F_i) = \int T(\boldsymbol{\theta'}|\boldsymbol{\theta})\pi(\boldsymbol{\theta}|F_i)d\boldsymbol{\theta} \tag{4.12}$$

Generating a sample from the proposal $S(\cdot|\boldsymbol{\theta})$, the probability of sample acceptance may be computed as the minimum value between 1 and

$$\begin{aligned} r(\boldsymbol{\theta'}) &= \frac{\pi(\boldsymbol{\theta'}|F_i)}{\pi(\boldsymbol{\theta}|F_i)} \cdot \frac{S(\boldsymbol{\theta}|\boldsymbol{\theta'})}{S(\boldsymbol{\theta'}|\boldsymbol{\theta})} \\ &= \frac{\frac{\pi(\boldsymbol{\theta'})I_F(\boldsymbol{\theta'})}{p_{Fi}}}{\frac{\pi(\boldsymbol{\theta})I_F(\boldsymbol{\theta})}{p_{Fi}}} \cdot \frac{S(\boldsymbol{\theta}|\boldsymbol{\theta'})}{S(\boldsymbol{\theta'}|\boldsymbol{\theta})} \\ &= \frac{\pi(\boldsymbol{\theta'})}{\pi(\boldsymbol{\theta})} \cdot \frac{I_F(\boldsymbol{\theta'})}{I_F(\boldsymbol{\theta})} \cdot \frac{S(\boldsymbol{\theta}|\boldsymbol{\theta'})}{S(\boldsymbol{\theta'}|\boldsymbol{\theta})} \end{aligned} \tag{4.13}$$

Seeing as the current state of the Markov chain already follows the target distribution, the acceptance ratio is simplified to

$$r(\boldsymbol{\theta'}) = \min\left\{1, \frac{\pi(\boldsymbol{\theta'})}{\pi(\boldsymbol{\theta})} \cdot \frac{S(\boldsymbol{\theta}|\boldsymbol{\theta'})}{S(\boldsymbol{\theta'}|\boldsymbol{\theta})}\right\} \cdot I_F(\boldsymbol{\theta'}) \tag{4.14}$$

This simplification of the acceptance probability results in a candidate sample being either accepted or rejected in a two stage approach. At the first step, a sample is generated from $S(\cdot|\boldsymbol{\theta})$ with the above acceptance probability. Secondly, it is checked whether the proposed sample lies in F_i. The standard MH algorithm suffers in high dimensions due to the increase in rejection rates at the first step for an increasing number of uncertain variables. A modified MH (MMH) algorithm

[11] has been proposed which allows SuS to efficiently generate samples in high dimensions.

MMH differs from the original MH in the manner in which the proposed candidate state is generated. Let $level_m$ represent the current intermediate level and b_m the subsequent intermediate threshold. Consider the current state of the Markov chain to be $\boldsymbol{\theta} = \{\theta_1, ..., \theta_d\}$. Instead of using a d-variate proposal PDF to directly obtain the candidate state $\boldsymbol{v} = \{v_1, ..., v_d\}$ as in MH, MMH expresses the proposal as a product of independent univariate PDFs. As outlined by Eq. 4.14, the sample generation is a two step process. Firstly, a pre-candidate sample w_j is drawn from a univariate proposal $S_j(\cdot|\theta_j)$ dependent on the component θ_j of the current state. The univariate proposal is user defined with common choices being a Gaussian or uniform [12]. The Metropolis ratio for w_j is then computed.

$$r(w_j) = \frac{\pi(w_j)}{\pi(\theta_j)} \cdot \frac{S_j(\theta_j|w_j)}{S_j(w_j|\theta_j)} \tag{4.15}$$

If w_j is accepted with respect to $r(w_j)$, v_j is set to w_j. Otherwise v_j is set to θ_j i.e the j^{th} component of the current state of the Markov chain. Consequently every time a w_j is rejected, v_j is assigned a component of a sample which already lives in the target failure domain. This process is repeated until all d components of $\boldsymbol{v}$ have been specified. Next, a check is made to identify whether $\boldsymbol{v} \in F$ through $g(\boldsymbol{v}) > b_m$. If the candidate sample belongs in the failure domain it is accepted as the next state of the Markov chain and $\boldsymbol{\theta'} = \boldsymbol{v}$. If not, the next state of the Markov chain is $\boldsymbol{\theta'} = \boldsymbol{\theta}$. To summarize, the MMH proceeds as follows in Algorithm 5:

Algorithm 5 Modified Metropolis-Hastings

Define the intermediate critical threshold b_m from the current intermediate $level_m$
Define the current state of the Markov chain $\boldsymbol{\theta} = \{\theta_1, ..., \theta_d\}$
Define the univariate proposal PDFs $S_j(\cdot|\theta_j)$ for $j = 1, ..., d$
for each $j = 1, ..., d$ **do**
    Generate a pre-candidate w_j from $S_j(\cdot|\theta_j)$
    Generate u_j uniformly on $[0, 1]$
    $r(w_j) = \frac{\pi(w_j)}{\pi(\theta_j)} \cdot \frac{S_j(\theta_j|w_j)}{S_j(w_j|\theta_j)}$
    Accept $v_j = w_j$ if $u_j < r(w_j)$, otherwise $v_j = \theta_j$
end for
$\boldsymbol{v} = \{v_1, ..., v_d\}$
Compute $g(\boldsymbol{v})$ for the proposed candidate sample $\boldsymbol{v}$
if $g(\boldsymbol{v}) > b_m$ **then**
    $\boldsymbol{\theta'} = \boldsymbol{v}$
else
    $\boldsymbol{\theta'} = \boldsymbol{\theta}$
end if

An illustration of the sampling process of the MMH sampler in the context of SuS is provided in Figure 4.2. In this case, $N = 10$ and $p_0 = 0.2$ resulting in $nc = 2$ and $ns = 5$. At $level_0$ the initial samples generated from the prior distribution are denoted in blue. At $level_1$, the red dots represent the samples generated at $level_0$ that have exceeded b_1. These samples are used as seeds to generate individual states of Markov chains denoted in green. This process is continued until termination at $level_m$. Examples of a rejected pre-candidate sample w_j (purple) and a proposed sample rejected by the second rejection event $\boldsymbol{v}$ (black) at $level_2$ are also shown. Note that in practice the progression of the Markov chain may not be as direct in terms of movements as shown in Figure 4.2. The purpose of this plot is to provide the reader with a visual depiction of the sampling process.

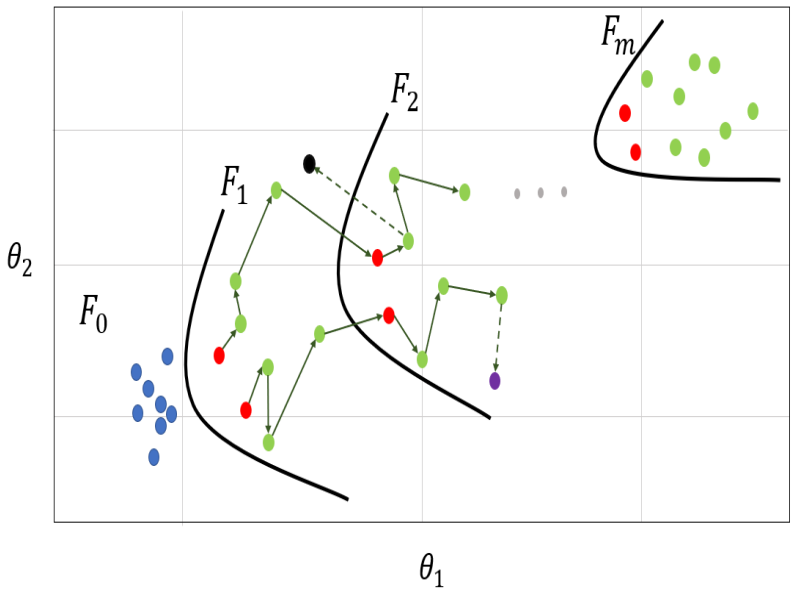


Figure 4.2: Illustration of the sample generation process by MMH. At $level_0$ the initial samples generated from the prior distribution are denoted in blue. At $level_1$, the red dots represent the samples generated at $level_0$ that have exceeded b_1. These samples are used as seeds to generate individual states of Markov chains denoted in green. This process is continued until termination at $level_m$. Examples of a rejected pre-candidate sample (purple) and a sample rejected by the second rejection event (black) at $level_2$ are also shown.

Numerous methods have since been proposed aimed at further improving the sampling procedure in *SuS* [10, 172, 149, 196]. For a detailed comparison on performance, the reader is referred to [172]. The construction of *SuS* as a nested sequence of failure events addresses some of the important issues which arise in MCMC as discussed in Chapter 3. By choosing the seeds of the Markov chain as samples which exceed the critical threshold at a given intermediate level, the resultant chains produce samples which follow the desired stationary distribution. This concept directly alleviates the requirement for a sample burn in period. The task of identifying convergence in MCMC methods has also been rectified by the combination of this seed choice along with the implementation of the stopping criterion in Algorithm 4. As discussed in section 4.1.1, *SuS* operates in standard normal space and requires a transformation of the generated samples to the physical parameter space. Allowing for the development of *SuS* tailored MCMC schemes which enable the framework to be robust to the number of uncertain parameters. In turn, ensuring that *SuS* is suitable for estimating the probability

of rare events in problems in high dimensions. This property, is a key principal for the SuS based Bayesian inference scheme presented later in this chapter.

4.3 BUS

Bayesian Updating with Structural reliability methods (BUS) [208] opens up the possibility of evaluating posterior densities through the use of engineering reliability methods. Consider a reliability problem with uncertain parameters $(\boldsymbol{\theta}, u)$ independent of one another with their joint PDF given by $Q(\boldsymbol{\theta})P(u)$. Define the target failure event by

$$F = \{\boldsymbol{\theta} \in \mathbb{R}^d : u - cL(D|\boldsymbol{\theta}) \leq 0\} \tag{4.16}$$

where $u \sim P(u)$ with support $[0, 1]$ and acknowledging that $P(u) = 1$. Suppose by some means (e.g. SuS), a failure sample distributed as $Q(\boldsymbol{\theta})P(u)$ conditional on the failure event F can be obtained. The subsequent PDF is

$$P(\boldsymbol{\theta}, u|F) = \frac{Q(\boldsymbol{\theta})I[u - cL(D|\boldsymbol{\theta}) \leq 0]}{p_F} \tag{4.17}$$

Integrating out the uniform random variable from the PDF of the failure sample yields

$$\begin{aligned} P(\boldsymbol{\theta}|F) &= \int_0^1 P(\boldsymbol{\theta}, u|F)du \\ &= \frac{Q(\boldsymbol{\theta})}{p_F} \int_0^1 I[u < cL(D|\boldsymbol{\theta})]du \\ &= \frac{Q(\boldsymbol{\theta})cL(D|\boldsymbol{\theta})}{p_F} \\ &= \frac{Q(\boldsymbol{\theta})cL(D|\boldsymbol{\theta})}{\int Q(\boldsymbol{\theta})cL(D|\boldsymbol{\theta})d\boldsymbol{\theta}} \end{aligned} \tag{4.18}$$

$$P(\boldsymbol{\theta}|F) \propto Q(\boldsymbol{\theta})cL(D|\boldsymbol{\theta}) \tag{4.19}$$

This is the distribution of the sample conditional on the failure event in Eq. 4.16. Similarly, consider again the sample acceptance step from rejection sampling in

Chapter 3 which determines whether or not $\boldsymbol{\theta} \sim P(\boldsymbol{\theta}|D)$. Let $\pi(\boldsymbol{\theta}|D)$ be the potentially unnormalized posterior, with the relationship between the normalized posterior and unnormalized posterior be given by $P(\boldsymbol{\theta}|D) \propto \pi(\boldsymbol{\theta}|D)$. Let $u \sim P(u)$ with support $[0, 1]$.

$$\begin{aligned} u &< \frac{\pi(\boldsymbol{\theta}|D)}{kQ(\boldsymbol{\theta})} \\ \implies u &< \frac{Q(\boldsymbol{\theta})L(D|\boldsymbol{\theta})}{kQ(\boldsymbol{\theta})} \\ \implies u &< k^{-1}L(D|\boldsymbol{\theta}) \\ \implies u - cL(D|\boldsymbol{\theta}) &< 0 \end{aligned} \tag{4.20}$$

with $c = k^{-1}$. While again acknowledging that $P(u) = 1$, under rejection sampling the PDF conditional on Eq. 4.20 for $\boldsymbol{\theta}$ and u is

$$P(\boldsymbol{\theta}, u|u - cL(D|\boldsymbol{\theta}) < 0) = \frac{Q(\boldsymbol{\theta})I[u - cL(D|\boldsymbol{\theta}) < 0]}{p_A} \tag{4.21}$$

Integrating out the uniform random variable from the PDF of the failure sample yields

$$\begin{aligned} P(\boldsymbol{\theta}|u - cL(D|\boldsymbol{\theta}) < 0) &= \frac{Q(\boldsymbol{\theta})}{p_A} \int_0^1 I[u < cL(D|\boldsymbol{\theta})]du \\ &= \frac{Q(\boldsymbol{\theta})cL(D|\boldsymbol{\theta})}{p_A} \\ &= \frac{Q(\boldsymbol{\theta})cL(D|\boldsymbol{\theta})}{\int Q(\boldsymbol{\theta})cL(D|\boldsymbol{\theta})d\boldsymbol{\theta}} \end{aligned} \tag{4.22}$$

$$P(\boldsymbol{\theta}|u - cL(D|\boldsymbol{\theta}) < 0) \propto Q(\boldsymbol{\theta})cL(D|\boldsymbol{\theta}) \tag{4.23}$$

Therefore, the choice of limit state function in Eq. 4.16, results in the PDF of the samples conditional on the specified failure event being equivalent to the PDF of the posterior distribution for rejection sampling. This means that samples conditional on the failure domain in Eq. 4.16 are distributed according to the

posterior distribution. Through the definition of the limit state function as

$$g(\boldsymbol{\theta}, u) = u - cL(D|\boldsymbol{\theta}) \tag{4.24}$$

BUS augments the sample space of $\boldsymbol{\theta}$ with the introduction of a uniform variable u on $[0, 1]$. This allows for the rejection sampling strategy presented in Chapter 3 to be combined methods from reliability engineering to provide an efficient solution to the Bayesian updating problem. By interpreting the sampling inefficiency of rejection sampling as a rare event, BUS acknowledges that the probability of sample acceptance p_A under rejection sampling is equivalent to the probability of failure p_F in reliability analysis when a uniform random variable on $[0, 1]$ is added to the sample space of the uncertain parameter. This is prevalent from the denominators of the expressions in Eq. 4.18 and Eq. 4.22, whereby p_F and p_A don't influence the distribution of the parameters but rather ensure the distribution is a valid PDF which integrates to 1. In BUS, p_F from Eq. 4.18 is directly linked to the model evidence P_D through c:

$$\begin{aligned} p_F &= \int Q(\boldsymbol{\theta}) cL(D|\boldsymbol{\theta}) d\boldsymbol{\theta} \\ &= cP_D \end{aligned} \tag{4.25}$$

Therefore, obtaining an appropriate value of c and computing p_F allows for the estimation of P_D as a by product of simulation. The link between the failure domain and the actual Bayesian updating task is that the samples which belong in the failure region follow the posterior distribution. With such samples being generated once an intermediate threshold less than 0 is produced. Consequently, the interest is in the samples conditional on F. While BUS is not constrained to any particular reliability method, for this study SuS is chosen. In terms of parameter estimation, as outlined in Section 4.2.1, SuS with BUS offers many advantages over MCMC.

In line with rejection sampling, difficulty lies in correctly selecting a suitable likelihood multiplier c for BUS. An appropriate choice of c in rejection sampling ensures the subsequent samples are distributed according to the posterior. Given the limit state function is dependent on c, the multiplier value is required to be

known prior to simulation. From the rejection principle

$$cL(D|\boldsymbol{\theta}) \leq 1 \tag{4.26}$$

the largest acceptable value of c is given by

$$c_{max} = \frac{1}{L(D|\boldsymbol{\theta})_{max}} \tag{4.27}$$

In the case of the value being greater than c_{max}, Eq. 4.26 is not satisfied resulting in incorrect samples being generated. A proof of this is contained in Appendix B. BUS provides an efficient solution to Bayesian updating tasks once c is chosen appropriately with respect to $L(D|\boldsymbol{\theta})_{max}$. However, in many cases this quantity is unknown in advance and as such, c cannot be computed. Two strategies developed to resolve this issue are discussed in the following.

4.3.1 Adaptive BUS (aBUS)

An adaptive strategy ($aBUS$) which selects a new value of c at each intermediate level of SuS has been proposed [33]. Adopting a modification firstly proposed in [67], [32] later introduced the natural logarithm to the limit state function of $aBUS$.

$$g(\boldsymbol{\theta}, u) = \ln(u) - \ln(c) - \ln L(D|\boldsymbol{\theta}) \tag{4.28}$$

The reasoning of which is discussed in more detail in Section 4.3.2. This updated $aBUS$ focuses solely on the selection of c, with the final multiplier value selected as

$$c = e^{-L(D|\boldsymbol{\theta})_{max}} \tag{4.29}$$

Utilizing information from the likelihood evaluations, c is set equal to the largest likelihood value generated from the set of samples. At the end of a subset level, it is checked whether a likelihood larger than the current value of c has been produced. If so, the current value of c is set equal to the largest observed likelihood. While [33] and [32] differ in two aspects: (i) the adoption of the natural logarithm and (ii) a minor correction of the intermediate threshold, both forms of $aBUS$ adopt the same stopping criterion as BUS. Under $aBUS$, the model evidence is

expressed by

$$P_D = p_F L(D|\boldsymbol{\theta})_{max} \tag{4.30}$$

In the event of the likelihood function reaching the theoretical maximum, the model evidence will be equivalent to the probability of failure. Given the finite nature of the sample set generated in a model run, it is highly unlikely that the true maximum of the likelihood function for a given problem will be simulated. Meaning, in the case of $\hat{L}(D|\boldsymbol{\theta})_{max} < L(D|\boldsymbol{\theta})_{max}$ with $\hat{L}(\cdot)$ denoting simulated likelihood function, the model evidence will be under estimated. This would have a further knock on effect in terms of model class selection whereby the model evidence will not accurately reflect the plausibility of the chosen model.

4.3.2 Nested BUS (nBUS)

A nested BUS formulation ($nBUS$) [67], rectifies the issue of multiplier choice by re-expressing the failure event of interest as follows. Consider again the acceptance criterion from rejection sampling.

$$cL(D|\boldsymbol{\theta}) > u \tag{4.31}$$

Expressing Eq. 4.31 on the natural log scale and rearranging yields:

$$\ln(cL(D|\boldsymbol{\theta})) > \ln(u)$$

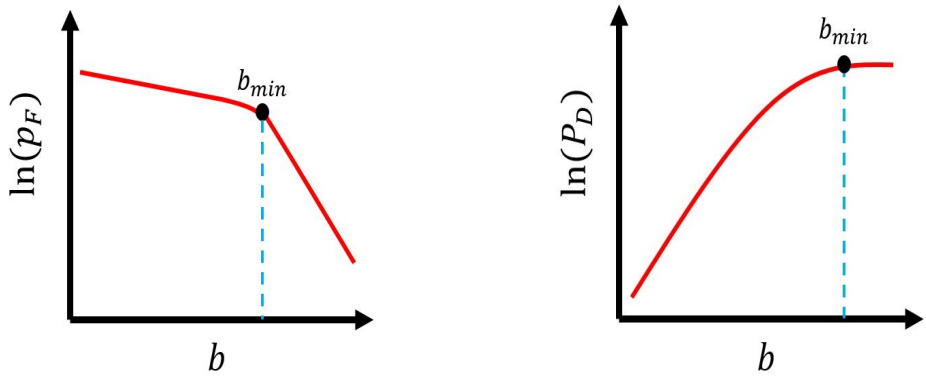
$$\ln L(D|\boldsymbol{\theta}) - \ln(u) > -\ln(c) \tag{4.32}$$

This reformulation of the failure event results in the critical threshold $b = -\ln c$ being expressed in terms of the likelihood multiplier. As SuS calculates this threshold adaptively at each intermediate level, its value is automatically computed. Let $g(\boldsymbol{\theta}, u) = \ln L(D|\boldsymbol{\theta}) - \ln(u)$, which is no longer dependent on the multiplier. In contrast to $aBUS$ and BUS, the task for $nBUS$ is to identify the minimum number of levels required to stop simulation. The details of which are discussed in Section 4.4. For a large enough intermediate threshold, the model evidence under $nBUS$ is given by

$$P_D = e^b p_F \qquad\qquad b > b_{min} \tag{4.33}$$

where b_{min} represents the minimum required level for $\boldsymbol{\theta} \sim P(\boldsymbol{\theta}|D)$. In contrast to $aBUS$, the evidence estimated by $nBUS$ is not dependent on the likelihood function but on the intermediate critical threshold. Therefore it does not suffer from the potential of not reaching the true maximum of the likelihood when using a finite sample size. The natural logarithm aids computational efficiency by ensuring that $g(\boldsymbol{\theta}, u)$ is a well defined random variable whilst also ensuring a smoother transition between intermediate levels. Additionally, it forms the basis of an automatic stopping condition on the premise of identifying the minimum required number of levels to draw samples from the posterior distribution. Examining the functional behaviour of $\ln(p_F)$ reveals a function with a slope equal to -1 for $b > b_{min}$. Similarly, $\ln(P_D)$ goes from a linearly increasing function as b increases through a transition stage to remaining constant equal to $\ln(P_D)$ once $b > b_{min}$.

Derivations for the characteristic trends illustrated in Figure 4.3 are given in Appendix B. In reality, the stochastic nature of $nBUS$ results in the trends being available on a sample basis only. As such, they contain statistical errors which in general decrease for an increasing number of samples. Given the sensitivity to this parameter, visual inspection of the characteristic trends based on the aforementioned functional behaviour is not enough to confidently conclude the generated samples are distributed according to the posterior



(a) (b)

Figure 4.3: Characteristics trends of $\ln(p_F)$ (left) and $\ln(P_D)$ (right).

4.4 Stopping Criteria

The manner of termination of the sampling process directly influences sample quality and computational expense. This section provides an overview of existing stopping conditions for BUS, $aBUS$ and $nBUS$.

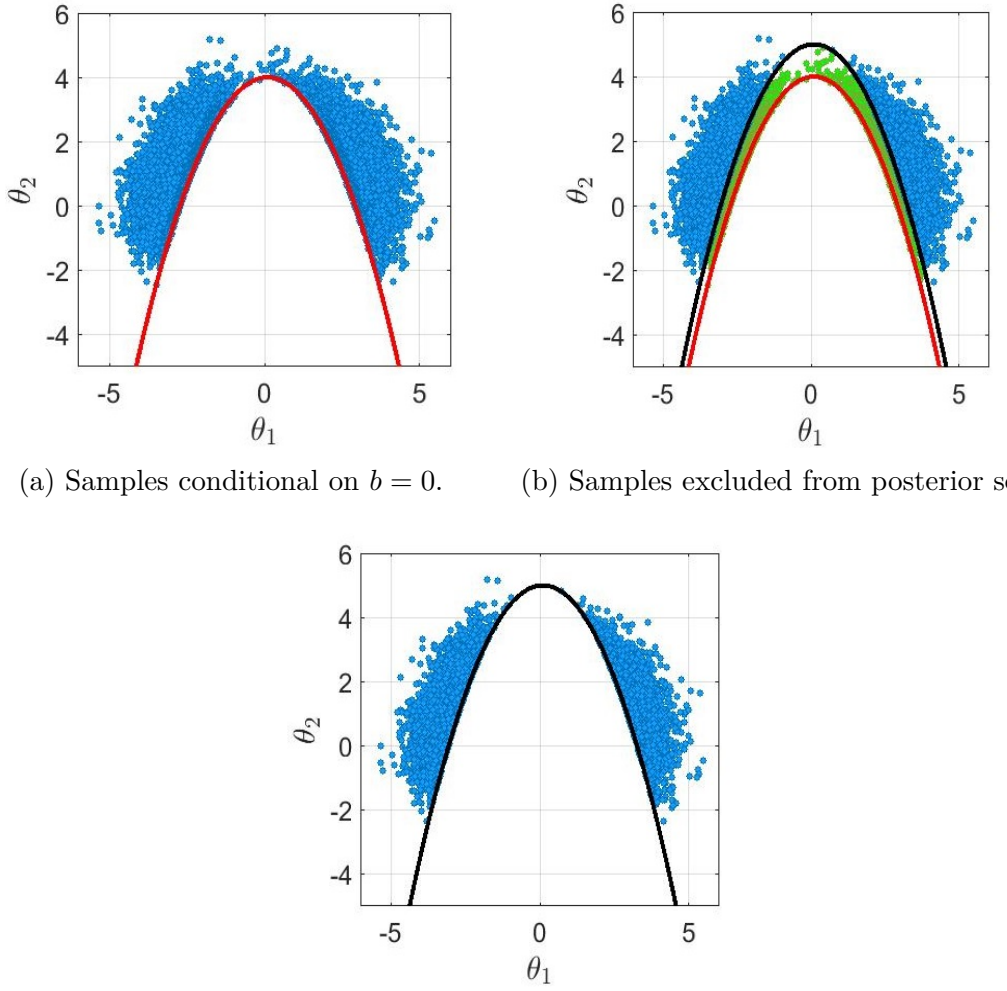
4.4.1 aBUS and BUS

At termination, both BUS and $aBUS$ generate $\boldsymbol{\theta} \sim P(\boldsymbol{\theta}|\{g(\boldsymbol{\theta}, u) \leq 0\})$. Given the stochastic nature of b, generating samples truly conditional on $b = 0$ is highly unlikely. To illustrate this sampling process consider a 2-dimensional problem with a parabolic limit state function. This example concerns the physical sample generation with respect to the limit state function. Figure 4.4 (a) presents the generated samples truly conditional on $b = 0$ with the final intermediate threshold given by the red parabola. In a single $aBUS$ run, suppose the final $b = -0.5$ is given by the black parabola. From Figure 4.4 (b) it is evident that a proportion of samples (green) located between the two boundaries are excluded from the posterior set conditional on $b = 0$. Where the true samples conditional on $b = -0.5$ (Figure 4.4 (c)) are the chosen posterior samples. Once an intermediate threshold value smaller than 0 is generated, it is suggested to stop simulation and rescale to $b = 0$, resulting in the chosen posterior samples not truly being conditional on the specified failure event, where a proportion of true posterior samples (green) have been excluded from the final set. Algorithm 6 provides a pseudo code for $aBUS$. Recall under $aBUS$ that c is chosen as

$$c = e^{-L(D|\boldsymbol{\theta})_{max}} \tag{4.34}$$

resulting in

$$\begin{aligned} g(\boldsymbol{\theta}, u) &= \ln(u) - \ln(c) - \ln L(D|\boldsymbol{\theta}) \\ &= \ln(u) - \ln(e^{-L(D|\boldsymbol{\theta})_{max}}) - \ln L(D|\boldsymbol{\theta}) \\ &= \ln(u) + L(D|\boldsymbol{\theta})_{max} - \ln L(D|\boldsymbol{\theta}) \\ &= \ln(u) + l - \ln L(D|\boldsymbol{\theta}) \end{aligned} \tag{4.35}$$



(a) Samples conditional on $b = 0$.

(b) Samples excluded from posterior set.

(c) Samples conditional on $b = 0$

Figure 4.4: Example of the sample generation process by $aBUS$ using a parabolic limit state function. Panel (a) illustrates the true posterior sample set with $b = 0$. Using a threshold of $b = -0.5$, panel (b) highlights the samples (green) which are excluded from the true posterior set by terminating simulation once a threshold less than $b = 0$ has been computed. Panel (c) shows the sample set conditional on $b = -0.5$ and assumed under $aBUS$ and BUS to be the true posterior set.

where l denotes $L(D|\boldsymbol{\theta})_{max}$. At $level_0$ N samples are drawn from the input PDF and at step 5 in Algorithm 6 the maximum of the likelihood function evaluated for all N samples is identified and $l_0 = \{\{L(D|\boldsymbol{\theta}_0^i)\}_{i=1}^N\}_{\max}$. The limit state function $\{g_0^i\}_{i=1}^N$ is evaluated for all N samples using l_0. The limit state function evaluations $\{g_0^i\}_{i=1}^N$ and $\{\boldsymbol{\theta}_0^i, u_0^i\}_{i=1}^N$ are placed in descending order with respect to $\{g_0^i\}_{i=1}^N$. At $level_1$, an intermediate failure threshold b_1 is computed such that $nc = p_0 \cdot N$ values of $\{g_0^i\}_{i=1}^N$ exceed its value. The purpose of this is to identify the samples

from $level_0$ to be used as seeds to populate F_1. The resulting sample set chosen based on $\{g_0^i\}_{i=1}^{nc}$ is $\{\boldsymbol{\theta}_0^i, u_0^i\}_{i=1}^{nc}$. Each sample in this set is used as a seed for an individual Markov chain to populate F_1 through the MMH sampler previously discussed. For each of the generated samples in the MMH sampler, the limit state function is evaluated using the updated value of $l_1 = \max(l_0, \{\{L(D|\boldsymbol{\theta}_1^i)\}_{i=1}^{N}\}_{\max})$. If $b_1 < 0$ the sampler stops and the samples generated at $level_1$ are taken as those from the posterior. If not, $aBUS$ proceeds to $level_2$. Simulation proceeds in the same manner as above until the computed intermediate threshold at any given $level_m$ is less than zero. The estimated model evidence is computed using the final l_m value and given by $P_D \approx \hat{p}_F l_m$.

Algorithm 6 *aBUS*

Define N, p_0, $nc = p_0 N$ and $ns = p_0^{-1}$

Initialize $m = 0$, where m is the current simulation level

Initialize $b_m = \infty$

Generate N samples $\{\boldsymbol{\theta}_m^i, u_m^i\}_{i=1}^N$ from the input PDF $\pi(\cdot)$

Set $l_m = \{\{L(D|\boldsymbol{\theta}_m^i)\}_{i=1}^N\}_{\max}$

For each $\{\boldsymbol{\theta}_m^i, u_m^i\}_{i=1}^N$ evaluate $g_m^i = \ln(u_m^i) - \ln L(D|\boldsymbol{\theta}_m^i) + l_m$

Sort the N values of g_m^i and the corresponding $\{\boldsymbol{\theta}_m^i, u_m^i\}_{i=1}^N$ in ascending order with respect to g_m^i

while $b_m > 0$ **do**

  $m = m + 1$

  Calculate $b_m = \frac{g_{m-1}^{nc} + g_{m-1}^{nc+1}}{2}$

  Store the first nc samples of the ordered set $\{\boldsymbol{\theta}_{m-1}^i, u_{m-1}^i\}_{i=1}^N$ as 'seeds'

  Using $\{\boldsymbol{\theta}_{m-1}^j, u_{m-1}^j\}_{j=1}^{nc}$ draw the remaining $N - nc$ samples from $\pi(\cdot|F_m)$ via MCMC as follows:

  for $j = 1, ..., nc$ **do**

    Starting with $\{\boldsymbol{\theta}_{m-1}^j, u_{m-1}^j\}$ as an initial seed, generate $\{\boldsymbol{\theta}_m^k, u_m^k\}_{k=1}^{ns-1} \sim \pi(\cdot|F_m)$ states of a Markov chain using the Modified Metropolis Hastings in Algorithm 5. In this algorithm, the sample acceptance criteria involves evaluating g_m with respect to b_m for each generated sample.

  end for

  Sort the N values of g_m^i and the corresponding $\{\boldsymbol{\theta}_m^i, u_m^i\}_{i=1}^N$ in descending order with respect to g_m^i

  Set $l_m = \max(l_{m-1}, L(D|\boldsymbol{\theta}_m^i)_{\max})$

  if $b_m < 0$ **then**

    Set $b_m = 0$

    $p_m = \frac{\sum_{i=1}^N (g_m^i < b_m)}{N}$

  else

    $p_m = p_0$

  end if

end while

$\hat{p}_F = \prod_{i=1}^m p_m$

Return $\boldsymbol{\theta} \sim P(\boldsymbol{\theta}|D)$ & $P_D \approx \hat{p}_F l_m$

4.4.2 nBUS

To ensure certainty that the minimum level has been surpassed, $nBUS$ proposes an automatic stopping condition. As the intermediate levels progress, the quality of samples decrease due to an increased correlation stemming from the MCMC process. Therefore, N generated posterior samples does not suggest N pieces of information are being received. $nBUS$ protects against this deterioration in sample quality by determining the probability of generating a sample from outside the posterior distribution. $nBUS$ proposes to compute this probability at every intermediate level. Once this probability is zero, the sampler stops as the generated samples are distributed according to the posterior. Recalling that $b = -\ln c$, the rejection principle from Chapter 3 may be expressed as:

$$e^{-b} L(D|\boldsymbol{\theta}) \leq 1 \tag{4.36}$$

For any $level_m$ such that $b_m < b_{min}$, where b_{min} is the minimum required threshold, $nBUS$ seeks to estimate the probability of $\boldsymbol{\theta}$ being in the set $A_m(\boldsymbol{\theta})$.

$$A_m(\boldsymbol{\theta}) = \{\boldsymbol{\theta} : e^{-b_m} L(D|\boldsymbol{\theta}) > 1\} \tag{4.37}$$

In this case $A_m(\boldsymbol{\theta})$ may be viewed as the set of inadmissible samples. Where the use of inadmissible refers to those which have not satisfied the rejection principle and are thus outside the posterior set. Regarding $A_m(\boldsymbol{\theta})$, consider the following theorem as presented in [67].

Theorem 1. *[67] There exists constants e^{-b_m} and a monotone decreasing sequence h_m, such that*

$$lim_{m\to\infty} \quad h_m = 0 \tag{4.38}$$

where h_m is the prior probability of the set $A_m(\boldsymbol{\theta}) = \{\boldsymbol{\theta} : e^{-b_m} L(D|\boldsymbol{\theta}) > 1\}$.

Theorem 1 reveals that for a large enough number of intermediate levels, the probability of generating a sample belonging outside the posterior set will decrease before converging to zero. By the definition of the set $A_m(\boldsymbol{\theta})$ as in Eq. 4.37, $A_m^c(\boldsymbol{\theta})$ is defined as:

$$A_m^c(\boldsymbol{\theta}) = \{\boldsymbol{\theta} : e^{-b_m} L(D|\boldsymbol{\theta}) \leq 1\} \tag{4.39}$$

This means that $A_m^c(\boldsymbol{\theta})$ contains all samples which satisfy the rejection principle in Eq. 4.36. Once h_m has converged to zero, $A_m(\boldsymbol{\theta})$ is the empty set and the samples drawn by $nBUS$ at $level_m$ belong in the set $A_m^c(\boldsymbol{\theta})$. By rearranging the expression for h_m in Theorem 1, allow for this probability measure to be expressed as:

$$h_m = p_{\boldsymbol{\theta}}(L(D|\boldsymbol{\theta}) > e^{b_m}) \tag{4.40}$$

The evaluation of Eq. 4.40 requires the computation of a high dimensional integral. Additionally, as its value will become increasingly small for an increasing number of intermediate levels, SuS is proposed to compute h_m. However as the samples generated during $nBUS$ are given by the limit state function

$$g(\boldsymbol{\theta}, u) = \ln L(D|\boldsymbol{\theta}) - \ln(u) \tag{4.41}$$

these samples cannot be used to compute h_m as the limit state function for computing h_m is given by

$$g(\boldsymbol{\theta}) = L(D|\boldsymbol{\theta}) \tag{4.42}$$

As the limit state functions differ, a separate implementation of SuS is required to generate the conditional failure samples needed for estimating h_m. This results in the algorithmic structure of $nBUS$ being nested. The automatic stopping condition of $nBUS$ proposes to compute h_m by SuS at every intermediate level of $nBUS$ until it has converged to zero. Once this has occurred, the samples generated at $level_m$ of $nBUS$ are taken as those drawn from the posterior. A pseudo-code for $nBUS$ is given in Algorithm 7. At $level_0$, h_0 is not computed as the sample set $\{\boldsymbol{\theta}_m^i, u_m^i\}_{i=1}^N$ is drawn from the prior and as such a default value of $h_0 = 1$ is initialized. Seeing as $h_{m\to 0}$ for an increasing m, in reality a very small tolerance ϵ must be chosen with a suggested value of $\epsilon = 10^{-8}$ given in [67]. At $level_0$ N samples are drawn from the input PDF and the limit state function $\{g_0^i\}_{i=1}^N$ is evaluated. The limit state function evaluations $\{g_0^i\}_{i=1}^N$ and $\{\boldsymbol{\theta}_0^i, u_0^i\}_{i=1}^N$ are placed in descending order with respect to $\{g_0^i\}_{i=1}^N$. At each intermediate level the stopping rule checks whether or not $h_m < \epsilon$. As $h_0 > \epsilon$ $nBUS$ proceeds to $level_1$. At $level_1$, an intermediate failure threshold b_1 is computed such that $nc = p_0 \cdot N$ values of $\{g_0^i\}_{i=1}^N$ exceed its value. The purpose of this is to identify

the samples from $level_0$ to be used as seeds to populate F_1. The resulting sample set chosen based on $\{g_0^i\}_{i=1}^{nc}$ is $\{\boldsymbol{\theta}_0^i, u_0^i\}_{i=1}^{nc}$. Each sample in this set is used as a seed for an individual Markov chain to populate F_1 through the MMH sampler previously discussed. For each of the generated samples in the MMH sampler, the limit state function is evaluated $\{g_1^i\}_{i=1}^{N}$. At step 17 in Algorithm 7 $nBUS$ enters an inner loop to compute h_1 via SuS in Algorithm 8.

Algorithm 7 $nBUS$- Outer Loop

1: Define N, p_0, $nc = p_0 N$, $ns = p_0^{-1}$
2: Define the stopping tolerance ϵ
3: Initialize $m = 0$, where m is the current simulation level
4: Initialize $h_m = 1$
5: Generate N samples $\{\boldsymbol{\theta}_m^i, u_m^i\}_{i=1}^{N}$ from the input PDF $\pi(\cdot)$
6: For each $\{\boldsymbol{\theta}_m^i, u_m^i\}_{i=1}^{N}$ evaluate $g_m^i = \ln L(D|\boldsymbol{\theta}_m^i) - \ln(u_m^i)$
7: Sort the N values of g_m^i and the corresponding $\{\boldsymbol{\theta}_m^i, u_m^i\}_{i=1}^{N}$ in descending order with respect to g_m^i
8: **while** $h_m > \epsilon$ **do**
9: $m = m + 1$
10: Calculate $b_m = \frac{g_{m-1}^{nc} + g_{m-1}^{nc+1}}{2}$
11: Store the first nc samples of the ordered set $\{\boldsymbol{\theta}_{m-1}^i, u_{m-1}^i\}_{i=1}^{N}$ as 'seeds'
12: Using $\{\boldsymbol{\theta}_{m-1}^j, u_{m-1}^j\}_{j=1}^{nc}$ draw the remaining $N - nc$ samples from $\pi(\cdot|F_m)$ via MCMC as follows:
13: **for** $j = 1, ..., nc$ **do**
14: Starting with $\{\boldsymbol{\theta}_{m-1}^j, u_{m-1}^j\}$ as an initial seed, generate $\{\boldsymbol{\theta}_m^k, u_m^k\}_{k=1}^{ns-1} \sim \pi(\cdot|F_m)$ states of a Markov chain using the Modified Metropolis Hastings in Algorithm 5. In this algorithm, the sample acceptance criteria involves evaluating g_m with respect to b_m for each generated sample.
15: **end for**
16: Sort the N values of g_m^i and the corresponding $\{\boldsymbol{\theta}_m^i, u_m^i\}_{i=1}^{N}$ in descending order with respect to g_m^i
17: Compute h_m as in Algorithm 8.
18: **end while**
19: Return $\boldsymbol{\theta} \sim P(\boldsymbol{\theta}|D)$ & $P_D \approx e^{b_m} p_0^m$

Allow for the vector of uncertain parameters used to compute h_1 to be denoted $\boldsymbol{v} \in \mathbb{R}^d$. Let the limit state function $\ln L(D|\boldsymbol{v})$ to be denoted $y(\boldsymbol{v})$ and the intermediate thresholds to be t_m. The reason for this is to avoid any potential confusion in terms of notation with quantities computed in the outer loop of Algorithm 7. In Algorithm 8, $\boldsymbol{v}$ may be viewed as an auxiliary variable with its sole purpose being for the estimation of h_1. The posterior samples $\boldsymbol{\theta}$ are generated in the outer loop of Algorithm 7 only. As in SuS, in Algorithm 8 N samples $\{\boldsymbol{v}_0^i\}_{i=1}^N$ are drawn from the prior before nc members of $\{\boldsymbol{v}_0^i\}_{i=1}^N$ are selected as seeds for nc independent Markov chains used for generating $\{\boldsymbol{v}_1^i\}_{i=1}^N$. At $level_1$, the intermediate threshold t_1 is computed using $\{y_0^i\}_{i=1}^N$. The samples $\{\boldsymbol{v}_1^i\}_{i=1}^N$ are drawn using the $\{\boldsymbol{v}_0^i\}_{i=1}^{nc}$ samples as seeds for the MMH sampler. This step involves the evaluation of $\{y_1^i\}_{i=1}^N$ for identifying whether or not a proposed sample belongs in F_1. Next, the number of failure samples at $level_1$, $n_F(1) = \sum_{i=1}^N (y_1^i > e^{b_1})$ is identified where the target critical threshold b_1 is the current intermediate threshold from Algorithm 7. The value of $h_1 = p_0^1(n_F(1)/N)$ is then estimated. The samples $\{\boldsymbol{v}_1^i\}_{i=1}^N$, the subsequent $\{y_1^i\}_{i=1}^N$ and the estimated h_1 are returned to Algorithm 7. If $h_1 < \epsilon$ the N samples generated at $level_1$ of the $nBUS$ outer loop in Algorithm 7 are drawn from the posterior. If not, $nBUS$ proceeds to $level_2$. The reasoning for returning $\{\boldsymbol{v}_1^i\}_{i=1}^N$ and $\{y_1^i\}_{i=1}^N$ at $level_1$ from Algorithm 8 is for computing h_2. When computing h_2, nc samples from $\{\boldsymbol{v}_1^i\}_{i=1}^N$ are used as seeds for the subsequent Markov chains to produce $\{\boldsymbol{v}_2^i\}_{i=1}^N$. As before, once $\{\boldsymbol{v}_2^i\}_{i=1}^N$ are drawn the number of failure samples at $level_2$, $n_F(2) = \sum_{i=1}^N (y_2^i > e^{b_2})$ is identified where the target critical threshold b_2 is the current intermediate threshold from Algorithm 7. The quantity $h_2 = p_0^2(n_F(2)/N)$ is then estimated and it is checked whether $h_2 < \epsilon$. If so, simulation stops and the samples $\boldsymbol{\theta}$ generated at $level_2$ in the outer loop of $nBUS$ are taken as the posterior set. If not, $nBUS$ progresses to $level_3$. For all $level_m$ where $m > 1$, Algorithm 8 proceeds in the same manner as above until h_m has converged below the chosen stopping tolerance ϵ and the $\boldsymbol{\theta}$ drawn at $level_m$ on the outer loop are the posterior samples. Overall, the nested nature of $nBUS$ results in the posterior samples $\boldsymbol{\theta}$ being produced in the outer loop (Algorithm 7) and h_m in the inner loop (Algorithm 8).

Algorithm 8 *nBUS*- Inner Loop

Define the current intermediate $level_m$
Input the current intermediate threshold value b_m from Algorithm 7
if $m = 1$ **then**
  $m = m - 1$
  Generate N samples $\{\boldsymbol{v}_m^i\}_{i=1}^N$ from the input PDF $\pi(\cdot)$
  For each $\{\boldsymbol{v}_m^i\}_{i=1}^N$ evaluate the limit state function $y_m^i = \ln L(D|\boldsymbol{v}_m^i)$
  Sort the N values of y_m^i and the corresponding $\{\boldsymbol{v}_m^i\}_{i=1}^N$ in descending order with respect to y_m^i
  $m = m + 1$
  Calculate $t_m = \frac{y_{m-1}^{nc} + y_{m-1}^{nc+1}}{2}$
  Store the first nc samples of the ordered set $\{\boldsymbol{v}_{m-1}^i\}_{i=1}^N$ as 'seeds'
  Using $\{\boldsymbol{v}_{m-1}^j\}_{j=1}^{nc}$ draw the remaining $N - nc$ samples from $\pi(\cdot|F_m)$ via MCMC as follows:
  for $j = 1, ..., nc$ **do**
    Starting with $\{\boldsymbol{v}_{m-1}^j\}$ as an initial seed, generate $\{\boldsymbol{v}_m^k\}_{k=1}^{ns} \sim \pi(\cdot|F_m)$ states of a Markov chain using the Modified Metropolis Hastings in Algorithm 5. In this algorithm, the sample acceptance criteria involves evaluating y_m with respect to t_m for each generated sample.
  end for
  Sort the N values of y_m^i and the corresponding $\{\boldsymbol{v}_m^i\}_{i=1}^N$ in descending order with respect to y_m^i
  Set $n_F(m) = \sum_{i=1}^N (y_m^i > e^{b_m})$
else if $m > 1$ **then**
  Define the current intermediate $level_m$
  Input the current intermediate threshold value b_m from Algorithm 7
  Input $\{\boldsymbol{v}_{m-1}^i\}_{i=1}^N$ and y_{m-1}^i
  Calculate $t_m = \frac{y_{m-1}^{nc} + y_{m-1}^{nc+1}}{2}$
  Store the first nc samples of the ordered set $\{\boldsymbol{v}_{m-1}^i\}_{i=1}^N$ as 'seeds'
  Using $\{\boldsymbol{v}_{m-1}^j\}_{j=1}^{nc}$ draw the remaining $N - nc$ samples from $\pi(\cdot|F_m)$ via MCMC as follows:
  for $j = 1, ..., nc$ **do**
    Starting with $\{\boldsymbol{v}_{m-1}^j\}$ as an initial seed, generate $\{\boldsymbol{v}_m^k\}_{k=1}^{ns-1} \sim \pi(\cdot|F_m)$ states of a Markov chain using the Modified Metropolis Hastings in Algorithm 5. In this algorithm, the sample acceptance criteria involves evaluating y_m with respect to t_m for each generated sample.
  end for
  Sort the N values of y_m^i and the corresponding $\{\boldsymbol{v}_m^i\}_{i=1}^N$ in descending order with respect to y_m^i
  Set $n_F(m) = \sum_{i=1}^N (y_m^i > e^{b_m})$
end if
Compute $h_m = p_0^m \frac{n_F(m)}{N}$
Return h_m, $\{\boldsymbol{v}_m^i\}_{i=1}^N$ and $\{y_m^i\}_{i=1}^N$ to Algorithm 7.

4.5 Numerical Applications

This section presents numerical applications of both $nBUS$ and $aBUS$. Each inference framework is firstly implemented on four benchmark examples taken from [32], before being used to estimate classification model parameters for high dimensional problems. The frameworks are analysed in terms of computational expense, posterior statistical estimation and model evidence estimation. The aim of this exercise is two-fold. Firstly, to compare $nBUS$ and $aBUS$ in terms of accuracy and computational efficiency. Secondly, to identify the sensitivity of the performance of $nBUS$ to different stopping tolerances. The stopping tolerances represent the value below which h_m must be for the $nBUS$ sampler to terminate. The chosen values are $\epsilon = 10^{-4}$, $\epsilon = 10^{-6}$ and $\epsilon = 10^{-8}$. For ease of representation each tolerance is denoted by $nBUS_4$, $nBUS_6$ and $nBUS_8$. The level probability p_0 is varied on the interval $[0.1, 0.3]$. The number of samples N is varied on the interval $[10^2, 10^4]$, with its value incrementally increasing. N is initialized as 10^2 and for each subsequent sampler run, N is increased by 10^2 samples before taking a final value of $N = 10^4$. This results in each sampler being implemented for 99 different values of N.

4.5.1 Benchmark Problems

For the sampler comparison, define $\mathbb{E}[\hat{\boldsymbol{\theta}}_1|D]$ and $\sigma[\hat{\boldsymbol{\theta}}_1|D]$ as the expectations of the posterior mean and posterior standard deviation respectively and let $\hat{P}_D$ represent the estimated evidence. The quantities were achieved through a large number of independent runs of each sampler. To aid the analysis of performance, consider the following metrics.

- $e_N = |\frac{\hat{P}_D - P_D}{P_D}|$
- $m_N = |\frac{\mathbb{E}[\hat{\boldsymbol{\theta}}_1|D] - \mathbb{E}[\boldsymbol{\theta}_1|D]}{\mathbb{E}[\boldsymbol{\theta}_1|D]}|$
- $k_N = |\frac{\sigma[\hat{\boldsymbol{\theta}}_1|D] - \sigma[\boldsymbol{\theta}_1|D]}{\sigma[\boldsymbol{\theta}_1|D]}|$

Let e_N, m_N and k_N denote the bias in the estimated model evidence, mean and standard deviation using a set of N posterior samples. Note that e_N, m_N and k_N are random variables for finite N. The bias represents the ratio of deviation of the estimated quantity with respect to the reference value. For the experimental set up, p_0 was varied in the interval $[0.1, 0.3]$ as suggested in [235]. While the number

of samples was incrementally increased from 10^2 to 10^4. The following introduces each benchmark example.

1. Consider two separate 1-dimensional problems. The first with a prior on $\boldsymbol{\theta}$ chosen as $Q(\boldsymbol{\theta}) \sim \mathcal{N}(0,1)$ and the likelihood given by $\mathcal{N}(3, 0.3)$.

2. The second of which defines $Q(\boldsymbol{\theta}) \sim \mathcal{N}(0,1)$ but with a likelihood of $\mathcal{N}(5, 0.2)$. The choice of prior and likelihood ensure an analytical solution is available to provide reference values for the analysis of statistical moment generation during simulation.

3. This problem concerns a posterior distribution which is represented by a 12-dimensional random vector. The prior distribution is $Q(\boldsymbol{\theta})$ is chosen as a multivariate standard normal distribution. The likelihood function may be expressed as

$$L(D|\boldsymbol{\theta}) = \prod_{i=1}^{d} \frac{1}{\sigma} \tau\Big(\frac{\theta_i - \mu}{\sigma}\Big) \tag{4.43}$$

   where $\sigma = 0.6$, $\mu = 0.462$ and τ is the PDF of the univariate standard normal distribution. Given that both the prior and likelihood are Gaussian, the first and second moments of the posterior are analytically tractable.

4. As previously discussed in [67, 208] this problem investigates the application of BUS to a two-storied building which is represented by a two-degree-of freedom shear building model. The objective of BUS is to identify the inter story stiffness parameters $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ which allows for the structural response of the mechanical model to be updated. The problem as originally presented in [20] using MH, models the prior distributions of $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ as the product of two Log normal distributions with modes 1.3 and 0.8 respectively, along with a unit standard deviation. The first and second story masses are given by $16.5 \cdot 10^3$ kg and $16.1 \cdot 10^3$ kg. Inter story stiffness's are specified as $k_1 = \boldsymbol{\theta}_1 \bar{k}_1$ and $k_2 = \boldsymbol{\theta}_2 \bar{k}_2$, where the nominal stiffness values are given by $k_1 = k_2 = 29.7 \cdot 10^6$ N/m. Bayesian updating is carried out having observed the modal data to be $\tilde{f}_1 = 3.13$Hz and $\tilde{f}_2 = 9.83$Hz. The likelihood function is formalised as

$$L(D|\boldsymbol{\theta}) = \exp\left[\frac{-J(\boldsymbol{\theta})}{2\epsilon^2}\right] \tag{4.44}$$

where ϵ represents the standard deviation of the prediction error and modal measure of fit ($J(\boldsymbol{\theta})$) is given by

$$J(\boldsymbol{\theta}) = \sum_{j=1}^{2} \lambda_j^2 \left[\frac{f_j^2(\boldsymbol{\theta})}{\tilde{f}_j^2} - 1 \right]^2 \tag{4.45}$$

Where, λ_1 and λ_2 are weights and $f_1(\boldsymbol{\theta})$ and $f_2(\boldsymbol{\theta})$ are the modal frequencies predicted by the corresponding finite element model.

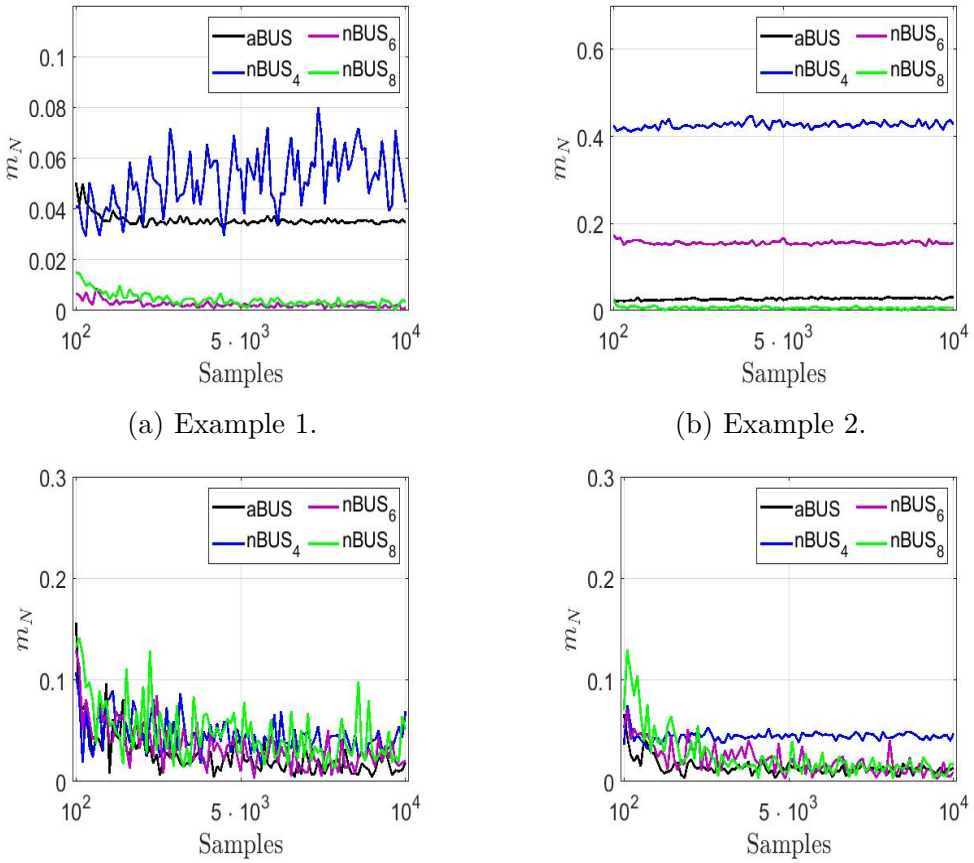
Table 4.1 contains the reference solutions for each problem. All reported results are taken as averages from 100 simulations.

	Example 1	**Example** 2	**Example** 3	**Example** 4
$\boldsymbol{P_D}$	$6.16 \cdot 10^{-3}$	$2.36 \cdot 10^{-6}$	$1 \cdot 10^{-6}$	$1.52 \cdot 10^{-3}$
$\boldsymbol{E[\theta_1 \mid D]}$	2.75	4.81	0.34	1.12
$\boldsymbol{\sigma[\theta_1 \mid D]}$	0.287	0.196	0.51	0.66

Table 4.1: Reference solutions for the four benchmark problems taken from [32].

4.5.2 Statistical and Evidence Estimation

Figure 4.5 and Figure 4.6 present the bias in the expectation and standard deviation of the posterior samples for $\hat{\boldsymbol{\theta}}_1$. For examples 1, 2 and 4, $nBUS_4$ (blue) introduces the largest error rate in statistical estimation. By selecting the stopping tolerance too large ($\epsilon = 10^{-4}$), it is apparent that $nBUS_4$ is terminating simulation too early, resulting in the quality of samples produced deteriorating. This indicates that the probability of drawing a sample from outside the posterior set has not converged to a small enough value when simulation has been terminated. Whereas $nBUS_8$ (green), produces the smallest level of bias in the generated samples. Aside from $nBUS_4$ there is a general decrease in the level of bias in both the expectation and standard deviation for each of the other BUS frameworks for an increasing number of samples. It is worth noting the superior performance of $nBUS_8$ over $aBUS$ (black) for 3 of the 4 benchmark problems.



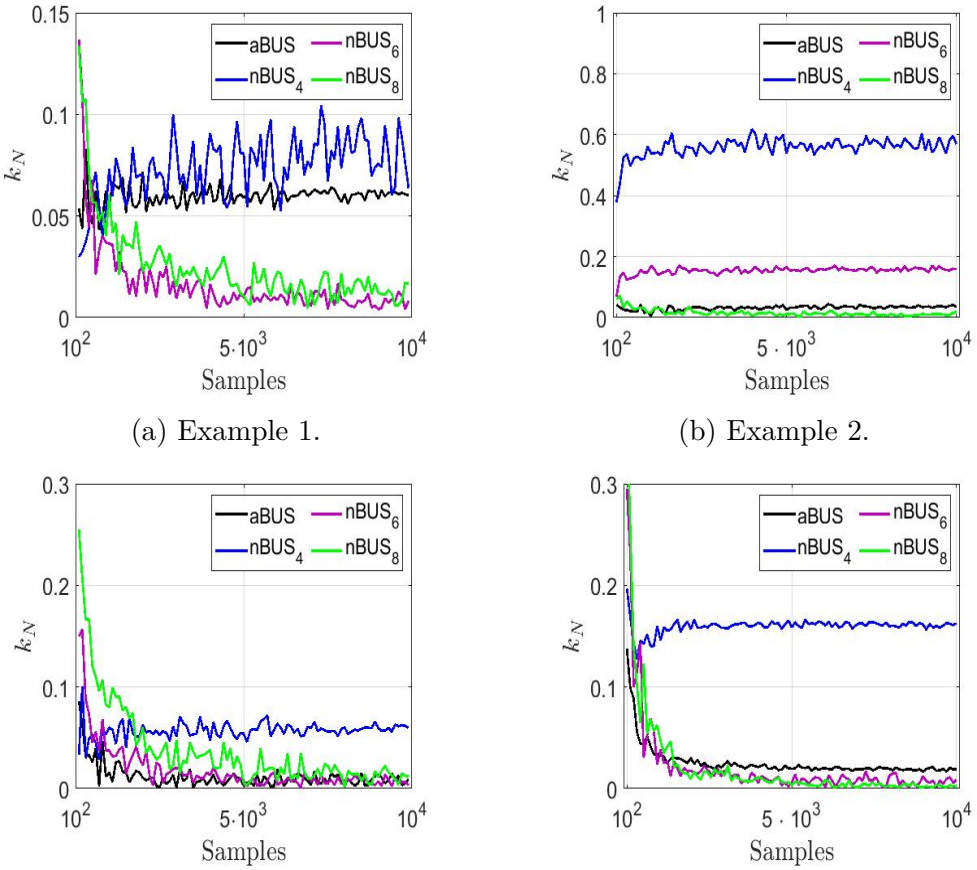
(a) Example 1.

(b) Example 2.

(c) Example 3.

(d) Example 4.

Figure 4.5: Bias in the mean of the posterior samples generated with using $aBUS$ (black), $nBUS_4$ (blue), $nBUS_6$ (purple) and $nBUS_8$ (green).



(a) Example 1.

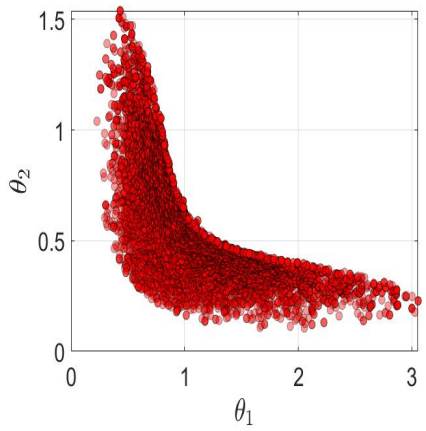
(b) Example 2.

(c) Example 3.

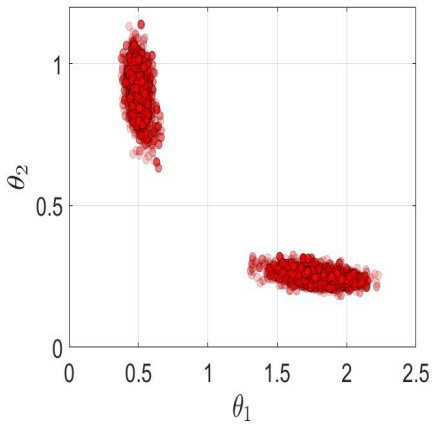
(d) Example 4.

Figure 4.6: Bias in the standard deviation of the posterior samples generated with using $aBUS$ (black), $nBUS_4$ (blue), $nBUS_6$ (purple) and $nBUS_8$ (green).

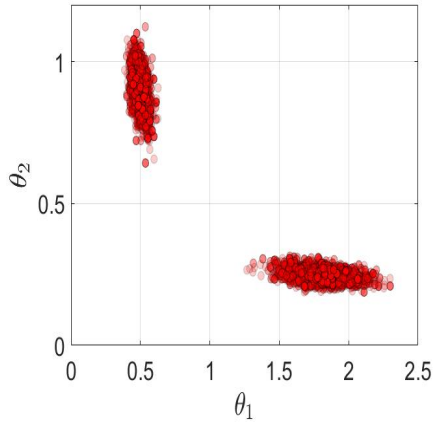
To investigate the influence of ϵ on the sample generation process of $nBUS$, Figure 4.7 presents the generated posterior samples by each of the four samplers for numerical example 4. It is evident that the reasoning behind the high level of bias in the posterior standard deviation estimate by $nBUS_4$ stems from its inability to identify the bi-modal nature of the posterior distribution. In contrast each of $aBUS$, $nBUS_6$ and $nBUS_8$ acknowledge the two modes of the distribution. This highlights the drawback of selecting ϵ too large.



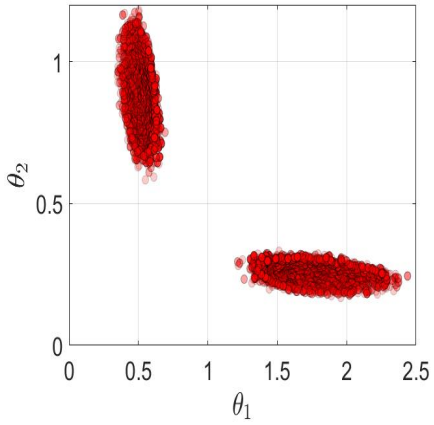
(a) Posterior samples of $nBUS_4$.



(b) Posterior samples of $nBUS_6$.



(c) Posterior samples of $nBUS_8$.

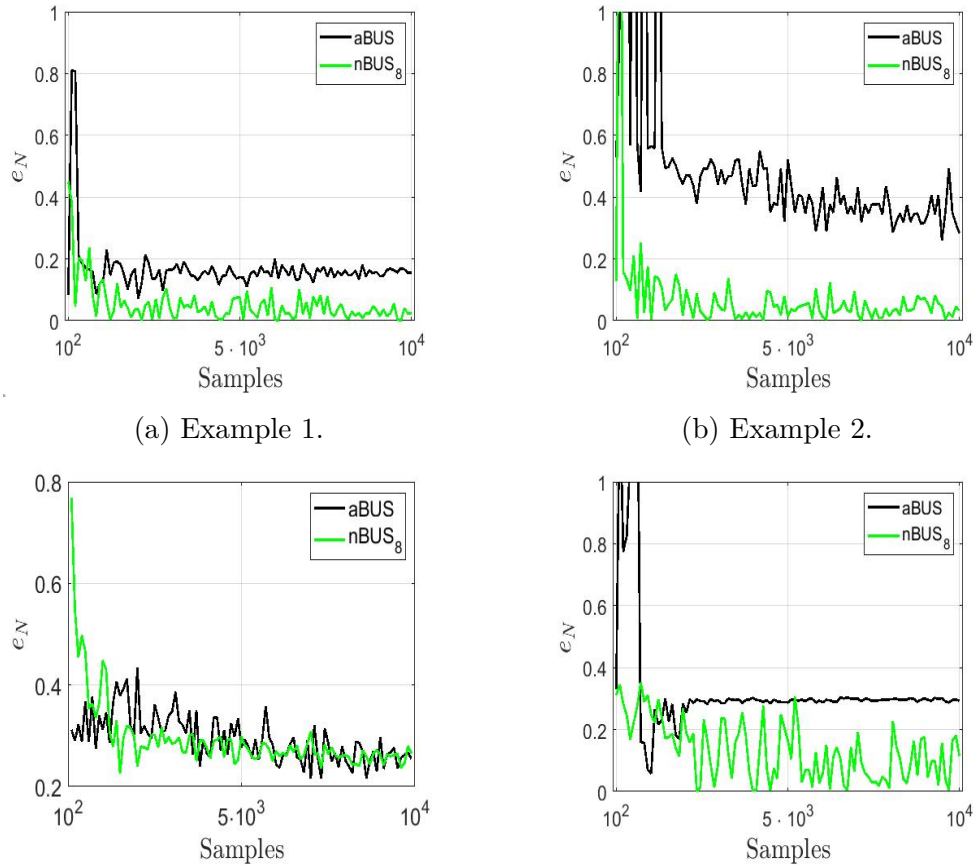


(d) Posterior samples of $aBUS$.

Figure 4.7: Example of sample generation on example 4 for $N = 5000$ and $p_0 = 0.1$. Panel (a) reveals that $nBUS_4$ has terminated the sampling process before the bi-modal nature of the target distribution has been realised. This is the reason for a large level of bias in the estimated standard deviation of the samples. In contrast, $nBUS_6$ (Panel (b)), $nBUS_8$ (Panel (c)) and $aBUS$ (Panel (d)) all populate the two modes of the posterior.

In terms of the model evidence which allows for competing models to be compared against under a Bayesian framework, Figure 4.8 compares the estimated evidence of $aBUS$ against $nBUS_8$. For numerical examples 1, 2 and 4, the model evidence estimated by $nBUS_8$ contains the smaller level of bias. While for numerical example 3 (Figure 4.8 (c)) the level of bias for both samplers is comparable. There is a general decreases in bias for an increasing number of samples. The coefficient of variation (c.o.v) of the model evidence was also computed which also showed a general decrease for an increase in the number of

samples. The analysis of this however, is omitted from this chapter. From both the statistical moment estimation and model evidence estimation for the problems considered, it is apparent that $nBUS$ results in a smaller degree of error in the posterior sample generation. This may stem from conditioning the final samples on the true target failure event.



(a) Example 1.

(b) Example 2.

(c) Example 3.

(d) Example 4.

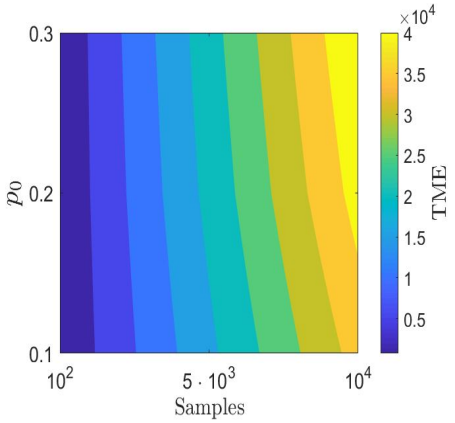
Figure 4.8: Bias in the estimated model evidence using $aBUS$ (black) and $nBUS_8$ (green).

4.5.3 Computational Expense

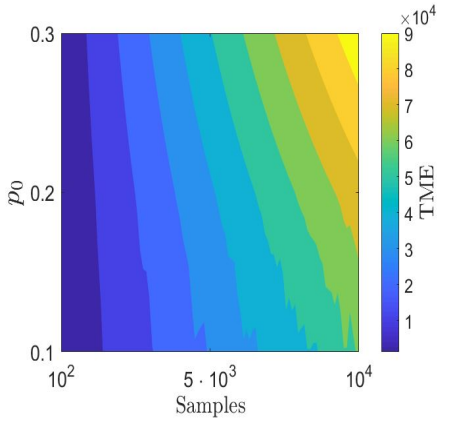
The number of likelihood evaluations acts as a measure of computational expense of the BUS approach. Intuitively, the computational cost of the BUS posterior sampling becomes greater with increasing intermediate levels due to the requirement of evaluating $L(D|\boldsymbol{\theta})$ for each proposed sample during the MCMC step. As such, one aims to sample from the correct probability distribution at the cheapest

possible cost. Total Model Evaluations (TME) represents the number of likelihood function evaluations and acts as a measure of the cost associated with sampling. As the computational cost of each evaluation of the likelihood function increases with the size of a data set, a desirable sampler will require the fewest TME whilst also maintaining accuracy in drawing samples from the posterior. Figure 4.9 presents the number of TME for each of the four samplers using numerical example 1. Note that p_0 is varied on the interval $[0.1, 0.3]$ and N on the interval $[10^2, 10^4]$ with an incremental increase of 10^2 samples. In the case of $N = 5 \times 10^3$, the TME for each framework is $aBUS = 2.15 \times 10^4$, $nBUS_4 = 3.87 \times 10^4$, $nBUS_6 = 6.09 \times 10^4$ and $nBUS_8 = 8.57 \times 10^4$. Intuitively, the smaller the stopping tolerance the more intermediate levels are simulated. Though a smaller tolerance ensures a greater level of confidence of not generating a sample from outside the posterior distribution, Figure 4.9 highlights the added computational expense in doing so. Further plots for the TME required for numerical example 2, 3 and 4 are given in Appendix B. The results however coincide with the findings from example 1.

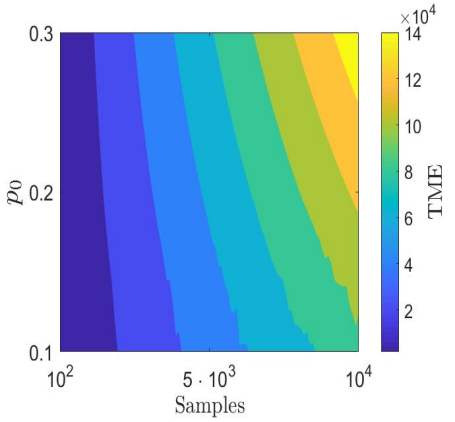
The $nBUS$ sampler requires the most TME for each chosen stopping tolerance due to the nested nature of the stopping condition. At each intermediate level, samples from the failure domain are generated in the outer loop using the limit state function dependent on the rejection principle. Once all N have been drawn, additional TME are required for the inner SuS loop where the probability of producing a draw from outside of the target set is computed. This involves the computation of a separate limit state function. Although this forms a robust stopping criterion, a clear trade off in terms of computational expense appears.



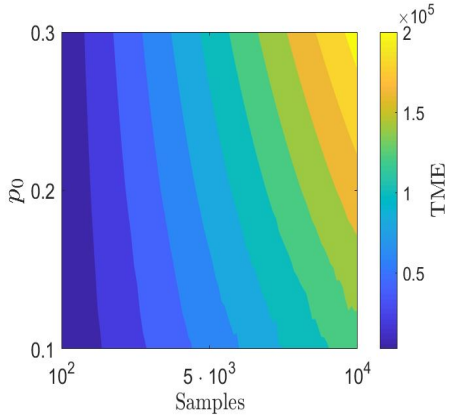
(a) Number of TME for $aBUS$.



(b) Number of TME for $nBUS_4$.



(c) Number of TME for $nBUS_6$.



(d) Number of TME for $nBUS_8$.

Figure 4.9: Illustration of the required TME for $aBUS$, $nBUS_4$, $nBUS_6$ and $nBUS_8$ for drawing posterior samples in example 1. Each value of TME was achieved as an average from 100 independent runs of each sampler.

4.5.4 Hand Written Digits and Image Classification

In order to investigate the applicability of BUS frameworks to examples differing in both dimensionality and the number of available data observations, both $nBUS_8$ and $aBUS$ are applied to two benchmark binary classification tasks. The classification model used is the logistic regression discussed in Chapter 2. The first problem uses the United States Postal Service (USPS) data set [69] as seen in Figure 4.10. This concerns the digital recognition of numbers on hand written envelopes. The data set consists of 10 individual classes in 256 dimensions with 1540 data observations. Each digit is represented by a 16×16 pixelated image. To transform the image pixels into a usable format for logistic regression, the pixel representation is given in vector form of length $16^2 = 256$. Therefore each

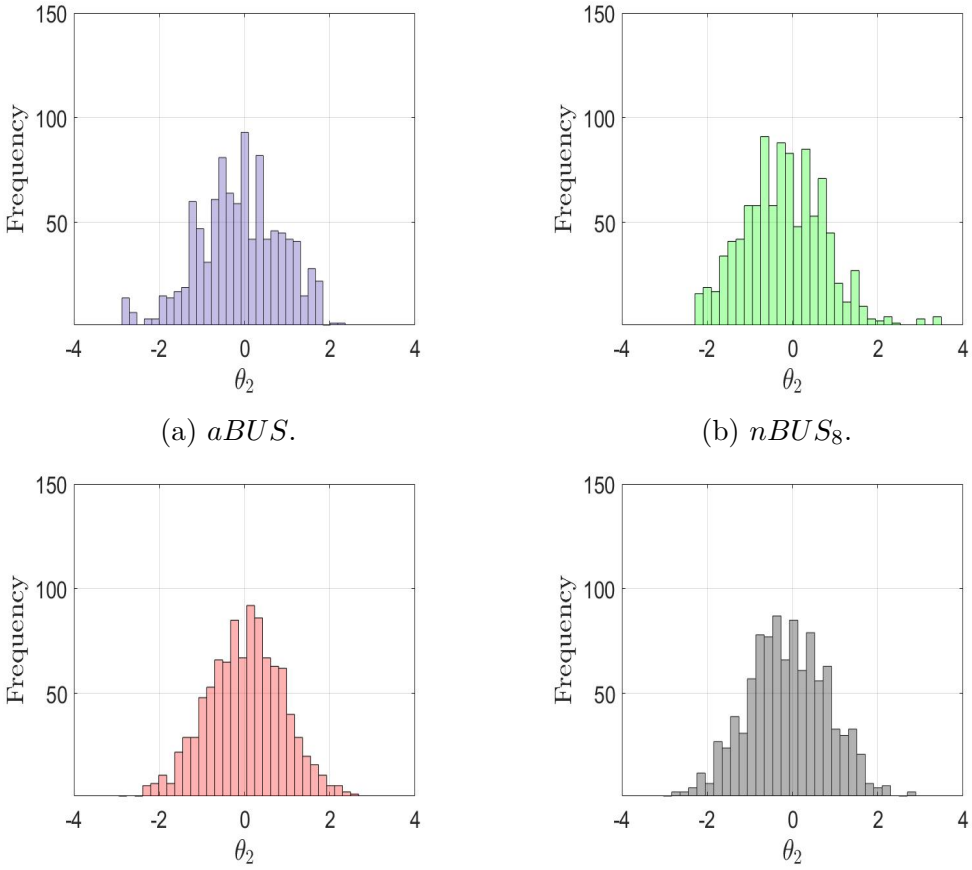
observation of a digit is represented in 256 dimensions. As the digits range from $0-9$, a binary sub-problem from the USPS digit data is defined by considering the problem of discriminating images showing the digits 3 and 5.

Figure 4.10: Examples of handwritten digits from the USPS data set [69].

The second problem deals with the classification of a data set containing 5 different class of image: (i) Hat (ii) Screwdriver (iii) Torch (iv) Rubik Cube and (v) Playing Cards (Figure 4.11). The data set consists of 75 observations, each represented by 4096 dimensions. Pre-processing of the images was carried out in the same manner as the USPS dataset. As the images in this data set are in colour, a colour normalization framework discussed in [173] was utilized to transform them to greyscale format. Each image was made up by 64×64 pixels and is expressed in terms of a vector of length $64^2 = 4096$. For this example the chosen target class is torch. For both examples 5-fold cross validation is implemented. The chosen classifier is the logistic regression model as discussed in Chapter 2. With the misclassification threshold set to 0.5.

Figure 4.11: Examples of images from the 5 classes taken from [65].

This example compares the quality of parameter estimation of Metropolis Hastings (MH), Laplace Approximation (LA) [215], $aBUS$ and $nBUS_8$ for the logistic regression classifier. The proposal for the MH sampler was chosen as a Gaussian centred at the current state of the Markov chain along with a unit step length. Similarly, the component-wise proposals for $aBUS$ and $nBUS_8$ were chosen as univariate Gaussian distributions centred at the current state of the respective Markov chains with a unit variance as suggested in [12]. The Raferty-Lewis metric [48] was implemented to establish a suitable sample burn in period for MH. The resultant burn in length for the USPS set was approximately 1200 samples while the image dataset did not show signs of convergence. The stopping tolerance for Laplace approximation was chosen as 10^{-3}. That is once the change in the moments of the Gaussian approximation between successive iterations is smaller than 10^{-3}, the algorithm stops and draws samples from a Gaussian with moments equal to the final approximated value. Figure 4.12 presents samples drawn by each method from the posterior marginals of θ_2 for the USPS data set. The sample distributions of Laplace and MH are more well peaked and symmetrical in comparison to $aBUS$ and $nBUS_D$. However, overall there is a general agreement between the distribution of samples drawn by all four algorithms. In each case 1000 samples were drawn for the marginals.



(a) $aBUS$.

(b) $nBUS_8$.

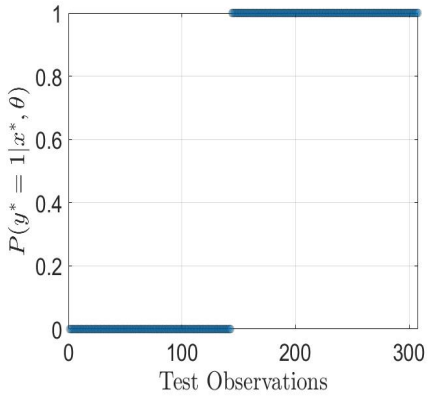
(c) Laplace.

(d) MH.

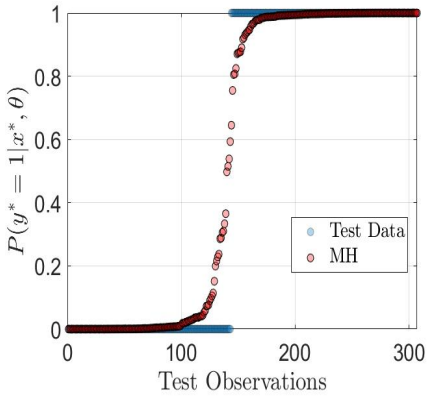
Figure 4.12: A comparison of samples generated from the posterior for θ_2 using the USPS data set. Each of the panels represent the marginal samples drawn by each of the four samplers. Panel (a) presents $aBUS$, panel (b) $nBUS_8$, panel (c) Laplace and panel (d) MH.

Let x^* denote the test observations and y^* the test observation class labels. Figure 4.13 investigates the predictive performance of the logistic regression model with parameters estimated by the four different samplers on the USPS dataset. Figure 4.13 (a) presents the ground truth of the test observations of the USPS data set. Using 5-fold cross validation and given that the USPS data set contains 1540 observations, the resulting test set consists of 308 observations. Figure 4.13 (b) presents the predictive probabilities for the logistic classifier using MH. Each of the remaining samplers is compared using MH as a reference benchmark. It is evident that the predictive probabilities of Laplace (Figure 4.13 (c)) differ greatly from MH. In comparison, $aBUS$ (Figure 4.13 (d)) forms a more accurate representation while the predictions of the classifier with parameters estimated by $nBUS_8$ (Figure

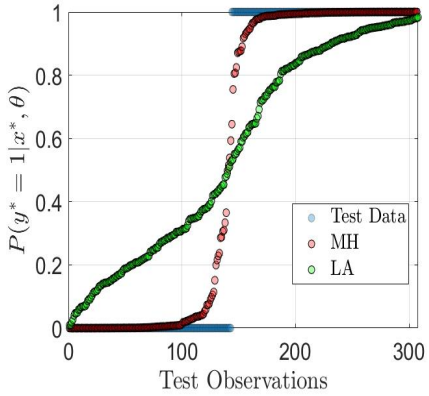
4.13 (d)) closely follow MH. In Table 4.2, the predictive performance is compared in terms of the Area Under the Curve (AUC) performance metric (see Chapter 2). The term in brackets denotes the standard error and CI the subsequent confidence interval. The quality of performance of each method is comparable. For the USPS data set, LA produces the largest standard error value which reveals that the variation in AUC for each cross validation fold is representative of the classification models lack of confidence in assigning class labels. Focusing on the performance of the BUS frameworks, with respect to $aBUS$, $nBUS_8$ results in a marginal better AUC rate and results in a better general predictive performance as shown in Figure 4.12. In terms of computational expense for USPS, $aBUS$ requires 2.8×10^3 TME with $nBUS_8$ requiring 1.3×10^4. In the case of the image problem, the TME are 4.9×10^3 and 1.3×10^4 for $aBUS$ and $nBUS_8$ respectively.



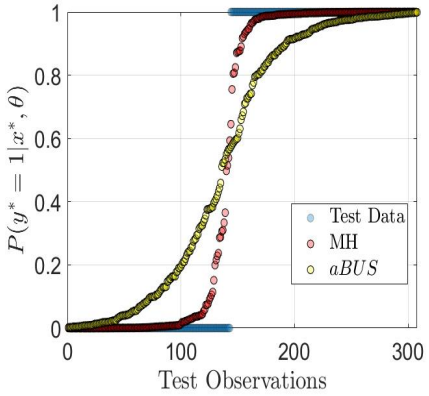
(a) Correct Classification.



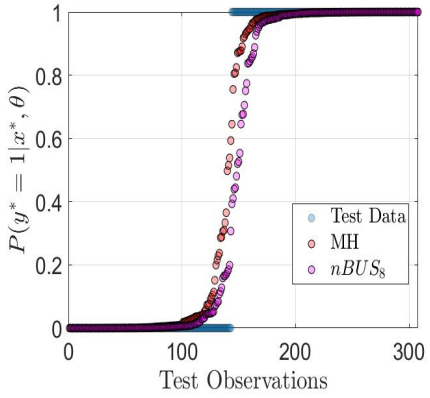
(b) MH.



(c) Laplace.



(d) $aBUS$.



(e) $nBUS_8$.

Figure 4.13: Predictive performance of logistic regression on the USPS data set using the four different samplers for parameter estimation. Panel (a) shows the ground truth, panel (b) MH, panel (c) Laplace, panel (d) $aBUS$ and panel (e) $nBUS_8$.

	USPS	**Images**
aBUS	0.96 (0.08)	0.94 (0.01)
CI	$(0.82, 0.99)$	$(0.91, 0.98)$
LA	0.96 (0.59)	0.94 (0.02)
CI	$(0.79, 1)$	$(0.91, 0.95)$
MH	0.98 (0.02)	0.93 (0.08)
CI	$(0.88, 1)$	$(0.90, 0.95)$
$\boldsymbol{nBUS_8}$	0.97 (0.08)	0.95 (0.01)
CI	$(0.85, 1)$	$(0.93, 0.98)$

Table 4.2: Performance metrics of MH, LA, $aBUS$, $nBUS_8$ for the USPS and image data set problems.

4.6 Chapter Summary

This chapter introduces reliability analysis along with the BUS framework for Bayesian updating tasks. BUS acknowledges the shortcomings of the rejection sampling algorithm discussed in Chapter 3, to efficiently sample from posterior distributions using rare event simulation techniques such as SuS. In its original form however, an input parameter which directly determines the distribution of the samples is required to be determined prior to simulation. Two solutions in the form of adaptive BUS ($aBUS$) and nested BUS ($nBUS$) have been presented which differ in both input parameter choice and stopping criteria.

Although $nBUS$ conditions the samples on the true failure event along with avoiding underestimating the model evidence, the automatic nested stopping criterion presents two issues. Firstly, choosing a stopping tolerance for the probability of generating a sample outside the target set requires careful thought as illustrated in Figure 4.5 and Figure 4.6. If chosen too large $nBUS$ will terminate simulation too early whilst a value too small will result in additional intermediate levels being required for simulation.

Secondly, the current stopping criteria greatly increases the computational cost of a sampler run. Solving two different reliability problems results in the subsequent $\boldsymbol{\theta}$ in the inner and outer SuS loops being conditional on different failure events. This means that the requirement of the nested SuS loop to compute additional $L(D|\boldsymbol{\theta})$ evaluations hinders the computational efficiency of the approach in comparison to $aBUS$ as seen by Figure 4.9 and the machine learning application. To allow for $nBUS$ to retain its advantages over $aBUS$ while also being scalable to real life machine learning tasks, two new stopping criteria aimed at reducing the number of likelihood evaluations in a sampler run are proposed in Chapter 5.

Chapter 5

New BUS Stopping Conditions

Having theoretically and numerically investigated the advantages of nested BUS ($nBUS$) over adaptive BUS ($aBUS$) in terms of sample quality and model evidence estimation, the added computational expense stemming from the nested stopping condition remains a major issue. To alleviate the increased number of model evaluations required for a sampler run along with maintaining the frameworks advantages, two alternative stopping criteria are presented in this chapter. The first criterion concerns the direct application of the rejection principle in areas of the sample space which are outside the target failure domain. In turn, reducing the number of likelihood evaluations by ensuring the nested loop is not computed at each intermediate level. The second criterion, totally avoids the use of the nested loop by exploiting the transition in the relationship between the model evidence and failure probability as discussed in Chapter 4. By accounting for the stochastic nature of the model evidence through the introduction of an entire distribution representing the probability of failure, BUS is not required to be run several times to confidently ensure simulation is correctly terminated. Both criteria are compared against $nBUS$ with a stopping tolerance of 10^{-8} ($nBUS_8$). Numerical experiments are performed using the benchmark and machine learning examples presented in Chapter 4. As the MCMC scheme implemented in BUS directly influences sample quality and the efficiency of sample generation, to further improve the sampling process a comparison of the performance of a number of MCMC variants specifically designed for SuS is also presented.

5.1 Direct Stopping

The first proposed stopping criterion reduces the number of likelihood function calls through the direct application of the rejection principle. At $level_m$ of $nBUS$ consider the rejection principle to be:

$$e^{-b_m} L(D|\boldsymbol{\theta}) \leq 1 \tag{5.1}$$

As discussed in Chapter 4, the task required by $nBUS$ is to identify when the generated intermediate threshold level b_m is large enough resulting in $\boldsymbol{\theta} \sim P(\boldsymbol{\theta}|D)$. Consider again the inadmissible set of samples generated at $level_m$ of $nBUS$.

$$A_m(\boldsymbol{\theta}) = \{\boldsymbol{\theta} : e^{-b_m} L(D|\boldsymbol{\theta}) > 1\} \tag{5.2}$$

Where $A_m(\boldsymbol{\theta})$ is the sample set outside the posterior set. It is observed in [67], that given an increasing sequence of failure levels, the sequence of inadmissible sets is monotone decreasing.

$$A_m \supset A_{m+1} \supset ... \supset \emptyset \tag{5.3}$$

Meaning for a large enough number of intermediate levels, the set of generated samples belonging outside the posterior will be empty. The original $nBUS$ stopping criterion exploits this property by computing $h_m = p_{\boldsymbol{\theta}}(L(D|\boldsymbol{\theta}) > e^{b_m})$ at each intermediate level. Once this probability has converged to a value lower than a very small tolerance ϵ, $nBUS$ generates $\boldsymbol{\theta} \sim P(\boldsymbol{\theta}|D)$. As discussed in Chapter 4, calculating h_m involves solving a separate reliability problem which can become very expensive to compute for a complex model. To alleviate this issue, consider the inequality determining membership of set $A_m(\boldsymbol{\theta})$.

$$e^{-b_m} L(D|\boldsymbol{\theta}) > 1 \tag{5.4}$$

Eq. 5.4 indicates that any sample satisfying this inequality has violated the rejection principle in Eq. 5.1 and is thus outside the posterior set. For an increasing number of intermediate levels, it is known that the number of samples satisfying Eq. 5.4 will be zero once the posterior distribution has been reached.

For all $\{\boldsymbol{\theta}_m^i\}_{i=1}^N$, the proposed stopping criterion directly computes the following:

$$A_m = \sum_{i=1}^{N} (e^{-b_m} L(D|\boldsymbol{\theta}_m^i) > 1) \tag{5.5}$$

If all N samples are rejected by Eq. 5.4, A_m will be zero and the subsequent samples will be from the posterior set. The resulting sampler is termed BUS_D due to the direct application of the rejection principle. Algorithm 9 presents a pseudo-code of BUS_D. At $level_0$, A_0 is assigned a default value of N as the samples are drawn from the prior. Next, the limit state function for the N samples is evaluated and the sampler progresses to $level_1$. Once N samples at $level_1$ are generated via the MMH algorithm, $A_1 = \sum_{i=1}^{N} (e^{-b_1} L(D|\boldsymbol{\theta}_1^i) > 1)$ is computed at step 16 for all $\{\boldsymbol{\theta}_1^i\}_{i=1}^N$. If $A_1 = 0$, simulation stops as all N samples generated at $level_1$ have been rejected as being members of the inadmissible set of samples. If $A_1 > 0$, BUS_D progresses to $level_2$. The simulation of intermediate levels of BUS_D is continued until $A_m = 0$. Once $A_m = 0$ for a given $level_m$ the sampler returns $\boldsymbol{\theta}_m \sim P(\boldsymbol{\theta}|D)$.

Algorithm 9 BUS_D

1: Define N, p_0, $nc = p_0 N$ and $ns = p_0^{-1}$
2: Initialize $m = 0$, where m is the current simulation level
3: Initialize $A_m = N$
4: Generate N samples $\{\boldsymbol{\theta}_m^i, u_m^i\}_{i=1}^N$ from the input PDF $\pi(\cdot)$
5: For each $\{\boldsymbol{\theta}_m^i, u_m^i\}_{i=1}^N$ evaluate $g_m^i = \ln L(D|\boldsymbol{\theta}_m^i) - \ln(u_m^i)$
6: Sort the N values of g_m^i and the corresponding $\{\boldsymbol{\theta}_m^i, u_m^i\}_{i=1}^N$ in descending order with respect to g_m^i
7: **while** $A_m > 0$ **do**
8: $m = m + 1$
9: Calculate $b_m = \frac{g_{m-1}^{nc} + g_{m-1}^{nc+1}}{2}$
10: Store the first nc samples of the ordered set $\{\boldsymbol{\theta}_{m-1}^i, u_{m-1}^i\}_{i=1}^N$ as 'seeds'
11: Using $\{\boldsymbol{\theta}_{m-1}^j, u_{m-1}^j\}_{j=1}^{nc}$ draw the remaining $N - nc$ samples from $\pi(\cdot|F_m)$ via MCMC as follows:
12: **for** $j = 1, ..., nc$ **do**
13: Starting with $\{\boldsymbol{\theta}_{m-1}^j, u_{m-1}^j\}$ as an initial seed, generate $\{\boldsymbol{\theta}_m^k, u_m^k\}_{k=1}^{ns-1} \sim \pi(\cdot|F_m)$ states of a Markov chain using the Modified Metropolis Hastings in Algorithm 5. In this algorithm, the sample acceptance criteria involves evaluating g_m with respect to b_m for each generated sample.
14: **end for**
15: Sort the N values of g_m^i and the corresponding $\{\boldsymbol{\theta}_m^i, u_m^i\}_{i=1}^N$ in descending order with respect to g_m^i
16: $A_m = \sum_{i=1}^N (e^{-b_m} L(D|\boldsymbol{\theta}_m^i) > 1)$ for all $\{\boldsymbol{\theta}_m^i\}_{i=1}^N$
17: **end while**
18: Return $\boldsymbol{\theta} \sim P(\boldsymbol{\theta}|D)$ & $P_D \approx e^{b_m} p_0^m$

5.1.1 Progression of Likelihood Multiplier

This section discusses the behaviour of the samples generated at each intermediate level of BUS_D prior to the target distribution being reached. By

$$e^{-b_m} L(D|\boldsymbol{\theta}) > 1 \tag{5.6}$$

samples generated at $level_m$ will either be accepted or rejected with respect to e^{-b_m}. To illustrate this property, consider benchmark problem 4 discussed in Chapter 4 concerning the identification of stiffness parameters of an inter story

building. This posterior distribution is known to be bi-modal. Figure 5.1 contains the samples drawn at $level_m$. Given that a proportion of samples are accepted (black) while the remaining being rejected (grey), another intermediate level is required. The minimum acceptable intermediate threshold b_{min} for all $\boldsymbol{\theta}$ to ensure

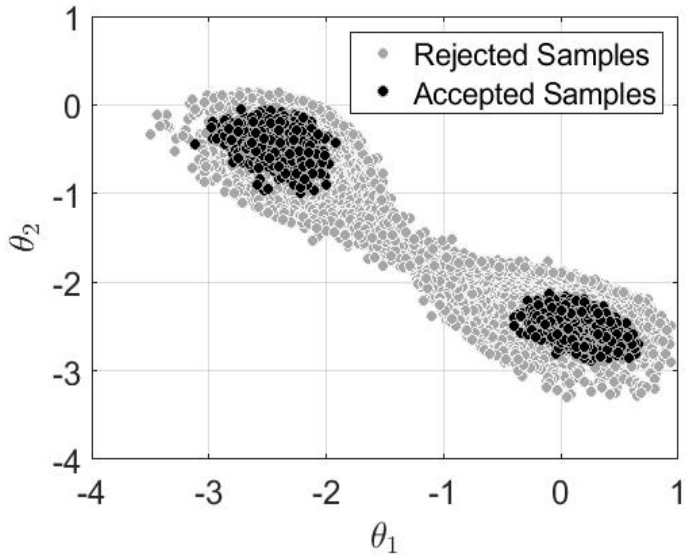


Figure 5.1: Samples generated by BUS_D for benchmark example 4 from Chapter 4. The bi-modal nature of the samples is beginning to appear with the rejected and accepted samples being represented by grey and black respectively.

the rejection principle holds is given by

$$b_{min} = \ln(L(D|\boldsymbol{\theta})_{max}) \tag{5.7}$$

Define $level_{min}$ as the minimum required level to generate samples from the posterior. For any $level_m < level_{min}$, given that $b_m < b_{min}$ the following holds:

$$b_m < \ln(L(D|\boldsymbol{\theta})_{max}) \tag{5.8}$$

Simply put b_m has not yet converged to a value large enough to satisfy the rejection principle for all $\boldsymbol{\theta}$. Therefore, for any $level_m < level_{min}$ we know that the selected b_m is too small. The samples rejected at each level prior to $level_{min}$ belong in the region of the highest likelihood. Under BUS_D one would expect an increase in intermediate levels to result in an increase in $L(D|\boldsymbol{\theta})$ as we tend towards the target failure domain. Stemming from the gradual approach of SuS towards the

failure domain, the generated $\boldsymbol{\theta}$ have an increasing probability of having generated the data with the theoretical maximum given by $L(D|\boldsymbol{\theta})_{max} = 1$.

Consider the set of samples generated at $level_m$ as in Figure 5.1. Seeing as those which best describe the data have not satisfied Eq. 5.6, b_m is deemed too small. This results in the set of rejected samples containing bias. Regarding the accepted samples satisfying Eq. 5.6, they have done so with an inadequate choice of b_m. Meaning the subsequent $\boldsymbol{\theta}$ are incorrectly accepted too often as the conditional failure distribution is too close to the prior. Despite Eq. 5.6 holding for some $\boldsymbol{\theta}$, the intermediate failure threshold has yet to converge to a large enough value. In this scenario, additional intermediate levels are required until all samples have been rejected with respect to Eq. 5.6.

5.1.2 Sampling Error

By identifying $b_m > b_{min}$, BUS_D ensures all N samples generated at $level_m$ satisfy the rejection principle. A possible drawback of BUS_D however is the finite nature of the sample set. Suppose due to sampling error and unknown to the sampler, the $(N+1)^{th}$ sample to be drawn at $level_m$ will not satisfy the rejection principle for the selected b_m . In its current form BUS_D provides no information beyond the finite set of samples. As such, the direct application of the rejection principle does not offer statistical guarantees that any subsequent samples beyond the current generated sample set will also follow the posterior distribution. To address this issue, the direct criterion is merged with the nested loop of $nBUS$ presented in Section 4.4.2. At $level_m$, as with BUD_D

$$A_m = \sum_{i=1}^{N} (e^{-b_m} L(D|\boldsymbol{\theta}_m^i) > 1) \tag{5.9}$$

is computed for all $\{\boldsymbol{\theta}_m^i\}_{i=1}^N$. Once $A_m = 0$, BUS_D is deemed to be generating posterior samples. To offer statistical assurances in terms of possible sampling error of BUS_D, once $A_m = 0$ it is proposed to compute h_m.

$$h_m = p_{\boldsymbol{\theta}}(L(D|\boldsymbol{\theta}) > e^{b_m}) \tag{5.10}$$

As previously outlined in Chapter 4, computing h_m at every intermediate level is computationally expensive. By only computing h_m once $A_m = 0$, additional unnecessary likelihood function calls have been avoided for all levels prior to

$level_m$. While A_m may be viewed as a computationally cheap proxy for h_m, the nested loop in return provides a robust assurance of stopping by computing Eq. 5.10. Then as with $nBUS$, if $h_m < \epsilon$ where ϵ is the chosen stopping tolerance, simulation stops as the probability of generating a sample from outside the target set is extremely small. If this is not the case, additional levels are generated until $h_m < \epsilon$ holds true. The subsequent stopping criterion is named $nBUS_D$. A pseudo-code for $nBUS_D$ is given in Algorithm 10. At $level_0$, as the samples are drawn from the prior, h_m is assigned a default value of 1 and A_m a default value of N. At $level_1$, the sampler generates N samples in F_1 by MMH. Next, it is checked whether $\{\boldsymbol{\theta}_1^i\}_{i=1}^N$ have resulted in $A_1 = 0$ in step 19. If $A_1 = 0$, h_1 is computed in the same manner as in $nBUS$ as outlined previously in Algorithm 8. It is then checked whether or not $h_1 < \epsilon$. If so, $\{\boldsymbol{\theta}_1^i\}_{i=1}^N$ are returned as the posterior set and if not $nBUS_D$ progresses to $level_2$. This is continued until for any m, $h_m < \epsilon$. Had $A_1 > 0$, the computation of h_1 would have been avoided and $nBUS_D$ would have proceeded directly to $level_2$. Therefore, h_m is is only computed once $A_m = 0$.

Algorithm 10 $nBUS_D$

Define N, p_0, $nc = p_0 N$, and $ns = p_0^{-1}$
Define the stopping tolerance ϵ
Initialize $m = 0$, where m is the current simulation level
Initialize $A_m = N$
Initialize $h_m = 1$
Generate N samples $\{\boldsymbol{\theta}_m^i, u_m^i\}_{i=1}^N$ from the input PDF $\pi(\cdot)$
For each $\{\boldsymbol{\theta}_m^i, u_m^i\}_{i=1}^N$ evaluate $g_m^i = \ln L(D|\boldsymbol{\theta}_m^i) - \ln(u_m^i)$
Sort the N values of g_m^i and the corresponding $\{\boldsymbol{\theta}_m^i, u_m^i\}_{i=1}^N$ in descending order with respect to g_m^i
while $h_m > \epsilon$ **do**
    $m = m + 1$
    Calculate $b_m = \frac{g_{m-1}^{nc} + g_{m-1}^{nc+1}}{2}$
    Store the first nc samples of the ordered set $\{\boldsymbol{\theta}_{m-1}^i, u_{m-1}^i\}_{i=1}^N$ as 'seeds'
    Using $\{\boldsymbol{\theta}_{m-1}^j, u_{m-1}^j\}_{j=1}^{nc}$ draw the remaining $N - nc$ samples from $\pi(\cdot|F_m)$ via MCMC as follows:
    for $j = 1, ..., nc$ **do**
        Starting with $\{\boldsymbol{\theta}_{m-1}^j, u_{m-1}^j\}$ as an initial seed, generate $\{\boldsymbol{\theta}_m^k, u_m^k\}_{k=1}^{ns-1} \sim \pi(\cdot|F_m)$ states of a Markov chain using the Modified Metropolis Hastings in Algorithm 5. In this algorithm, the sample acceptance criteria involves evaluating g_m with respect to b_m for each generated sample.
    end for
    Sort the N values of g_m^i and the corresponding $\{\boldsymbol{\theta}_m^i, u_m^i\}_{i=1}^N$ in descending order with respect to g_m^i
    $A_m = \sum_{i=1}^N (e^{-b_m} L(D|\boldsymbol{\theta}_m^i) > 1)$ for all $\{\boldsymbol{\theta}_m^i\}_{i=1}^N$
    if $A_m = 0$ **then**
        Compute h_m as in Algorithm 8
    else
        $h_m = 1$
    end if
end while
Return $\boldsymbol{\theta} \sim P(\boldsymbol{\theta}|D)$ & $P_D \approx e^{b_m} p_0^m$

5.2 Robust Stopping

Having reduced the computational expense associated with $nBUS$ through the introduction of $nBUS_D$, this section aims to further enhance algorithmic efficiency by avoiding the computation of the nested loop at all simulation levels. Recall that the relationship between the model evidence P_D and failure probability p_F for $nBUS$

$$P_D = e^{b_m} p_F \qquad\qquad b_m > b_{min} \tag{5.11}$$

holds only in the event of an intermediate threshold at $level_m$ greater than the minimum required value being generated. The proposed stopping criterion exploits this relationship by tracking the model evidence produced by $nBUS$ at each intermediate level. Once the relationship is satisfied, the resulting samples are conditional on the target failure event. As highlighted in Chapter 4 however, this quantity is available on a sample basis only, and as such contains statistical error.

$$P_D \approx \hat{P}_D = e^{b_m} p_0^m \tag{5.12}$$

Furthermore, p_F in SuS is a point estimate with no uncertainty measurements readily available. To address this issue, [235] proposed a Bayesian approach to generate a posterior PDF for p_F using sample information at each intermediate level. The subsequent PDF quantifies the uncertainty in the SuS estimations of the probability of failure. This uncertainty takes into account both prior information and information contained in the samples produced.

5.2.1 Bayesian Post Processor for Subset Simulation

Under SuS the failure probability is represented as a product of conditional probabilities $p_m = p(F_m|F_{m-1})$, each of which is estimated by

$$p_m \approx \hat{p}_m = \frac{1}{N} \sum_{i=1}^{N} I[\boldsymbol{\theta}_{m-1}^{(i)}] \tag{5.13}$$

At the end of simulation, SuS combines the intermediate probabilities to form an estimate for the sample failure probability. Given b_m is chosen such that nc limit state function outputs exceed its value, allow the indicator sum in Eq. 5.13 to be

replaced by $\{p_0 \cdot N\}$. The final p_F estimate is given by

$$p_F \approx \hat{p}_F = \prod_{i=1}^{m} \hat{p}_m = (p_0)^m \tag{5.14}$$

Zuev et al. [235] substitute the frequentist estimates of Eq. 5.13 and Eq. 5.14 by their Bayesian counterparts. The prior is chosen as a uniform on $[0, 1]$ while a sensible choice of likelihood from the indicator function is the binomial distribution. Conditioning the intermediate probabilities on information contained in the number of failure sample results in a beta distribution with parameters α and β.

$$P(p_m|D_{m-1}) = \frac{p_m^{p_0 \cdot N}(1-p_m)^{N-p_0 \cdot N}}{B(p_0 \cdot N + 1, N - p_0 \cdot N + 1)} \tag{5.15}$$

where D_{m-1} is the observed failure sample data from $level_{m-1}$ and $B(\cdot)$ the beta function. The MAP value (or mode) of this beta PDF is given by

$$\begin{aligned} P(p_m|D_{m-1})_{MAP} &= \frac{p_0 \cdot N + 1 - 1}{p_0 \cdot N + 1 + N - p_0 \cdot N + 1 - 2} \\ &= p_0 \end{aligned} \tag{5.16}$$

This is simply the level probability. It is worth noting that Eq. 5.15 assumes the subsequent MCMC samples are independent. Since m is the total number of SuS levels, constructing a distribution to replace Eq. 5.14 requires taking the product of m beta distributions. An exact representation for the PDF of a product of independent beta variables had been proposed [212]. However, this derivation requires the computation of an infinite sum which must be approximated. An alternative route is to approximate the product of m independent beta variables with a single beta variable such that the corresponding moments match by choice of the beta shape parameters α and β [217]. This simplified approach has been shown [76] to accurately approximate the target density even if the product of beta variables does not itself follow a beta distribution. The authors of [76] proved this by the following theorem.

Theorem 2. *[76] Let $X_1, ..., X_m$ be independent beta variables, $X_i \sim B(\alpha_i, \beta_i)$ and $P_F = \prod_{i=1}^{m} X_i$. P_F is approximately distributed as $\hat{P}_F \sim B(\alpha, \beta)$ with α and*

β given by

$$\alpha = \mu_1 \frac{\mu_1 - \mu_2}{\mu_2 - \mu_1^2} \qquad \beta = (1 - \mu_1)\frac{\mu_1 - \mu_2}{\mu_2 - \mu_1^2}$$

$$\mu_1 = \mathbb{E}[\hat{P}_F] = \prod_{i=1}^{m} \frac{\alpha_i}{\alpha_i + \beta_i} \qquad \mu_2 = \mathbb{E}[\hat{P}_F^2] = \prod_{i=1}^{m} \frac{\alpha_i(1 - \alpha_i)}{(\alpha_i + \beta_i)(\alpha_i + \beta_i + 1)}$$

A proof of Theorem 2 is outlined in [76]. Using the result of Theorem 2, Zuev et al. later derived expressions for α and β in the context of *SuS* [235] where $\hat{P}_F$ is the posterior PDF of the failure probability and $\hat{P}_F \sim B(\alpha, \beta)$. The resulting α and β expressions for generating a posterior distribution of the failure probability are given by

$$\alpha = \frac{(\frac{p_0 \cdot N+1}{N+2})^m \cdot (1 - (\frac{p_0 \cdot N+2}{N+3})^m)}{(\frac{p_0 \cdot N+2}{N+3})^m - (\frac{p_0 \cdot N+1}{N+2})^m}$$

$$\beta = \frac{(1 - (\frac{p_0 \cdot N+1}{N+2})^m) \cdot (1 - (\frac{p_0 \cdot N+2}{N+3})^m)}{(\frac{p_0 \cdot N+2}{N+3})^m - (\frac{p_0 \cdot N+1}{N+2})^m} \tag{5.17}$$

With m denoting the number of intermediate levels of *SuS*, N the number of samples and p_0 the level probability. To ensure that the beta PDF defined by these expressions will be bounded, the following presents a proposition which proves that both α and β in Eq. 5.17 are greater than 1.

Proposition 1. *If $\hat{P}_F \sim B(\alpha, \beta)$ with α and β defined as*

$$\alpha = \frac{(\frac{p_0 \cdot N+1}{N+2})^m \cdot (1 - (\frac{p_0 \cdot N+2}{N+3})^m)}{(\frac{p_0 \cdot N+2}{N+3})^m - (\frac{p_0 \cdot N+1}{N+2})^m} \qquad \beta = \frac{(1 - (\frac{p_0 \cdot N+1}{N+2})^m) \cdot (1 - (\frac{p_0 \cdot N+2}{N+3})^m)}{(\frac{p_0 \cdot N+2}{N+3})^m - (\frac{p_0 \cdot N+1}{N+2})^m}$$

then α and β are both greater than 1.

To prove $\alpha > 1$ and $\beta > 1$, it is suffices to show that the numerator is greater than the denominator in the expressions of Eq. 5.17. Consequently for α it is required to prove that

$$\left(\frac{p_0 \cdot N+1}{N+2}\right)^m \cdot \left(1 - \left(\frac{p_0 \cdot N+2}{N+3}\right)^m\right) > \left(\frac{p_0 \cdot N+2}{N+3}\right)^m - \left(\frac{p_0 \cdot N+1}{N+2}\right)^m \tag{5.18}$$

Similarly for β it is required to prove that

$$\left(1-\left(\frac{p_0\cdot N+1}{N+2}\right)^m\right)\cdot\left(1-\left(\frac{p_0\cdot N+2}{N+3}\right)^m\right)>\left(\frac{p_0\cdot N+2}{N+3}\right)^m-\left(\frac{p_0\cdot N+1}{N+2}\right)^m \tag{5.19}$$

In terms of notation in the proof, let the LHS denote the left hand side of the expressions of Eq. 5.18 and Eq. 5.19. Specifically when proving $\alpha > 1$ LHS refers to the left hand side of Eq. 5.18 and for $\beta > 1$ LHS refers to the left hand side of Eq. 5.19. Similarly, when proving $\alpha > 1$ RHS refers to the right hand side of Eq. 5.18 and for $\beta > 1$ RHS refers to the right hand side of Eq. 5.19. Let N denote the number of generated samples, p_0 the level probability and m the number of intermediate levels.

Proof. Firstly, consider the case of α. Let the LHS be given by

$$\text{LHS}=\left(\frac{(p_0\cdot N+1)(N+3)}{(N+2)(N+3)}\right)^m-\left(\frac{(p_0\cdot N+1)(p_0\cdot N+2)}{(N+2)(N+3)}\right)^m \tag{5.20}$$

On the other hand let the RHS be given by,

$$\text{RHS}=\left(\frac{(p_0\cdot N+2)(N+2)-(p_0\cdot N+1)(N+3)}{(N+2)(N+3)}\right)^m \tag{5.21}$$

To prove that the numerator is indeed greater than the denominator of the α expression, setting the LHS > RHS and using the above expressions implies that

$$\left(\frac{(p_0\cdot N+1)(N+3)}{(N+2)(N+3)}\right)^m-\left(\frac{(p_0\cdot N+1)(p_0\cdot N+2)}{(N+2)(N+3)}\right)^m>\left(\frac{(p_0\cdot N+2)(N+2)-(p_0\cdot N+1)(N+3)}{(N+2)(N+3)}\right)^m \tag{5.22}$$

Given all quantities in the denominator are positive this expression may be simplified to

$$\begin{aligned}&((p_0\cdot N+1)(N+3))^m-((p_0\cdot N+1)(p_0\cdot N+2))^m\\&\quad>((p_0\cdot N+2)(N+2)-(p_0\cdot N+1)(N+3))^m\end{aligned}$$

$$\begin{aligned}\implies\ &2\cdot((p_0\cdot N+1)^m\cdot(N+3)^m)>(p_0\cdot N+2)^m\cdot(N+2)^m\\&+(p_0\cdot N+1)^m\cdot(p_0\cdot N+2)^m\end{aligned}$$

$$\implies\ 2\cdot((p_0\cdot N+1)^m\cdot(N+3)^m)>(1+\tfrac{(p_0\cdot N+1)^m}{(N+2)^m}) \tag{5.23}$$

Introducing the logarithm for computational convenience expresses this as

$$\log(2 \cdot ((p_0 \cdot N + 1)^m \cdot (N + 3)^m)) > \log(1 + \tfrac{(p_0 \cdot N+1)^m}{(N+2)^m}) \tag{5.24}$$

For $x > -1$ we know that $x \geq \log(1 + x)$. Therefore, to prove that α is greater than 1, it suffices to show that

$$\log(2 \cdot ((p_0 \cdot N + 1)^m \cdot (N + 3)^m)) > \tfrac{(p_0 \cdot N+1)^m}{(N+2)^m}$$

$$\implies \log(2) + m\log(p_0 \cdot N + 1) + m\log(N + 3) > \tfrac{(p_0 \cdot N+1)^m}{(N+2)^m} \tag{5.25}$$

Given all components on the LHS are individually and collectively greater than 1 while the RHS ≈ 0, for the expression given by the post processor, it suffices that $\alpha > 1$ for all N and p_0. Similarly for β, it is enough to show that the numerator is greater than the denominator for the post processor expression. Using the same approach as with α consider

$$\text{LHS} = ((\tfrac{N+2}{N+2})^m - (\tfrac{p_0 \cdot N+1}{N+2})^m) \cdot ((\tfrac{N+3}{N+3})^m - (\tfrac{p_0 \cdot N+2}{N+3})^m) \tag{5.26}$$

Expanding this expression results in the LHS being

$$\text{LHS} = (\tfrac{(N+2)(N+3)}{(N+2)(N+3)})^m - (\tfrac{(p_0 \cdot N+1)(N+3)}{(N+2)(N+3)})^m - (\tfrac{(p_0 \cdot N+2)(N+2)}{(N+2)(N+3)})^m + (\tfrac{(p_0 \cdot N+2)(p_0 \cdot N+1)}{(N+2)(N+3)})^m \tag{5.27}$$

Similarly, consider the RHS of the beta expression to be given by

$$\text{RHS} = (\tfrac{(p_0 \cdot N+2)(N+2) - (p_0 \cdot N+1)(N+3)}{(N+2)(N+3)})^m \tag{5.28}$$

To prove that the numerator is indeed greater than the denominator of the α expression, setting the LHS > RHS and using the above expressions implies that

$$\begin{aligned}(\tfrac{(N+2)(N+3)}{(N+2)(N+3)})^m - (\tfrac{(p_0 \cdot N+1)(N+3)}{(N+2)(N+3)})^m - (\tfrac{(p_0 \cdot N+2)(N+2)}{(N+2)(N+3)})^m \\ +(\tfrac{(p_0 \cdot N+2)(p_0 \cdot N+1)}{(N+2)(N+3)})^m > (\tfrac{(p_0 \cdot N+2)(N+2) - (p_0 \cdot N+1)(N+3)}{(N+2)(N+3)})^m\end{aligned} \tag{5.29}$$

Given all quantities in the denominator are positive this expression may be simplified to

$$\begin{aligned}(N + 2)^m \cdot (N + 3)^m - (p_0 \cdot N + 1)^m \cdot (N + 3)^m - (p_0 \cdot N + 2)^m \cdot (N + 2)^m \\ +(p_0 \cdot N + 2)^m \cdot (p_0 \cdot N + 1)^m > (p_0 \cdot N + 2)^m (N + 2)^m - (p_0 \cdot N + 1)^m (N + 3)^m\end{aligned}$$

$$\implies (N+2)^m \cdot (N+3)^m + (p_0 \cdot N + 2)^m + (p_0 \cdot N + 1)^m > 2 \cdot ((p_0 \cdot N + 2)^m \cdot (N+2)^m) \tag{5.30}$$

Similar to α, for ease of computation the logarithm is introduced resulting in

$$m\log(N+2) + m\log(N+3) + m\log(p_0 \cdot N + 2) + m\log(p_0 \cdot N + 1) > \log(2) + m\log(p_0 \cdot N + 1) + m\log(N+2)$$

$$\implies m\log(N+3) + m\log(p_0 \cdot N + 2) > \log(2) \tag{5.31}$$

Since all components are greater than 1 while the arguments on the LHS > RHS for all N and p_0, it is concluded that $\beta > 1$. □

Allow for the process of generating a beta distribution of the failure probability defined in terms of α and β to be referred to as the post processor. Let P_F^+ represent the posterior distribution of the failure probability generated through the post processor and $\hat{P}_F^+$ the subsequent approximation by a single beta distribution. Proposition 1 provides a significant consequence in proving both α and β are greater than 1 ensures that the beta PDF P_F^+ will be bounded. If $\hat{P}_F^+ \sim B(\alpha, \beta)$ and α and β are given by Eq. 5.17, the first two moments of P_F^+ and $\hat{P}_F^+$ are equivalent. In turn resulting in an approximation, the details of which are further discussed in [76, 235]. In terms of the relationship between the SuS probability of failure estimate ($\hat{p}_F$) and $\hat{P}_F^+$, it can be shown that the MAP estimate of $\hat{P}_F^+$ (i.e. $\hat{P}_{MAP}^+$), is equivalent to $\hat{p}_F$. Given the mode of a product of independent distributions is equal to the product of the modes and the MAP expression of each intermediate level beta distribution defined in Eq. 5.16, the MAP of the post processor is

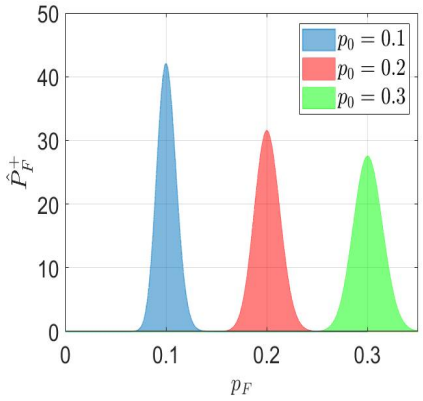
$$\begin{aligned} \hat{P}_{MAP}^+ &= \prod_{i=1}^{m} P(p_i|D_{i-1})_{MAP} \\ &= (p_0)^m \end{aligned} \tag{5.32}$$

This is equivalent to the expression for $\hat{p}_F$ in Eq. 5.14. Therefore, the post processor applies a Bayesian approach to quantify the uncertainty in the initial $\hat{p}_F$ SuS estimate. By extracting information from the number of failure samples at a given level, a single beta PDF ($\hat{P}_F^+$) is generated of which the MAP value coincides with $\hat{p}_F$.

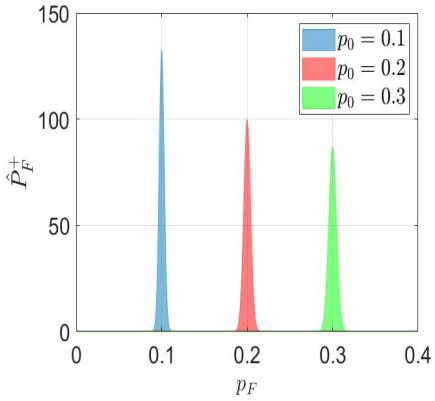
The beta PDF is defined in terms of the shape parameters α and β. From Eq. 5.17, the expressions for α and β depend on the SuS parameters N, p_0 and m. Figure 5.2 investigates the influence of the parameter settings on the shape of the beta PDF. As the probability of failure estimate corresponds to the mode of the beta distribution, for $m = 1$ the distribution will have a mode located at p_0. This stems from the choice of b at every level to ensure $p_0 \cdot N$ samples belong in the intermediate failure domain. Intuitively, selecting a larger p_0 results in a greater number of generated samples belonging to the intermediate failure domain. The distributions become increasingly positively skewed for an increase in m. This may be explained by the expressions for α and β given in Eq. 5.17. The following corollary reveals that for $p_0 < 0.5$, $\beta > \alpha$. A derivation of corollary 1 is given in Appendix C.

Corollary 1. *Let* $P_F^+ \sim B(\alpha, \beta)$ *with* $\alpha > 1$ *and* $\beta > 1$. *Then for any* $p_0 < 0.5$, $\beta > \alpha$.

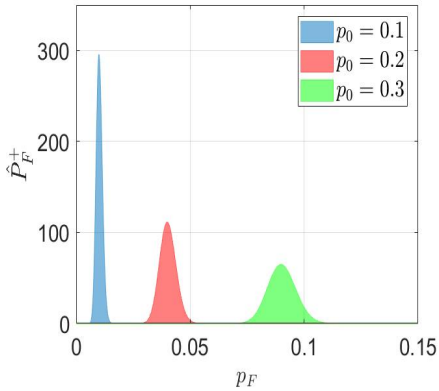
In Figure 5.2, for a greater number of samples the probability density increases for all p_0 considered. This means that their is an increased likelihood of the sample event (i.e. failure) occurring within this neighbourhood. Intuitively, this coincides with the concept of the Direct Monet Carlo (DMC) estimator which stated that the empirical average of the probability of failure tends towards its true value as N increases to infinity.



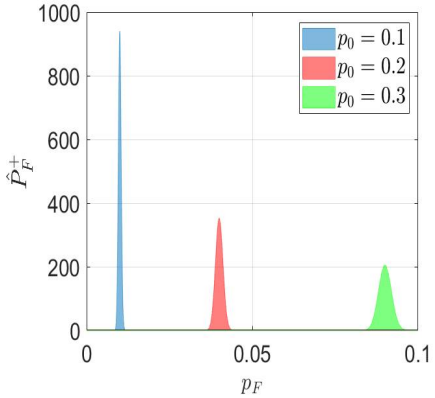
(a) $m = 1$, $N = 1000$



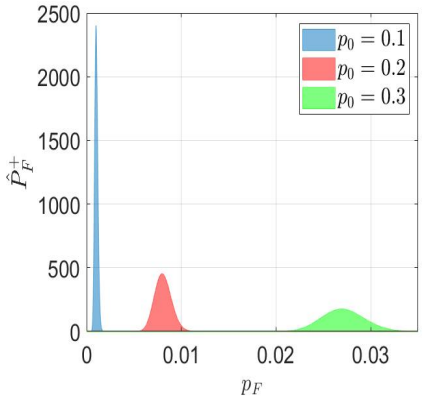
(b) $m = 1$, $N = 10000$



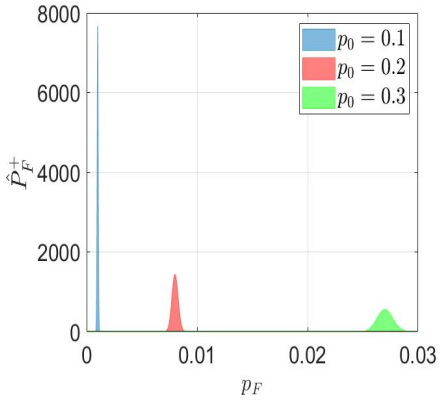
(c) $m = 2$, $N = 1000$



(d) $m = 2$, $N = 10000$



(e) $m = 3$, $N = 1000$



(f) $m = 3$, $N = 10000$

Figure 5.2: Influence of the choice of N, p_0 and m on the beta PDF using the shape parameters defined by the post processor.

5.2.2 Bayesian Post Processor with BUS

This section introduces a stopping condition for $nBUS$ which makes use of the post processor discussed in the previous section. Consider the expression for the estimation of the log evidence produced by $nBUS$.

$$\ln(\hat{P}_D) = b_m + m \cdot \ln(p_0) \qquad b_m > b_{min} \tag{5.33}$$

This stopping criterion proposes to track the progression of $\ln(\hat{P}_D)$ at each intermediate level to identify when Eq. 5.33 holds i.e. $b_m > b_{min}$. As previously discussed, at a given $level_m$ the limit state function values $\{g^i_{m-1}\}^N_{i=1}$ are placed in descending order with the critical threshold b_m chosen by

$$b_m = \frac{g^{nc}_{m-1} + g^{nc+1}_{m-1}}{2} \tag{5.34}$$

Where as before $nc = p_0 \cdot N$. Given that nc samples are chosen as seeds for Markov chains to generate N samples at $level_m$, the respective limit state function values of these samples $\{g^j_{m-1}\}^{nc}_{j=1}$ have exceeded b_m. Using Eq. 5.34, for each pair of limit state function evaluations in $\{g^j_{m-1}\}^{nc}_{j=1}$ it is possible to compute a sequence of failure thresholds beyond b_m.

$$b^j_m = \frac{g^j_{m-1} + g^{j+1}_{m-1}}{2} \qquad \text{for} \quad j = 1, ..., nc \tag{5.35}$$

With the final value of the sequence b^{nc}_m denoting the original critical threshold value assigned at each level as in Eq. 5.34. An illustration of the progression of the critical threshold at $level_m$ is provided in Figure 5.3. Note that as previously outlined the limit state function values are placed in descending order prior to the computation of the intermediate failure threshold. Consider the case of $N = 1000$ and $p_0 = 0.1$ with $nc = 100$. The blue line represents the sequence of critical thresholds computed using $\{g^j_{m-1}\}^N_{j=nc+1}$. The black line denotes b^{nc}_m as in Eq. 5.34. The sequence $\{b^j_m\}^{nc}_{j=1}$ is given by the red line and is computed using $\{g^j_{m-1}\}^{nc}_{j=1}$. The proposed criterion examines the behaviour of the log evidence using this sequence of critical thresholds.

$$\ln(\hat{P}_D)^j_m = b^j_m + m \cdot \ln(p_0) \qquad \text{for} \quad j = 1, ..., nc \tag{5.36}$$

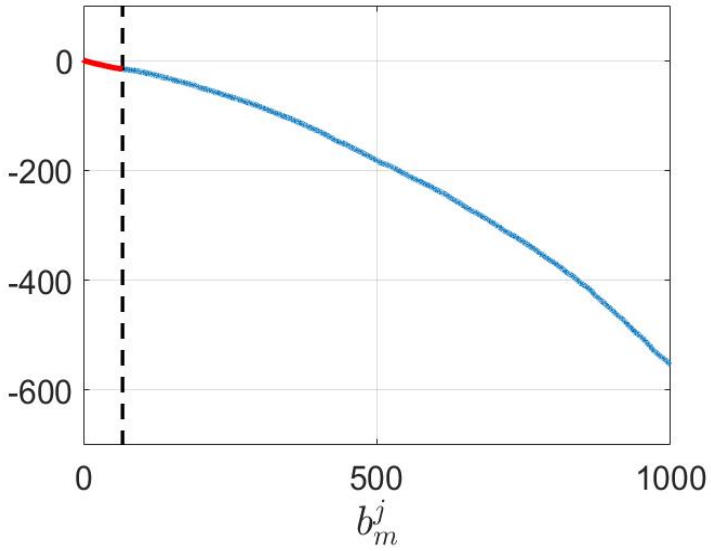


Figure 5.3: Illustration of the critical threshold progression at $level_m$ for $N = 1000$ and $p_0 = 0.1$. The blue line represents the sequence of critical thresholds computed using $\{g_{m-1}^j\}_{j=nc+1}^{N}$. The black line denotes b_m^{nc} as in Eq. 5.34. The sequence $\{b_m^j\}_{j=1}^{nc}$ is given by the red line beyond b_m^{nc} and is computed using $\{g_{m-1}^j\}_{j=1}^{nc}$.

Eq. 5.36 represents the trend of the log evidence for the sequence of b_m^j at $level_m$. Given that b_m^j is a stochastic quantity the resulting $\ln(\hat{P}_D)_m^j$ will also be stochastic. To account for this stochasticity, the Bayesian post processor discussed in the previous section is introduced. Due to the characteristic trends discussed in Chapter 4 being in log space, the post processor PDF must be transformed. As previously discussed, SuS simulates a sequence of intermediate failure events which have a decreasing likelihood of occurrence. Resulting in the subsequent failure probability estimates becoming smaller and smaller. In terms of the post processor, this entails that theoretically the failure PDF $\hat{P}_F^+$ will become increasingly positively skewed as the algorithm progresses. The introduction of the log transformation results in this extremely skewed beta distribution becoming more symmetrical. Derivations of the log transformation of the PDF and Cumulative Distribution Function (CDF) of $\hat{P}_F^+$ are given in Appendix C. At $level_m$, Eq. 5.36 is computed for all nc values of $\ln(\hat{P}_D)_m^j$. Next, $\ln(\hat{P}_D)_m^j$ is compared against the MAP of the log transformed beta distribution

$\ln(\hat{P}_F^+)$ which is derived in Appendix C.

$$\ln(\hat{P}_{MAP}^+) = \ln\left(\frac{\alpha}{(\alpha+\beta-1)}\right) \tag{5.37}$$

With α and β being chosen as expressed in Proposition 1. Though it was shown in the previous section that $\hat{p}_F = \hat{P}_{MAP}^+$, working with $\hat{P}_{MAP}^+$ results in information regarding the uncertainty in the estimated probability of failure being readily available. It is proposed to take advantage of this information to allow for statistical error stemming from the log evidence to be accounted for. At every intermediate level, the criterion generates $\ln(\hat{P}_{MAP}^+)$ using the post processor. To allow for the stochastic behaviour of the log evidence, a 99% credible interval of the distribution $\ln(\hat{P}_F^+)$ is defined. If all $\ln(\hat{P}_D)_m^j$ for $j = 1, ..., nc$ evaluations lie within this interval, the transition in the relationship between the model evidence and probability of failure has occurred i.e. $b_m > b_{min}$. Indicating the minimum required level to draw samples from the posterior distribution has been reached. Let δ_m represent the proportion of log evidence evaluations out of a possible nc lying within the specified credible interval.

$$\delta_m = \frac{\sum_{j=1}^{nc}(a \leq \ln(\hat{P}_D)_m^j \leq h)}{nc} \tag{5.38}$$

Let $[a, h]$ denote the range of the 99% credible interval of $\ln(\hat{P}_F^+)$. In the case that $\delta_m = 1$, the log evidence is approximately equal to $\ln(\hat{P}_{MAP}^+)$ as all $\ln(\hat{P}_D)_m^j$ lie within the specified credible interval and the samples generated at $level_m$ are taken as the posterior set. For $\delta_m < 1$, a proportion of log evidence evaluations are outside the credible interval so another intermediate level is required. In practice, to account for potential sampling error, a tolerance t is chosen whereby if $\delta_m > t$ simulation terminates. The authors recommend a value of $t = 0.95$ i.e. 95% of log evidence evaluations lie in the specified credible interval of $\ln(\hat{P}_F^+)$. In this case, if

$$\delta_m > 0.95 \tag{5.39}$$

the samples generated at the corresponding $level_m$ have been drawn from the target density. The proposed stopping condition is referred to as BUS_+ due to the application of the post processor along with the avoidance of computing the nested loop. This is summarised in Algorithm 11. At $level_0$, δ_0 is set a default value

of 0 as samples are being drawn from the prior. At $level_1$ the sequence $\{b_1^j\}_{j=1}^{nc}$ is computed with the value b_1^{nc} being equivalent to the original intermediate threshold specified in $nBUS$. Samples are generated via MMH, where b_1^{nc} is used to determine whether a proposed sample is accepted or rejected. Once the N samples at $level_1$ have been drawn, $\{\ln(\hat{P}_D)_1^j\}_{j=1}^{nc}$ is evaluated for all $\{b_1^j\}_{j=1}^{nc}$ at step 19. At step 20, the beta distribution parameters α and β are computed before being used to compute $\ln(\hat{P}_{MAP}^+)$. Once the credible interval has been computed in step 22, δ_1 is evaluated for all $\{\ln(\hat{P}_D)_1^j\}_{j=1}^{nc}$. If $\delta_1 > t$, simulation stops and the samples drawn at $level_1$ are taken as those from the posterior. If not, BUS_+ progresses to $level_2$ and sequentially generates intermediate levels in the same manner as the above steps until $\delta_m > t$ holds.

Algorithm 11 BUS_+

Define N, p_0, $nc = p_0 N$ and $ns = p_0^{-1}$

Define the tolerance t

Initialize $m = 0$, where m is the current simulation level

Initialize $\delta_m = 0$

Generate N samples $\{\boldsymbol{\theta}_m^i, u_m^i\}_{i=1}^N$ from the input PDF $\pi(\cdot)$

For each $\{\boldsymbol{\theta}_m^i, u_m^i\}_{i=1}^N$ evaluate $g_m^i = \ln L(D|\boldsymbol{\theta}_m^i) - \ln(u_m^i)$

Sort the N values of g_m^i and the corresponding $\{\boldsymbol{\theta}_m^i, u_m^i\}_{i=1}^N$ in descending order with respect to g_m^i

while $\delta_m < t$ **do**

  $m = m + 1$

  for $j = 1, ..., nc$ **do**

    Calculate: $b_m^j = \frac{g_{m-1}^j + g_{m-1}^{j+1}}{2}$

  end for

  Store the first nc samples of the ordered set $\{\boldsymbol{\theta}_{m-1}^i, u_{m-1}^i\}_{i=1}^N$ as 'seeds'

  Using $\{\boldsymbol{\theta}_{m-1}^j, u_{m-1}^j\}_{j=1}^{nc}$ draw the remaining $N - nc$ samples from $\pi(\cdot|F_m)$ via MCMC as follows:

  for $j = 1, ..., nc$ **do**

    Starting with $\{\boldsymbol{\theta}_{m-1}^j, u_{m-1}^j\}$ as an initial seed, generate $\{\boldsymbol{\theta}_m^k, u_m^k\}_{k=1}^{ns-1} \sim \pi(\cdot|F_m)$ states of a Markov chain using the Modified Metropolis Hastings in Algorithm 5. In this algorithm, the sample acceptance criteria involves evaluating g_m with respect to b_m^{nc} for each generated sample.

  end for

  Sort the N values of g_m^i and the corresponding $\{\boldsymbol{\theta}_m^i, u_m^i\}_{i=1}^N$ in descending order with respect to g_m^i

  For each b_m^j evaluate $\ln(\hat{P}_D)_m^j = b_m^j + m \cdot \ln(p_0)$

  Compute $\alpha = \frac{(\frac{p_0 \cdot N+1}{N+2})^m \cdot (1-(\frac{p_0 \cdot N+2}{N+3})^m)}{(\frac{p_0 \cdot N+2}{N+3})^m - (\frac{p_0 \cdot N+1}{N+2})^m}$ and $\beta = \frac{(1-(\frac{p_0 \cdot N+1}{N+2})^m) \cdot (1-(\frac{p_0 \cdot N+2}{N+3})^m)}{(\frac{p_0 \cdot N+2}{N+3})^m - (\frac{p_0 \cdot N+1}{N+2})^m}$

  Compute $\ln(\hat{P}_{MAP}^+) = \ln(\frac{\alpha}{(\alpha+\beta-1)})$

  Compute the credible interval: $0.99 = p(a \leq \ln(\hat{P}_{MAP}^+) \leq h)$ with $a \leq h$. Where $[a, h]$ is referred to as the range of the 0.99 credible interval.

  Compute $\delta_m = \frac{\sum_{j=1}^{nc} (a \leq \ln(\hat{P}_D)_m^j \leq h)}{nc}$

end while

Return $\boldsymbol{\theta} \sim P(\boldsymbol{\theta}|D)$ & $P_D \approx e^{b_m^{nc}} p_0^m$

In terms of added computational expense, $\{\ln(\hat{P}_D)_m^j\}_{j=1}^{nc}$ is computed as a by product of simulation, while the post processor involves generating a single beta PDF using the readily available data. This ensures that the task of computing $\ln(\hat{P}_{MAP}^+)$ in terms of additional cost is negligible. The use of the post processor also ensures that the algorithm is not required to be run a large number of times in order to obtain an average of the log evidence characteristic trend. Utilizing the uncertainty information regarding the probability of failure estimate allows for this to be achieved with confidence in a single run of the sampler.

5.2.3 BUS Variations

Table 5.1 provides an overview comparison between the variations of the BUS frameworks presented in this Chapter and Chapter 4. In terms of the limit state function Y, each of the modified BUS frameworks introduce the natural logarithm as first proposed by [67], to provide numerical stability and aid the transition between intermediate levels. Aside from the original BUS algorithm, both $aBUS$ and $nBUS$ compute the likelihood multiplier c during simulation. $aBUS$ relates the final c to the maximum of the likelihood evaluations observed during a sampler run such that $c = e^{-L(D|\boldsymbol{\theta})_{max}}$. Given the finite choice of sample size N, this empirical solution raises the question of how likely is it that the true likelihood maximum will be reached. $nBUS$ reformulates the failure event, allowing for c to be expressed in terms of the failure threshold $b = -\ln c$. As b is computed during a sequentially during a sampler run, identification of the likelihood multiplier is no longer an issue.

In terms of stopping criteria, BUS and $aBUS$ terminate simulation once an intermediate threshold less than zero has been produced. The stochastic nature of b however, results in the samples not being conditional on the true failure event. In contrast, $nBUS$ ensures all samples are conditional on the true failure event but requires the minimum level to generate posterior samples to be identified. The nested loop stopping criterion presented in [67], terminates once the probability of drawing samples h_m from outside the target density is less than a small tolerance ϵ. The computation of h_m results in a large number of additional likelihood evaluations which greatly affect model efficiency.

To rectify this issue, $nBUS_D$, combines the nested architecture in [67] with the direct application of the rejection principle to reduce the number of required likelihood evaluations. Through the direct use of the rejection principle, the inner

loop of $nBUS$ is only called upon once all samples have been accepted. This allows for the likelihood of generating a sample outside the posterior to be quantified while unnecessary likelihood functions calls in areas of the input space far away from the failure domain are avoided.

To further reduce computational expense, BUS_+ proposes utilizing a Bayesian interpretation of the probability of failure to identify when the relationship with the model evidence has transitioned. By accounting for the statistical variation in the model evidence, BUS_+ avoids the requirement of running $nBUS$ multiple times in order to achieve an average of the log evidence characteristic trend. In turn, this ensures a robust estimate is computed in a single sampler run. The two criteria presented reduce the computational expense of $nBUS$ while providing statistical assurances of concluding simulation.

Model	**Limit State Function**	**Likelihood Multiplier**	**Criterion**
BUS	$u - cL(D\|\boldsymbol{\theta})$		$b_m \leq 0$
aBUS	$\ln(u) - \ln(c) - \ln(L(D\|\boldsymbol{\theta}))$	$e^{-L(D\|\boldsymbol{\theta})_{max}}$	$b_m \leq 0$
nBUS	$\ln(L(D\|\boldsymbol{\theta})) - \ln(u)$	e^{-b_m}	$h_m < \epsilon$
nBUS$_D$	$\ln(L(D\|\boldsymbol{\theta})) - \ln(u)$	e^{-b_m}	$A_m = 0$ & $h_m < \epsilon$
BUS$_+$	$\ln(L(D\|\boldsymbol{\theta})) - \ln(u)$	e^{-b_m}	$\delta_m > t$

Table 5.1: Comparison of different BUS frameworks in terms of the limit state function, computed likelihood multiplier and stopping criterion.

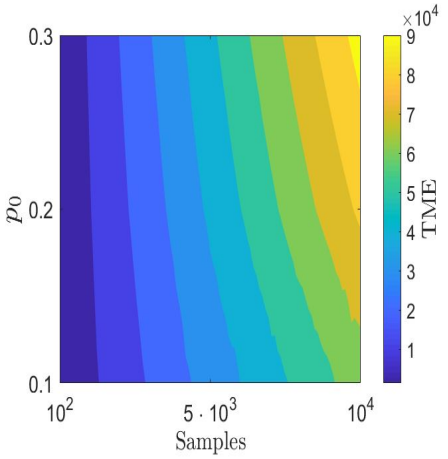
5.3 Numerical Applications

In this section, $nBUS_D$ and BUS_+ are implemented on the four benchmark problems and supervised machine learning task discussed in Chapter 4. The computational costs are firstly compared against nested BUS with a stopping tolerance of $\epsilon = 10^{-8}$ ($nBUS_8$) and adaptive BUS ($aBUS$). In line with $nBUS_8$, the stopping tolerance for $nBUS_D$ is also set to $\epsilon = 10^{-8}$. The ratio of log evidence evaluations lying within the post processor credible interval for BUS_+ is selected as $t = 0.95$. The experimental set up and performance metrics are the same

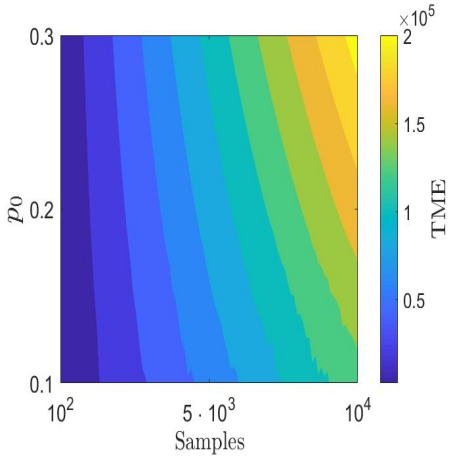
as those used in Chapter 4. The level probability p_0 is varied on the interval $[0.1, 0.3]$. The number of samples N is varied on the interval $[10^2, 10^4]$, with its value incrementally increasing. Initially N is set to 10^2. For each subsequent sampler run, N is increased by 10^2 samples before taking a final value of $N = 10^4$. This results in each sampler being implemented for 99 different values of N.

5.3.1 Computational Expense

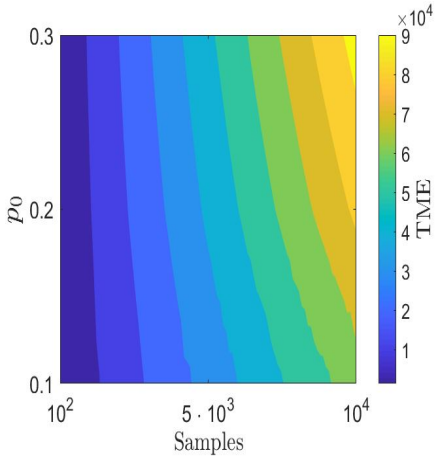
The main focus for the development of $nBUS_D$ and BUS_+ was the reduction in the required computational cost of $nBUS$ for drawing samples from the posterior. As such, the first comparison of the numerical experiments is in terms of the number of total model evaluations (TME) of each sampler. Both $nBUS_D$ and BUS_+ result in a reduction in the number of likelihood function calls with respect to $nBUS_8$ for all numerical examples. With respect to $nBUS_8$, for $N = 5 \times 10^3$, the average reduction in TME is 50% for $nBUS_D$ and 73% for BUS_+. Figure 5.4 presents the number of TME for each sampler for example 4. From Figure 5.4 (a) and (c) it is apparent that the both $aBUS$ and $nBUS_D$ require a similar number of TME for the p_0 and N considered. Similarly, BUS_+ exhibits similar behaviour in terms of TME up to $N = 9 \cdot 10^3$. For $N \in [9 \cdot 10^3, 10^4]$ however, BUS_+ requires 10% fewer TME for $p_0 = 0.1$ in comparison to $aBUS$ and $nBUS_D$. While for $p_0 = 0.3$, the reduction is approximately 18% with respect to $aBUS$ and $nBUS_D$. In terms of $nBUS_8$, $nBUS_D$ and BUS_+ require 54% and 49% less TME than $nBUS_8$. This added computational cost of $nBUS_8$ stems from the computation of h_m at each intermediate level. From a comparison between $aBUS$, $nBUS_D$ and BUS_+ it is noted that the difference in computational cost between each frameworks varies from example to example. However, avoiding the nested architecture of $nBUS$ at every level guarantees a more efficient framework by using either $nBUS_D$ or BUS_+. Further plots for the TME required by each sampler for numericals example 1, 2 and 3 are given in Appendix C.



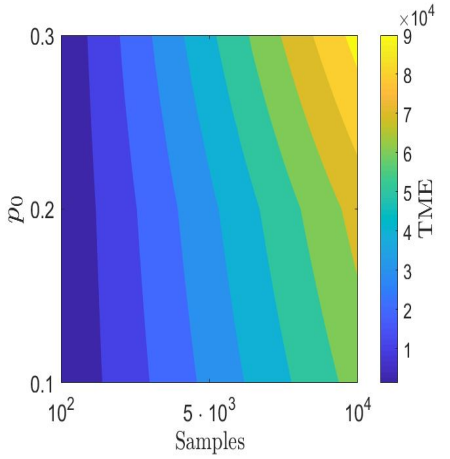
(a) Number of TME for $aBUS$.



(b) Number of TME for $nBUS_8$.



(c) Number of TME for $nBUS_D$.

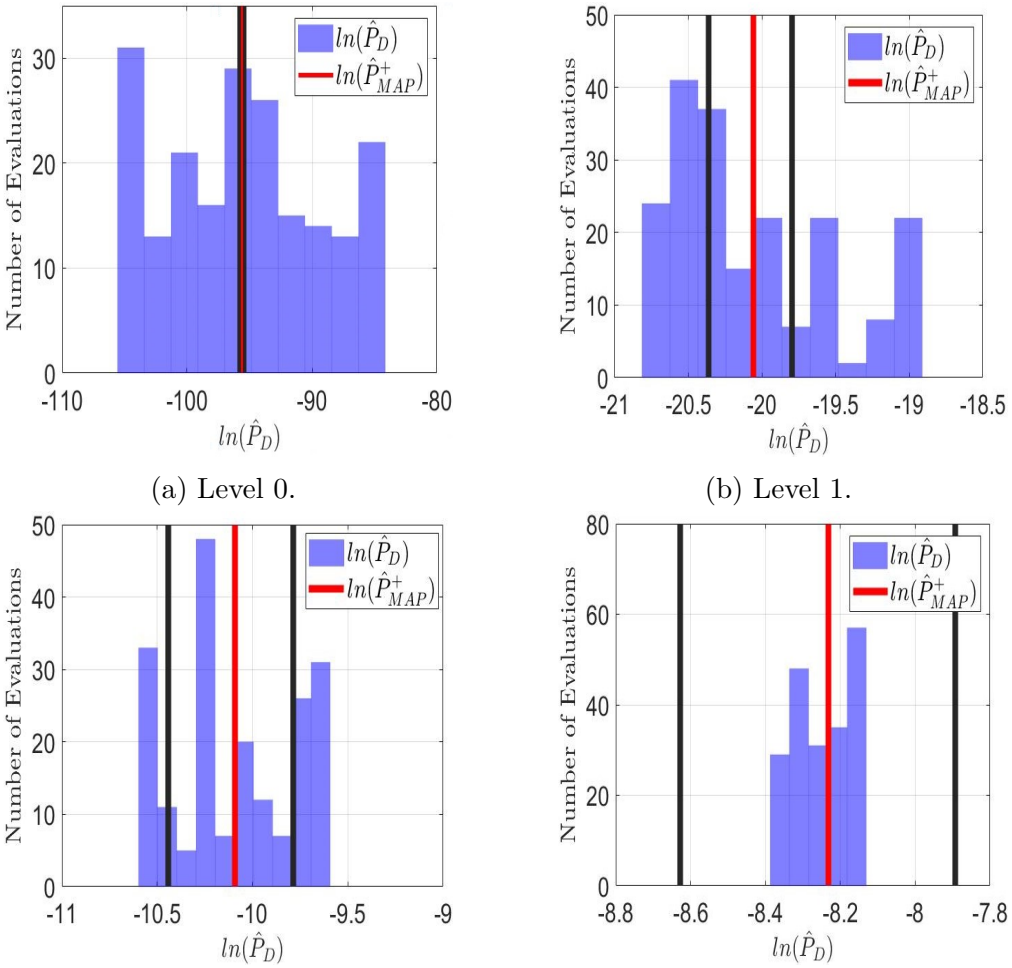


(d) Number of TME for BUS_+.

Figure 5.4: Illustration of the required TME for $aBUS$, $nBUS_8$, $nBUS_D$ and BUS_+ for drawing posterior samples in example 4. Each value of TME was achieved as an average from 100 independent runs of each sampler.

To further investigate the behaviour of BUS_+ consider the histograms of log evidence evaluations produced for example 4 in Figure 5.5. As the levels progress, the distribution of the sample evaluations of the log evidence becomes closer and closer to $\ln\hat{P}^{+}_{MAP}$ (red) before remaining settled inside the chosen credible interval (black). The credible interval allows for the statistical variation in the log evidence. For an increase in N or p_0, a larger number of log evidence evaluations are required to fall within this interval. From the experimental results in Figure 5.4, it is noted that the rate of increase of the number of TME became much larger in the case of $N = 10^4$ and $p_0 = 0.3$. This increase in computational cost stems from a greater number of levels required to ensure all log evidence evaluations

are captured by the credible interval. This may suggest that BUS_+ may be best suited with smaller level probabilities in the presence of a very expensive model.



(a) Level 0.

(b) Level 1.

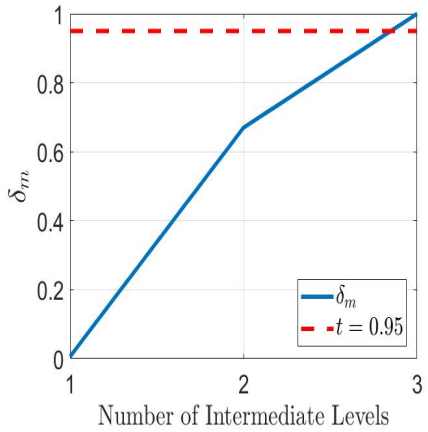
(c) Level 2.

(d) Level 3.

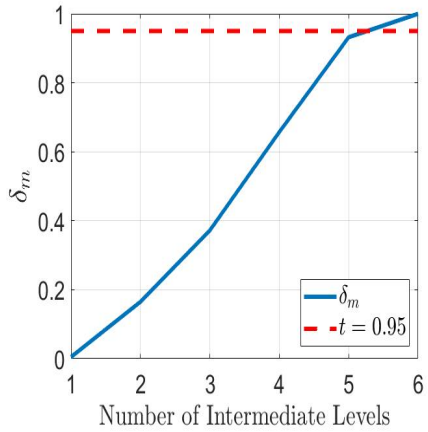
Figure 5.5: Log evidence progression using example 4 for $level_{0-3}$ (a)-(d). The chosen parameters being $N = 1000$, $p_0 = 0.1$, 99% credible interval and a sample ratio of 0.95. A histogram of the log evidence evaluations is denoted in blue, MAP of the post processor in red and the subsequent credible interval in black.

Figure 5.6 examines the behaviour of the ratio of log evidence evaluations δ_m within the specified credible interval for an increasing p_0 for example 4. For $p_0 \in [0.1, 0.3]$, as expected δ_m increases for a growing number of intermediate levels. In each case, δ_m surpasses the ratio threshold $t = 0.95$ prior to stopping the sampling process. Notably for $p_0 = 0.3$, the rate of increase of δ_m is much slower in comparison to $p_0 \in [0.1, 0.2]$. As a greater number of intermediate levels is generated, the behaviour of δ_m for $p_0 = 0.3$ coincides with the above point

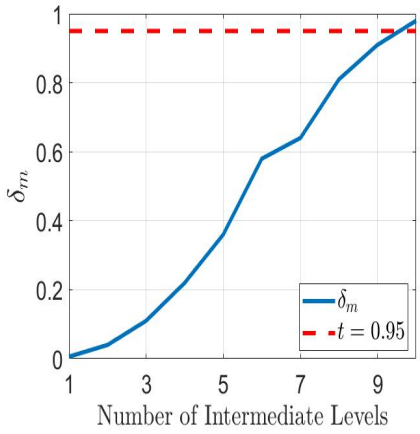
raised regarding the increase in the number of TME required for drawing posterior samples.



(a) $p_0 = 0.1$.



(b) $p_0 = 0.2$.



(c) $p_0 = 0.3$.

Figure 5.6: BUS_+ progression of δ_m for varying p_0 and $N = 1000$ for example 4. As expected, all panels show an increase in δ_m as the number of intermediate levels progresses. For each value of p_0, δ_m surpasses the chosen threshold $t = 0.95$ before stopping the sampling process.

The use of the credible interval avoids the requirement of having to run BUS_+ a large number of times to achieve an average for the log evidence. Figure 5.7 shows multiple log evidence realisations from 100 independent runs of BUS_+ for example 4. As expected, the variation in estimates falls within the credible region of a single sampler run which terminated using the same number of levels. The advantage of allowing for sampling error is highlighted by a proportion of the log

evidence evaluations from the sampler runs falling outside the credible interval upper bound.

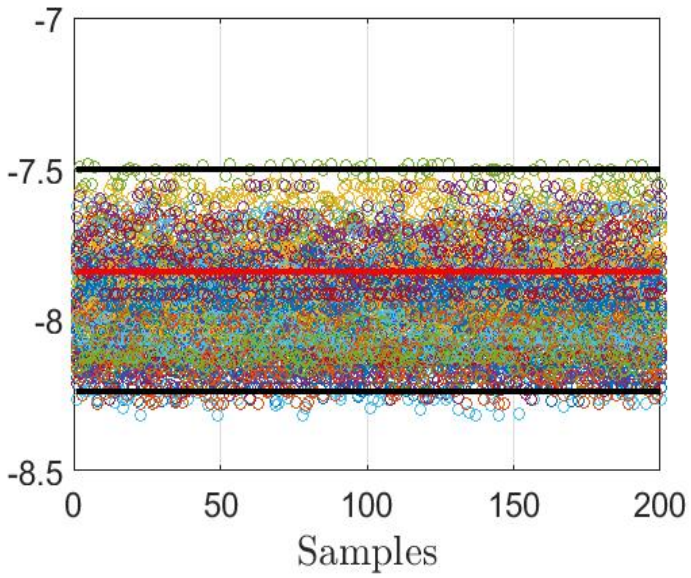


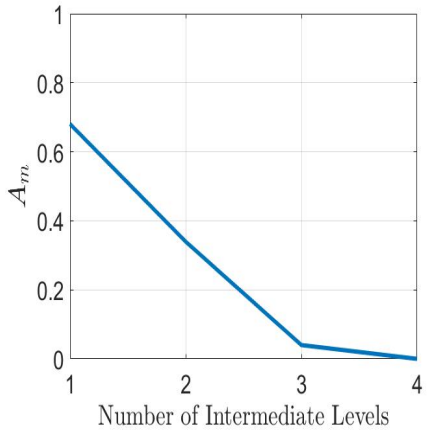
Figure 5.7: Log evidence estimates from 100 independent runs of BUS_+. The majority of estimates fall within the credible interval of a single sampler run which terminated using the same number of intermediate levels. This avoids the requirement of running the model a large number of times to achieve an average for the log evidence.

In terms of $nBUS_D$, Figure 5.8 investigates the behaviour of the number of samples within the inadmissible set A_m with respect to p_0 for example 4 using $N = 1000$.

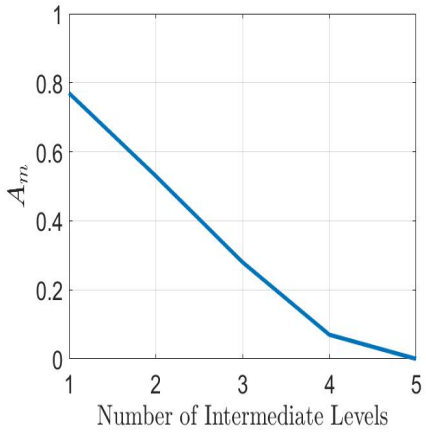
$$A_m = \frac{\sum_{i=1}^{N}(e^{-b_m}L(D|\boldsymbol{\theta}_m^i) > 1)}{N} \tag{5.40}$$

Here A_m is expressed as a proportion of the total N samples generated at $level_m$. As a reminder, $A_m = 0$ indicates the N generated samples have been rejected as being members of the inadmissible set of samples. Panels (a)-(c) of Figure 5.8 present the trends of A_m for $p_0 = 0.1$, $p_0 = 0.2$ and $p_0 = 0.3$. As expected, for an increasing number of intermediate levels, A_m decreases before converging to zero. The rate of decrease in A_m slows down for an increase in p_0, where $p_0 = 0.3$ requires the greatest number of intermediate levels. This coincides with the behaviour of δ_m for BUS_+ and the general architecture of SuS, whereby

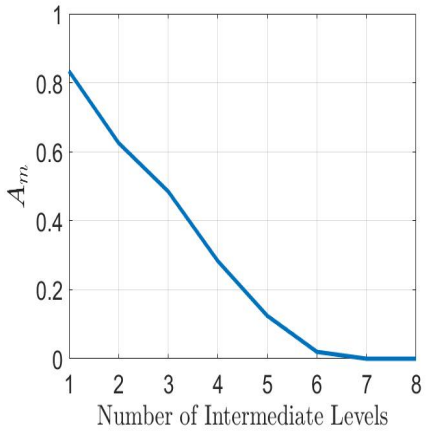
an increase in p_0 results in a greater number of TME for a sampler run. With $nBUS_D$ once $A_m = 0$, h_m is computed via the nested loop of $nBUS$.



(a) $p_0 = 0.1$.



(b) $p_0 = 0.2$.

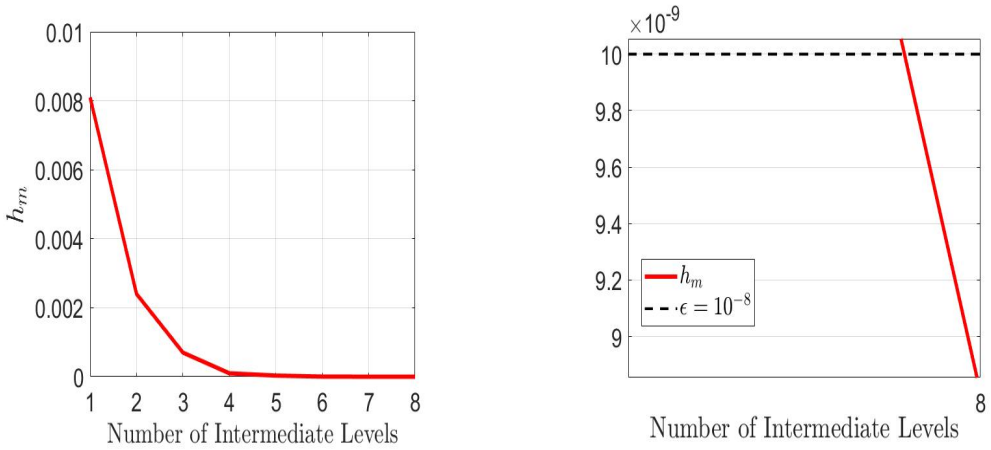


(c) h_m for $p_0 = 0.3$.

Figure 5.8: Progression of A_m in $nBUS_D$ for example 4. Panel (a) shows the behaviour of A_m for an increase in the number of intermediate levels with $p_0 = 0.1$. As expected, A_m decreases before converging to 0 at $level_4$. Panel (b) shows the behaviour of A_m for $p_0 = 0.2$ where sampling stops at $level_5$. Panel (c) shows the behaviour of A_m for $p_0 = 0.3$ where sampling stops at $level_8$.

Figure 5.9 (a) presents the trend of h_m for each intermediate level using $p_0 = 0.3$. In practice, though h_m is not computed at levels prior to $A_m = 0$, the purpose of this plot is to illustrate the behaviour of h_m for an increase in m. As $A_m = 0$ at $level_8$ as shown in Figure 5.8 (c), in practice h_m would be computed at $level_8$. For a chosen stopping tolerance of $\epsilon = 10^{-8}$, Figure 5.9 (b) shows that

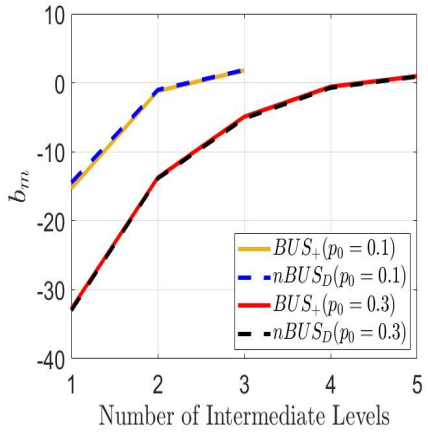
h_m transitions to a value lower than ϵ at $level_8$. Therefore the samples drawn at $level_8$ by $nBUS_D$ are taken as those from the posterior distribution.



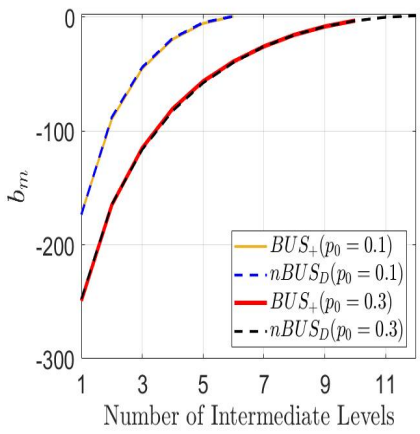
(a) (b)

Figure 5.9: Progression of h_m in $nBUS_D$ for $p = 0.3$ and $N = 1000$ in example 4. Panel (a) shows the behaviour of h_m for an increase in the number of intermediate levels. As expected, h_m decreases at each level before falling below the desired threshold of $\epsilon = 10^{-8}$ at $level_8$ as shown in panel (b). The resulting samples generated at $level_8$ are taken as those from the posterior.

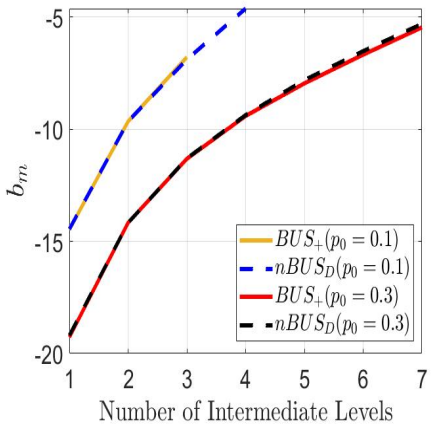
In terms of the progression of the intermediate failure thresholds during a sampler run, Figure 5.10 provides a comparison between $nBUS_D$ and BUS_+ using $N = 1000$ for all four examples. Note that these trends were achieved as averages from 100 independent runs of each sampler. Firstly considering $p_0 = 0.1$, for examples 1 and 2 both $nBUS_D$ and BUS_+ require the same number of intermediate levels. As the underlying sampling process between $nBUS_D$ and BUS_+ does not change, one would expect the deviation in trends produced to be down to the treatment of b_m as a stochastic term. The manner in which the two samplers differ in how they terminate the sampling process is visible in examples 3 and 4 whereby both $nBUS_D$ and BUS_+ require a different number of intermediate levels. For $p_0 = 0.3$ the mentioned points also hold true in terms of trend similarity and also the possibility of a differing number of intermediate levels.



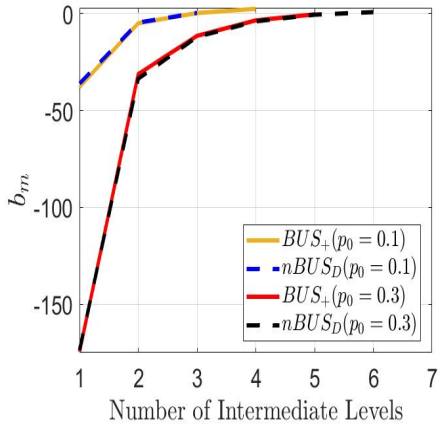
(a) Example 1.



(b) Example 2.



(c) Example 3.



(d) Example 4.

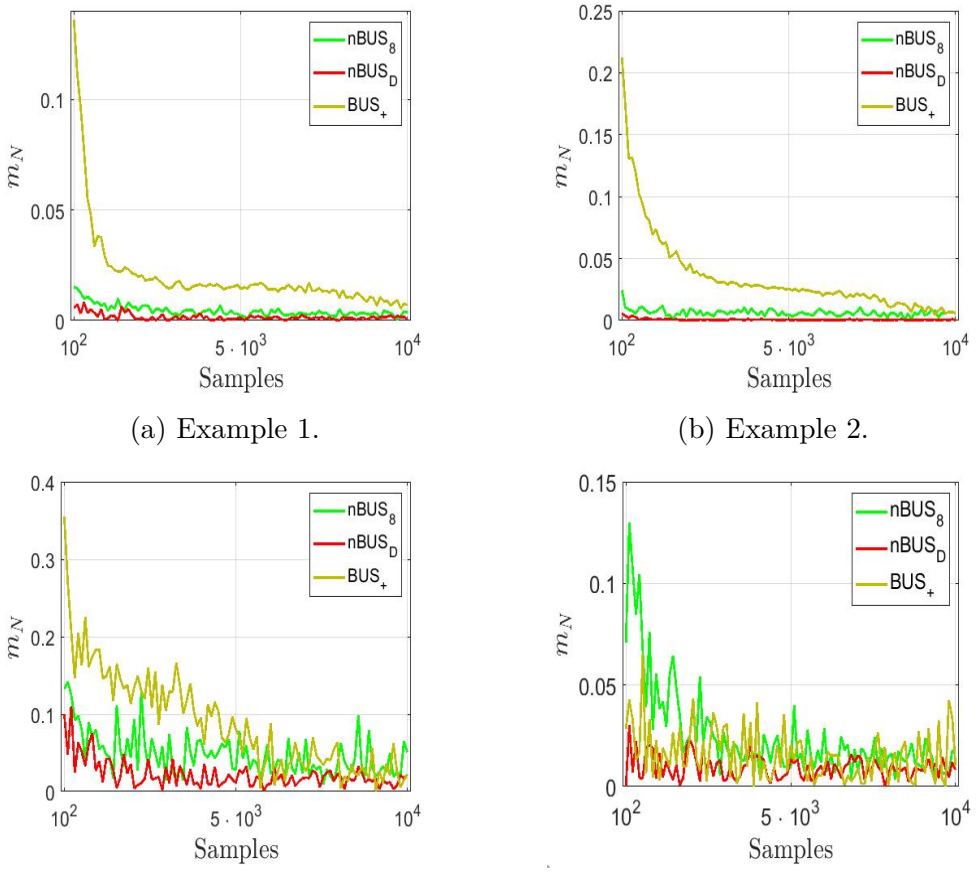
Figure 5.10: Comparison of intermediate thresholds of $nBUS_D$ and BUS_+. Each of the samplers is implemented for $N = 1000$ while using $p_0 = 0.1$ and $p_0 = 0.3$. Panel (a) contains the trends for example 1, panel (b) example 2, panel (c) example 3 and panel (d) example 4. All trends were achieved as an average from 100 independent runs of each sampler.

In summary, the findings in this section reveal the computational savings achieved using $nBUS_D$ and BUS_+ over $nBUS$. By avoiding the computation of the nested loop at every intermediate level, $nBUS_D$ and BUS_+ result in a large decrease of TME for each of the four considered numerical examples. In terms of BUS_+, an investigation of the behaviour of δ_m highlights the added computational cost associated with an increase of p_0. Additionally, the advantage of using the credible interval to avoid running BUS_+ a large number of times to estimate the average log evidence was also discussed. Similar to δ_m, for an increasing p_0 the quantity A_m in $nBUS_D$ requires a greater number of levels to converge. The

computation of h_m once $A_m = 0$ provides assurances that the samples generated at $level_m$ are indeed distributed according to the posterior.

5.3.2 Statistical and Evidence Estimation

Given the clear computational savings over $nBUS_8$, both proposed stopping criteria are compared against the nested framework in terms of statistical moment and model evidence estimation. A comparison with $aBUS$ is omitted due to the superior performance of $nBUS_8$ in Chapter 4. Figure 5.11 presents the bias in the expectation of the posterior samples for $\hat{\boldsymbol{\theta}}_1$. An inverse relationship between the number of samples and the level of bias in the expectation for BUS_+ (yellow) is apparent for the considered examples. For an increase in N, the bias in the estimated expectation decreases. This coincides with numerical investigations presented in [32] that reveal that the level of bias in the posterior estimates of BUS is directly influenced by the number of generated samples. For each of the numerical examples there is a level of bias still present in the estimates to the mean of the posterior once they have terminated according to there respective stopping conditions. On this point, it is noted in [20] that even if the generated samples were distributed exactly as the posterior distribution, discrepancies between the moment estimations would still be apparent due to the finite number of samples drawn. However, as an in increase in sample size requires additional likelihood evaluations, a balance between an acceptable level of bias and computational cost is needed. In general, the difference in the level of bias between $nBUS_D$ (red) and $nBUS_8$ (green) is insignificant.



(a) Example 1.

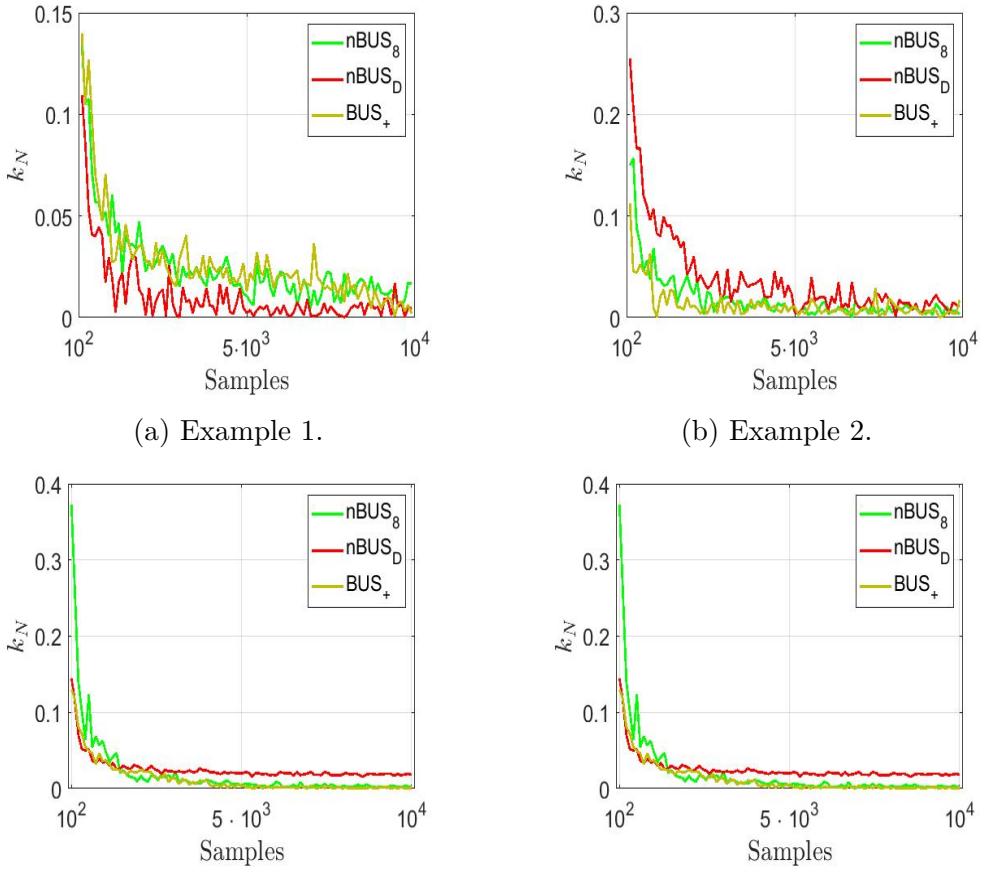
(b) Example 2.

(c) Example 3.

(d) Example 4.

Figure 5.11: Bias in the mean of the posterior samples generated with $nBUS_8$, $nBUS_D$ and BUS_+.

The bias in the standard deviation of the posterior samples is depicted in Figure 5.12. In this case, the level of bias in the standard deviation of the BUS_+ samples is comparable with both $nBUS_D$ and $nBUS$. Again with an improvement in accuracy with an increase in N. All be it the relationship is not as extreme in the case of the expectation.



(a) Example 1.

(b) Example 2.

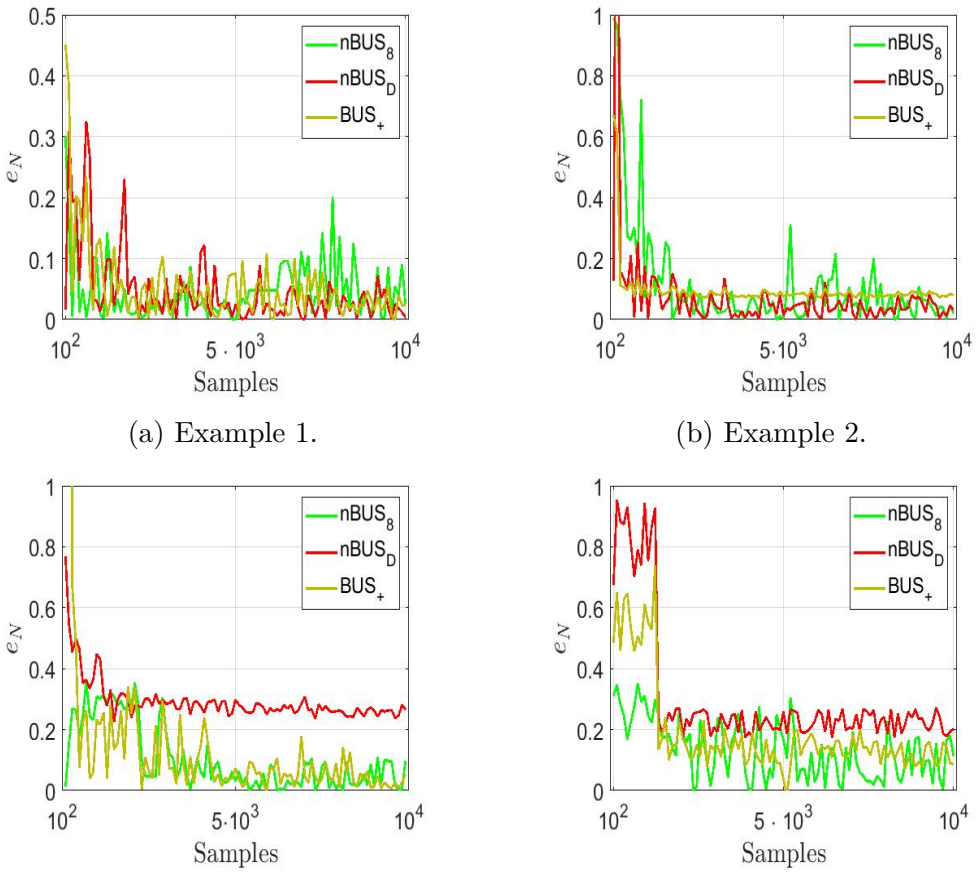
(c) Example 3.

(d) Example.4

Figure 5.12: Bias in the standard deviation of the posterior samples generated with $nBUS_8$, $nBUS_D$ and BUS_+.

Allowing for $nBUS_D$ and BUS_+ to generate samples within the $nBUS$ framework ensures as outlined in Chapter 4, that the model evidence estimation does not suffer from possible finite sample size of likelihood function evaluations as is the case with $aBUS$. Instead, the model evidence is directly influenced by the final critical threshold value used to generate posterior samples. Figure 5.13, presents their performance in estimating the model evidence. For an increase in N, as there are a greater number of limit state function evaluations available, one would expect a more accurate evidence estimate due to the accuracy of the final critical threshold. Their is a general agreement between the three frameworks in terms of the accuracy of the estimated quantity. In the case of example 3 in Figure 5.13 (c), it is apparent that for up to 10^3 samples the bias in the model evidence of $nBUS_D$ decreases before remaining approximately equal to the same level of bias for the remaining sample sizes considered. Further examination of

the final critical threshold used for $N > 10^3$ revealed that it had not converged to a large enough value at termination of $nBUS_D$. Consequently, the bias in the estimated evidence fails to decrease for an increasing N. In example 4 in Figure 5.13 (d) a sudden drop in the amount of bias for $nBUS_D$ and BUS_+ appears within a similar range of sample sizes. Given the expression for the estimated log evidence takes $p_F \approx p_0^m$, further analysis showed that both $nBUS_D$ and BUS_+ terminated simulation at the same number of $level_m$. In this case however, m appeared to have been too small for the bias in the evidence to decrease. Once $N > 7 \cdot 10^2$, $nBUS_D$ and BUS_+ take a greater number of intermediate levels and thus the bias decreases as a more appropriate $level_m$ has appeared to have been reached. This further emphasises the influence of N on the accuracy of all BUS samplers. The coefficient of variation was also computed for all N and p_0 combinations, showing a similar decrease in the variation of evidence values for an increase in N.



(a) Example 1.

(b) Example 2.

(c) Example 3.

(d) Example 4.

Figure 5.13: Bias in the estimated evidence with $nBUS_8$, $nBUS_D$ and BUS_+.

5.3.3 Hand Written Digits and Image Classification

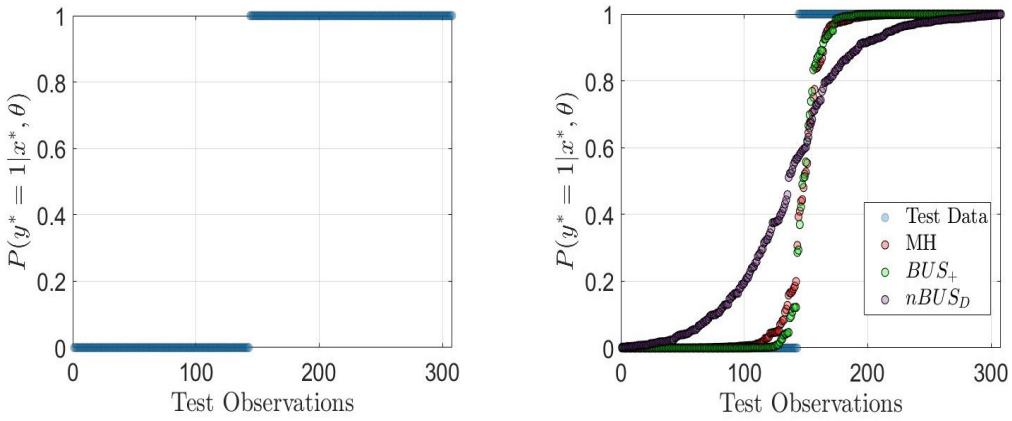
Table 5.2 presents the classification predictions using the high dimensional USPS and image data sets discussed in Chapter 4. The pre-processing of the images is done in the same manner as outlined in Chapter 4. Logistic regression is the chosen classification model and a misclassification threshold of 0.5 is used. For each data set 5- fold cross validation is used. In table 5.2 the AUC is reported with the standard errors in brackets. Let CI denote the subsequent confidence intervals. The AUC values predicted by $nBUS_D$ and BUS_+ are similar to $nBUS_8$ with slight variations in the standard error. For the USPS, the TME were 1.3×10^4, 1.1×10^4 and 2.8×10^3 for $nBUS_8$, $nBUS_D$ and BUS_+. While for the image data set, the respective TME were 1.3×10^4, 1.2×10^4 and 3.7×10^3. This shows a vast reducton in computational expense.

	USPS	**Images**
$\boldsymbol{nBUS_8}$	0.97 (0.08)	0.95 (0.01)
CI	$(0.85, 1)$	$(0.93, 0.98)$
$\boldsymbol{nBUS_D}$	0.97 (0.01)	0.98 (0.15)
CI	$(0.95, 0.99)$	$(0.84, 1)$
$\boldsymbol{BUS_+}$	0.98 (0.001)	0.98 (0.07)
CI	$(0.97, 1)$	$(0.84, 1)$

Table 5.2: Performance metrics of $nBUS_8$, $nBUS_D$ and BUS_+ on the USPS and image data sets.

Let x^* denote the test observations and y^* the test observation class labels. Figure 5.14 further investigates the predictive performance of the logistic regression model with parameters estimated by BUS_+ and $nBUS_D$. Figure 5.14 (a) presents the ground truth of the test observations of the USPS data set. Figure 4.13 (b) presents the predictive probabilities for the logistic classifier using MH (red), BUS_+ (green) and $nBUS_D$ (purple). It is evident that there is more variation in the difference of predictive probabilities between MH and $nBUS_D$ in comparison to MH and BUS_+. This also reveals that the classification accuracy of $nBUS_D$ for

this example would be more sensitive to the choice of misclassification threshold as a smaller proportion of test predictions have been mapped to the extrema values of 0 and 1.



(a) Correct Classification. (b) Classification Comparison.

Figure 5.14: Predictive performance of logistic regression on the image dataset using the four different samplers for parameter estimation. Panel (a) shows the ground truth, panel (b) shows a comparison between MH, BUS_+ and $nBUS_D$.

5.4 Comments on Sampler Convergence

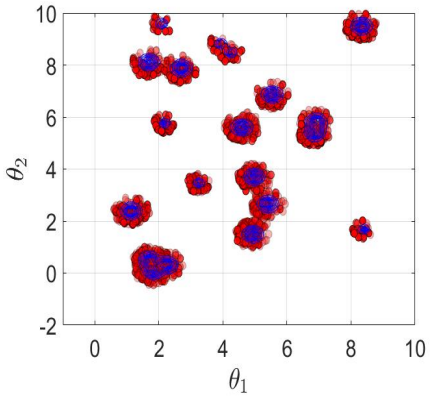
The stopping conditions proposed in this chapter aim to reduce the computational burden of $nBUS$ while also offering statistical guarantees of sampling from the correct probability distribution. In terms of convergence, the sample sets generated by both samplers may contain a degree of bias due to the nature of the conditions themselves. Firstly, $nBUS_D$ makes use of the nested loop of $nBUS$ once the set of inadmissible samples is empty at any given $level_m$. Theorem 1 discussed in Chapter 4 and presented in [67] proves that as $m \to \infty$ the probability measure h_m tends to zero. Therefore each sample is drawn from the posterior with certainty as the sampler has converged to the correct distribution. As previously discussed however, in practice due to the increased correlation among samples for an increasing m along with the precision restrictions of computing programmes, a very small stopping tolerance ϵ is chosen. The larger the choice of ϵ the greater the difference between ϵ and zero becomes i.e. $|\epsilon - 0|$. As $nBUS_D$ makes use of the nested loop, the choice of ϵ directly influences the sampler for any $\epsilon > 0$. This means that like $nBUS$, a degree of bias may be present in the sampling process for any $\epsilon > 0$. By the $nBUS$ formulation, it is known that once a b_m has been

generated such that $b_m > b_{min}$, the samples are always distributed according to the target posterior distribution [67]. With respect to BUS_+, the generation of intermediate levels is halted once a transition in the relationship between the model evidence and probability of failure has occurred. This signifies that at $level_m$ $b_m > b_{min}$. The use of the post processor in BUS_+ enables simulation to be terminated with statistical assurances of $b_m > b_{min}$. The ratio threshold t chosen allows for potential sampling error to be accounted for. However with this comes the introduction of bias into the posterior set. The larger the value, theoretically the smaller the level of bias but the larger the potential computational cost. However a smaller value of t will introduce a larger level of bias into the sampling process. While $t < 1$, in principle BUS_+ will not offer exact perfect sampling of the posterior distribution. An investigation of the sampling quality of $nBUS_D$ and BUS_+ with respect to the values of ϵ and t is presented in the following. Consider the target distribution as a mixture of 20 bivariate Gaussian's.

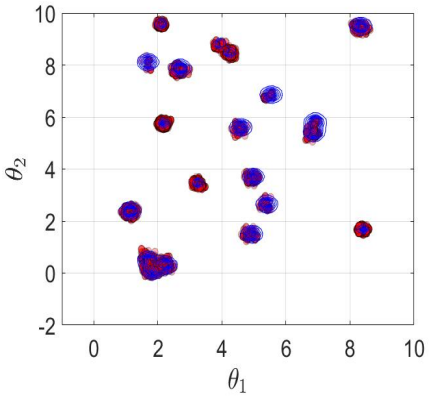
$$f_{\boldsymbol{\Theta}}(\boldsymbol{\theta}) = \sum_{i=1}^{20} \frac{w_i}{2\pi\sigma^2} e^{\left(\frac{-1}{2\sigma^2}(\boldsymbol{\theta}-\mu_i)^T(\boldsymbol{\theta}-\mu_i)\right)} \tag{5.41}$$

where $\{\mu_1, ...\mu_{20}\}$ are specified in [129]. Each mixture component is assigned a weight w of 0.05. Unlike in [129], a variance 0.09 is assigned to the first four Gaussians and 0.025 to the remainder. The reasoning for this is to establish the performance of BUS_+ and $nBUS_D$ with respect to t and ϵ on a multi-modal distribution with varying supports. Firstly consider the case for BUS_+ using $N = 5000$ and $p_0 = 0.1$. Figure 5.15 presents the samples produced by BUS_+ using $t = 0.5$ and $t = 1$. It is evident that for $t = 1$, a larger proportion of samples are concentrated within the individual contour plots of the posterior distribution. Whereas for $t = 0.5$ the majority of samples appear to lie outside the target density regions. To establish the proportion of samples within each mode, a variation of the *mode-coverage* metric presented in [231] is implemented for quantitative purposes. This is defined as once the number of samples falling within a specified radius of the mode centre is greater than a predefined threshold, the mode is considered covered. Given the equal weightings of each mixture component, a threshold of $N/20$ is proposed, where N is the number of generated samples. In line with [231] the radius value is set to 0.25. If each mode has an equal proportion of the generated samples, all modes are considered to have been adequately populated. A value of 0 denotes no mode has been fully populated

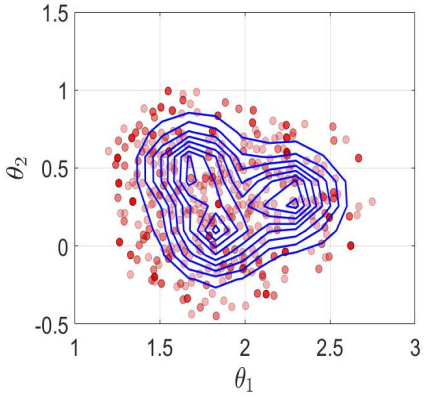
and a value of 1 that each mode contains an equal number of samples. For $t = 0.5$ the total mode coverage is 0. Even though a large proportion of samples have populated each mode, the samples Figure 5.15 (a) are widely dispersed. In contrast, $t = 1$ the total mode coverage is 0.75. This is evident from Figure 5.15 (b) where the majority of samples are concentrated at the individual modes. Further analysis of the samples are presented in Figure 5.15 (c) and Figure 5.15 (d) for the mode located closest to the origin of the sample space. Evidently for $t = 0.5$ a large proportion of samples lie outside the supports of this posterior mode. This is an indication that additional levels of BUS_+ are required to ensure the vast majority of samples lie within the density region. In contrast, BUS_+ with $t = 1$ required an additional intermediate level with the samples presented Figure 5.15 (d).



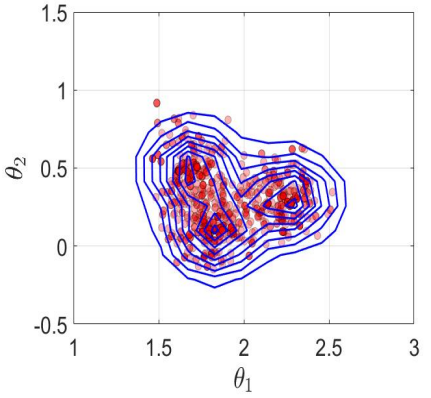
(a) Posterior samples using $t = 0.5$.



(b) Posterior samples using $t = 1$.



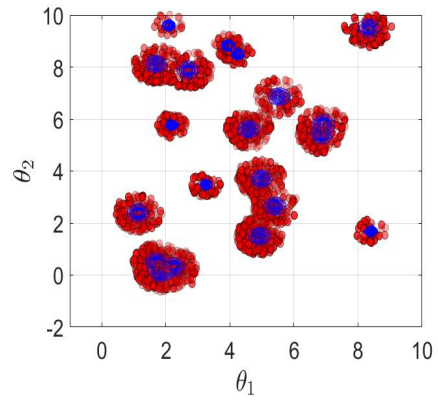
(c) Analysis of mode using $t = 0.5$.



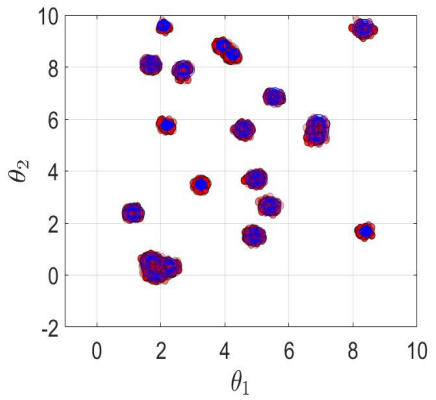
(d) Analysis of mode using $t = 1$.

Figure 5.15: Generation of posterior samples with BUS_+ using different values for t. Panel (a) and panel (b) present the posterior samples for $t = 0.5$ and $t = 1$. Panel (c) and panel (d) present the population of posterior samples located in the mode closest to the origin of the sample space.

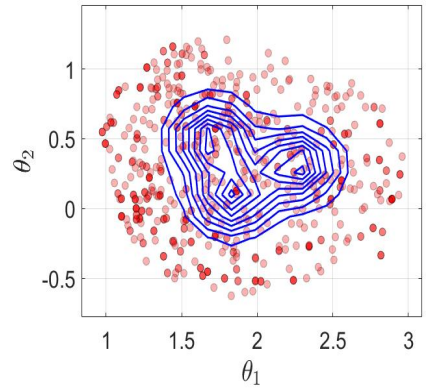
Figure 5.16 presents the behaviour of the sampling process in $nBUS_D$ for a varying ϵ. As previously discussed, once the set of inadmissible samples is empty at any given level h_m is computed with respect to ϵ. For $\epsilon = 10^{-4}$, the average mode coverage is 0 while for $\epsilon = 10^{-8}$ the average mode coverage is 0.79. Note that for $\epsilon = 10^{-8}$ $nBUS_D$ required an additional intermediate level as even though the inadmissible set of samples was empty, h_m was larger than $\epsilon = 10^{-8}$. The influence of this additional level can be seen in the sampling accuracy in Figure 5.16 (d) in comparison to Figure 5.16 (c).



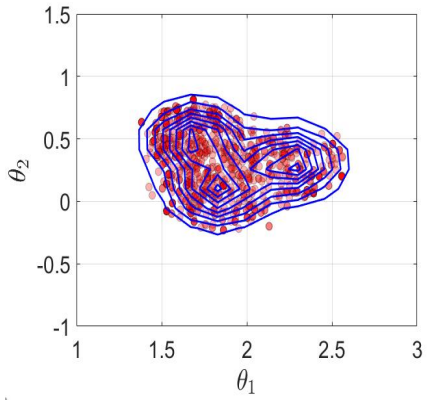
(a) Posterior samples using $\epsilon = 10^{-4}$.



(b) Posterior samples using $\epsilon = 10^{-8}$.



(c) Analysis of mode using $\epsilon = 10^{-4}$.



(d) Analysis of mode using $\epsilon = 10^{-8}$.

Figure 5.16: Generation of posterior samples with $nBUS_D$ using different values for ϵ. Panel (a) and panel (b) present the posterior samples for $\epsilon = 10^{-4}$ and $\epsilon = 10^{-8}$. Panel (c) and panel (d) present the population of posterior samples located in the mode closest to the origin of the sample space.

From the above it is evident that the choices of t and ϵ directly influences the sampling convergence properties of BUS_+ and $nBUS_D$. For a larger t and smaller ϵ, each sampler becomes more and more accurate. However this comes at a cost of an increased number of TME.

5.5 MCMC Schemes: Influence on Efficiency

Aside from the stopping criterion, the generation of samples at each intermediate level of BUS greatly influences the efficiency and sample quality of the framework. In general, as a Markov Chain progresses, the correlation between samples increases and thus less information may be extracted from the generated

posterior samples. To protect against this potential loss of information, aside from identifying the minimal required level to sample from the correct distribution, an efficient exploration of the failure domain is also of great importance. As outlined in Chapter 4, the sample generation in SuS may be carried out in a two step process. At the first step, a sample is proposed being either rejected or accepted. While at the second step, it is checked whether the sample is indeed a failure sample.

As discussed in Chapter 4, the original MCMC framework proposed for SuS (MMH) [11] allows for the efficient generation of samples in high dimensions by assigning a one dimensional proposal PDF to the proposed sample for each dimension. MMH generates sample candidates using proposal PDFs whose mean values change corresponding to the last accepted sample. The candidate being accepted if the output of the limit state function is greater than the corresponding intermediate threshold. A number of alternative MCMC methods for SuS have been developed. Designed to avoid low acceptance rates while achieving an efficient exploration of the failure domain, the methods differ in their treatment of the candidate sample.

To investigate the potential of further enhancing the performance of BUS, three MCMC frameworks are compared against the MMH algorithm. As outlined in Chapter 3, the choice of proposal distribution directly influences the efficiency of MH based samplers. Delayed rejection essentially finds new proposals and amends the Metropolis acceptance ratio accordingly to obtain more efficient mixing [97]. Suppose the initial pre-candidate sample rejected in the SuS MCMC process, the first method concerns repeated pre-candidate generation. In the event that the initial pre-candidate sample is rejected, the Delayed MMH (DMMH) [149] algorithm generates a second pre-candidate from an updated proposal distribution. This updated proposal is conditioned on the first pre-candidate sample which has been rejected. Therefore, reducing chain correlation since fewer repeated pre-candidate samples will occur. Consider the current state of the Markov chain to be $\boldsymbol{\theta} = \{\theta_1, ..., \theta_d\}$. As with MMH the proposal distribution is expressed as a product of independent univariate PDFs to directly obtain the candidate state $\boldsymbol{v} = \{v_1, ..., v_d\}$. Firstly, a pre-candidate sample w_j is drawn from a univariate proposal $S_j(\cdot|\theta_j)$ dependent on the component θ_j of the current state. The Metropolis ratio $r(w_j)$ is computed with w_j either being accepted or rejected as the pre-candidate sample. If w_j is accepted with respect to $r(w_j)$, v_j is set to w_j.

Otherwise a second pre-candidate sample $w_{j,2}$ is drawn from $S_j(\cdot|\theta_j, w_j)$ Again a Metropolis ratio $r(w_{j,2})$ is computed with $w_{j,2}$ either being accepted or rejected as the pre-candidate sample. If $w_{j,2}$ is accepted v_j is set to $w_{j,2}$. Otherwise v_j is set to θ_j i.e the j^{th} component of the current state of the Markov chain. Once all j components of $\boldsymbol{v}$ have been populated, it is checked whether $\boldsymbol{v} \in F$. If the candidate sample belongs in the failure domain it is accepted as the next state of the Markov chain and $\boldsymbol{\theta'} = \boldsymbol{v}$. If not, the next state of the Markov chain is $\boldsymbol{\theta'} = \boldsymbol{\theta}$. By repeating the sample generation at the first step of the MCMC process for SuS, no additional calls to the likelihood function are required. Similar to MMH, DMMH adopts the component wise update of the parameter vector with the only difference being this repeated pre-candidate generation. A pseudo-code for DMMH is provided in Algorithm 12.

Algorithm 12 Delayed Modified Metropolis-Hastings [149]

Define the current state of the Markov chain $\boldsymbol{\theta} = \{\theta_1, ..., \theta_d\}$
Define the univariate proposal PDFs $S_j(\cdot|\theta_j)$ for $j = 1, ..., d$
for each $j = 1, ..., d$ **do**
    Generate a pre-candidate w_j from $S_j(\cdot|\theta_j)$
    Generate u_j uniformly on $[0, 1]$
    $r(w_j) = \frac{\pi(w_j)}{\pi(\theta_j)} \cdot \frac{S_j(\theta_j|w_j)}{S_j(w_j|\theta_j)}$
    if $u_j < r(w_j)$ **then**
        Accept $v_j = w_j$
    else
        Generate a pre-candidate $w_{j,2}$ from $S_j(\cdot|\theta_j, w_{j,1})$
        Generate $u_{j,2}$ uniformly on $[0, 1]$
        $r(w_{j,2}) = \frac{\pi(w_{j,2})}{\pi(\theta_j)} \cdot \frac{S_j(w_j|w_{j,2})}{S_j(w_j|\theta_j)} \cdot \frac{S_j(\theta_j|w_j,w_{j,2})}{S_j(w_{j,2}|\theta_j,w_j)} \cdot \frac{1-r(w_j,w_{j,2})}{1-r(w_j)}$
        where $r(w_j, w_{j,2}) = \frac{\pi(w_j)}{\pi(w_{j,2})} \cdot \frac{S_j(w_{j,2}|w_j)}{S_j(w_j|w_{j,2})}$
        Accept $v_j = w_{j,2}$ if $u_{j,2} < r(w_{j,2})$, otherwise $v_j = \theta_j$
    end if
end for
$\boldsymbol{v} = \{v_1, ..., v_d\}$
Accept $\boldsymbol{\theta'} = \boldsymbol{v}$ if $\boldsymbol{v} \in F$, otherwise $\boldsymbol{\theta'} = \boldsymbol{\theta}$

As discussed in the section on adaptive MCMC methods in Chapter 3, a proportion of developed MCMC algorithms make use of the findings in [188, 187] regarding the optimal sample acceptance rate in MH. In a similar fashion, Adaptive

Conditional Sampling (ACS) [172], adaptively scales the standard deviation of the proposal distribution for each proposed sample. As MMH updates the parameter vector in a component wise manner, ACS adaptively tunes the univariate proposal distributions to ensure that the sample acceptance rate of the Markov chain is close to 0.44. In [187] it was shown that 0.44 is the optimal acceptance rate for Markov chains with one dimensional Gaussian proposal distributions. Directly utilizing this optimal acceptance rate, ACS nullifies the requirement for defining the spread of a proposal distribution through the use of information contained in the standard deviation of the samples. Consider the current state of the Markov chain to be $\boldsymbol{\theta} = \{\theta_1, ..., \theta_d\}$. As with DMMH and MMH the proposal distribution is expressed as a product of independent univariate PDFs to directly obtain the candidate state $\boldsymbol{v} = \{v_1, ..., v_d\}$. ACS seperates the set of nc seeds for the independent Markov chains of SuS into smaller subsets of seeds for the sample generation process. Firstly, a pre-candidate sample w_j is drawn from a univariate Gaussian proposal $\mathcal{N}_j(\rho \cdot \theta_j, z)$ where ρ is a correlation parameter and z the distributions standard deviation. Next, w_j is accepted as v_j. Once all j components of $\boldsymbol{v}$ have been populated, it is checked whether $\boldsymbol{v} \in F$. If the candidate sample belongs in the failure domain it is accepted as the next state of the Markov chain and $\boldsymbol{\theta'} = \boldsymbol{v}$. If not, the next state of the Markov chain is $\boldsymbol{\theta'} = \boldsymbol{\theta}$. The standard deviation z and correlation parameter ρ of the proposal distribution is then updated with the above process repeated for the remaining number of Markov seeds. As the parameters of the proposal distribution are tuned on the fly, the pre-candidate acceptance/rejection step of MMH is no longer required. Therefore, ACS draws a sample in a component wise manner from a sequence of adaptively tuned proposal distributions before checking whether or not the proposed sample belongs in the failure domain. A pseudo-code for ACS is provided in Algorithm 13.

Algorithm 13 Adaptive Conditional Sampling [172]

Define the current state of the Markov chain $\boldsymbol{\theta} = \{\theta_1, ..., \theta_d\}$
Define the length of each chain to be updated: $N_a = p_a \cdot nc$, where $p_a \in [0.1, 0.2]$ [172] and $nc = p_0 N$ as in SuS
Define the initial proposal distribution standard deviation: $z_0 = 1$
Define the correlation parameter scaling parameter: $\lambda = 0.6$
Compute the proposal distribution standard deviation: $z = \min(\lambda z_0, 1)$
Define the Gaussian univariate proposal PDFs $\mathcal{N}_j(\theta_j, z)$ for $j = 1, ..., d$
Compute the correlation parameter: $\rho = \sqrt{1 - z^2}$
Permute randomly the nc seeds into R groups each containing N_a seeds
for each $r = 1, ..., R$ **do**
    for each $j = 1, ..., d$ **do**
        Generate a pre-candidate $w_j \sim \mathcal{N}_j(\rho \cdot \theta_j, z)$
        Accept $v_j = w_j$
    end for
    $\boldsymbol{v} = \{v_1, ..., v_d\}$
    Accept $\boldsymbol{\theta'} = \boldsymbol{v}$ if $\boldsymbol{v} \in F$, otherwise $\boldsymbol{\theta'} = \boldsymbol{\theta}$
    Compute the average chain acceptance rate: $A = \frac{1}{N_a} \sum_{j=1}^{N_a} I_F[\theta_r]$
    Update the scaling parameter: $\lambda = 10^{(\log_{10}(\lambda) + r^{-0.5}(A - 0.44))}$
    Update the standard deviation: $z = \min(\lambda z_0, 1)$
    Update the correlation parameter: $\rho = \sqrt{1 - z^2}$
end for

Like ACS, the final MCMC variant to be considered introduces a user defined standard deviation to control the individual spreads of the one dimensional proposal distributions. Subset Infinity (SuS-Inf) [10], expresses the candidate sample as a Gaussian vector whose statistics depend on the last accepted sample. As SuS functions in standard normal space, SuS-Inf takes advantage of the property that a linear combination of Gaussian variables is simply a Gaussian. The name of the sampler stems from the possible infinite number of Gaussian univariate PDFs being combined to produce a Gaussian proposal distribution. Unlike ACS however, the spread of the proposal PDFs in SuS-Inf are user defined with the mean of the proposals adaptively defined with respect to the current state of the Markov chain. Consider the current state of the Markov chain to be $\boldsymbol{\theta} = \{\theta_1, ..., \theta_d\}$. As with DMMH, MMH and ACS, the proposal distribution is expressed as a product of independent univariate PDFs to directly obtain the

candidate state $\boldsymbol{v} = \{v_1, ..., v_d\}$. Firstly, a pre-candidate sample w_j is drawn from a univariate Gaussian proposal $\mathcal{N}_j(t \cdot \theta_j, z)$ where t is a scaling parameter and z the distributions standard deviation. Next, w_j is accepted as v_j. Once all j components of $\boldsymbol{v}$ have been populated, it is checked whether $\boldsymbol{v} \in F$. If the candidate sample belongs in the failure domain it is accepted as the next state of the Markov chain and $\boldsymbol{\theta'} = \boldsymbol{v}$. If not, the next state of the Markov chain is $\boldsymbol{\theta'} = \boldsymbol{\theta}$. Unlike ACS, *SuS*-Inf requires a user defined standard deviation z. Note that the scaling parameter t is equivalent to the correlation parameter in ACS. The difference is that in *SuS*-Inf z does not change from its user defined value. As with ACS, no pre-candidate acceptance/rejection step is required. A pseudo-code for *SuS*-Inf is provided in Algorithm 13.

Algorithm 14 Subset Simulation Infinity [10]

1: Define the current state of the Markov chain $\boldsymbol{\theta} = \{\theta_1, ..., \theta_d\}$
2: Define the proposal distribution standard deviation: $z = 0.5$ [10]
3: Define the Gaussian univariate proposal PDFs $\mathcal{N}_j(\theta_j, z)$ for $j = 1, ..., d$
4: Compute the scaling parameter for the mean of the proposal distribution: $t = \sqrt{1 - z^2}$
5: **for** each $j = 1, ..., d$ **do**
6: Generate a pre-candidate $w_j \sim \mathcal{N}_j(t_j \cdot \theta_j, z)$
7: Accept $v_j = w_j$
8: **end for**
9: $\boldsymbol{v} = \{v_1, ..., v_d\}$
10: Accept $\boldsymbol{\theta'} = \boldsymbol{v}$ if $\boldsymbol{v} \in F$, otherwise $\boldsymbol{\theta'} = \boldsymbol{\theta}$

It is worth noting that all MCMC variants discussed, adopt the same second step of MCMC for *SuS* by checking whether the candidate sample belongs in the failure domain. They are primarily concerned with improving the sample quality and acceptance efficiency. The issue of potentially reducing computational cost through the limit state function at the second step is addressed in Chapter 6. The study in [45] is extended to analyse the performance of the MCMC schemes in *BUS*. The quality of performance is measured in terms of TME, posterior sample correlation and the number of effective independent samples. The expression in *SuS* for the correlation between samples is given by [12]

$$\gamma_m = 2 \sum_{i=1}^{N_s - 1} \left(1 - \frac{i}{N_s}\right) \frac{R_m^i}{R_m^0} \tag{5.42}$$

$$R_m^i = \mathbb{E}[I_m(\theta_{m-1}^{(1)(k)}) I_m(\theta_{m-1}^{(1+i)(k)})] - p_0^2 \tag{5.43}$$

where R_m^i is the auto-covariance between samples at lag i, k the number of Markov chains in the MCMC phase and m the number of required levels. In SuS literature [235] it is widely acknowledged that the smaller the γ the more information contained in the samples and thus the higher the model efficiency. As expected, the greater the number of levels the higher the correlation between samples. To capture the number of effectively independent samples (N_{eff}) [32] the following metric is used

$$N_{eff} = \frac{N}{1 + \gamma_m} \tag{5.44}$$

In the event of samples having a correlation of 0, Eq. 5.44 reduces to N. In general, the smaller the N_{eff} the greater the sample dependence. Table 5.3 presents a comparison study for each MCMC variant for all six examples discussed in this chapter. The results were achieved using BUS_+ with $N = 1000$ and $p_0 = 0.1$ averaged over a large number of independent runs. The term in brackets for the sample correlation represents the respective c.o.v.. Stemming from the delayed acceptance of a candidate sample at the first step of sampling, DMMH requires a similar number of TME with respect to MMH. Overall the difference in computational expense for all methods appears negligible. This is expected as they primarily differ in the initial generation of the candidate sample. In terms of correlation, DMMH outperforms MMH for examples (1-4) in small dimensions but the performance deteriorates for the higher dimensional cases. In all cases, ACS and SuS-Inf produce a greater number of effectively independent samples, which stems from the lower correlation between the posterior set. It should be noted, given the spread of the proposal distribution is initialized by a user defined standard deviation z in SuS-Inf, the resulting sample acceptance rate is highly influenced by the chosen value. ACS relates the proposal distribution to the optimal acceptance of 0.44 [187, 188]. With respect to this and the similarity of the experimental results between the two methods, ACS is recommended to be implemented in conjunction with BUS to avoid the issue of selecting a standard deviation value for SuS-Inf which directly influences the algorithms efficiency.

	Metric	Ex 1	Ex 2	Ex 3	Ex 4	USPS	Images
MMH	TME	8.5×10^3	8.2×10^3	5.9×10^3	3.5×10^3	1.5×10^4	1.3×10^4
	γ	3.95(0.1)	2.22(0.2)	3.88(0.1)	9.53(0.1)	4.71(0.2)	4.26(0.3)
	N_{eff}	202	310	170	95	175	190
DMMH	TME	8.6×10^3	8.3×10^3	5.9×10^3	3.6×10^3	1.5×10^4	1.4×10^4
	γ	2.84(0.2)	2.17(0.1)	3.17(0.3)	8.81(0.2)	4.81(0.3)	4.35(0.3)
	N_{eff}	260	315	240	102	172	187
ACS	TME	8.2×10^3	8.7×10^3	5.4×10^3	4.3×10^3	1.1×10^4	1.2×10^4
	γ	1.76(0.3)	2.09(0.2)	2.12(0.2)	5.12(0.2)	4.29(0.1)	3.87(0.3)
	N_{eff}	362	323	320	164	189	205
***SuS*-Inf**	TME	8.1×10^3	8.5×10^3	5.6×10^3	4.3×10^3	1.2×10^4	1.2×10^4
	γ	1.66(0.2)	1.78(0.3)	2.57(0.1)	5.41(0.1)	4.52(0.2)	5.72(0.2)
	N_{eff}	375	360	280	156	181	149

Table 5.3: Performance comparison for MCMC variants using BUS_+ with $N = 1000$ and $p_0 = 0.1$. All results were achieved using 100 independent model runs.

5.6 Chapter Summary

Motivated by the added computational expense stemming from the nested loop stopping criterion presented in Chapter 4, this chapter presents two alternative stopping criteria which provide statistical assurances of termination. Both methods are aimed at reducing the number of Total Model Evaluations (TME) to solve the Bayesian updating task, while retaining the sample and model evidence correctness of $nBUS$.

The first criterion, adopts the direct application of the rejection principle in areas of the parameter space which are far away from the target domain. By doing so, avoiding the computation of additional likelihood function calls. To

account for the potential sampling error stemming from the finite set of generated samples, once all samples at a given level have been accepted as being conditional on the target failure event, the nested loop stopping condition is implemented. This in turn provides a sense of statistical assurance around the sampling process. The resultant framework is termed $nBUS_D$.

The second criterion, takes advantage of a transition in the relationship between the probability of failure and model evidence to identify when generated samples are from the posterior distribution. Through the introduction of a Bayesian interpretation of the probability of failure, stochastic variation in the model evidence along with possible sampling error is accounted for, allowing for simulation to be terminated with a degree of confidence. In turn, ensuring that a large number of sampler runs are not required in order to compute an average of the log model evidence. The resulting framework is termed BUS_+. Both methods were found to perform comparably well in terms of statistical estimation and model evidence estimation with respect to $nBUS$ while drastically reducing the cost of a model run.

Aside from determining the level of termination within BUS, the choice of MCMC method directly influences sample quality and the efficiency of sample generation. Three alternative MCMC schemes specifically designed for SuS were compared against MMH to identify any further potential efficiency advances when used with the proposed stopping criteria. It was found that for the examples considered, ACS proved to be the most efficient sampling scheme along with SuS-Inf. However, given the dependence of SuS-Inf on choice of proposal distribution spread, ACS is the preferred MCMC scheme implemented for the remaining chapters of this dissertation.

With respect to the BUS sampling process, is noted that the reformulated MCMC schemes for SuS primarily focus on the first stage of sampling to improve sample quality and overall model efficiency. As previously discussed however, the majority of computational expense in BUS stems from the number of limit state function calls. Similar to general MCMC frameworks, this involves the computation of the likelihood function for each proposed sample using the available data. In the presence of large data sets, the computational time for each likelihood calculation increases greatly, resulting in MCMC based methods becoming very expensive to estimate model parameters. The following chapter presents an approach, aimed at making BUS more scalable to such problems.

Chapter 6

BUS for Big Data

The stopping criteria proposed in the previous chapter improve the computational efficiency of $nBUS$ whilst providing statistical guarantees. In its current form, the computation of the likelihood function for each proposed sample renders the framework unsuitable for large datasets. To improve its scalability to such scenarios, Support Vector Machines (SVM) as discussed in Chapter 2 are integrated into the BUS approach. The proposed method reduces the computational expense by treating the engineering reliability problem as a binary classification problem. For the benefit of the reader, some concepts previously discussed are reintroduced in this chapter.

6.1 Introduction

Advancements in computational resources have allowed for MCMC sampling to become a practical strategy for Bayesian computation. The emergence of large datasets however, has decreased the suitability of MCMC based methods. Each proposed candidate sample requires an evaluation of the likelihood function for the entire data set. This means that for a large number of observations, many individual computations are required to gain one bit of information, namely whether to accept or reject the sample. This issue arises in BUS at the second step of the MCMC sampling strategy, where in order to determine whether a sample belongs in the failure domain, the likelihood function for the entire training set is required to be computed.

In the past decade, interest in the area of scalable MCMC algorithms has grown immensely. One line of research has focused on the interpretation of

MCMC sampling as a stochastic optimization problem which directly removes the computation of the ratio of likelihood functions at a cost of producing approximate samples [223, 52]. The majority of methods however, can be broadly categorized in terms of divide-and-conquer and sub-sampling approaches [15]. A natural way to approach the MCMC challenge is to divide the dataset into a series of batches, run MCMC on each batch separately and then combine results [198, 164]. However, the question of how the batch posterior approximations can be efficiently combined remains a challenge. Sub-sampling approaches are primarily concerned with reducing the number of data observations required to compute a value of the likelihood function for each sample. By incrementally sampling random mini batches of the data, these methods aim to provide a degree of confidence that the error between the computed likelihood and true likelihood is very small [128, 14]. However, the amount of data consumed for the likelihood computation from mini batch to mini batch varies significantly.

Differing from standard MCMC methods, the two step sampling process in *BUS* results in the computational expense stemming from the evaluation of the limit state function. To address this issue, a widely adopted approach in reliability analysis is to introduce a function approximation referred to as a *surrogate*. A surrogate may be viewed as an approximation of the underlying function of interest aimed at mimicking the behaviour of the true function as closely as possible while greatly reducing computational expense. It should be noted that the area of surrogate models in reliability analysis is vast and as such the focus of this chapter is primarily on methods which increase computational efficiency with a particular emphasis placed on SVM (see Chapter 2) inspired approaches. The pioneering work of [189] and [111] has resulted in the application of machine learning methods to reliability tasks. By outlining a clear analogy between a binary classification problem and the engineering reliability problem, methods such as SVM may be used as approximations to the costly limit state function. In turn, allowing for non-expensive approximates of the probability of failure to be obtained.

An importance sampling and SVM inspired framework has been proposed [112], aimed at reducing the number of limit state function evaluations. SVM is utilised to select samples evaluated with the true function while importance sampling narrows the region of interest in the input domain. Simulation is terminated once the variation between successive probability of failure estimates is below a chosen tolerance. In line with [112], [169] extract information contained in the samples

belonging in the margin of the SVM, before an adaptive strategy using Latin Hypercube Sampling (LHS) is implemented. The stopping criterion is dependent on the number of samples located within the margin. A similar adaptive strategy is proposed in [72] with an adaptive support vector regression framework presented in [39].

To enable the application of SVM to problems with disjoint failure domains, [17] combine LHS with K-means clustering to allow for SVM to be trained in different locations within the input domain. The kernel parameter is estimated using a grid search scheme, while an appropriate training set size is considered using an information metric. In terms of quantifying the uncertainty in the SVM output, [16] address the short comings of [176] probabilistic extension of SVM for reliability analysis tasks. By considering the spatial variation between samples, [16] present a modification allowing for the probability of misclassifying the class membership of samples to be estimated.

Aside from SVM, numerous machine learning techniques have been integrated into reliability analysis frameworks. In particular but not limited to, Gaussian processes [59, 23, 108], neural networks [171, 170, 90] and clustering [229]. A number of authors have provided comparisons between different surrogate model approaches [111, 181]. Given the relative infancy of the application of reliability methods to Bayesian updating tasks, the vast majority of pattern recognition based surrogates have been developed with the primary quantity of interest being the probability of failure. Taking advantage of the interpretation of the engineering reliability problem as a classification problem, a surrogate based framework which approximates the limit state function is proposed in this chapter. SVM are integrated into the *BUS* sampling scheme to reduce the required number of likelihood evaluations for a given problem.

6.1.1 Problem Formulation

As previously discussed in Chapter 4, the engineering reliability problem can be formulated by a limit state function which represents the behaviour of a complex system. The system's reliability is studied through the characterisation of a failure domain. In this context, the failure domain is defined as the system response exceeding a given capacity. The objective is the estimation of the probability of the system being in a state of unacceptable performance (i.e. failure). Despite *SuS* bringing significant improvements in comparison to Direct Monte Carlo (DMC)

in terms of computational cost and required sample size, direct evaluation of the limit state function can still be quite expensive. For example, in the case of a complex finite element model. In the context of BUS, as stated in Chapter 5, each call of the limit state function requires an evaluation of the likelihood function. The cost of this is dependent on the size of the underlying observed data set. As such, the computational complexity of identifying appropriate parameter values increases with the number of data observations. To reduce the cost of the limit state function evaluation, an approximation to the response output can be made.

The identification of a system's state can be interpreted as a binary response of being either acceptable (safe) or unacceptable (failure). This observation allows for the estimated $\hat{p}_F$ to be expressed in terms of an indicator function $I[\cdot]$. Recall from Chapter 4, the expression for the limit state function $g(\boldsymbol{\theta})$ with the estimated $\hat{p}_F$ given by

$$\hat{p}_F = \frac{1}{N}\sum_{i=1}^{N} I[g(\boldsymbol{\theta}_i) < b] \tag{6.1}$$

This indicator function $I[\cdot]$ is equal to 1 if the argument is true and 0 otherwise. The formulation of the failure probability in Eq. 6.1, suggests that the outcome of the calculation is dominated by the sign of the limit state function. Adopting this viewpoint has enabled the reliability problem to be interpreted as a binary classification task [189, 111]. This observation has widened the spectrum of methods suitable for reliability analysis.

The classical interpretation of the reliability engineering problem estimates the probability of the limit state output response being less than or equal to zero, that is $p(g(\boldsymbol{\theta}) \leq 0)$. In essence, the optimal threshold boundary is at exactly zero. Hurtado et al. [111] acknowledge that a margin based classifier such as SVM for which the optimal hyperplane equals zero forms the basis of a natural function approximation for the engineering reliability problem. Interpreting the failure domain as a binary task with the safe domain defines a classification function of the form

$$c(\boldsymbol{\theta}) = sgn(g(\boldsymbol{\theta})) \tag{6.2}$$

In which the sign of the limit state function is equal to 1 for the safe domain and -1 for the failure domain. The subsequent implementation of SVM to estimate

the probability of failure relies on the sign of the approximated performance function i.e. whether or not the input has resulted in failure in the output. By constructing a boundary between the two classes, Hurtado [110] showed the suitability of SVM to reliability analysis over other classification techniques for numerous reasons. The definition of the margin allows for a natural relationship with reliability analysis to be formed. Maximization of the margin may be treated as a quadratic programming problem (see Chapter 2), which can be solved through standard techniques. This is in contrast to neural networks which require non-linear optimization techniques to compute model weights. SVM are also scalable to high dimensions while they offer flexibility in terms of kernel choice to allow for non-linear limit state functions to be approximated.

6.2 SVM Inspired Reliability Samplers

The following section discusses the possibility of SVM reducing the computational cost of the BUS samplers. Firstly an existing SVM-SuS framework for reliability analysis is reviewed. The method, termed Subset simulation by Support vector Margin Algorithm for Reliability esTimation (2SMART) [40] makes use of SVM to replace the computationally intense limit state function evaluations in SuS. At every intermediate level of SuS, 2SMART trains a large number of SVM classifiers to replace the limit state function evaluation of new proposed samples in the MCMC process. By defining selection criteria for training sets, it is argued in [40] that 2SMART forms an accurate surrogate while reducing computational cost in comparison to SuS. A detailed description and a critical appraisal of 2SMART is offered. As an alternative to 2SMART, a new sampling algorithm termed BUS_{SVM} for Bayesian inference tasks is presented. Unlike 2SMART, BUS_{SVM} trains a single SVM classifier at each intermediate level of SuS and offers the use of the automatic stopping conditions presented in Chapter 5 to correctly stop the sampling process. Additionally, BUS_{SVM} significantly simplifies the coding implementation aspect of the sampler.

6.2.1 2SMART Method Overview

In this section, properties of 2SMART are discussed and potential improvements are identified for the application to Bayesian updating. 2SMART seeks to build an analytical surrogate to the limit state function $g(\boldsymbol{\theta})$ through the SVM output

$\hat{g}(\boldsymbol{\theta})$. Under SuS, it is very unlikely that samples which constitute a rare event will be generated at $level_0$ [108]. As such, a data driven surrogate like SVM may not be of sufficient quality around the estimated failure regions. Under 2SMART the failure event of interest is

$$F = \{\boldsymbol{\theta} \in \mathbb{R}^d : \hat{g}(\boldsymbol{\theta}) \leq b\} \tag{6.3}$$

where $\hat{g}(\boldsymbol{\theta})$ is the output of the SVM and b the target failure threshold. By assuming that their exists a perfect classifier at $\hat{g}(\boldsymbol{\theta}) = b$, 2SMART iteratively generates an approximation to the limit state function at each intermediate level of SuS to estimate $\hat{p}_F$. A pseudo-code for the 2SMART framework is presented in Algorithm 15. At $level_0$ akin to SuS, $\{\boldsymbol{\theta}_0^i\}_{i=1}^{N_n}$ samples are drawn from the input PDF $\pi(\cdot)$ while $\{\boldsymbol{e}_0^i\}_{i=1}^{N_u}$ samples are drawn uniformly from a d-dimensional sphere centred at $(0, 0)$. The variable $\boldsymbol{e}$ may be viewed as an auxiliary variable generated to act as a space filling strategy. The parameter of interest for the calculation of $\hat{p}_F$ is $\boldsymbol{\theta}$. The parameters N_n and N_u are user defined sample sizes with values chosen as $N_n = 100$ and $N_u = 50$ in [40]. For each $\{\boldsymbol{\theta}_0^i\}_{i=1}^{N_n}$ and $\{\boldsymbol{e}_0^i\}_{i=1}^{N_u}$ the limit state function $g(\boldsymbol{\theta}_0)$ and $g(\boldsymbol{e}_0)$ is evaluated. In 2SMART the intermediate failure threshold b_m is chosen in the same manner as SuS by using the output of $g(\boldsymbol{\theta})$ only. A training set D_0 is defined with $\{\boldsymbol{\theta}_0^i\}_{i=1}^{N_n}$ and $\{\boldsymbol{e}_0^i\}_{i=1}^{N_u}$ representing the training observations. The subsequent training labels are given by $sgn(g(\boldsymbol{\theta}_0))$ and $sgn(g(\boldsymbol{e}_0))$ with the sgn determined with respect to b_0. Using D_0, the first SVM classifier $\hat{g}_0$ at $level_0$ is trained. Once trained, $\hat{g}_0$ is iteratively updated over a predefined number of iterations $k_3 = 16 + 12 \cdot (\frac{d}{2})^{0.2}$, where d denotes the dimensionality of the parameter vector $\boldsymbol{\theta}$. The expression for k_3 is defined arbitrarily with no theoretical justification given in [40].

Algorithm 15 2SMART

Define N_n, N_u and p_0

Initialize $m = 0$, where m is the current simulation level

Define the maximum number of iterations as $k_3 = 16 + 12 \cdot (\frac{d}{2})^{0.2}$, where d is the dimension of the parameter space

Initialize $b_m = \infty$

while $b_m > 0$ **do**

  Generate N_n samples $\{\boldsymbol{\theta}_m^i\}_{i=1}^{N_n}$ from the input PDF $\pi(\cdot)$

  Generate N_u samples $\{\boldsymbol{e}_m^i\}_{i=1}^{N_u}$ uniformly from a d-dimensional sphere centred at $(0, 0)$

  For each $\{\boldsymbol{\theta}_m^i\}_{i=1}^{N_n}$ and $\{\boldsymbol{e}_m^i\}_{i=1}^{N_u}$ evaluate $g(\boldsymbol{\theta}_m^i)$ and $g(\boldsymbol{e}_m^i)$

  Calculate $b_m = \frac{g(\boldsymbol{\theta}_m)^{p_0 \cdot N_n} + g(\boldsymbol{\theta}_m)^{p_0 \cdot N_n + 1}}{2}$

  $D_m = \{\{\boldsymbol{\theta}_m, \boldsymbol{e}_m\}, \{sgn(g(\boldsymbol{\theta}_m)), sgn(g(\boldsymbol{e}_m))\}\}$

  Train a SVM classifier $\hat{g}_m$ on D_m

  for $k = 1 : k_3$ **do**

    Retrain $\hat{g}_m$ over the predefined number of iterations k_3 to generate samples from the intermediate failure domain using Algorithm 16. The number of failure samples $n_F(m)$ is computed during this step.

  end for

  if $b_m > 0$ **then**

    $m = m + 1$

  end if

end while

Return $p_F \approx p_0^{m-1} \cdot n_F(m)$

The updating process of $\hat{g}_0$ is presented in Algorithm 16. Updating in this sense refers to the addition of new training points to D_0 for training a new classifier at each iteration. As iterations progress 2SMART increases the number of samples being generated (N) and the size of the training set D_0. For iteration k at $level_0$, N samples $\{\boldsymbol{\theta}^i_{0,k}\}^N_{i=1}$ are drawn from the input PDF $\pi(\cdot)$ and evaluated using the trained SVM classifier $\hat{g}_0$. Next, training points are selected from $\{\boldsymbol{\theta}^i_{0,k}\}^N_{i=1}$ to be added to D_0. These training points fall into the following three different categories:

1. **Margin Points (m_A).** The set of samples belonging in the margin (m_A).

$$m_A = \{\boldsymbol{\theta} \in \mathbb{R}^d : -1 < \hat{g}(\boldsymbol{\theta}_{m,k}) < 1\} \tag{6.4}$$

2. **Distance to Hyperplane (h_D).** The second category concerns ordering samples $\{\boldsymbol{\theta}^i_{m,k}\}^N_{i=1}$ in terms of Euclidean distance to the hyperplane.

3. **Switching Points (s_P).** The final category concerns samples $\{\boldsymbol{\theta}^i_{m,k}\}^N_{i=1}$ assigned different class labels between iterations k and $k+1$ i.e. samples assigned class 1 by $\hat{g}_{0,k}$ and class -1 by $\hat{g}_{0,k+1}$. It is argued in [40] that these samples represent areas of the sample space where the limit state function has been poorly approximated. Therefore, they are used to retrain the SVM classifier.

As outlined in step 9 of Algorithm 16, the number of training points (TP) identified for updating the training set D_0 increases with each iteration. The training points $\{\boldsymbol{\theta}^i_{0,k}\}^{TP}_{i=1}$ consist of samples belonging in the categories m_A, h_D and s_P selected from $\{\boldsymbol{\theta}^i_{0,k}\}^N_{i=1}$. Note that each of the TP training points are chosen as cluster centres from m_A, h_D and s_P by k-means clustering. Once $\{\boldsymbol{\theta}^i_{0,k}\}^{TP}_{i=1}$ has been identified, the true limit state function g is evaluated for each $\{\boldsymbol{\theta}^i_{0,k}\}^{TP}_{i=1}$ with the subsequent training set updated by $D_{0,k} = D_0 \cup \{\boldsymbol{\theta}^i_{0,k}, sgn(g^i_{0,k})\}^{TP}_{i=1}$ for training $\hat{g}_{0,k}$. At each iteration k of $level_0$, the above process is repeated with the training set $D_{0,k}$ being updated with the chosen TP training points and there subsequent limit state function evaluations. At the final iteration k_3 at $level_0$, the trained classifier $\hat{g}_{m,k_3}$ is taken as the approximation to the limit state function at $level_0$ with the sample set $\{\boldsymbol{\theta}^i_{m,k_3}\}^{N_3}_{i=1}$ the generated failure samples at $level_0$. The number of failure samples at $level_0$ is computed as $n_F(0) = \frac{\sum^{N_3}_{i=1} \hat{g}(\boldsymbol{\theta}^i_{0,k_3}) \leq 0}{N_3}$ and are returned to Algorithm 15.

Algorithm 16 Iterative Updating of the SVM Training Set

1: Input $\hat{g}_m$ and D_m

2: Define $k_1 = 6 \cdot (\frac{d}{2})^{0.2}$, $k_2 = 12 \cdot (\frac{d}{2})^{0.2}$ and $k_3 = k_2 + 16$

3: **for** $k = 1 : k_3$ **do**

4: Define the number of samples:

$$N = \begin{cases} N_1, & \text{if} \quad k < k_1 \\ N_2, & \text{if} \quad k_1 \leq k < k_2 \\ N_3, & \text{if} \quad k_2 \leq k \leq k_3 \end{cases}$$

In [40] $N_1 = 10^4$, $N_2 = 5 \cdot 10^4$ and $N_3 = 2 \cdot 10^5$

5: Generate N samples $\{\boldsymbol{\theta}^i_{m,k}\}_{i=1}^N$ from the input PDF $\pi(\cdot)$

6: For each $\{\boldsymbol{\theta}^i_{m,k}\}_{i=1}^N$ evaluate $\hat{g}^i_{m,k}$

7: Identify samples from $\{\boldsymbol{\theta}^i_{m,k}\}_{i=1}^N \in m_A$, h_D and s_P

8: Define the number of training points to be selected:

$$TP = \begin{cases} 3 + \frac{k-1}{k_1-1} \cdot \sqrt{d}, & \text{if} \quad k < k_1 \\ 4 + \frac{k-1}{k_2-1} \cdot \sqrt{d}, & \text{if} \quad k_1 \leq k < k_2 \\ 5 \cdot \sqrt{d}, & \text{if} \quad k_2 \leq k \leq k_3 \end{cases}$$

9: From each sample categories m_A, h_D and s_P identified in step 8 select TP training points $\{\boldsymbol{\theta}^i_{m,k}\}_{i=1}^{TP}$ as cluster centres by k-means clustering

10: Evaluate the true limit state function $g^i_{m,k}$ for all $\{\boldsymbol{\theta}^i_{m,k}\}_{i=1}^{TP}$

11: Update the training set: $D_{m,k} = D_m \cup \{\boldsymbol{\theta}^i_{m,k}, sgn(g^i_{m,k})\}_{i=1}^{TP}$

12: Train a SVM classifier $\hat{g}_{m,k}$ on $D_{m,k}$

13: **end for**

14: Store $p_0 \cdot N_3$ samples as seeds for $level_{m+1}$

15: $n_F(m) = \frac{\sum_{i=1}^{N_3} \hat{g}(\boldsymbol{\theta}^i_{m,k_3}) \leq 0}{N_3}$

16: Return $\hat{g}_{m,k_3}$, $n_F(m)$ & $\{\boldsymbol{\theta}^i_{m,k_3}\}_{i=1}^{p_0 \cdot N_3}$

For all $level_{1,\ldots,m}$, simulation proceeds in the same manner as above with the only difference the use of an adjusted Modified Metropolis Hastings (AMMH) sampler to generate samples in step 6 of Algorithm 16. At the final iteration k_3 at $level_m$, $p_0 \cdot N_3$ samples are stored as seeds for the AMMH sampler to generate samples at $level_{m+1}$. The AMMH sampler is given in Algorithm 17. AMMH uses a scaling factor λ defined by the practitioner, to increase the probability of the pre-candidate sample being accepted. The authors in [40] argue that decreasing λ for an increasing k avoids an extremely high repeated pre-candidate generation

rate. While the trained SVM classifier is used to determine whether or not the proposed samples belongs in the subsequent failure domain.

Algorithm 17 Adjusted Modified Metropolis-Hastings

Input $\hat{g}_{m,k}$ and b_m
Define the current state of the Markov chain $\boldsymbol{\theta} = \{\theta_1, ..., \theta_d\}$
Define the univariate proposal PDFs $S_j(\cdot|\theta_j)$ for $j = 1, ..., d$
Select a scaling factor such that:

$$\lambda = \begin{cases} 7, & \text{if} \quad k < k_1 \\ 3.5, & \text{if} \quad k_1 \leq k < k_2 \\ 1, & \text{if} \quad k_2 \leq k \leq k_3 \end{cases}$$

for each $j = 1, ..., d$ **do**
    Generate a pre-candidate w_j from $S_j(\cdot|\theta_j)$
    Generate u_j uniformly on $[0, 1]$
    $r(w_j) = \frac{\pi(w_j)}{\pi(\theta_j)} \cdot \frac{S_j(\theta_j|w_j)}{S_j(w_j|\theta_j)}$
    Accept $v_j = w_j$ if $u_j < \lambda r(w_j)$, otherwise $v_j = \theta_j$
end for
$\boldsymbol{v} = \{v_1, ..., v_d\}$
if $\hat{g}_{m,k}(\boldsymbol{v}) \leq b_m$ **then**
    Accept $\boldsymbol{\theta'} = \boldsymbol{v}$
else
    Accept $\boldsymbol{\theta'} = \boldsymbol{\theta}$
end if

In line with standard BUS, simulation in 2SMART progresses as in Algorithm 15 and 16 before terminating at $level_m$ once $b_m < 0$. Overall 2SMART relies on the training of a large number of SVM classifiers at each intermediate level to replace the computationally expense limit state function evaluations.

6.2.2 Observations and Potential Improvements

The computational savings of 2SMART over SuS stem from the limit state function being evaluated only on the selected training points of each iteration at every level. To help aid performance for the use of 2SMART in the context of BUS, some modifications are proposed.

Observation 1: Sample Size and Selection of Training Points

At the final iterations $k_2 \leq k \leq k_3$, the suggested value for N given in [40] is $2 \cdot 10^5$. Large values for N are justified to ensure each of the three category points m_A, h_D and s_P are readily available in the sample space. However, this poses problems for the AMMH step implemented from $level_1$. The first problem is computational expense in terms of pre-candidate sample generation for each sample of each chain. To alleviate the potential computational cost of generating a large number of samples, each chain could be sent to an individual computing cluster for sample generation. This could allow for a decrease in the wall clock time of simulation.

The second problem concerns sample correlation. Consider the case of $N = 2 \cdot 10^5$, where a selection of $p_0 \cdot N$ seeds is required. Intuitively, having a large number of chains may help distribute the samples more evenly in the failure domain. In reality, the correlation between chains along with the correlation between samples at successive levels will increase. Essentially, the large choice of sample size acts as a type of space filling strategy. The obvious remedy would be to reduce the number of samples. However, with this comes the possibility of not all category of training points existing in the sample space. With respect to those switching classes between iterations, a decrease in sample size reduces the likelihood of these unstable data points existing. In this case the 2SMART framework will not work. As such for smaller sample sizes, it is proposed to remove this category of training point and replace them by additional points which lie in the margin set.

$$m_A = \{\boldsymbol{\theta} \in \mathbb{R}^d : -1 < \hat{g}(\boldsymbol{\theta}) < 1\} \tag{6.5}$$

Observation 2: Hyperparameter Selection

As discussed in Chapter 3, the SVM class decision rule is given by the following. For the i^{th} instance let x_i denote the data observation, y_i the corresponding class label, $\boldsymbol{\theta}$ the unknown parameter, a the bias term and $\langle \cdot, \cdot \rangle$ the dot product operation.

$$y_i(\langle \boldsymbol{\theta}, x_i \rangle - a) \geq 1 \tag{6.6}$$

To allow for potentially misclassified points, a penalty term named a *slack* variable $\xi_i \geq 0$ where $i = 1, ..., n$ is considered [58]. The slack variable represents the

distance of the misclassified observation to the hyperplane. The further the misclassified data observation is from the hyperplane, the greater the penalty. These are defined as $\xi_i = 0$ for data observations on or inside the correct margin boundary and $\xi_i = |y_i - \langle \boldsymbol{\theta}, x_i \rangle - a|$ for other observations. Thus, an observation located on the separating hyperplane will have $\xi_i = 1$ and points with $\xi_i > 1$ will be misclassified. The reformulated optimization problem is

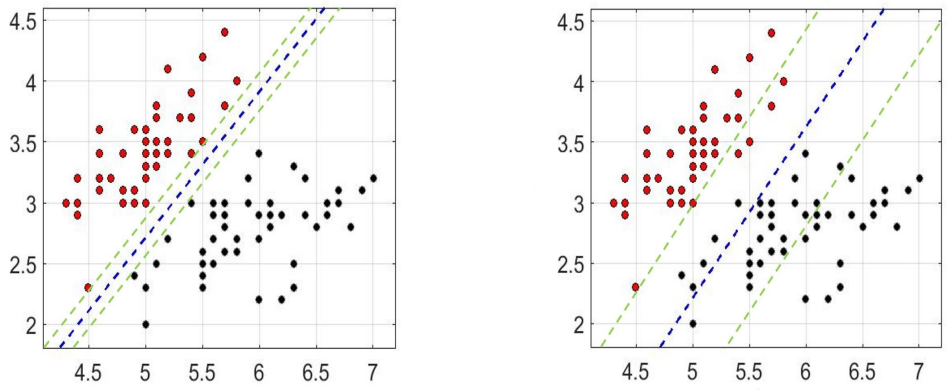
$$\min_{\boldsymbol{\theta},b,\xi} \quad W \sum_{i=1}^{n} \xi_i + \frac{||\boldsymbol{\theta}||^2}{2} \quad \text{s.t} \quad y_i(\langle \boldsymbol{\theta}, x_i \rangle - a) \geq 1 - \xi_i \quad \text{for} \quad i = 1, ..., n \quad (6.7)$$

with the parameter $W > 0$ referred to as the regularization parameter. As the slack variable is sensitive to outliers, W controls the trade-off between minimizing training errors and controlling model complexity. From Eq. 6.7, values of W will influence the margin width in different ways. A small W decreases the influence of the constraints which leads to a large margin. Whereas, a larger value for W increases the influence of the constraints and results in a small margin. Assigning the regularization parameter W a default value of infinity ensures all constraints are strictly enforced.

To investigate the importance of choosing parameters appropriately, consider a two dimensional example using the iris flower data set [69] as presented in Figure 6.1. Selecting the length and width of the flower petals as features, the binary classification task is to distinguish between an iris sertosa (red) and iris veriscolor (black). The chosen kernel is the RBF kernel given by

$$k(x_1, x_2) = \exp\left(-\frac{1}{2\sigma^2} ||x_1 - x_2||^2 \right) \quad (6.8)$$

Figure 6.1 highlights the influence of W on the margin width.

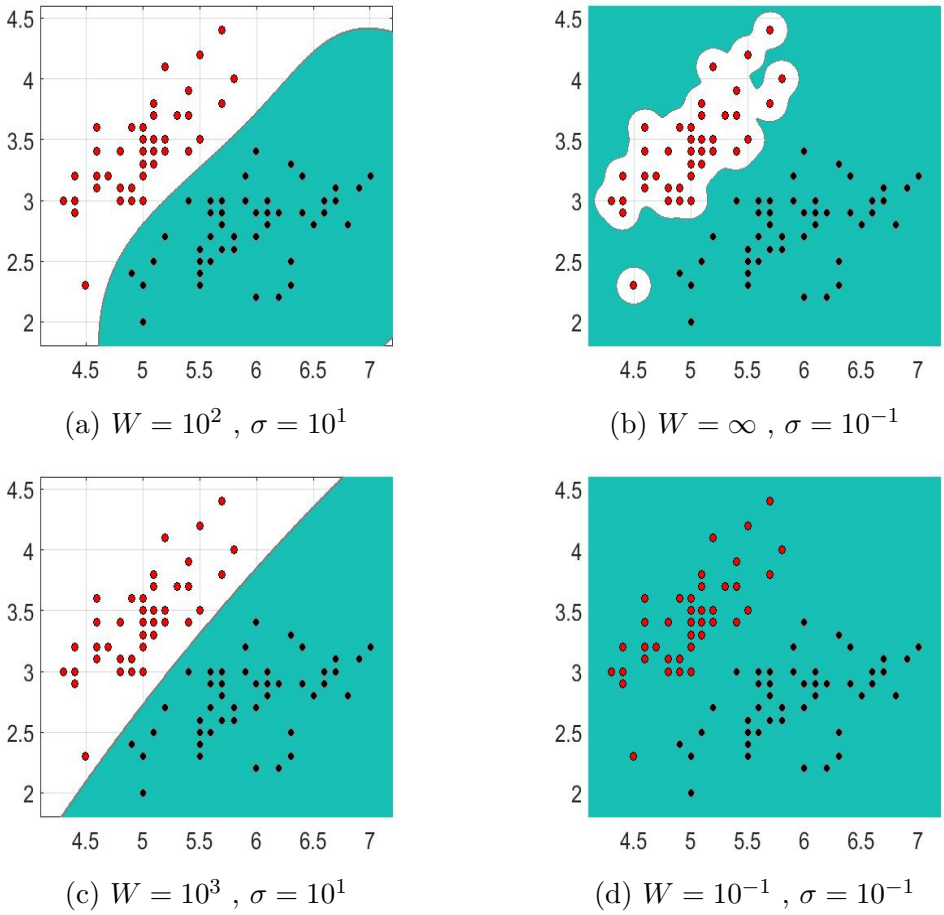


(a) $W = \infty$ (b) $W = 10^{-2}$

Figure 6.1: Influence of W on the margin width.

As outlined in Chapter 2, SVM offers the advantage of flexibility through the use of kernel functions $k(\cdot, \cdot)$. Non-linear kernel mappings enable the SVM to approximate different limit state function behaviours accurately. Such functions require the estimation of hyperparameters from the available data. One strategy for performing hyperparameter selection as adopted in [40], is by an exhaustive *grid search*. The practitioner manually specifies an interval of possible hyperparameter values with the performance of the interval values recorded by some performance criterion. In machine learning, typically this is cross validation.

Figure 6.2 reveals the influence of W and the hyperparameter σ on the SVM decision boundary. The shape of the decision boundary (green) of the target class veriscolor (black) varies greatly. An appropriate class separation is achieved in Figure 6.2 (a), while the parameter settings in Figure 6.2 (d) have resulted in the classifier incorrectly allocating veriscolor class membership to the entire sample space. In the case of $W = \infty$ in Figure 6.2 (b), the decision rule has been learned to an extent that it lacks generalization properties. Consequently, for this example such a value for W results in the model being susceptible to over-fitting.



(a) $W = 10^2$, $\sigma = 10^1$ (b) $W = \infty$, $\sigma = 10^{-1}$

(c) $W = 10^3$, $\sigma = 10^1$ (d) $W = 10^{-1}$, $\sigma = 10^{-1}$

Figure 6.2: Influence of W and kernel hyperparameters on the SVM decision boundary.

Figure 6.3 highlights the influence of W and the RBF kernel parameter σ on the SVM accuracy for the iris example. This example emphasises the importance of selecting problem dependent parameter values for a SVM classifier. Currently in 2SMART a default value of $W = \infty$ and choosing σ via a grid search method present the potential issues of over-fitting the margin width and assigning inappropriate decision boundaries. This directly influences the quality and the accuracy of the surrogates constructed at each iteration for every intermediate level. To achieve a balance in the choice of W and hyperparameters, we propose the implementation of a Bayesian optimization scheme [206] for determining appropriate parameter values in 2SMART.

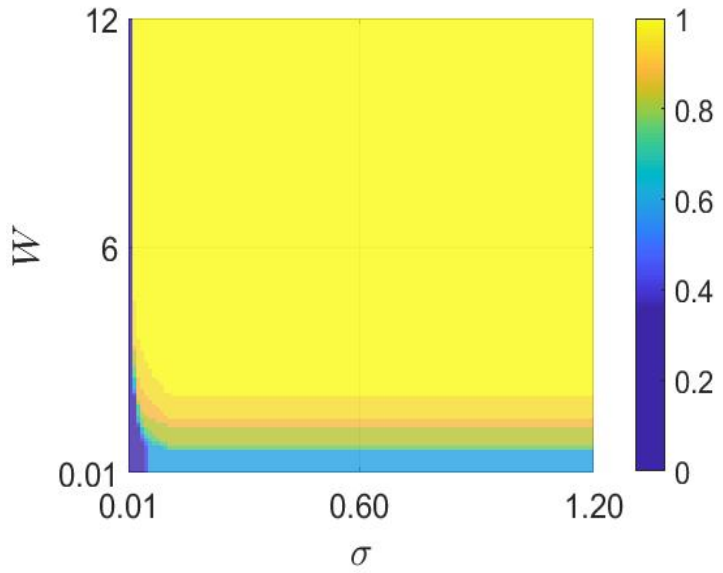


Figure 6.3: Influence of the regularization parameter W and kernel parameter σ on SVM accuracy.

Observation 3: MCMC Sampling

The final observation regards the choice of MCMC scheme. In Algorithm 17, λ is defined by the practitioner. The purpose of this is to increase the sample acceptance ratio $r(w_j)$ for a pre-candidate sample w_j. This scaling factor directly influences the entire sampling process. As the sensitivity of the MCMC scheme to λ is unknown, we propose the use of the adaptive Conditional Sampling (ACS) [172] method. As discussed in Chapter 5, ACS avoids the pre-candidate step by adaptively scaling the standard deviation of the sample proposal distribution.

6.2.3 Remaining Issues

Having identified improvements for the 2SMART method to be applicable to Bayesian updating problems, shortcomings of the framework are still present. Firstly, surrogate modelling is predominately required to form a computationally cheap approximation to an otherwise computationally expensive function. By introducing SVM, 2SMART seeks to decrease the number of TME in a model run by forming an approximation to the limit state function. The assumption is often made however, that training many surrogate models is preferable over the direct computation of the initial expensive function. 2SMART requires the training of

$m \cdot k_3$ SVM models, where m is the number of required intermediate levels. While this strategy results in a decreased number of TME, a subsequent negative effect on simulation running time is potentially apparent. Additionally, no theoretical justification is given for the choice of k_3, the number of training points (TP) or the sample sizes.

Secondly, by a combination of a training point selection strategy involving clustering methods along with a specified number of iterations in terms of the dimensionality of the data, the accuracy of the surrogate is assumed to be at a satisfactory level once simulation terminates according to the choice of intermediate threshold. Without any modelling error control mechanism, errors in the surrogate model will be propagated throughout the entire sampling process. Given that the primary interest is in the samples conditional on failure as opposed to the failure probability, sample quality is of paramount importance in this work.

Finally, in line with the outlined potential for deterioration in sample quality, stopping conditions which guarantee convergence to the correct posterior distribution (Chapter 5) are required.

6.3 BUS with Support Vector Machines

This section presents an alternative SVM-inspired SUS framework aimed at adopting some of the modifications suggested in Section 6.2.2 while also addressing the issues outlined in Section 6.2.3. The method is a combination between SVM and BUS, termed BUS_{SVM} and is developed for solving Bayesian updating problems on large data sets. BUS_{SVM} initializes simulation in the same manner as $nBUS$ at $level_0$ through the generation of N i.i.d samples from the input PDF $\pi(\cdot)$ by Direct Monte Carlo (DMC) using the limit state function

$$g(\boldsymbol{\theta}, u) = \ln L(D|\boldsymbol{\theta}) - \ln(u) \tag{6.9}$$

As with BUS_+ and $nBUS_D$, $nc = p_0 \cdot N$ seeds are identified from $\{\boldsymbol{\theta}_0^i, u_0^i\}_{i=1}^N$ for sample generation at $level_1$ by choice of level probability p_0. Under this framework, a total of N TME are required at $level_0$. For $level_{1,\ldots,m}$, with m being the final level, a single SVM surrogate is constructed at each level. The SVM output $\hat{g}(\boldsymbol{\theta}, u)$ is used as an approximation for the true $\ln L(D|\boldsymbol{\theta}) - \ln(u)$ for identifying whether or not a proposed sample belongs in the failure domain. In line with BUS_+

and $nBUS_D$ the intermediate failure thresholds b_m are chosen as an increasing sequence resulting in the intermediate failure event being

$$F_m = \{\hat{g}(\boldsymbol{\theta}_m, u_m) > b_m\} \tag{6.10}$$

By the definition of Eq. 6.10 and through placing $g(\boldsymbol{\theta}_{m-1}, u_{m-1})$ in descending order, the resulting intermediate failure thresholds computed at each intermediate level form an increasing sequence. To reduce the total TME, a user defined parameter q is introduced during the modelling process, where $q \in [0, 1]$. The purpose of this parameter is to determine the number of training points to be generated by the chosen MCMC scheme using $\{\boldsymbol{\theta}_0^i, u_0^i\}_{i=1}^{nc}$ as seeds. At $level_1$ the training points and their corresponding class labels for the training set D_1 are given by

$$D_1 = \{\{\boldsymbol{\theta}_1^i, u_1^i\}, sgn(g(\boldsymbol{\theta}_1^i, u_1^i))\}_{i=1}^{q \cdot N} \tag{6.11}$$

Figure 6.4 illustrates the training process on the benchmark example 4 problem from Chapter 5. The red points are samples which constitute the safe domain and the black the failure. The subsequent SVM decision boundary learned is represented by the green surface. Common practice in surrogate modelling for reliability analysis is the use of clustering methods to identify disjoint failure domains or in this case multi-modal distributions. Clusters are identified in the training points before surrogates are locally trained on each cluster [90, 59]. In contrast, BUS_{SVM} extracts information contained within the training points to identify different modes. An advantage of treating the reliability problem as a classification task, is the ability to identify strict boundaries between domains and in turn class labels. By initially populating the sample space using MCMC, a single SVM is trained to identify different class regions as highlighted in Figure 6.4.

Having discussed the importance of hyperparameter selection in Section 6.2.2, hyperparameters of the chosen kernel (if any) and the penalty term W, are tuned using a Bayesian optimization scheme termed *expected improvement* (EI) [206, 154]. For large data sets, the direct computation of the objective function can be highly computationally intensive. Bayesian optimization techniques deploy surrogate models aimed at finding a global optimum of the objective function in a minimum number of steps. In this case, a Gaussian process [183] is used as an approximation

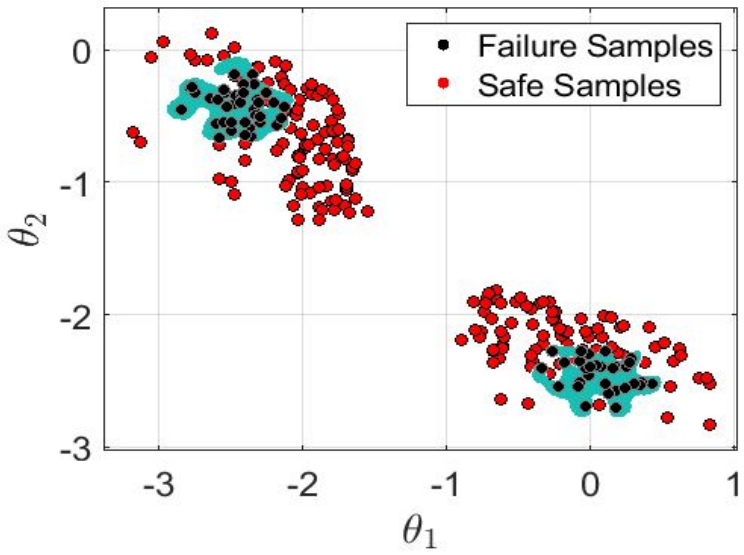


Figure 6.4: SVM training process using $q = 0.5$ and $N = 1000$ for benchmark example 4. Samples in the safe domain are represented in red and samples in the failure domain in black. The decision boundary formed by the SVM is given by the green contours. The RBF kernel is used for this example.

for the specified objective function. EI directly samples from the Gaussian process function in areas where an improvement over the current best observation is likely. EI balances exploitation of the Gaussian process predictions and exploration at the locations where the predictions uncertainty is high. The classification error rate is chosen as the objective function to be minimized. The correct tuning of W allows for regularization, in turn preventing the SVM from over-fitting. The convex optimization problem (Chapter 2) is solved using Sequential Minimal Optimization (SMO) [175].

Once training is complete, the samples in D_1 are used as seeds for the generation of the remaining $\{\boldsymbol{\theta}_1^j, u_1^j\}_{j=1}^{N-q\cdot N}$ samples at $level_1$. Whether or not a generated MCMC sample belongs in F_1 is determined by

$$sgn(\hat{g}(\boldsymbol{\theta}_1, u_1)) = \begin{cases} -1, & \text{if} \quad \hat{g}(\boldsymbol{\theta}_1, u_1) > b_1 \\ 1, & \text{Otherwise} \end{cases} \tag{6.12}$$

This entire process is repeated for $level_{2,\ldots,m}$. In terms of stopping simulation, BUS_{SVM} proposes the use of the stopping criteria presented in Chapter 5. The total number of TME will be $N + m \cdot q \cdot N$ with m being the required number of levels.

6.3.1 Controlling the Error of SVM

Training a SVM on points from the true intermediate conditional distributions is not enough to confidently ensure Eq. 6.12 makes the correct decision. A series of precautionary steps are taken to control the level of error in the decision making process.

1. **SVM Output.** For a proposed sample $\{\boldsymbol{\theta}', u'\}$ to be accepted, the value of $\hat{g}(\boldsymbol{\theta}', u')$ is checked to verify whether its predicted value is within the 95% quantile of $D = \{\{\boldsymbol{\theta}^i, u^i\}, sgn(g(\boldsymbol{\theta}^i, u^i))\}_{i=1}^{q \cdot N}$ denoted $D_{0.95}$. This rule aims to control the error in the sampling procedure and the subsequent sample acceptance rate of the MCMC mechanism [7, 139]. Consider the Metropolis step for SuS as discussed in Chapter 4. Let $S(\cdot)$ denote the proposal distribution, $\{\boldsymbol{\theta}, u\}$ the current state of the Markov chain, $\{\boldsymbol{\theta}', u'\}$ the proposed sample and $\pi(\cdot)$ the target density. The acceptance ratio for $\{\boldsymbol{\theta}', u'\}$ is given by

$$r(\boldsymbol{\theta}', u') = \min\left\{1, \frac{\pi(\boldsymbol{\theta}', u')}{\pi(\boldsymbol{\theta}, u)} \cdot \frac{S(\boldsymbol{\theta}, u|\boldsymbol{\theta}', u')}{S(\boldsymbol{\theta}', u'|\boldsymbol{\theta}, u)}\right\} \cdot I_F(\boldsymbol{\theta}', u') \tag{6.13}$$

BUS_{SVM} approximates the limit state function such that the sample acceptance ratio becomes

$$r(\boldsymbol{\theta}', u') = \min\left\{1, \frac{\pi(\boldsymbol{\theta}', u')}{\pi(\boldsymbol{\theta}, u)} \cdot \frac{S(\boldsymbol{\theta}, u|\boldsymbol{\theta}', u')}{S(\boldsymbol{\theta}', u'|\boldsymbol{\theta}, u)}\right\} \cdot \hat{I}_F(\boldsymbol{\theta}', u') \tag{6.14}$$

where $\hat{I}_F(\boldsymbol{\theta}', u')$ represents the process of checking whether the proposed $\{\boldsymbol{\theta}', u'\}$ is accepted with respect to the SVM approximation. The indicator function may be interpreted as a binary variable, equal to 1 if $\{\boldsymbol{\theta}', u'\} \in F$ and 0 otherwise. By adopting the initial Metropolis ratio

$$\min\left\{1, \frac{\pi(\boldsymbol{\theta}', u')}{\pi(\boldsymbol{\theta}, u)} \cdot \frac{S(\boldsymbol{\theta}, u|\boldsymbol{\theta}', u')}{S(\boldsymbol{\theta}', u'|\boldsymbol{\theta}, u)}\right\} \tag{6.15}$$

and only targeting $I_F(\boldsymbol{\theta}', u')$ this represents a valid Metropolis step. Checking whether $\hat{g}(\boldsymbol{\theta}', u') \in D_{0.95}$ allows for the potential sampling error to be controlled. Additionally, accurately identifying whether $\{\boldsymbol{\theta}', u'\} \in F$ at

intermediate levels of SuS helps maintain the samplers property that no burn in period is required to initialize the Markov chains at each level.

2. **Markov Chain Seeds.** At $level_m$, the nc samples to be chosen as seeds from $\{\boldsymbol{\theta}^j_{m-1}, u^j_{m-1}\}^N_{j=1}$ are identified. To ensure that errors are not propagated between levels by means of incorrect initiation of the Markov chains, the seeds are re-evaluated by the true limit state function g. This allows for an accurate intermediate threshold to be identified.

$$b_m = \frac{g(\boldsymbol{\theta}^{nc}_{m-1}, u^{nc}_{m-1}) + g(\boldsymbol{\theta}^{nc+1}_{m-1}, u^{nc+1}_{m-1})}{2} \tag{6.16}$$

Recomputing the true limit state function for the nc sample seeds allows for the final samples being conditional on the true failure event. Additionally the model evidence estimate is dependent on b_m.

$$P_D \approx \hat{P}_D = p_0^m e^{b_m} \qquad\qquad b_m > b_{\min} \tag{6.17}$$

At $level_m$ the maximum added computational expense for re-evaluating the true limit state function is nc additional TME. However, apriori it is not known whether all nc sample seeds will require to be re-evaluated by the true limit state function g. Seeing as the limit state function evaluations are placed in descending order for computing the intermediate failure threshold, a proportion of the nc evaluations may have already been evaluated by g at the second step of the MCMC process in BUS when generating the training set D_m. Therefore only the samples in the set of seeds accepted using the SVM approximation $\hat{g}$ are required to be re-evaluated by g. To quantify the additional computational cost allow

$$r_m = \sum_{j=1}^{nc} I(\boldsymbol{\theta}^j_{m-1}, u^j_{m-1}) \tag{6.18}$$

to represent the number of seeds which were evaluated with the SVM approximation $\hat{g}$ at $level_m$ i.e. the samples for which g is required to be re-evaluated. Therefore for m intermediate levels let the total computational cost of BUS_{SVM} be given by

$$N + m \cdot q \cdot N + R \tag{6.19}$$

where $R = \sum_{i=1}^{m} r_i$. At any given level there is the possibility of a maximum of nc additional TME stemming from the step of re-evaluating the Markov chains seeds. The variable R controls the number number of additional TME in the final computational cost of BUS_{SVM}. In the unlikely event that $R = 0$ no additional TME are required.

Unlike probabilistic models such as logistic regression, SVM is deterministic in its class prediction. A post processing extension to SVM has been proposed [176], which allows for posterior probabilities to be attached to the classifiers output. By mapping the model output to the unit interval by means of the sigmoid function and a prudent choice of parameters, class conditional probabilities are derived. It has been noted however [216], that the implementation of such a post processing step is an unreliable representation of the desired posterior probability and as such is not introduced to this model. This is another reason for the implementation of the above rules aimed at controlling errors.

6.3.2 Selection of q

This section discusses the constraints on the choice of q to ensure BUS_{SVM} is computationally cheaper than its BUS competitors. Recall that the total TME for a single run of BUS_+ is $N + m \cdot (N - nc)$. Note that the total TME for $nBUS_D$ will vary as it is dependent on whether or not the nested loop is required to be computed at additional levels beyond $level_m$. To investigate the influence of q on the computational savings of BUS_{SVM} over BUS_+ consider Proposition 2.

Proposition 2. *Let p_0 denote the level probability and q a scaling parameter introduced to identify the number of training points for BUS_{SVM}. Let m denote the total number of intermediate levels, N the number of generated samples and R the number of additional total model evaluations for all levels. For BUS_{SVM} to require fewer total model evaluations than BUS_+, q must be chosen such that*

$$1 - q > p_0 - \frac{R}{N \cdot m} \tag{6.20}$$

holds.

Assuming that both BUS_{SVM} and BUS_+ terminate at the same $level_m$, proving Proposition 2 allows for constraints on the selection of q to be established.

This in turn ensures that the total number of TME for a single run of BUS_{SVM}

$$N + m \cdot q \cdot N + R \tag{6.21}$$

is less than the total number of TME for a single run of BUS_{+}

$$N + m \cdot (N - nc) \tag{6.22}$$

In terms of notation let N represent the number of samples, m the number of intermediate levels, p_0 the level probability and q a scaling parameter introduced to identify the number of training points for BUS_{SVM}. Also allow for $nc = p_0 \cdot N$.

Proof.

$$\begin{aligned} & N + m \cdot (N - p_0 \cdot N) > N + m \cdot q \cdot N + R \\ \implies & m \cdot (N - p_0 \cdot N) > m \cdot q \cdot N + R \\ \implies & m \cdot N - p_0 \cdot N \cdot m > m \cdot q \cdot N + R \\ \implies & 1 - q > p_0 + \frac{R}{N \cdot m} \end{aligned} \tag{6.23}$$

□

Therefore to ensure that BUS_{SVM} is computationally cheaper than BUS_{+}, q needs to be chosen according to Proposition 2. The values of R and m are unknown when selecting q prior to simulation. For $m = 1$, and considering the maximum value of $R = (m - 1) \cdot N \cdot p_0$ Eq. 6.23 becomes

$$1 - q > p_0 \tag{6.24}$$

For $m > 1$ Eq. 6.23 becomes

$$\begin{aligned} & 1 - q > p_0 + \frac{(m-1) \cdot N \cdot p_0}{N \cdot m} \\ \implies & 1 - q > p_0 + \frac{m \cdot N \cdot p_0 - N \cdot p_0}{N \cdot m} \\ \implies & 1 - q > p_0 + \frac{m \cdot p_0 - p_0}{m} \\ \implies & 1 - q > p_0 + \frac{m \cdot p_0}{m} - \frac{p_0}{m} \\ \implies & 1 - q > 2 \cdot p_0 - \frac{p_0}{m} \end{aligned} \tag{6.25}$$

For an increase in m the right hand side of Eq. 6.23 becomes larger and larger, resulting in a smaller q being able to be selected. As m is unknown prior to simulation, for the work in this thesis q will be selected such that

$$1 - q > 2 \cdot p_0 \tag{6.26}$$

holds. This is chosen under the assumption that R takes its maximum value during simulation. Assuming that q is chosen according to Proposition 2, the reduction of TME by using BUS_{SVM} over BUS_+ is

$$\begin{aligned} & N + m \cdot (N - p_0 \cdot N) - (N + m \cdot q \cdot N + R) \\ \implies & N + m \cdot (N - p_0 \cdot N) - N - m \cdot q \cdot N - R \\ \implies & m \cdot (N - p_0 \cdot N - q \cdot N) - R \end{aligned} \tag{6.27}$$

A pseudo-code for BUS_{SVM} using the stopping condition from BUS_+ is shown in Algorithm 18. Firstly q and p_0 are selected such that Proposition 2 holds. $Level_0$ progresses in the same manner as BUS_+. At $level_1$, the sequence $\{b_1^j\}_{j=1}^{nc}$ is computed with the value b_1^{nc} being equivalent to the original intermediate threshold specified in $nBUS$. Next, as nc seeds are already in F_1, $q \cdot N - nc$ samples are drawn by the chosen MCMC sampler in step 16. Once $\{\boldsymbol{\theta}_1^i, u_1^i\}_{i=1}^{q \cdot N}$ have been drawn, the training set D_1 is specified with $\{\boldsymbol{\theta}_1^i, u_1^i\}_{i=1}^{q \cdot N}$ denoting the training observations and the subsequent $\{sgn(g_1^i)\}_{i=1}^{q \cdot N}$ the class labels. Note that $\{g_1^i\}_{i=1}^{q \cdot N}$ is computed as part of the MCMC sampling process in step 16. Using D_1, a SVM

classifier $\hat{g}_1$ is trained as an approximation to the underlying limit state function. As the training observations $\{\boldsymbol{\theta}_1^i, u_1^i\}_{i=1}^{q \cdot N}$ already belong in F_1, they are used as seeds to generate the remaining $\{\boldsymbol{\theta}_1^i, u_1^i\}_{i=1}^{N-q \cdot N}$ at $level_1$ in step 22 via Algorithm 19. In this sampler $\hat{g}_1$ is used to identify whether or not proposed samples belong in F_1. As in BUS_+, once all N samples at $level_1$ have been generated, the post processor quantities are next computed and it is checked whether $\delta_1 > t$. If so, $\{\boldsymbol{\theta}_1^i\}_{i=1}^N$ are returned as the posterior samples. If not, r_m is computed and the resulting samples are re-evaluated using the true limit state function g. Next, BUS_{SVM} progresses to $level_2$ and continues simulation as outlined above until for any $level_m$ $\delta_m > t$ is true.

Algorithm 18 BUS_{SVM}

Define N, p_0, $nc = p_0 N$, $ns = p_0^{-1}$ and q
Initialize $m = 0$, where m is the current simulation level
Initialize $\delta_m = 0$
Define the tolerance t
Generate N samples $\{\boldsymbol{\theta}_m^i, u_m^i\}_{i=1}^N$ from the input PDF $\pi(\cdot)$
For each $\{\boldsymbol{\theta}_m^i, u_m^i\}_{i=1}^N$ evaluate $g_m^i = \ln L(D|\boldsymbol{\theta}_m^i) - \ln(u_m^i)$
Sort the N values of g_m^i and the corresponding $\{\boldsymbol{\theta}_m^i, u_m^i\}_{i=1}^N$ in descending order with respect to g_m^i
while $\delta_m < t$ **do**
  $m = m + 1$
  for $j = 1, ..., nc$ **do**
    Calculate: $b_m^j = \frac{g_{m-1}^j + g_{m-1}^{j+1}}{2}$
  end for
  Store the first nc samples of the ordered set $\{\boldsymbol{\theta}_{m-1}^i, u_{m-1}^i\}_{i=1}^N$ as 'seeds'
  Using $\{\boldsymbol{\theta}_{m-1}^j, u_{m-1}^j\}_{j=1}^{nc}$ draw $q \cdot N - nc$ samples from $\pi(\cdot|F_m)$ via MCMC as follows:
  for $j = 1, ..., nc$ **do**
    Starting with $\{\boldsymbol{\theta}_m^j, u_m^j\}$ as an initial seed, generate $\{\boldsymbol{\theta}_m^k, u_m^k\}_{k=1}^{q \cdot ns} \sim \pi(\cdot|F_m)$ states of a Markov chain using the chosen MCMC sampler from those discussed in Chapter 5. In this algorithm, the sample acceptance criteria involves evaluating g_m with respect to b_m^{nc} for each generated sample.
  end for
  $D_m = \{\{\boldsymbol{\theta}_m^i, u_m^i\}, sgn(g_m^i)\}_{i=1}^{q \cdot N}$
  Train a SVM $\hat{g}_m$ using D_m
  Using $\{\boldsymbol{\theta}_m^j, u_m^j\}_{j=1}^{q \cdot N}$ as seeds draw the remaining $N - q \cdot N$ samples from $\pi(\cdot|F_m)$ via MCMC as follows:
  for $j = 1, ..., nc$ **do**
    Starting with $\{\boldsymbol{\theta}_{m-1}^j, u_{m-1}^j\}$ as an initial seed, generate $\{\boldsymbol{\theta}_m^k, u_m^k\}_{k=1}^{ns - q \cdot ns} \sim \pi(\cdot|F_m)$ states of a Markov chain using Algorithm 19. In this algorithm, the sample acceptance criteria involves evaluating $\hat{g}_m$ with respect to b_m^{nc} for each generated sample.
  end for
  Sort $\{g_m^i\}_{i=1}^{q \cdot N} \cup \{\hat{g}_m^i\}_{i=1}^{N - q \cdot N}$ and the corresponding $\{\boldsymbol{\theta}_m^i, u_m^i\}_{i=1}^N$ in descending order with respect to $\{g_m^i\}_{i=1}^{q \cdot N} \cup \{\hat{g}_m^i\}_{i=1}^{N - q \cdot N}$
  For each b_m^j evaluate $\ln(\hat{P}_D)_m^j = b_m^j + m \cdot \ln(p_0)$
  Compute $\alpha = \frac{(\frac{p_0 \cdot N+1}{N+2})^m \cdot (1 - (\frac{p_0 \cdot N+2}{N+3})^m)}{(\frac{p_0 \cdot N+2}{N+3})^m - (\frac{p_0 \cdot N+1}{N+2})^m}$ and $\beta = \frac{(1 - (\frac{p_0 \cdot N+1}{N+2})^m) \cdot (1 - (\frac{p_0 \cdot N+2}{N+3})^m)}{(\frac{p_0 \cdot N+2}{N+3})^m - (\frac{p_0 \cdot N+1}{N+2})^m}$
  Compute $\ln(\hat{P}_{MAP}^+) = \ln(\frac{\alpha}{(\alpha + \beta - 1)})$
  Compute the credible interval: $0.99 = p(a \leq \ln(\hat{P}_{MAP}^+) \leq h)$ with $a \leq h$. Where $[a, h]$ is referred to as the range of the 0.99 credible interval.
  Compute $\delta_m = \frac{\sum_{j=1}^{nc} (a \leq \ln(\hat{P}_D)_m^j \leq h)}{nc}$
  if $\delta_m < t$ **then**
    $r_m = \sum_{j=1}^{nc} I(\boldsymbol{\theta}_m^j, u_m^j)$
    Re-evaluate g_m^i for all r_m
  else
    $r_m = 0$
  end if
end while
Return $\boldsymbol{\theta} \sim P(\boldsymbol{\theta}|D)$ & $P_D \approx e^{b_m^{nc}} p_0^m$

Algorithm 19 presents a pseudo-code example for MMH under the BUS_{SVM} framework. Note that as before, the space of the parameter vector $\boldsymbol{\theta}$ in BUS is augmented by a uniform random variable such that the parameter vector $\boldsymbol{\theta} \in \mathbb{R}^{d+1}$. The sampler progresses in the same manner as MMH before differing in the process of checking whether the proposed sample $\boldsymbol{v}$ belongs in the failure domain. At step 11, it is checked whether $\hat{g}(\boldsymbol{v}) > b_m^{nc}$. If this is the case and $\hat{g}(\boldsymbol{v}) \in D_{0.95}$ the next state of the Markov chain is $\boldsymbol{\theta'} = \boldsymbol{v}$. If this is not the case the next state of the Markov chain equals the current state $\boldsymbol{\theta'} = \boldsymbol{\theta}$. These steps capture the computational savings in BUS_{SVM} by using the approximated $\hat{g}$ over g. Note that any of the SuS tailored samplers discussed in Chapter 5 can be easily modified in the same manner. MMH was chosen to illustrate the differences with adjusted MMH (AMMH) in 2SMART.

Algorithm 19 Modified Metropolis-Hastings for BUS_{SVM}

1: Input D_m, $\hat{g}_m$, g_m and b_m^{nc}
2: Define the current state of the Markov chain $\boldsymbol{\theta} = \{\theta_1, ..., \theta_{d+1}\}$
3: Define the univariate proposal PDFs $S_j(\cdot|\theta_j)$ for $j = 1, ..., d+1$
4: **for** each $j = 1, ..., d+1$ **do**
5: Generate a pre-candidate w_j from $S_j(\cdot|\theta_j)$
6: Generate u_j uniformly on $[0, 1]$
7: $r(w_j) = \frac{\pi(w_j)}{\pi(\theta_j)} \cdot \frac{S_j(\theta_j|w_j)}{S_j(w_j|\theta_j)}$
8: Accept $v_j = w_j$ if $u_j < r(w_j)$, otherwise $v_j = \theta_j$
9: **end for**
10: $\boldsymbol{v} = \{v_1, ..., v_{d+1}\}$
11: **if** $\hat{g}(\boldsymbol{v}) > b_m^{nc}$ & $\hat{g}(\boldsymbol{v}) \in D_{0.95}$ **then**
12: Accept $\boldsymbol{\theta'} = \boldsymbol{v}$
13: **else**
14: Accept $\boldsymbol{\theta'} = \boldsymbol{\theta}$
15: **end if**

6.3.3 Potential Computational Cost of The Surrogate

The computational expense associated with the likelihood function within BUS stems from the requirement to make n computations for every proposed sample to identify whether it belongs in the failure domain. This results in a total of $N \cdot n$ computations at each intermediate level. In studies which introduce a surrogate to replace a computationally intensive function, it is often the case that the computational savings in the underlying function are highlighted while

the computational expense of using a surrogate is neglected [39, 170, 40]. An example of this can be seen in running a SVM a large number of times in 2SMART. For BUS_{SVM}, the choice of q dictates the size of the training set for the SVM approximation. Intuitively, the larger the selected value, the greater number of calls to the true limit state function are made. By construction, the number of likelihood function evaluations has an inverse relationship with the number of training instances. Non-linear mappings raise the issue of computing a $n \times n$ kernel. When n is large, it can be computationally expensive. For the case of BUS_{SVM} this involves a $(N - q \cdot N) \times (N - q \cdot N)$ matrix. Therefore, finding a balance between the computational expense of the surrogate, the number of likelihood functions calls and the surrogate accuracy is paramount. To avoid this scenario, it has been argued if the number of dimensions in a given problem is large, one may not need to map the data to a higher dimensional space [109]. Whereas in this case, the linear kernel may be used. On the other hand, it has been shown that the linear kernel is a degenerate version of the RBF kernel and hence is never more accurate than a properly tuned RBF [120]. This may suggest that the use of a non-linear kernel within BUS_{SVM} may be dependent on both the dimensionality and number of observations of a given data set.

6.4 Numerical Applications

The purpose of this section is to numerically illustrate the capabilities of BUS_{SVM} in comparison to BUS_{+}. The first problem concerns the identification of the inter story stiffness parameters for example 4 discussed in Chapter 4 and 5. The second problem aims to sample from a posterior distribution consisting of a mixture of 20 Gaussian distributions, in order to identify the ability of BUS_{SVM} in populating multiple modes with an equal number of samples. The final problem is a binary classification task using 10 data sets varying in dimensionality and the number of observations. The capabilities of BUS_{SVM} is illustrated against BUS_{+} in terms of predictive accuracy, TME and wall clock time. For each problem, the RBF kernel is used while all results are taken as averages from a large number independent model runs unless otherwise stated.

6.4.1 Inter-Story Stiffness Parameters

The first problem concerns the identification of the inter story stiffness parameters as discussed in example 4 in Chapter 4. In line with the experimental set up in previous chapters, the bias in the quantities of interest are used as metrics for model accuracy. The parameters of BUS_{SVM} are chosen as $N = 10^3$ and $p_0 = 0.1$. As the number of likelihood function calls is dependent on q, careful selection is required. To investigate the influence of q, the top two panels of Figure 6.5 presents a sensitivity study for q with values ranging from $[0.2 - 0.9]$. Even though q must be chosen such that Proposition 2 holds and this entails that $1 - q > 0.2$ (i.e. the largest possible value is less than 0.8), the interval $[0.2 - 0.9]$ is chosen to examine the behaviour of the quantities of interest. Both the expectation and standard deviation of $\hat{\boldsymbol{\theta}}_1$ are compared against the respective reference values. For an increase in q, it is apparent that a trend in the variation of values is not present. Remembering that an increase in q reduces the proportion of samples from N generated by means of SVM, one might have expected a decrease in the variation of values. With respect to the reference values, for this problem an appropriate value for q is in the interval $[0.4 - 0.6]$ which coincides with the constraints placed on q by Proposition 2. For this purpose, q is set to 0.5 while BUS_+ is used as a benchmark for performance. In terms of computational savings, BUS_{SVM} results in a reduction of TME of 61% and 59% for 10^3 and 10^4 respectively for $q = 0.5$. For inspection of statistical estimation, the bottom two panels of Figure 6.5 illustrates a general decrease in the variation of estimates for both BUS_{SVM} (blue) and BUS_+ (orange) as the number of MCMC samples increases. For $N < 10^3$ the variation in the estimated expectation by BUS_{SVM} is less than that of BUS_+. Closer inspection of Figure 6.5 (d) shows that BUS_{SVM} appears to be underestimating the standard deviation for all N considered. Further analysis revealed that this underestimation stemmed from the quality of learning of BUS_{SVM} at individual modes of the posterior distribution.

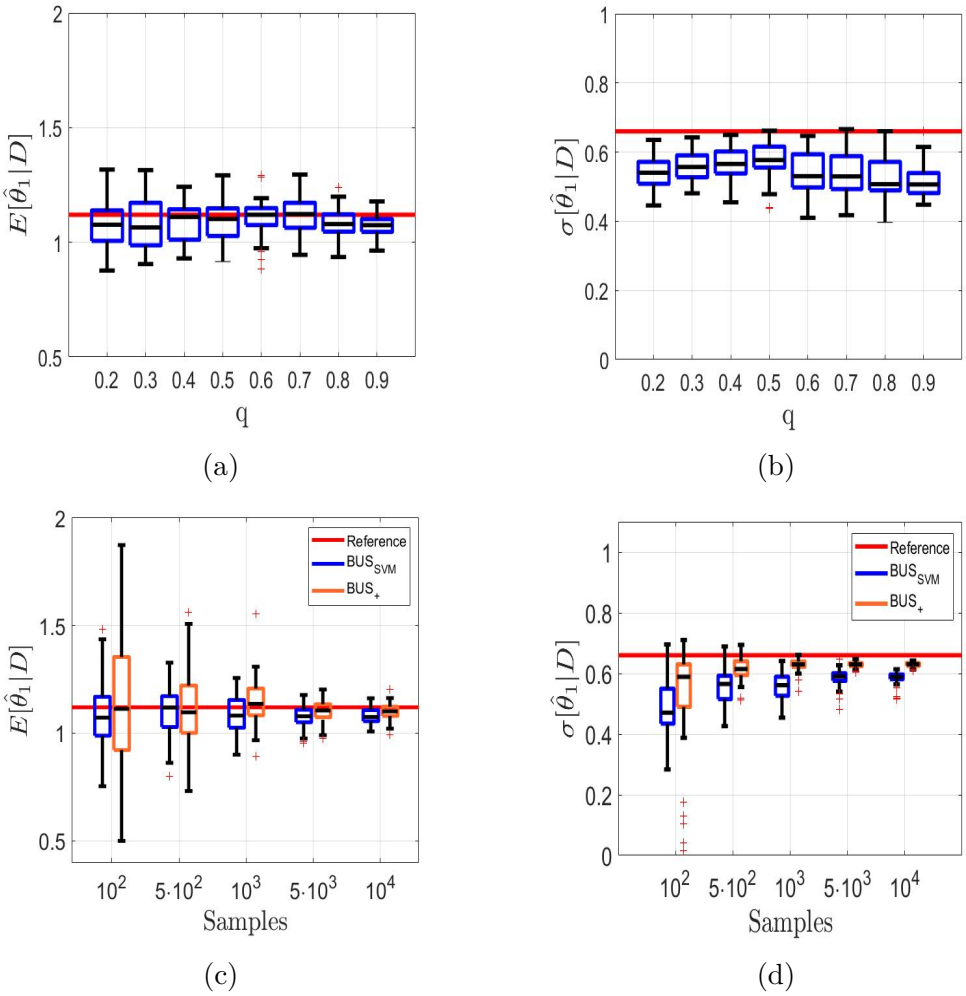
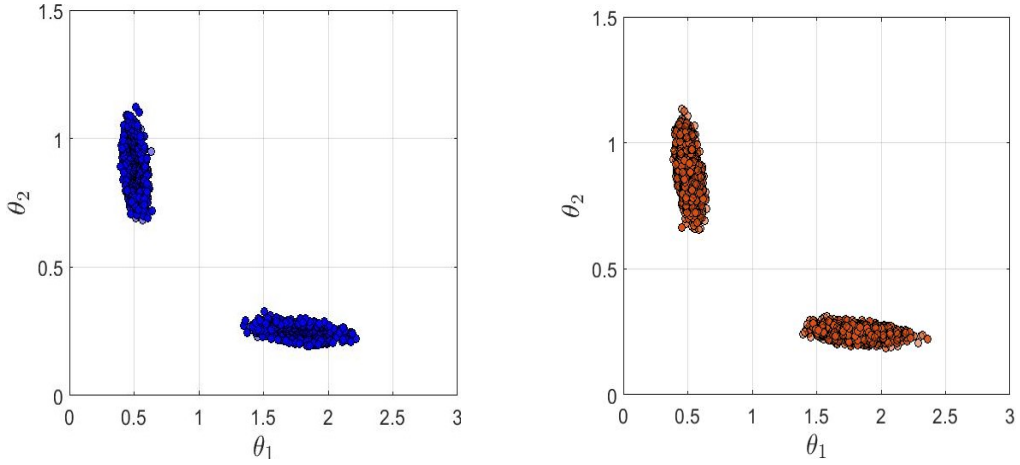


Figure 6.5: The top row shows the sensitivity in the estimates of the expectation (a) and standard deviation (b) for an increasing q. The bottom row compares BUS_{SVM} (blue) and BUS_+ (orange) for estimating the expectation (c) and standard deviation (d). The reference solutions 1.12 and 0.66 are given by the corresponding red lines.

To further investigate the quality of learning of BUS_{SVM} at individual modes, the geometrical behaviour of the samples drawn from the posterior are analysed. Figure 6.6 presents the generated posterior parameter samples in lognormal space. BUS_{SVM} has successfully identified the bi-modal distribution while populating the posterior regions to the same extent as BUS_+.



(a) (b)

Figure 6.6: Posterior samples in the lognormal space for the stiffness parameters generated using BUS_{SVM} (a) and BUS_{+} (b).

The target marginal distributions for both parameters in Figure 6.7, were obtained by numerically integrating the expression for the posterior PDF, which is still feasible for this two-dimensional example. The histogram of samples for BUS_{SVM} and BUS_{+} are comparable with their numerical counterparts. In the case of $\hat{\boldsymbol{\theta}}_1$ however, the samples of BUS_{SVM} fail to populate the modal region to the same extent as BUS_{+}. It is noted in [20] that even if the generated samples were distributed exactly as the posterior distribution, discrepancies between the numerical marginals would still be apparent due to the finite number of sample in the approximation.

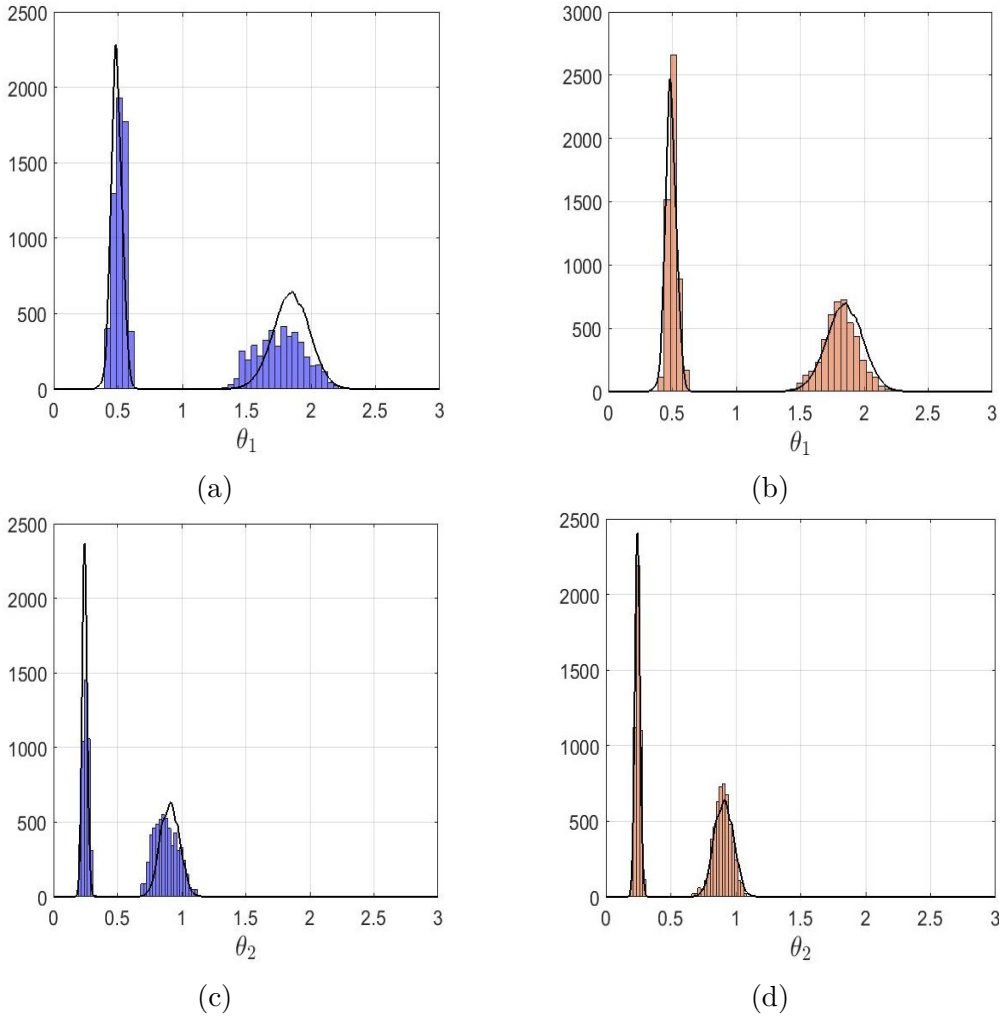


Figure 6.7: Posterior marginal PDFs for $\boldsymbol{\theta_1}$ and $\boldsymbol{\theta_2}$ with there respective estimations using BUS_{SVM} (blue) and BUS_{+} (orange).

Similar to the marginal PDF, the marginal CDF for $\hat{\boldsymbol{\theta}}_1$ is computed via numerical integration as outlined in Figure 6.8 (red) along with the empirical results from 100 independent sampler runs of BUS_{SVM} (blue). The numerical CDF can be seen as being much smoother than its PDF counterpart as the integration step has removed some of the error in the estimation.

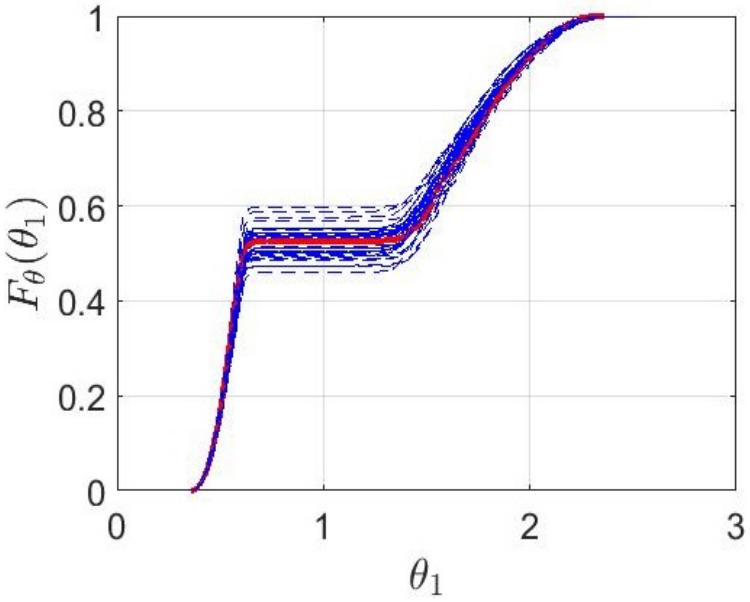


Figure 6.8: Empirical CDF estimate of $\boldsymbol{\theta_1}$ using BUS_{SVM}. CDF estimates from a large number of independent sampler runs are outlined in blue with the reference solution in red.

6.4.2 Mixture of Gaussians

Traditional MCMC methods tend to cope poorly with multi-modal distributions as they can become stuck at the local minimum of a single mode. By utilizing SuS, which can efficiently explore disjoint failure domains, BUS_+ is in turn suitable for problems involving complex multi-modal distributions. A variant of the multi-modal example discussed in Chapter 5 for sampler convergence is considered to benchmark BUS_{SVM} against BUS_+. Consider the target distribution as a mixture of 20 bivariate Gaussians.

$$f_{\boldsymbol{\Theta}}(\boldsymbol{\theta}) = \sum_{i=1}^{20} \frac{w_i}{2\pi\sigma^2} e^{\left(\frac{-1}{2\sigma^2}(\boldsymbol{\theta}-\mu_i)^T(\boldsymbol{\theta}-\mu_i)\right)} \tag{6.28}$$

where $\{\mu_1, ...\mu_{20}\}$ are specified in [129]. However, each mixture component is assigned a weight w of 0.05 and equal variance 0.001. Under this scenario, the modes are well separated where most of the modes are more than 15 standard deviations apart. This makes it challenging for standard MCMC algorithms to explore all of the modes. The reference solutions for the statistical moment estimation are given by $\mathbb{E}[\hat{\boldsymbol{\theta}}_1|D] = 4.48$, $\mathbb{E}[\hat{\boldsymbol{\theta}}_2|D] = 4.91$, $\sigma[\hat{\boldsymbol{\theta}}_1|D] = 2.36$ and

$\sigma[\hat{\boldsymbol{\theta}}_2|D] = 3.14$. Similar to the example discussed in Section 6.4.1, for an increase in q no trend in the variation of moments values is apparent from Figure 6.9 (a)-(b). As such, q is selected as 0.5.

Boxplots of the estimated expectation and standard deviation of $\boldsymbol{\theta}$ in Figure 6.9 (c)-(f) illustrate the ability of BUS_{SVM} in estimating the distribution of parameters which contain multiple modes. For each quantity, the variation in the estimations by BUS_{SVM} is greater than BUS_{+} for all N considered while again for both frameworks the uncertainty in the estimates decreases for an increasing N. For $N = 10^3$ and $N = 10^4$, BUS_{SVM} with $q = 0.5$ requires 49% and 36% less TME than BUS_{+}.

Figure 6.10 (a) presents the posterior samples (red) generated by BUS_{SVM} with contours of the mixture of Gaussian's in black. Each of the distribution modes appears to be populated by the samples. Aside from a qualitative inspection of the ability of BUS_{SVM} to populate individual modes. As before, the mode coverage metric is considered whereby once the number of samples falling within a specified radius of the mode centre is greater than a predefined threshold, the mode is considered covered. To identify whether each mode is adequately populated, the mode coverage is computed for an increasing number of samples. A value of 0 denotes no mode has been fully populated and a value of 1 that each mode contains an equal number of samples. It is worth noting that the interpretation of this metric changes in the case that the mixture of Gaussian's are not equally weighted. From Figure 6.10 (b) as expected, the percentage of mode coverage increases for an increase in N. The mode coverage of BUS_{+} (orange) is greater than BUS_{SVM} (blue). However, with respect to the computational savings in terms of TME the difference is deemed negligible. For 10^4 samples, both methods adequately populate approximately 50% of the modes. The five modes which the framework had difficulty in populating in terms of mode coverage are denoted by blue crosses in Figure 6.10 (a). While the four most populated modes were those lying closest to the origin.

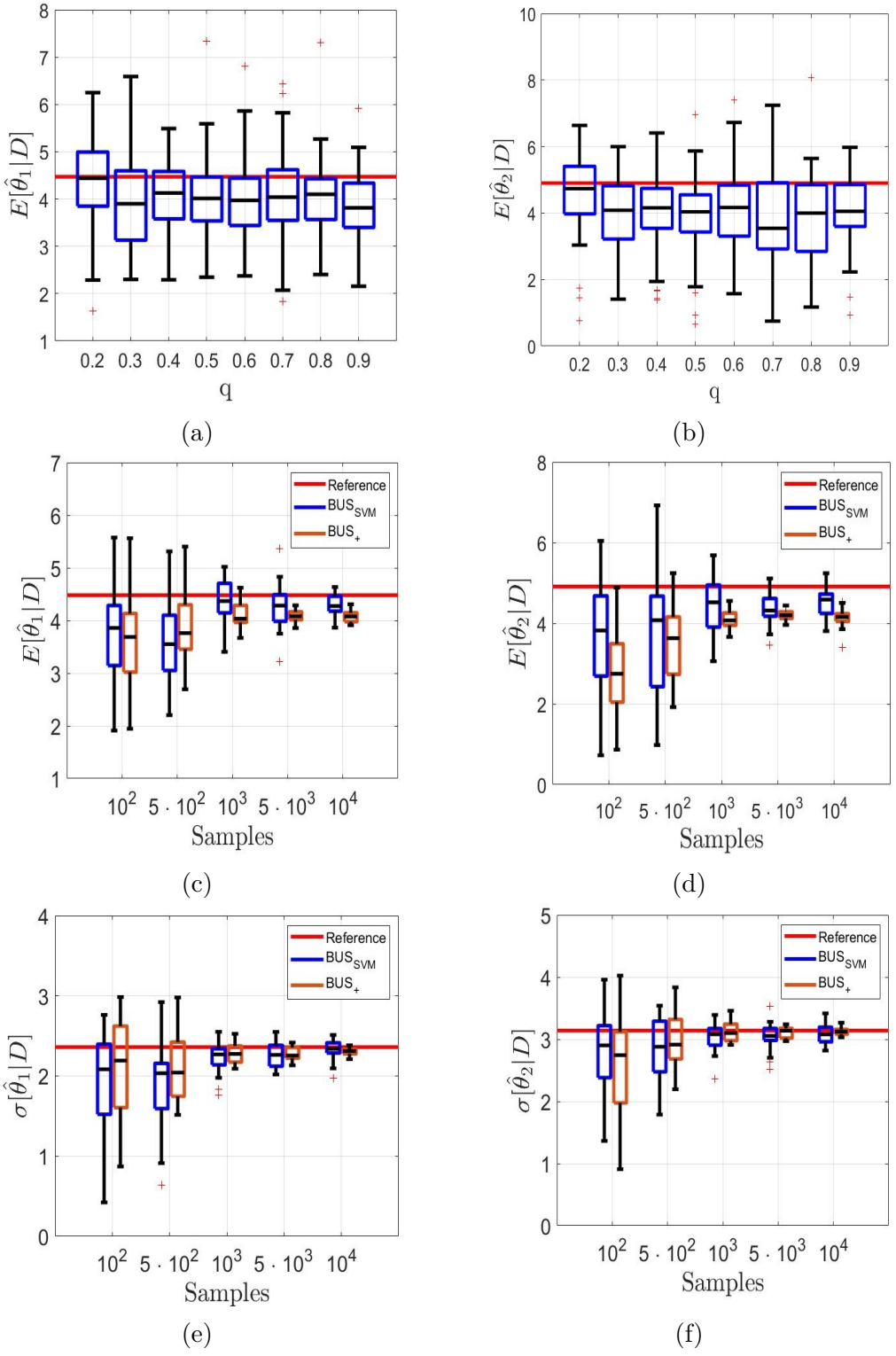
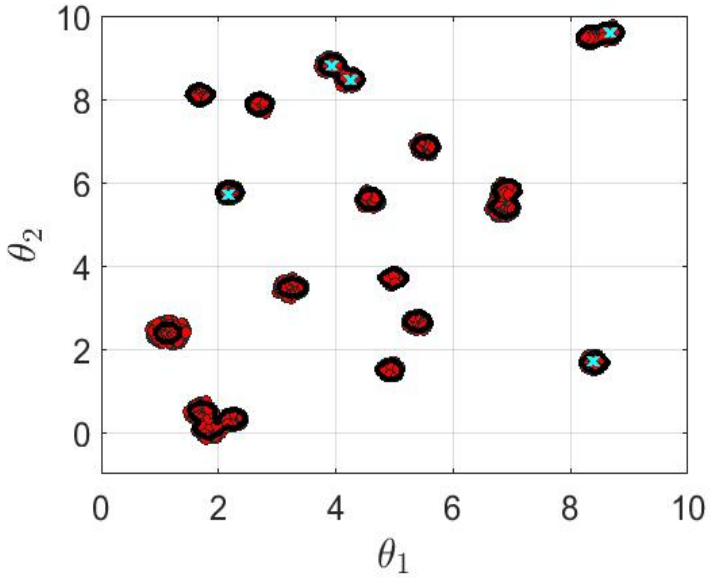
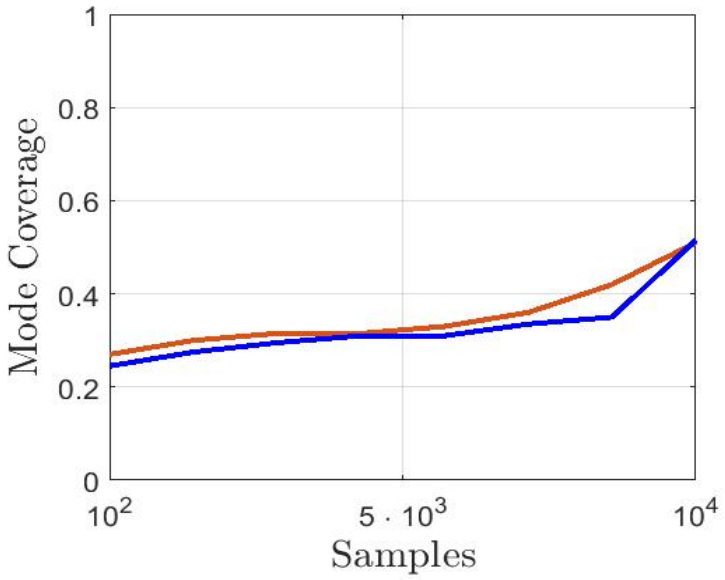


Figure 6.9: Top two panels show the expectations for θ_1 (a) and θ_2 (b) for an increasing q. Comparisons in moment estimation between BUS_{SVM} and BUS_+ are shown in panels (c)-(f).



(a)



(b)

Figure 6.10: Panel (a) presents the posterior samples (red) for the mixture of Gaussians posterior distribution. The blue crosses denote the five modes BUS_{SVM} had most difficulty in populating. Panel (b) shows the trend of mode coverage for BUS_{SVM} (blue) and BUS_{+} (orange) for an increasing number of samples.

To further empirically investigate the convergence of both samplers in terms of mode coverage, the samplers were run for increasing values of N up to a maximum of $5 \cdot 10^4$. The level probability was chosen as $p_0 = 0.1$. It was found that BUS_{+}

converged to a mode coverage of 1 with approximately $N = 4.3 \cdot 10^4$ samples whereas for $N = 5 \cdot 10^5$ the maximum mode coverage achieved by BUS_{SVM} was 0.93. Future work may entail the use of clustering methods to improve learning at individual modes of distributions by BUS_{SVM}.

6.4.3 Supervised Machine Learning

The purpose of developing BUS_{SVM} is to alleviate the computational cost associated with likelihood evaluations in the presence of a large number of data observations. In this example, 10 benchmark classification problems are considered. Both BUS_{+} and BUS_{SVM} are compared in terms of classification Area Under the Curve (AUC), Total Model Evaluations (TME) and clock running time. To justify the use of clock running time as a comparison metric, all experiments were run a large number of times using the same i5 core vPro processor and programmed in MATLAB®. Table 6.1 presents details of 8 benchmark data sets as both the USPS and image data sets discussed in Chapter 4 are also included in this study. The covtype [69] data set concerns the identification of forest cover type from four different areas of the Roosevelt national forest in northern Colorado, USA. The financial data [168] describes the financial ratios of various business firms aimed at establishing whether or not they are credit worthy. Two different cases of whether or not a given observation is drawn from an overlapping normal distribution is the classification task for ringnorm and twonorm [1]. The identification of two classes of splice junction in a DNA sequence is considered in the splice [69] example while the waveform data set [69] concerns differentiating between two distinct wave patterns in a signal. The analysis of the thyroid data set [69] is to determine whether or not the gland is functioning correctly. The X-Ray problem involves the identification of two different lung conditions from X-Rays taken at the national institutes of health clinical centre in Maryland, USA [221]. The original dataset consisted of several different diseases but for this study the task of simply whether or not a patient has healthy lungs is considered.

	Observations	Features	Area
Covtype (C)	495141	10	Forestry
Finance (F)	17108	93	Finance
Ringnorm (R)	7400	20	Probability Theory
Splice (S)	3190	59	Biology
Thyroid (Th)	215	5	Medicine
Twonorm (Tw)	7400	20	Signal Processing
Waveform (W)	5000	21	Probability Theory
X-Ray (X)	69908	9	Medicine

Table 6.1: Classification benchmark data sets. For space making purposes the following data set acronyms are used: Covtype (C), Finance (F), Image (I), Ringnorm (R), Splice (S), Thyroid (Th), Twonorm (Tw), USPS (U), Waveform (W) and X-Ray (X).

For consistency, cross validation is used along with a misclassification threshold of 0.5. To investigate the influence of q on larger datasets, two values of 0.2 and 0.5 are chosen for this example. For the sake of brevity let the resulting methods be denoted by BUS_{SVM2} and BUS_{SVM5}. Table 6.2 presents the results for BUS_{+}, BUS_{SVM2} and BUS_{SVM5}. Each first row contains the AUC, the second row TME and the final row the average wall clock time in seconds for training and prediction. For the covtype, image, ringnorm, USPS and X-Ray datasets it is clear that BUS_{SVM2} performs poorly in terms of AUC with respect to BUS_{+}. In all cases aside from the image data set, the standard error of the estimate is similar to BUS_{+}. Whereas BUS_{SVM5} performs comparably well with BUS_{+}. The reduction in the number of TME when using SVM within BUS_{+} is evident in all cases. As expected, BUS_{SVM2} requires fewer TME than BUS_{SVM5}. However, the great deterioration in model accuracy reveals this comes at a cost. The trade-off between limiting the number of TME and increasing the training set size for the SVM is evident from the wall clock times. Even though the number of required TME is smaller for all data sets, the running time of BUS_{SVM2} is greater than

BUS_{SVM5} in each case. By decreasing the value of q, the SVM is required to compute a larger kernel matrix. This highlights the importance of balancing the computational savings of using a function approximation with the cost of the function approximation itself.

	Metric	**C**	**F**	**I**	**R**	**S**
$\boldsymbol{BUS_+}$	AUC	0.75(0.001)	0.73(0.03)	0.98(0.01)	0.77(0.01)	0.64(0.03)
	TME	4.7×10^3	4.4×10^3	3.3×10^3	4.4×10^3	3.8×10^3
	Time	471.8	46.1	9.5	3.4	2.4
$\boldsymbol{BUS_{SVM2}}$	AUC	0.59(0.002)	0.70(0.03)	0.74(0.18)	0.61(0.02)	0.61(0.03)
	TME	1.8×10^3	2.3×10^3	2.6×10^3	2.1×10^3	2.3×10^3
	Time	269.8	27.5	34.2	3.1	3.2
$\boldsymbol{BUS_{SVM5}}$	AUC	0.73(0.001)	0.74(0.03)	0.95(0.11)	0.74(0.02)	0.76(0.03)
	TME	2.2×10^3	2.5×10^3	2.6×10^3	2.2×10^3	2.5×10^3
	Time	242.6	20.9	27.8	2.7	2.6
	Metric	**Th**	**Tw**	**U**	**W**	**X**
$\boldsymbol{BUS_+}$	AUC	0.79(0.09)	0.99(0.01)	0.97(0.01)	0.92(0.01)	0.65(0.01)
	TME	4.7×10^3	5.1×10^3	$3. \times 10^3$	4.7×10^3	4.1×10^3
	Time	1.2	4.2	10.1	3.1	43.8
$\boldsymbol{BUS_{SVM2}}$	AUC	0.83(0.09)	0.99(0.002)	0.79(0.03)	0.87(0.02)	0.48(0.05)
	TME	1.6×10^3	1.7×10^3	2.3×10^3	2.1×10^3	2.9×10^3
	Time	10.1	3.1	9.6	2.9	25.8
$\boldsymbol{BUS_{SVM5}}$	AUC	0.85(0.08)	0.99(0.002)	0.97(0.01)	0.89(0.02)	0.62(0.008)
	TME	2.1×10^3	2.2×10^3	2.5×10^3	2.9×10^3	3.2×10^3
	Time	1.7	2.5	7.8	2.4	18.6

Table 6.2: Experimental results for the 10 benchmark data sets. For each method the first row represents the AUC, the second row the number of TME and the third row the average wall clock running time.

6.5 Chapter Summary

Motivated by the task concerning MCMC methods in the presence of large data sets, the question of improving the scalability of BUS was addressed in this chapter. By interpreting the engineering reliability problem as a binary classification problem, the evaluation of the likelihood function for proposed MCMC samples is replaced by a functional approximation. Two SVM inspired frameworks are presented to reduce the number of TME in a model run. Firstly, observations and potential improvements for 2SMART [40] are outlined to increase its suitability for Bayesian updating tasks along with improving algorithmic stability for smaller MCMC sample sizes.

Secondly, a new SVM inspired method is proposed which avoids the requirement of training multiple SVM at each intermediate level of SuS. BUS_{SVM} selects training points from the set of MCMC samples generated at each level based on a user defined parameter to train a SVM model. From the numerical experiments a suggested value of 0.5 allows for a balance between the likelihood function and training the SVM model to be achieved. To control the error introduced by the functional approximation, a set of rules must be followed for a candidate sample to be accepted. The adoption of the radial basis function allows for the SVM to accurately approximate non linear failure boundaries. However, the framework does offer flexibility in terms of kernel choice. Numerical results illustrate the capabilities of BUS_{SVM} for multi-modal problems along with classification tasks involving hundreds of thousands of observations. To establish the suitability of BUS_{SVM} along with BUS_{+} and $nBUS_{D}$ for different data types, the next chapter concerns a real world application of classification for the detection of breast cancer.

Chapter 7

Breast Cancer Detection

From the outset of this dissertation, the goal was the development of BUS techniques suitable for supervised machine learning problems varying in data complexity. Given the theoretical underpinnings of BUS, the numerical experiments discussed in previous chapters have primarily stemmed from the engineering community. The purpose of this was to compare the accuracy and stability of the proposed frameworks to existing methods. In this chapter, BUS_{SVM}, $nBUS_D$ and BUS_+ are applied to two real world problems in the area of breast cancer detection. The first problem concerns the identification of cancerous tissue in Whole Slide Images (WSI) of patients, with the classification task whether or not a given specimen is benign or malignant. Additionally, the scenario where incorrect training labels have been provided by the pathologist is also considered. The second problem focuses on the identification of molecular biomarkers for predicting five year breast cancer relapse rates. Biomarker datasets are traditionally high dimensional and contain very few observations. Consequently, the BUS approaches are compared against existing Bayesian inference methods to establish their suitability in this setting. The work presented in this chapter was undertaken whilst on placement at IBM Research Zürich.

7.1 Classification in Healthcare

As outlined in Chapter 2, classification methods have become one of the most widely applied forms of machine learning for real world problems. In UK healthcare for example, the National Health Service (NHS) has adopted a data driven approach towards improving service efficiency. The creation of a machine learning

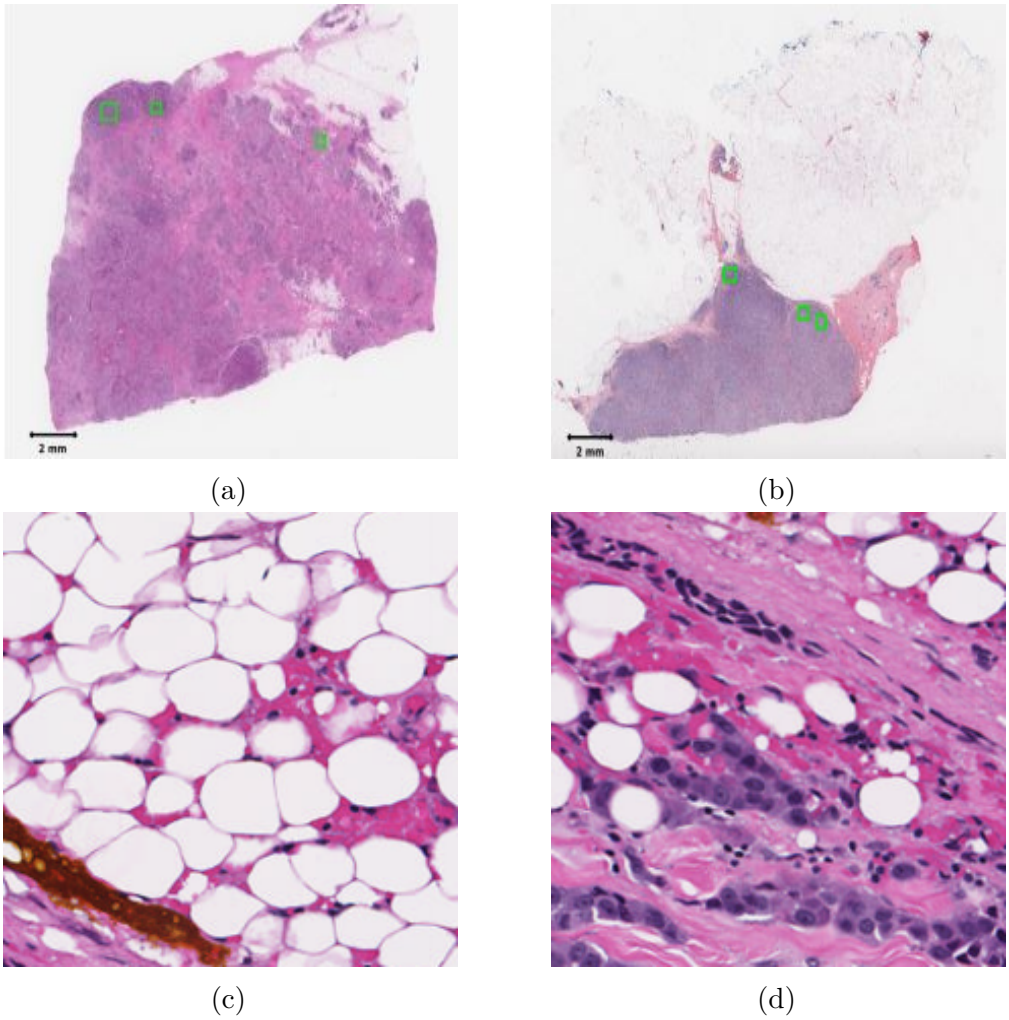
driven phone application *GP at hand*, has allowed potential patients to seek professional consultation without having to attend the practice in person. The phone application exploits classification techniques to build an interactive chatbot which identifies whether the prospective patient requires a consultation with a physician or whether the symptoms may be cured with over the counter medicine. Ensuring a reduction in the number common illnesses (flu, colds etc.) cases required to be assessed in person allows for a physicians time to used more optimally with patients suffering from serious illnesses. Aside from improving general business practice in terms of efficiency, classification frameworks have also attracted major interest for applications in cancer identification. In general, when cancer is detected early, treatment is more effective. An on-going line of research is investigating the potential of classification techniques in providing this early diagnosis. At IBM Zürich, a team of scientists are actively researching the use of machine learning techniques aimed at identifying particular cancer drivers along with providing a tool to analyse histopathological images for breast cancer diagnosis.

In terms of the latter, visual inspection of Whole Slide Images (WSI) is a commonly used for the identification of cancerous cells. During this process, a biopsy sample is acquired from a patient before being evaluated by a pathologist through the use of a microscope. Intuitively, two issues for both patient and practitioner arise during this process. Firstly, visual inspection is extremely time consuming. Given the aggressive nature of the disease, time has been identified as a key factor in improving survival rates. Theoretically if correctly implemented, machine learning has the potential to reduce the waiting time between obtaining the biopsy sample and returning the determined diagnosis. In turn, allowing for possible treatments to be started at an earlier time. The second issue with visual inspection is the possible variation in diagnosis between pathologists. Two different types of variabilities can potentially occur. In the case of *inter-variability*, suppose ten pathologists have been asked to identify potential cancerous regions in biopsy's taken from five different patients. Inter-variability refers to the amount the ten observers may vary from one another in terms of their concluding remarks on the presence of the disease in each of the patients. In the second case, consider one of the pathologists is chosen to re-examine the same five tissue biopsys at a later date. *Intra-variability* may appear in the event that the chosen pathologists updated recommended diagnosis differs from their initial conclusions drawn. Both

inter and intra-variability result in uncertainty appearing within the diagnosis procedure. In this aspect, probabilistic machine learning frameworks quantify the degree of certainty surrounding the recommended diagnosis. Given the machine learning algorithm learns a functional mapping from the tissue sample to diagnosis, once correctly trained the resultant framework could be potentially used in an online learning capacity. In this context machine learning offers the potential to increase diagnosis efficiency on top of quantifying the uncertainty in the predicted diagnosis. Note that the aim of incorporating machine learning frameworks into the medical domain is not concerned with the removal of the practitioner from the decision-making process, but to provide them with tools to make better informed decisions.

7.1.1 Breast Cancer Biopsy Samples

To continue the exploration of machine learning in cancer diagnosis, the inference frameworks presented in Chapter 5 and 6 are implemented on breast cancer WSI data sets which were publicly released as part of the *MACCAI 2016* challenge [173]. Due to the aforementioned issues in the variability between expert diagnosis, a voting system between multiple experts was utilized to obtain the labels for the data set. The binary task considered is the identification of whether a given sample is benign (non-cancerous) or malignant (cancerous). The initial dataset consisted of 10 patients. Examples of WSI are given in Figure 7.1 (a) and (b) with the green squares highlighting regions of interest identified by a pathologist. However, given the small sample size and a single WSI tending to be $100,000 \times 100,000$ pixels in size, each image is discretized into a series of *patches* [179]. Figure 7.1 (c) and (d) present examples of benign and malignant patches respectively. The advantage of which is creating a large data set from an initial small number of patients.



(a) (b)

(c) (d)

Figure 7.1: Panels (a) and (b) present examples of a WSI of a breast biopsy. The green squares denote regions of interest identified by a pathologist. Given the size of each image, panels (c) and (d) provide an example of a benign and malignant patch from the same WSI.

The resulting data set details are given in Table 7.1, which also contains the class split for each patient. Note that the number of observations represents the number of patches for a given patient.

Patient	**Observations**	**B/M**
$\boldsymbol{P_1}$	4019	1110/3309
$\boldsymbol{P_2}$	2532	1243/1289
$\boldsymbol{P_3}$	3592	1702/1890
$\boldsymbol{P_4}$	2864	1459/1405
$\boldsymbol{P_5}$	2133	1184/949
$\boldsymbol{P_6}$	2468	1125/1343
$\boldsymbol{P_7}$	3029	119/2910
$\boldsymbol{P_8}$	1491	818/673
$\boldsymbol{P_9}$	1228	276/952
$\boldsymbol{P_{10}}$	3485	1565/1920

Table 7.1: Whole slide image data set for ten patients. Each patient differs in the number of observations and balance of class labels. In this case the number of observations signifies the number of patches. For this task, the class labels are benign (B) and malignant (M).

The data pre-processing of the images was carried out in the same manner as the image classification problems discussed in Chapter 4. As with the image data set in Chapter 4, a colour normalization algorithm discussed in [173] was used to convert the images to greyscale before each pixel of the WSI images was used as a dimensions. This resulted in each observation being represented in 1024 dimensions. In turn, allowing for the performance of the BUS frameworks to be analysed in high dimensional real life examples. As discussed in Chapter 2, the AUC, TP (true positive), FP (false positive), TN (true negative) and FN (false negative) rates are used as metrics for the predictive performance. For cancer detection the risk of falsely classifying a sick patient as healthy far out weighs that of falsely classifying a healthy patient as sick. Of course the latter scenario has unnecessary ramifications for the patient in terms of treatment and monetary costs. However, the main concern is correctly identifying individuals with cancer. As such, one aims to minimize the FN rate.

The AUC, TME and FN rates for $nBUS_D$ (blue), BUS_+ (orange) and BUS_{SVM} (yellow) are given in Figure 7.2 (a)-(c). As expected, the number of TME for BUS_{SVM} is less than both $nBUS_D$ and BUS_+. With BUS_{SVM} resulting in a reduction of 48% and 34% with respect to $nBUS_D$ and BUS_+. All results were obtained with 5-fold cross validation using the logistic regression model. In terms of AUC, aside from patient 1, the posterior samples generated by each inference framework produce competitive classification models. The standard errors for the AUC were of the same order for each of the three BUS frameworks. Each BUS approach is compared with Laplace approximation (LA), variational inference (VI) and Metropolis Hastings (MH) in Figure 7.2 (d) for patient 5 in terms of AUC and wall clock time in seconds. The vertical error bar represents the AUC standard error while the horizontal error bar the variation in wall clock time for each fold. For parameter identification, VI runs the quickest followed closely by BUS_{SVM}. Additional intermediate levels generated by $nBUS_D$ result in the greatest computational time of the BUS methods. As expected, MH takes the longest to converge to the posterior but produces the highest AUC rate. While the usual efficiency associated with LA, suffers in the presence of the computation of a Hessian matrix for a large number of dimensions.

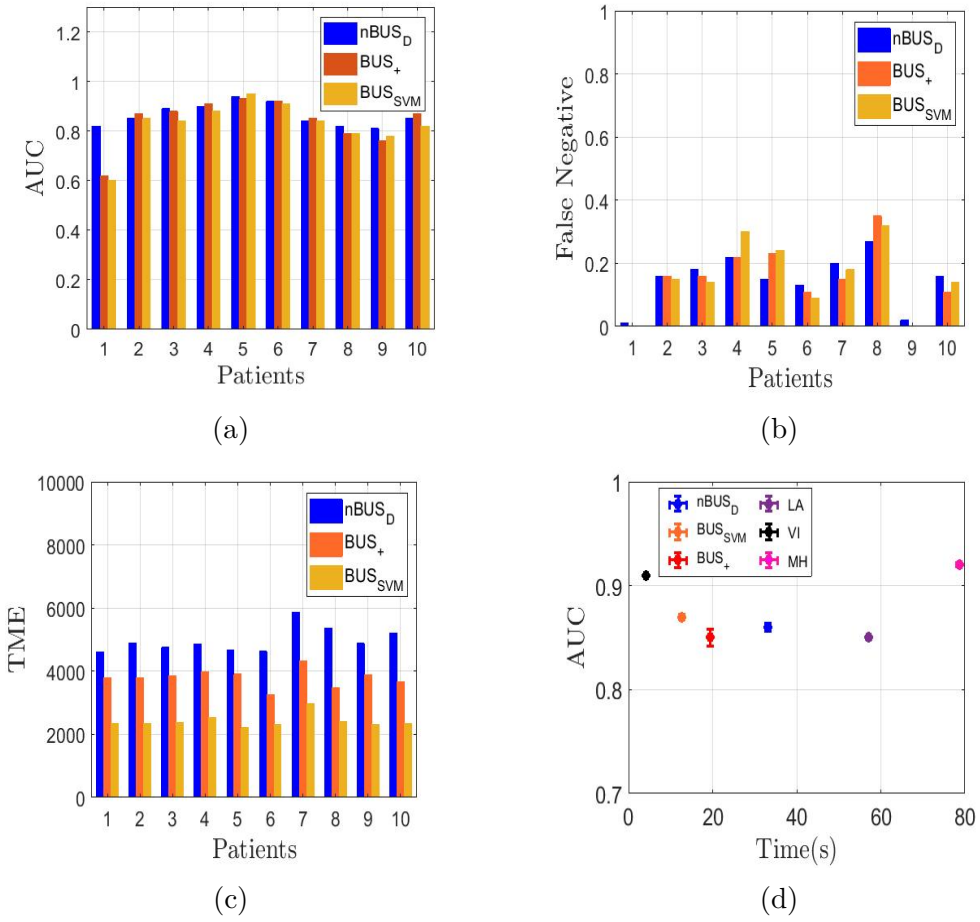


Figure 7.2: Panels (a), (b) and (c) present the AUC, false negative rate and TME using each BUS framework for the 10 patients in the WSI data set. Each method performs comparably well while BUS_{SVM} requires a fraction of the TME to sample form the posterior of model parameters. Panel (d) benchmarks the BUS methods against three competing inference techniques in terms of wall clock time and AUC.

Figure 7.3 presents an example of a malignant WSI patch. The predictive contours reveal which regions of the tissue sample that the classifier has identified as a region of interest. Aside from high accuracy and confident decision making, an interpretable model is highly desirable. Under visual inspection, pathologists may identify different areas of a sample to be a region of interest based on similar characteristics to a previously examined specimen. Classification models which produce posterior probabilities allow for a degree of certainty to be attached to different regions of the WSI. As the entire WSI is representative of a malignant specimen, the classifier is correctly attaching a high level of confidence of cancer

identification. The separation of probabilities along the diagonal of the contour plot coincides with the morphology of this specific tissue sample. It should be noted that an interesting future line of research may be the comparison between what denotes a region of interest for the pathologist and what the model defines as a region of interest. Both may come to the same diagnosis, but the information which lead to the decision being made may differ greatly. This however, is beyond the scope of this current work.

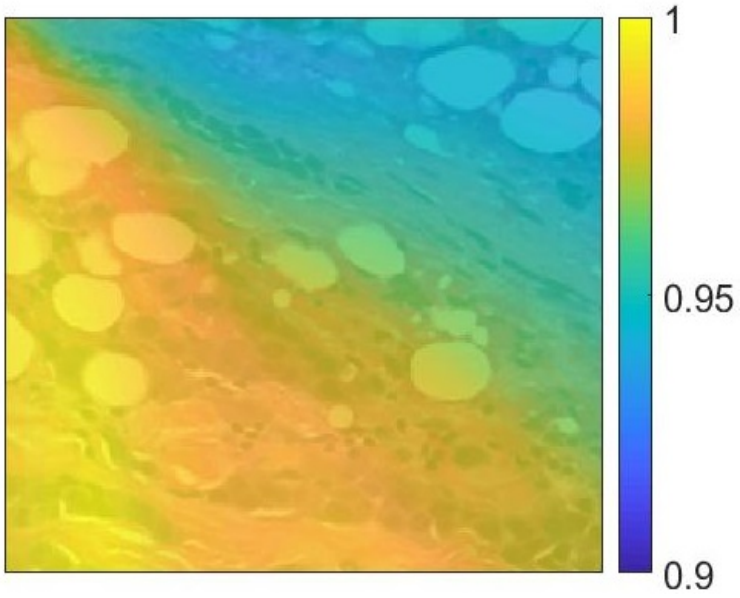
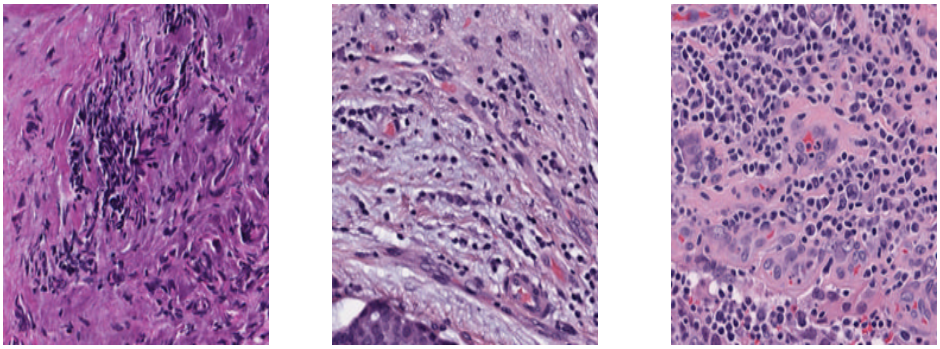


Figure 7.3: Predictive contour of a malignant patch. The algorithm has identified the left area of the image as the main region of interest with a probability of 1. Given the entire image is an example of a malignant specimen, the classifier recognizes with a high probability the presence of cancerous tissue.

Aside from the binary task of identifying whether or not a given tissue sample is benign or malignant, in the case of a malignant sample the aggressiveness of the disease should also be considered. A common practice for representing this aggressiveness is by a proliferation score. A widely applied technique among pathologists for tumour identification is mitosis counting which is dependent on the number of overlapping cells occurring in the sample. The proliferation score acts as a frequency metric by combining mitosis counting, comparison of tumour cell structure and comparison of the individual cells. Each of these categories are assigned a proliferation score ranging from 1 to 3. For example, in the case of mitosis counting a score of 3 indicates a large number of cells are overlapping (i.e. the cancer is extremely aggressive). On the other hand, a score of 1 indicates

a small number of overlapping cells in the sample. In essence, a score of 3 may be viewed as a worst case scenario. For each category, the proliferation score is computed and summed together with the total number referred to as a Nottingham score [74]. A score of $0-5$ is representative of grade one cancer, $6-7$ grade two cancer and >8 grade three cancer.

Figure 7.4 presents examples of biopsy samples with Nottingham scores of $0-5$ (a), $6-7$ (b) and >8 (c) respectively. Using the category of mitosis counting as a visual aid, evidently there are a greater number of overlapping cells (dark purple regions) in Figure 7.4 (c) in comparison to the remaining two samples. Combining the patient data into a single data set forms a difficult binary classification task due to the varying phenotype of a mitotic cell as this goes through a mitosis cycle which is part of the cells life cycle. This means that observations contained in the same class can potentially differ greatly. For the 26241 observations considered $nBUS_D$, BUS_+ and BUS_{SVM} produce FN rates of 0.17, 0.08 and 0.04. The number of TME was 5.1×10^3, 3.9×10^3 and 2.4×10^3. The misclassification threshold dictates the corresponding TP, TN, FP and FR. An advantage of which is setting its value to minimize or maximize one of the quantities. In the interest of fairness, for this study a threshold of 0.5 was used in all cases with no further analysis done in terms of an optimal value for each patient.



(a) Score $0-5$. (b) Score $6-7$. (c) Score >8.

Figure 7.4: Nottingham scores for the breast tissue samples with examples of a score of $0-5$ (a), $6-7$ (b) and >8 (c). For a score greater than 8 it is visually evident that a greater number of overlapping cells are occurring. This means that the cancer is at a more advanced stage in comparison to panel (a) and panel(b).

7.1.2 Incorrect Labels

In healthcare, as most medical tests are not 100% accurate the resulting data cannot be assumed to be a noise free gold standard. This issue also appears in the case of WSI. Stemming from the aforementioned potential variation between pathologists diagnosis, the resulting labelled set is susceptible to containing noise. A noisy label is one which has been incorrectly assigned a given class label. Supervised machine learning functions on the assumption that the labelled data is a correct representation of reality. However, reliably labelled data are often expensive and time consuming to obtain. With the increase in the application of machine learning techniques to solve real world problems has come the need to incorporate such uncertainties into models. Corruption of the instances themselves often referred to as feature noise, results in the input data not being a fair representation of reality. The importance of label noise over feature noise has been highlighted in [77] whom argue that there may be many features in a data set but only one class label. As such, the importance of each feature varies whereas the class labels always play a major role on model training.

Numerous approaches varying in theoretical underpinnings, have been proposed to address the issue of class label noise. Classification filtering, concerns the identification of possible noisy labels through model predictions [122, 113, 152]. A classifier is learned using the training data, resulting in the removal of all misclassified instances from the test set. As the noise identification is dependent on the classifier itself, deleting instances and labels from the training data in such a manner may result in vital information being removed. Forming an unsupervised task has also been addressed whereby clusters are defined representing a new set of class labels [41, 78, 227].

Probabilistic approaches have been developed [134, 37, 36], which incorporate label noise into the modelling process through the use of probability tables. Although similar in their theoretical foundations, each of the approaches are aimed at specific classification models. The use of unreliable labels, directly influences the set of generated posterior samples. As the likelihood function, represents the probability of $\boldsymbol{\theta}$ generating the observed D, in the event of the data containing incorrect labels, this relationship will change. The estimated parameter values will be incorrect due to the information contained in D. A method is presented which incorporates the uncertainty surrounding the class labels into the Bayesian inference framework by the alteration of the likelihood function.

Consequently, the framework is flexible in terms of both inference methods and classification models.

Suppose y is randomly corrupted with noise which results in $\hat{D} = \{(x, \hat{y})\}$ being generated as the training dataset. In this context 'noise' refers to the mislabelling of data instances due to for example administrative errors during data collection or lack of information leading to inter-variability between pathologist diagnosis. The noise is treated as a stochastic process which is independent of the input data. Under this framework the assumption regarding the label noise is that the true value has been 'flipped' to the opposing value randomly with a uniform probability across the entire label set.

Under a Bayesian approach, a latent variable z is introduced which represents the likelihood of noise being present in the training set. The prior distribution over z is assumed to be a beta distribution with parameters, α and β, which are chosen prior to simulation. The assumption that the training set contains a small level is noise is made by choosing $\beta >> \alpha$. In this case, the beta distribution is a sensible choice as its values are bounded on $[0, 1]$ along with the distribution parameters allowing for flexibility in terms of shape. In a binary setting, with the class labels being discrete quantities, the Bernoulli distribution is assigned as the likelihood function, resulting in the distribution $P(z|\hat{y})$ representing the presence of noise in the training set. Accordingly, as new data becomes available, the distribution over the noisy labels is updated.

In the case of allowing for noise to be incorporated into the modelling process the likelihood function

$$L(D|\boldsymbol{\theta}) = \prod_{i=1}^{n} p(x_i, y_i|\boldsymbol{\theta}) \tag{7.1}$$

is required to be modified. In a noisy setting however it is assumed that y has been corrupted and is represented by $\hat{y}$. Thus the relationship between the unknown parameters and the input data has changed and consequently the likelihood function to include the possibility of noise being present must be altered appropriately.

$$L(\hat{D}|\boldsymbol{\theta}, z) = \prod_{i=1}^{n} p(x_i, \hat{y}_i|\boldsymbol{\theta}, z_i) \tag{7.2}$$

This incorporates an individual label noise term z_i for each data observation. Adopting this modified likelihood function, the posterior distribution for learning both z and $\boldsymbol{\theta}$ based on observing $\hat{D}$ may be given by

$$P(\boldsymbol{\theta}, z|\hat{D}) = \frac{L(\hat{D}|\boldsymbol{\theta}, z)Q(\boldsymbol{\theta})L(\hat{y}|z)Q(z)}{P_{\hat{D}}} \tag{7.3}$$

where $Q(z)$ denotes the beta prior over z and P_D the model evidence. As a first step the placement partners sought a simplistic framework which provided a global summary of the labels potentially being incorrect. Given this, the use of Eq. 7.3 was not desirable from the placement advisors point of view. However, for future work a full Bayesian approach of jointly learning z and $\boldsymbol{\theta}$ would be of great interest. Instead, $P(z|\hat{y})$ was directly utilized for this preliminary study. For the purpose of the placement visit and as a preliminary step of modelling potentially noisy labels, it was assumed that the noisy labelled set had been generated by some predictive process. In this case, the human pathologist is viewed as a predictive classification model with their corresponding diagnosis' the potentially noisy class labelled set. Following this a confusion matrix may be formed which summarises pathologists' predictions. In the case of class 1, to incorporate this information directly into the likelihood function, a flipping factor is introduced as:

$$\hat{G}_1 = \frac{\mathbb{E}[P(z|\hat{y}=0)]}{\mathbb{E}[P(z|\hat{y}=0)] + (1 - \mathbb{E}[P(z|\hat{y}=0)])} \tag{7.4}$$

This quantity may be viewed as the false negative rate. In the case of class 0, to directly incorporate this information into the likelihood function, a flipping factor is introduced as:

$$\hat{G}_0 = \frac{\mathbb{E}[P(z|\hat{y}=1)]}{\mathbb{E}[P(z|\hat{y}=1)] + (1 - \mathbb{E}[P(z|\hat{y}=1)])} \tag{7.5}$$

This quantity may be viewed as the false positive rate. For the sake of brevity consider the log likelihood to be given by

$$\log(L(\hat{D}|\boldsymbol{\theta})) = \sum_{i=1}^{n} \hat{y}_i \log(p(\hat{y}_i = 1|x_i, \boldsymbol{\theta})) + (1 - \hat{y}_i)\log(p(\hat{y}_i = 0|x_i, \boldsymbol{\theta})) \tag{7.6}$$

This line of thought results in the log likelihood function being expressed as follows

$$\log(L(\hat{D}|\boldsymbol{\theta}, z)) = \sum_{i=1}^{n} \hat{y}_i \log(\hat{p}_1) + (1 - \hat{y}_i)\log(\hat{p}_0) \tag{7.7}$$

where

$$\hat{p}_1 = (1 - \hat{G}_1)p(\hat{y}_i = 1|x_i, \boldsymbol{\theta}) \tag{7.8}$$

$$\hat{p}_0 = (1 - \hat{G}_0)p(\hat{y}_i = 0|x_i, \boldsymbol{\theta}) \tag{7.9}$$

In the case of $\hat{G}_1 = 1$, all observed labels with class 1 are assumed to be incorrect. This means that $\hat{p}_1$ reduces to 0, allowing for $p(\hat{y} = 0|x, \boldsymbol{\theta})$ to be computed. Note that $(1 - \hat{G}_1)$ may be viewed as the true positive rate or sensitivity. This measures the proportion of actual class 1 labels that are correctly identified as such i.e. if equalled to 0, $\hat{p}_1$ reduces to 0 as none of the class 1 labels have been correctly assigned. Similarly, $(1 - \hat{G}_0)$ may be viewed as the true negative rate or specificity. This measures the proportion of actual class 0 labels that are correctly identified as such i.e. if equalled to 0, $\hat{p}_0$ reduces to 0 as none of the class 0 labels have been correctly assigned.

It should be noted that the flipping factor is not updated for each observation but computed globally. With respect to the Bayesian inference step, the proposed framework is flexible in terms of implementation with different methods. By introducing uncertainty into the relationship between the parameters and the observed data, the confidence in assigning a probability of class membership changes. Predictive probabilities for the test set of patient 2 as shown in Figure 7.5, reveal the influence of the flipping probability on the resulting classifiers decision making. For the 352 test observations, the robust step for label noise (orange) results in 11% more observations not being assigned a probability of 1 or 0 in comparison to standard inference (blue). The introduction of the flipping factors directly reduce the likelihood of the model parameters generating the data in the event that noise is deemed present in the data set. This in turn reduces the level of confidence in assigning an observation to a given class labels in the presence of potentially incorrect training labels. The aim of incorporating uncertainty regarding the class labels is to slow down the rate of deterioration of classifier performance for increasingly noisy labelled sets.

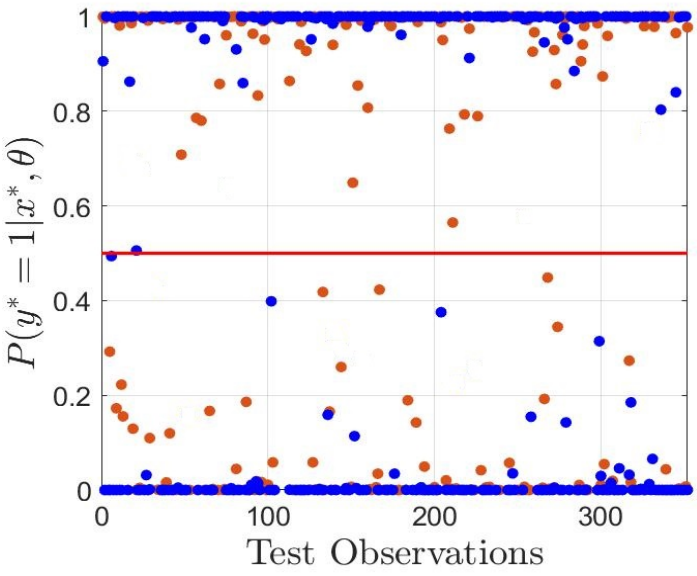


Figure 7.5: Predictive probabilities for the test set of patient 2. A larger proportion of predictions based on standard inference (blue) have been mapped to the probability upper and lower bounds. The introduction of the robust step (orange) results in a reduction in the classifiers confidence in assigning class labels in the presence of potentially incorrect training labels. The misclassification threshold of 0.5 is denoted by the red line.

Using the standard inference approach as a benchmark, the proposed robust inference framework is applied on three of the patient data sets along with the entire combination of WSI. Each test set remains untouched and not contaminated with noise in order to allow for the approaches to be validated. For an increasing level of noise the output of each classifier using both inference approaches are compared. It is worth noting that the aim of this exercise is not to optimize the classifier performance but to illustrate the potential advantages of utilizing the proposed approach in the presence of label noise. The label noise for this study is uniformly distributed across both class with each selected label 'flipped' with equal probability. Unknown to the classifier, a predefined number of observation labels were changed for the noise generation process. Figure 7.6 presents the competing FN rates between the robust (blue) and standard inference frameworks using logistic regression along with BUS_+. In the case of patient 2 (a), patient 3 (b) and the entire combined patient database (d), for an increasing proportion of noisy training labels, the introduction of the robust approach has resulted in lower false negative rates. For patient 4 (c), the change in false negative rates

is negligible. Similar to the experimental set up in Section 7.1.1, all values were taken as averages from a large number of simulations each using 5 fold cross validation. The sample standard deviations of the values were all of the order 10^{-1}.

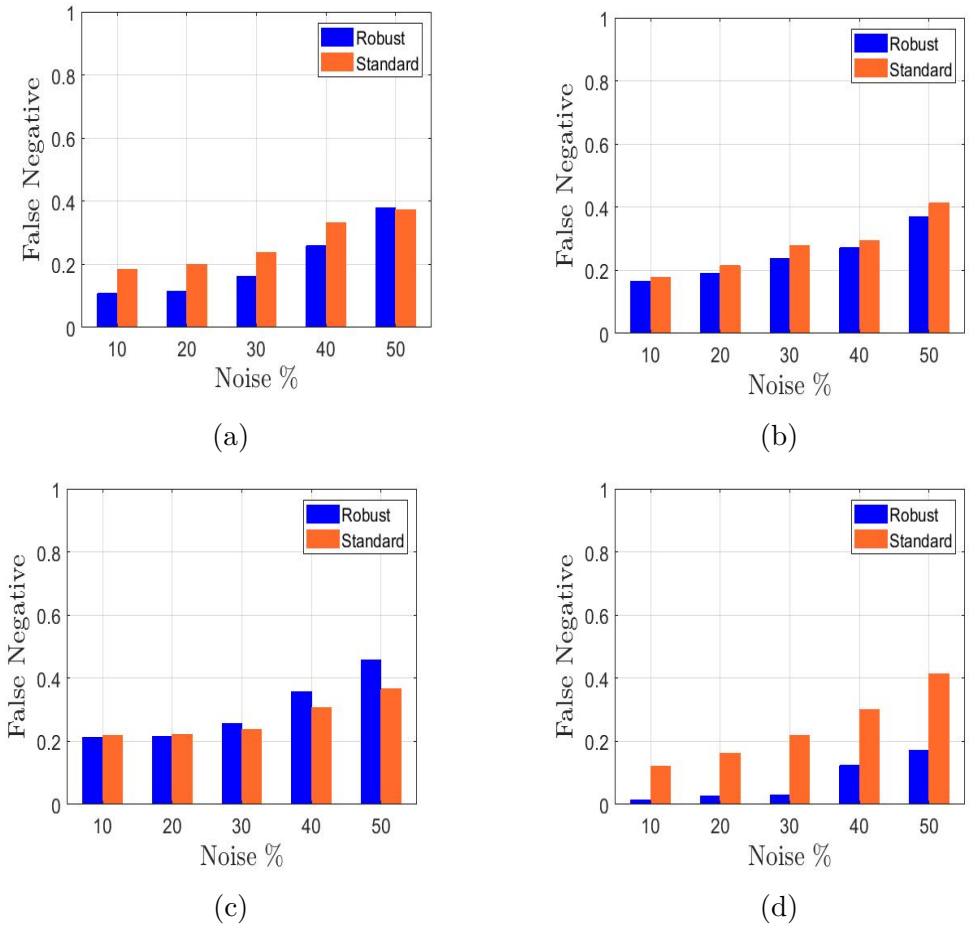


Figure 7.6: False negative rates for increasing levels of incorrect training labels for patient 2 (a), patient 3 (b), patient (4) (c) and combined patient data sets (d).

In summary, the introduction of the flipping probabilities results in label uncertainty being incorporated into the model. Altering the relationship between the data and model through a revision of the likelihood function influences the resultant model parameter values learned during the training phase. The framework offers flexibility in terms of implementation alongside different inference schemes, while the choice of prior distribution and likelihood function for the flipping probabilities ensures the added computational cost to simulation is negligible. Given the importance of modelling label uncertainty in applications such

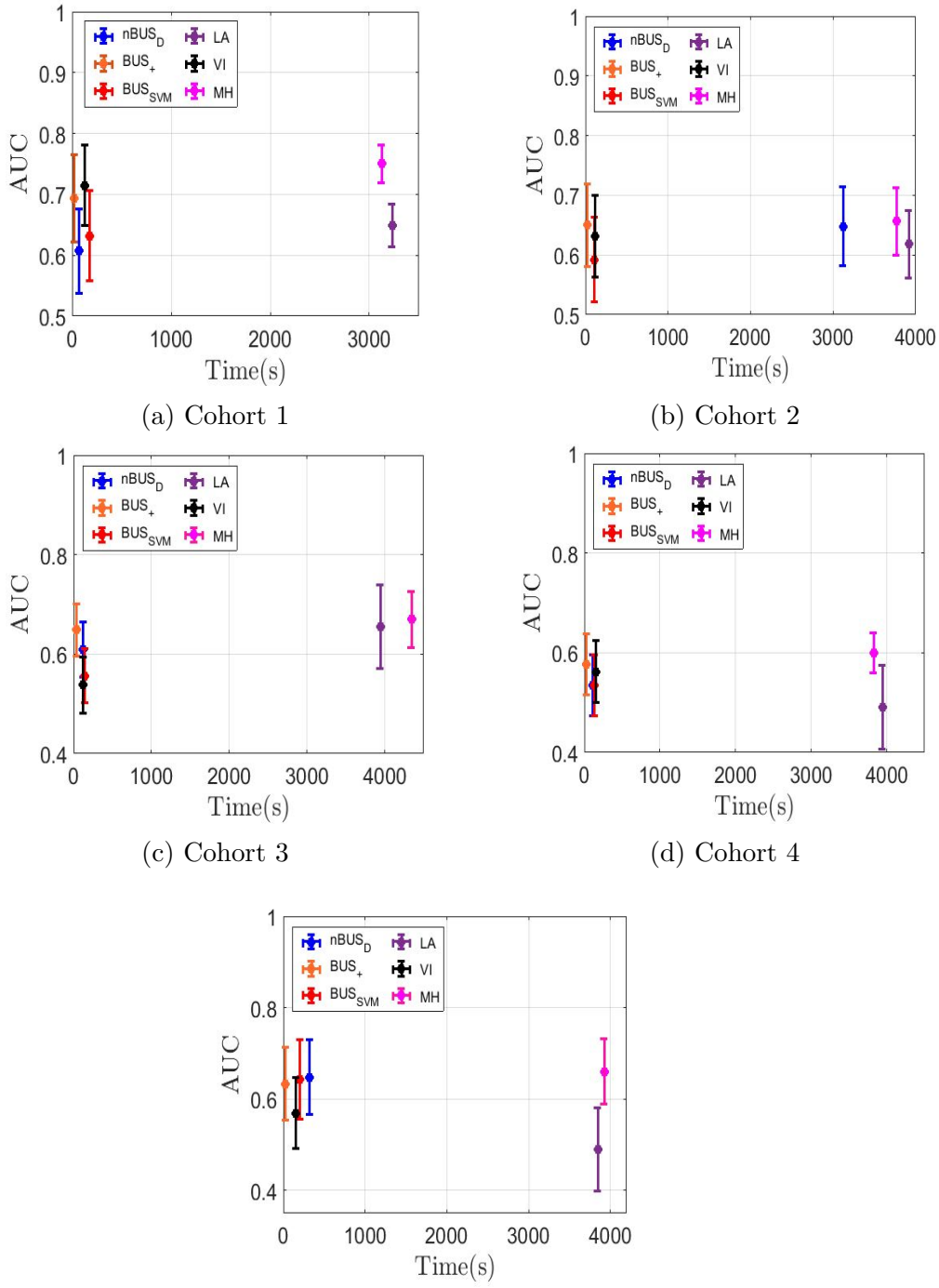
as WSI detection, we acknowledge work is required to allow for a more useful robust approach to be formed. In particular the use of Eq. 7.3 for a full Bayesian treatment of noise in the labelled data. The details are outlined in Chapter 8

7.2 Breast Cancer Biomarker Identification

Aside from the identification of breast cancer through the analysis of WSI, another on-going line of research at IBM Research Zürich is the investigation of the possible use of biomarkers for establishing cancer relapse rates. The identification of molecular biomarkers play an important role in clinical genomics [60]. Stratification of patients according to their clinical prognosis is a desirable goal in cancer treatment in order to achieve better personalized medicine. Modern technologies such as DNA microarrays measure thousands of gene expression profiles at a time, allowing for the potential of using this information for the identification of patterns in gene activity that might provide criteria for individual risk assessment in cancer patients. While reliable predictions on the basis of gene signatures could support doctors on selecting the right therapeutic strategy, biomarker discovery poses a great challenge in bioinformatics due to the very high dimensionality of the data combined with a typically small number of observations. For this purpose, the BUS frameworks are implemented on 5 separate microarray gene expression cohorts which concern the relapse rates in breast cancer patients. The classification task considered is the identification of gene signatures which constitute a relapse within 5 years of initial diagnosis. Each of the data sets taken from [144], are in 12442 dimensions with the largest number of observations in a given data set being 286. Having identified the suitability of BUS_{SVM} to data sets with a large number of observations, the purpose of this exercise is to identify which of the methods is best suited to sparse data.

Each BUS approach is compared with LA, VI and MH for the given data sets with the experimental set up the same as in Section 7.1.1. Using MH as a benchmark, Figure 7.7 presents a comparison of the methods in terms of the AUC and running time. As expected, for 4 of the 5 data sets, MH requires the largest computational budget. The limitations of LA can again be seen through the requirement of inverting a Hessian matrix which for these datasets is extremely expensive to compute. Therefore in the likely event of having limited computation time for a given task, LA is not a suitable method for such problems.

The performance of VI is comparable to that of the BUS methods in terms of predictive accuracy. However, for all datasets BUS_+ converges quicker than VI with a reduction in running time of 81% for Figure 7.7 (e). Interestingly, the performance between $nBUS_D$ and BUS_+ differs greatly in terms of running time. For all examples considered, BUS_+ terminated simulation in the shortest amount of time. In Figure 7.7 (b), the convergence of the probability of generating a sample from outside the posterior set being below 10^{-8} resulted in a large number of intermediate levels for $nBUS_D$. In terms of BUS_{SVM}, for the examples considered it is apparent that the trade off between the computational savings of the true model and the computational costs of training a surrogate are prevalent. In the case of Figure 7.7 (e), using BUS_+ over BUS_{SVM} reduces the running time by 85%. This may indicate that BUS_{SVM} may be best suited to problems which are moderately high dimensional and contain many data observations as considered in Section 7.1.1. The standard errors of all methods considered are all less than $2 \cdot 10^{-1}$.



(a) Cohort 1

(b) Cohort 2

(c) Cohort 3

(d) Cohort 4

(e) Cohort 5

Figure 7.7: Comparison of inference frameworks in terms of predictive accuracy and wall clock time for each of the biomarker datasets. The standard errors of the AUC are given by the vertical error bars.

7.2.1 Gaussian Process Classification: Model Evidence Estimation

As outlined in Chapter 4 and 5, aside from predictive capabilities through parameter estimation, *BUS* also provides estimates for the model evidence. To investigate the accuracy of the *BUS* methods for model estimation on high dimensional real life data, the Gaussian process classification model discussed in Chapter 2 is applied to the biomarker data set. The reason for doing is twofold. Firstly, to compare the respective model evidence estimations made by different inference approaches in terms of hyperparameter selection for the Gaussian process covariance function. Secondly, to allow for a non-linear decision boundary.

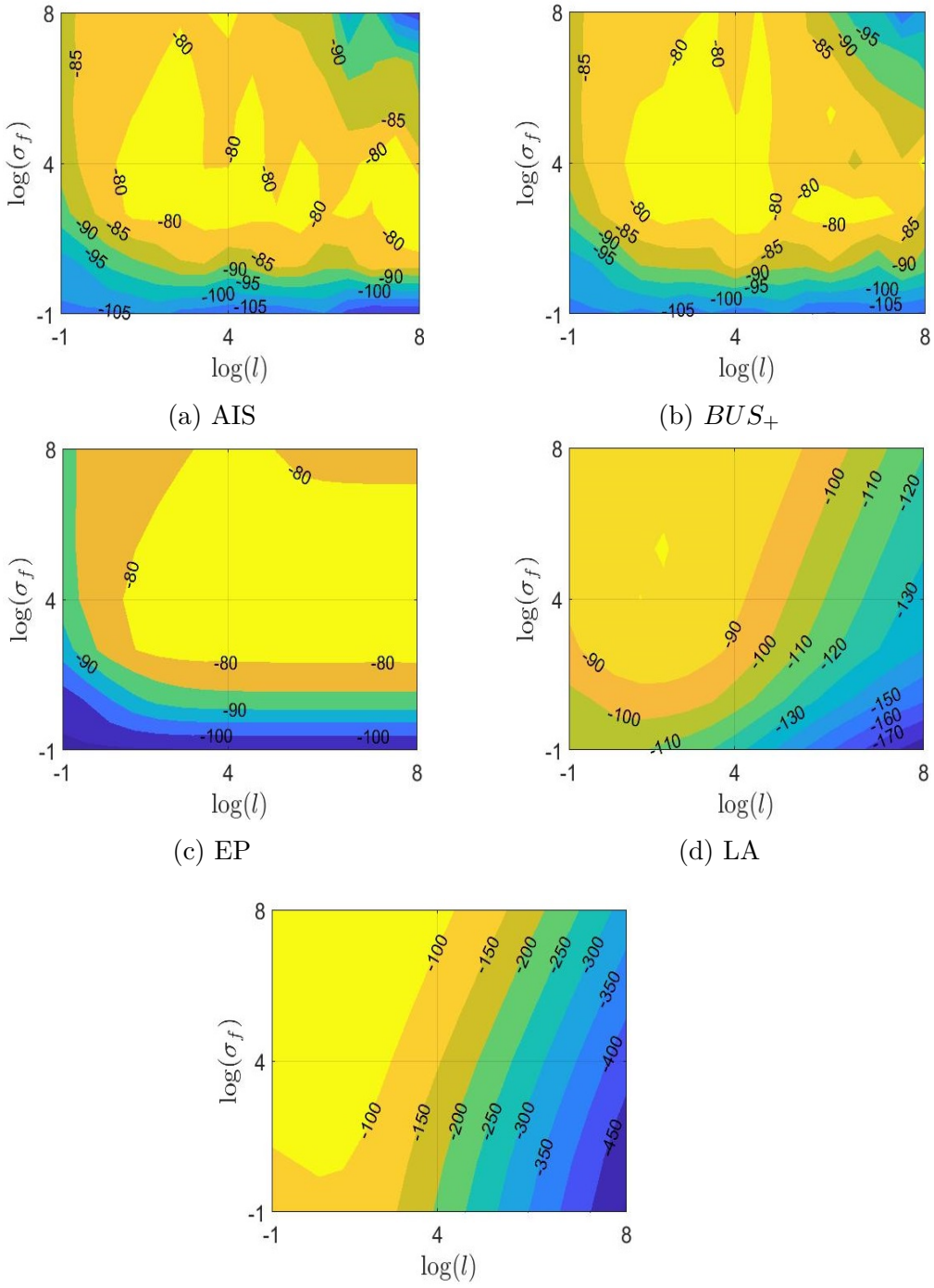
Finding the optimal hyperparameter values for the covariance function under a Bayesian approach may entail minimizing an objective function such as classification accuracy. For the purpose of this study the identification of the optimal hyperparameter values is not of interest but the estimation of the model evidence for each inference framework with respect to changes in the hyperparameters. Given that MH does not provide an estimate for the model evidence, in line with [132] and [165], Annealed Importance Sampling (AIS) [162] is used as a benchmark. For this study, the chosen covariance function is the squared exponential

$$k(x, x^{'}) = \sigma_f^2 exp\left(-\frac{1}{2l^2}||x - x^{'}||^2 \right) \tag{7.10}$$

with σ_f denoting the signal variance and l the length scale parameter of the covariance. Note that for many classification tasks it may be reasonable to use an individual length scale parameter for every input dimension, for example an automatic relevance determination (ARD) [183] function. However, for the sake of presentability the above covariance function is used. For this example the chosen likelihood function is that of the cumulative Gaussian distribution also termed the probit model. The Gaussian process was chosen for this example due to the low number of observations in the data sets. Similar to non-linear SVM models, the Gaussian process can suffer in terms of computational expense in the presence of a large number of observations when computing the covariance function. The computation involves inverting a matrix as large as the number of training points and as the number of these grow, the problem becomes more and more expensive. Sparse extensions of the Gaussian process have been developed to allow for this

issue to be solved, however they are not addressed in this study [183, 135] . For this reason, it was not implemented on the WSI data set.

An exhaustive investigation on a 16×16 grid of values for the log hyperparameters is carried out. The log is introduced to allow for a more peaked distribution to aid visual comparisons. For each hyperparameter combination, the approximated log evidence by AIS , BUS_+, EP, LA and VI is computed. The method of Expectation Propagation (EP) [151] has been highlighted to perform comparably well to AIS for Gaussian process models [132] and as such is also introduced for performance comparison. Given the optimal performance of BUS_+ over $nBUS_D$ and BUS_{SVM} on the biomarker data set, the latter methods are omitted from this study. Figure 7.8 presents the estimated log evidences for the first dataset of the biomarker study. Using panel (a) (AIS) as the benchmark it is apparent that the largest log evidences for this example have stemmed from hyperparameter values roughly in the intervals $2 \leq \log(\sigma_f) \leq 7$ and $2 \leq \log(l) \leq 7$. It is visually apparent that BUS_+ (b) shows a general agreement with AIS, with a slight variation in values towards the highest point of the distribution. Both LA (d) and VI (e) produce extremely skewed approximations and underestimate the model evidence.



(a) AIS (b) BUS_+

(c) EP (d) LA

(e) VI

Figure 7.8: Comparison of the log evidence for AIS (a), BUS_+ (b), EP (c), LA (d) and VI (e) using the first data set from the breast cancer biomarker problem.

In terms of the fifth biomarker dataset, Figure 7.9 reveals a flatter distribution of values in comparison to Figure 7.8. As before, BUS_+ (b) follows the contours

of the AIS (a) estimates closely while again EP (c) proves more accurate than LA (d) and VI (e). Again both LA and VI produce a skewed distribution with the model evidence being underestimated. This is in line with results in [165] and [132] who acknowledge the same shortcomings with LA. In the case of BUS_+ and AIS the optimal hyperparameter values in terms of evidence supporting the observed data were found to belong on the intervals $4 \leq \log(\sigma_f) \leq 5$ and $4 \leq \log(l) \leq 6$. From the visual inspection of the model evidence with varying hyperparameter values, its is apparent that BUS_+ produces a similar level of evidence in support of the respective model with respect to AIS. Each subsequent optimal hyperparameter pairing which produced both the highest and lowest model evidence values were identified and used for making predictions. As expected, all inference methods resulted in a classifier of random guessing with respect to poorly selected hyperparameters. In terms of the optimal paring, where optimal in this case refers to the best performing pairing of the observed values, all methods resulted in AUC values greater than 0.90 and standard errors of the order of 10^{-1}. This is a much better performance than the logistic regression model as is discussed in Section 7.2.

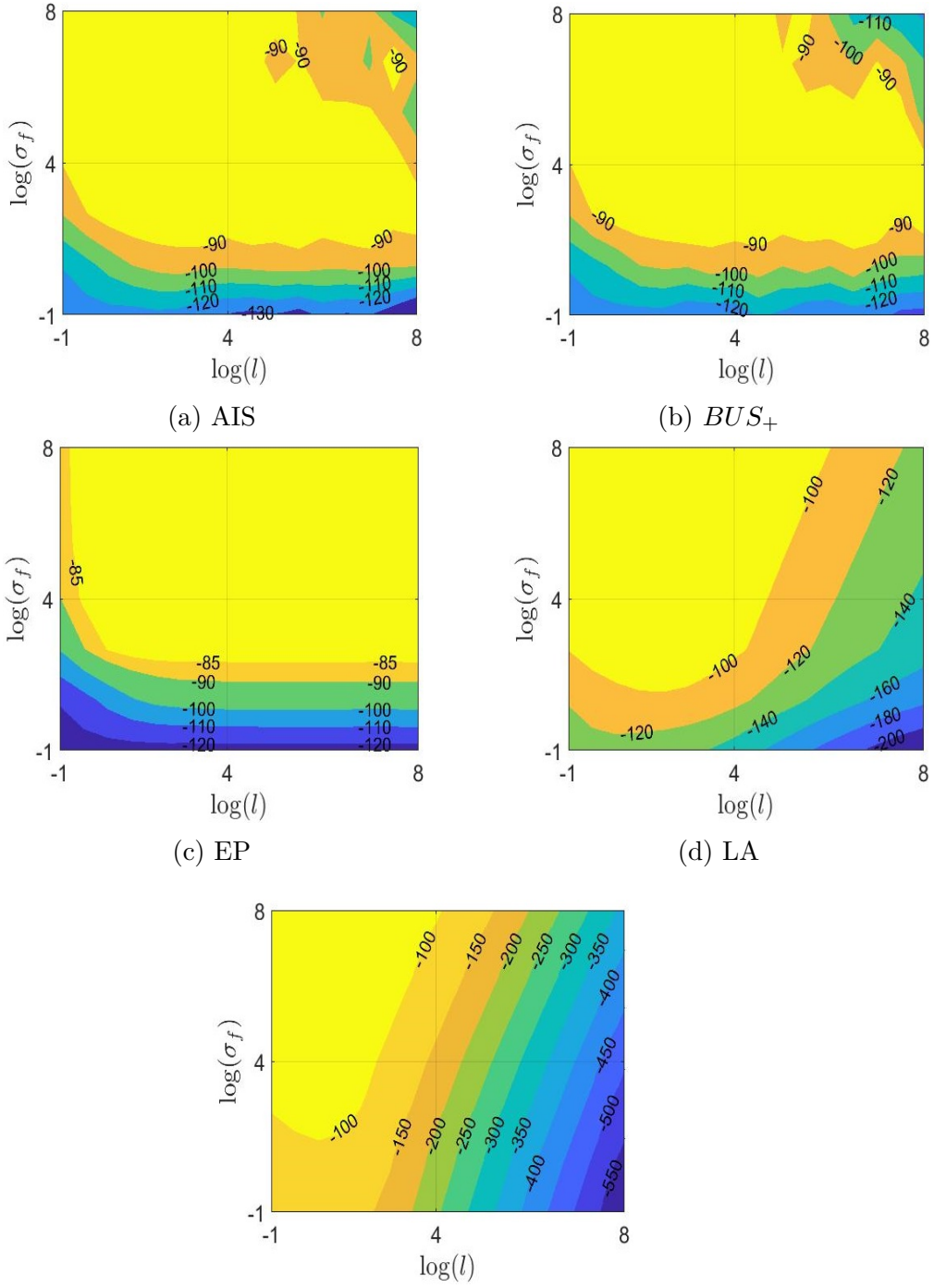
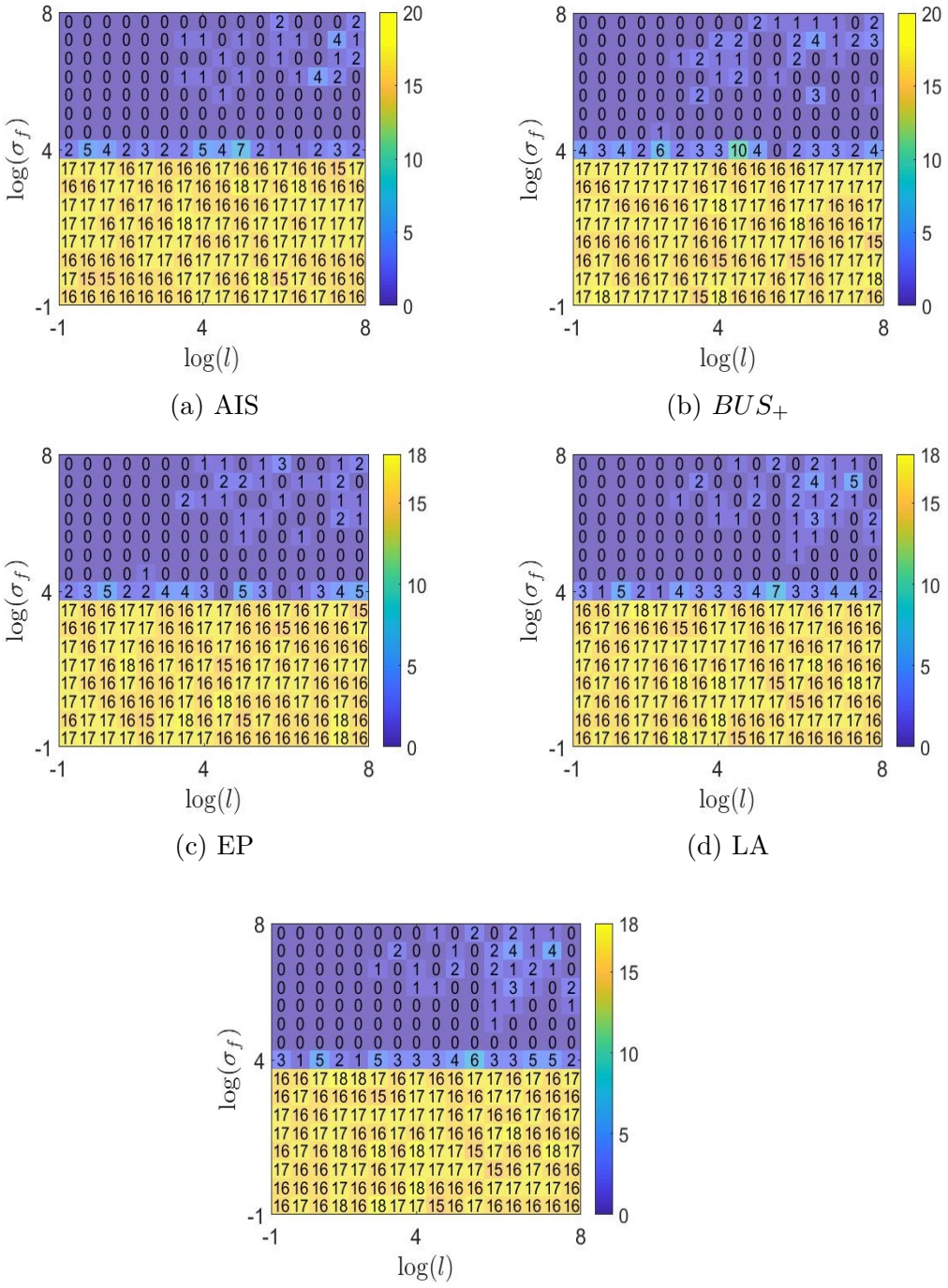


Figure 7.9: Comparison of the log evidence for AIS (a), BUS_+ (b), EP (c), LA (d) and VI (e) using the fifth data set from the biomarker problem.

As previously discussed, in terms of the classification task, the minimization of FN is of great importance due to the potential ramifications for the patient.

To investigate the sensitivity of this measure to the choice of hyperparameters, Figures 7.10 and 7.11 present the number of false negatives for each combination of hyperparameters. The results are presented in this format following work carried out in [183]. Despite the variation in model evidence estimation between inference methods, in all cases the same regions of the hyperparameter space result in low and high false negative rates. With FN rates reaching maxima of approximately 57% in Figure 7.10 and 61% in Figure 7.10. In the case of Figure 7.11, the hyperparameter intervals highlighted previously for producing the highest model evidence values also correspond to the smallest number of FN for BUS_+ and AIS. For all inference approaches, the sensitivity of the FN rate to the selection of $\log(\sigma_f)$ is evident through the drastic increase in rates for $-1 \leq \log(\sigma_f) \leq 4$. Particularly for $3.5 \leq \log(\sigma_f) \leq 4$, where FN rates decrease by 33% and 48% for each data set. Whereas, the predictive capabilities of the model appear invariant to the choice of $\log(l)$ with respect to $\log(\sigma_f)$ for the chosen examples.



(a) AIS

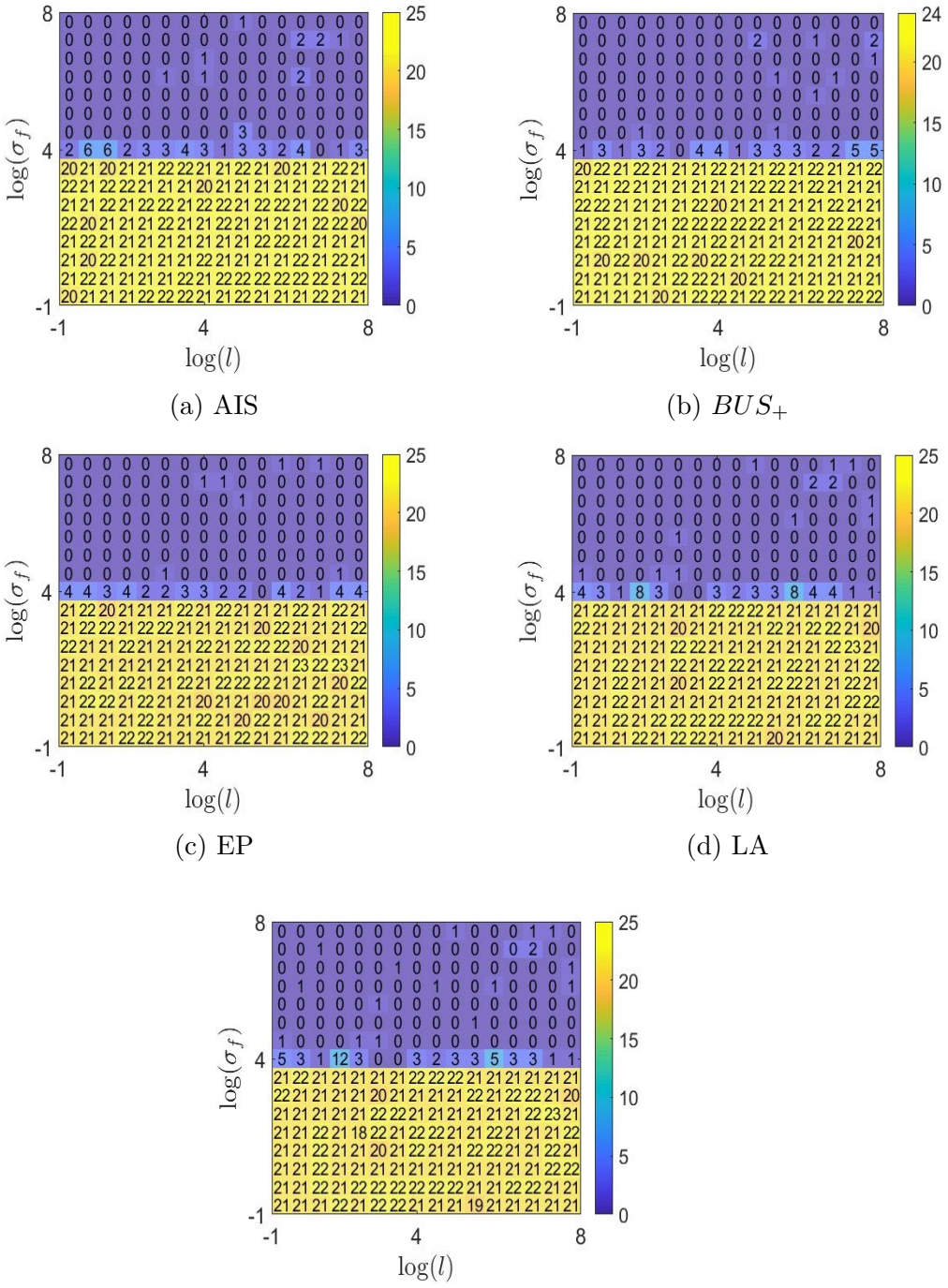
(b) BUS_+

(c) EP

(d) LA

(e) VI

Figure 7.10: Comparison of the number of false negatives for each combination of hyperparameters for AIS (a), BUS_+ (b), EP (c), LA (d) and VI (e) using the first data set from the biomarker problem



(a) AIS

(b) BUS_+

(c) EP

(d) LA

(e) VI

Figure 7.11: Comparison of the number of false negatives for each combination of hyperparameters for AIS (a), BUS_+ (b), EP (c), LA (d) and VI (e) using the fifth data set from the biomarker problem.

7.3 Chapter Summary

Motivated by the potential of machine learning techniques in the area of breast cancer detection, this chapter has presented in-depth applications of the methods developed during this dissertation. Two cancer detection tasks were considered, with both data sets differing greatly in dimensionality and number of observations. The first problem concerns WSI of breast tissue samples. The analysis of tissue biopsies has traditionally been done through visual inspection with a microscope. In its current form however, this method is inefficient in terms of diagnosis turn around time and suffers from variations in the diagnosis itself. Regarding the latter, two forms of variability can occur among pathologists which result in uncertainty appearing in the suggested diagnosis. To illustrate the potential of probabilistic classifiers in offering a degree of confidence to biopsy diagnosis, the BUS frameworks are implemented in conjunction with logistic regression using data from ten different patients. Each of BUS_+, $nBUS_D$ and BUS_{SVM} are compared in terms of predictive performance, computational expense and against other widely applied inference approaches. Results highlight the suitability of BUS_{SVM} over BUS_+ and $nBUS_D$ to problems containing a large number of observations.

Aside from uncertainty in the model parameters and predictive output, the case incorporating uncertainty of the observation class labels is also considered. Given the aforementioned potential variation between pathologists recommended diagnosis, the possibility of incorrectly labelled data arises. Inspired by this, a probabilistic approach which incorporates uncertainty into the parameter estimations is discussed. To protect against the rate of deterioration in classifier performance for an increasing number of incorrect labels, the relationship between the model parameters and observed data is altered resulting in a more robust decision making process. In its current form however, only probabilities of incorrect labels for the entire training set are provided. The authors acknowledge that future work is required to ensure a realistic application in the area of cancer diagnosis.

The second classification task studied, concerns the identification of five year breast cancer relapse rates based on molecular biomarkers. Biomarker discovery poses a great challenge in bioinformatics due to the very high dimensionality of the data combined with a typically small number of observations. In practice, a Bayesian inference method has to satisfy a wide range of requirements. From the

experimental results, if runtime and predictive accuracy are a major concerns, then VI and BUS_+ should be considered. Gaussian process classifiers were introduced to investigate the estimation of the model evidence of each inference method. By varying the values of the covariance function hyperparameters, a distribution of the estimated model evidences is generated. If model evidence estimation is of concern, then BUS_+ has been shown to perform comparably well to AIS, while both LA and VI appear to underestimate this quantity. In terms of false negative rates, the importance of choosing appropriate hyperparameter values has also been highlighted.

Chapter 8

Summary and Conclusions

The work carried out in this dissertation was motivated by the requirement of supervised machine learning algorithms to have unknown parameters which are representative of the data generation process as inputs. The importance of this can be seen in the surge of interest in supervised machine learning as a viable solution to difficult real world problems. In reality, tasks involve different forms of data, some contain many observations while others expressed in very high dimensions. As such, the computational efficiency of the modelling process is paramount to ensure that supervised machine learning is a realistic aid for tasks which are otherwise done through human interaction. Efficient Bayesian inference methods suitable for identifying unknown parameters have been proposed. Moreover, a probabilistic framework has been discussed which accounts for incorrectly labelled data which directly influences the unknown parameter values. This chapter provides an overview of the developed ideas, main findings, potential future research avenues and a summary of publications and awards stemming from this work.

8.1 Summary of Completed Work

Chapter 2 provided an overview of machine learning with a particular emphasis placed on supervised learning. Supervised learning extracts information from a training dataset, in order to infer a function which maps the inputs to the desired outputs. The form of output differs in terms of being discrete or continuous with classification a representation of the former and regression the latter. Focusing on the task of classification, the influence of poor parameter estimation on model accuracy was highlighted.

In Chapter 3, the concept of Bayesian inference as a tool for measuring uncertainties in model parameters was introduced. While providing a solid mathematical framework for updating probabilities which are representative of the parameter uncertainties, a conditional probability distribution referred to as the posterior is required to be computed. The posterior is a powerful distribution which allows for unknown parameters to be identified, competing models to be compared against one another based on the underlying data and predictions to be made. Due to the complex mathematical formulation of the posterior, sampling methods like rejection sampling and Markov Chain Monte Carlo (MCMC) have been used for this computation. The advantages and shortcomings of both methods were discussed.

In Chapter 4, a framework stemming from an area of engineering referred to as reliability analysis, was presented for sampling from the posterior distribution. Through the use of an advanced MCMC method termed Subset Simulation (SuS), Bayesian Updating using Structural reliability methods (BUS) offers a robust solution which is suitable to high dimensional tasks and can handle multi-modal distributions. BUS takes advantage of the low sample acceptance rates in rejection sampling to interpret the Bayesian updating task as a rare event. Like rejection sampling, BUS requires the prudent choice of a an input parameter prior to simulation which dictates both sampling efficiency and the distribution of the generated samples. A reformulation termed nested BUS ($nBUS$) relaxes this constraint by learning the parameter automatically during simulation. Additionally, automatic stopping is performed to protect against a deterioration in sample quality. A numerical investigation using engineering and classification examples reveal the added computational cost associated with $nBUS$.

To address this issue, two new stopping criteria which greatly reduce the number of Total Model Evaluations (TME) during a model run while offering statistical guarantees of convergence to the posterior distribution are proposed in Chapter 5. The first criterion, allows for the direct computation of the rejection principle in regions of the sample space far away from the target domain before computing the probability of generating a sample from outside the posterior set once the posterior has been judged to have been reached. In turn, avoiding extra calls of the likelihood function at every intermediate level. The resulting method is named $nBUS_D$. The second condition exploits a transition in the

relationship between the probability of failure and model evidence to identify when to terminate simulation. A distribution is generated for the probability of failure to allow for the statistical variation stemming from the stochastic nature of the model evidence. Once all evaluations of the model evidence belong in a specified interval the algorithm stops. The resulting model is named BUS_+. Computational savings over $nBUS$ along with the proposed methods ability to maintain sample quality are shown using several benchmark problems.

The emergence of large data sets has reduced the suitability of MCMC based frameworks to such Bayesian inference tasks. In Chapter 6, Support Vector Machines (SVM) are incorporated into the BUS framework to reduce the number of likelihood function evaluations required during a model run. Modifications for an existing framework named 2SMART which has been developed for reliability analysis tasks have been proposed while a novel BUS_{SVM} method is presented. The ability of which to maintain sampling accuracy while requiring fewer TME is illustrated on problems of complex topology.

To identify the suitability of the developed methods to different data set types, real life data from problems concerning breast cancer detection are addressed in Chapter 7. The importance of correctly labelled data is highlighted resulting in the formulation of a probabilistic approach for incorporating uncertainty surrounding the class labels into the posterior samples. All methods are compared against widely applied Bayesian inference techniques using a number of different classification models.

8.2 Summary of Contributions

- **Observations of *nBUS*.** As discussed in Chapter 4, the correct choice of stopping tolerance for the nested loop stopping condition is vital. If chosen too large, simulation can be terminated before the posterior distribution is reached. While a value too small, may result in added computational expense. Additionally, adaptive Conditional Sampling (ACS) is determined to be the preferred MCMC scheme. This is due to its ability to automatically tune the standard deviation of the proposal distribution.

- **New Stopping Criteria.** To alleviate the computational cost associated with $nBUS$, two methods referred to as $nBUS_D$ and BUS_+ are proposed in Chapter 5. In both cases the requirement to compute additional like-

lihood function calls at every level is avoided while statistical guarantees of termination are offered. From the numerical experiments in Chapter 7, in terms of computational expense both methods are best suited to very high dimensional examples with small numbers of data observations. It is revealed that BUS_+ is the method of choice as $nBUS_D$ may require additional intermediate levels due to satisfying the chosen tolerance of sampling error.

- **Scalability for Large Data Sets.** Through the introduction of Support Vector Machines (SVM), the added computational expense of using MCMC based methods on large datasets is addressed in Chapter 6. Firstly, modifications are proposed for the Subset simulation by Support vector Margin Algorithm for Reliability esTimation (2SMART) method to improve algorithmic stability and sampling accuracy for Bayesian inference tasks. Secondly, a new SVM inspired method named BUS_{SVM} is presented. From the numerical experiments in Chapter 7, in terms of computational expense BUS_{SVM} is identified as being best suited to data sets containing a large number of observations.

- **Dealing with Incorrectly Labelled Data**. A preliminary probabilistic approach which introduces uncertainty surrounding the class labels of the data set is discussed in Chapter 7. The framework updates the relationship between the data and unknown parameters to allow for the posterior samples to be informed about the possibility of incorrect labels being present in the data.

- **Potential of Machine Learning in Breast Cancer Detection**. An in depth analysis of Whole Slide Image (WSI) and biomarker identification for breast cancer detection is presented in Chapter 7. Aside from illustrating the capabilities of the frameworks developed in this dissertation, the potential of models which are probabilistic in there output for addressing the uncertainty in predictions done by traditional means is also highlighted. An exhaustive investigation of model evidence estimation in terms of hyperparameter choice for Gaussian process classification models reveals the ability of BUS_+ in estimation accuracy with respect to MCMC.

8.3 Research Outlook

Several research directions can be followed from the work presented in this dissertation. In the case of BUS, a potential third stopping condition could be formulated based on the computation of the gradient of the log evidence. As this quantity becomes constant once the minimum required level has been surpassed, identifying when the slope is zero would ensure termination of simulation. Given the stochastic nature of the log evidence, methods from the area of stochastic optimization [184] would be required to compute the respective approximation.

In terms of the post processor of SuS, relaxing the independent assumption between the product of beta variables would in theory allow for a more accurate representation of the uncertainty surrounding the probability of failure estimate. This may first involve sorting out the potential dependency between the seeds at each intermediate level. This could potentially allow for the sampling ratio accounting for sampling error in the credible interval of BUS_+ to be relaxed.

Improvements to sample quality will primarily stem from the chosen MCMC scheme. The possibility of further reducing correlation between generated samples or developing an analogous framework which efficiently produces independent samples from the target posterior distribution would be advantageous. With respect to algorithmic efficiency, alternative approaches for reducing the computational cost of the likelihood function for large datasets would allow BUS to be further scalable to such problems. The incorporation of methods existing within the area of stochastic MCMC [128, 15] may provide the basis for future research. The highly parallelisable nature of sequential monte carlo (SMC) [63] methods, if integrated into BUS may also allow for savings in terms of wall clock time.

It is worth noting that BUS with SuS has some similarities with nested sampling [204] discussed in Chapter 3. Notably, in both algorithms sampling is performed by means of nested subsets, starting from the prior distribution. Nested sampling focuses on shrinking intermediate subsets of the prior domain that are nested. Unlike BUS, nested sampling does not augment the parameter space and directly operates in the space spanned by the uncertain parameter vector $\boldsymbol{\theta}$ while the manner in which they each estimate the model evidence also differs. Future work could possibly include a combination of the BUS and nested sampler but is beyond the scope of this thesis.

A number of extensions to BUS_{SVM} could be made. Firstly, aside from relying on the class labels to guide the SVM for learning in disjoint areas of the sample

space, the identification of individual clusters and subsequent learning of an SVM on each cluster could improve localised surrogate quality. Secondly, an investigation into the suitability of relevance vector machines [216] (RVM) to replace SVM would allow the uncertainty in the sample acceptance step to be quantified. Thirdly, automatic choice of parameter which dictates the surrogates training set size.

The issue of mislabelled training data is clearly a very exciting area of research in terms of application to cancer diagnosis. A full Bayesian framework, which learns a local flipping probability parameter during simulation for each data observation would improve the practicality of the proposed method. By having the ability to compute local flipping estimates instead of a single global estimate, potentially mislabelled WSI could be re-analysed by the pathologist.

8.4 Published Work

The following literature and awards were generated from the present dissertation.

8.4.1 Conference Papers

1. P.G.Byrnes, F.A.DiazDelaO, *Bayesian Updating for Probabilistic Classification Using Reliability Methods*, Proceedings of the 2^{nd} International Conference on Uncertainty Quantification in Computational Sciences and Engineering, UNCECOMP, Rhodes Island, Greece, June $15^{th} - 17^{th}$, 2017.

2. P.G.Byrnes, F.A.DiazDelaO, *Reliability Based Bayesian Inference for Probabilistic Classification: An Overview of Sampling Schemes*, Proceedings of the 37^{th} International Conference on Innovative Techniques and applications of Artificial Intelligence, Cambridge, UK, December $12^{th} - 14^{th}$, 2017.

3. P.G.Byrnes, F.A.DiazDelaO, *Efficient Bayesian Inference by Reliability Methods: Applications in Supervised Machine Learning*, Proceedings of the International Conference on Pattern Recognition and Artificial Intelligence, PRAI, New York, USA, August $15^{th} - 17^{th}$, 2018.

4. P.G.Byrnes, F.A.DiazDelaO, *Kernel Logistic Regression: A Robust Weighting for Imbalanced Classes with Noisy Labels*, Proceedings of the 4^{th} International

Conference on Machine Learning and Data Engineering, iCMLDE, Sydney, Australia, December $3^{rd} - 7^{th}$, 2018.

8.4.2 Awards

1. Awarded " University of Georgia, USA /University of Liverpool, UK, Doctoral Student Short-Term International Research Fellowship" April 2018. Hosted by Dr. Roberto Perdisci at the Department of Computer Science.

2. "Runner Up Best Student Paper Award" for *Kernel Logistic Regression: A Robust Weighting for Imbalanced Classes with Noisy Labels*, Proceedings of the 4^{th} International Conference on Machine Learning and Data Engineering, iCMLDE, Sydney, Australia, December $3^{rd} - 7^{th}$, 2018.

8.5 Work Under Review

1. P.G.Byrnes, F.A.DiazDelaO, *Stopping Criteria for Bayesian Inference by Reliability Methods*, Under Review in *Computer Methods in Applied Mechanics and Engineering* .

2. P.G.Byrnes, F.A.DiazDelaO, *Scalable Bayesian Inference by Support Vector Machines*, In Preparation.

Appendix A

A.1 Logistic Regression Log-Likelihood

For the derivations in the Appendix allow for some of the notation used during this dissertation to be reintroduced for the convenience of the reader. Let $D = \{(x_i, y_i)\}_{i=1}^{n}$ denote the observed data set comprising of observations x and class labels y. Let $\boldsymbol{\theta}$ denote the unknown model parameters to be inferred and p_i the probability of observation i belonging to class 1. Let the log likelihood of the logistic regression model be given by

$$\log(L(D|\boldsymbol{\theta})) = \sum_{i=1}^{n}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)] \tag{A.1}$$

Reintroducing the expression for class membership in terms of the sigmoid function yields

$$\begin{aligned}
\log(L(D|\boldsymbol{\theta})) &= \sum_{i=1}^{n}\left[y_i \log\left(\frac{1}{1+e^{-x_i^T\boldsymbol{\theta}}}\right) + (1-y_i)\log\left(\frac{e^{-x_i^T\boldsymbol{\theta}}}{1+e^{-x_i^T\boldsymbol{\theta}}}\right)\right] \\
&= \sum_{i=1}^{n}\left[\log\left(\frac{e^{-x_i^T\boldsymbol{\theta}}}{1+e^{-x_i^T\boldsymbol{\theta}}}\right) + y_i\left[\log\left(\frac{1}{1+e^{-x_i^T\boldsymbol{\theta}}}\right) - \log\left(\frac{e^{-x_i^T\boldsymbol{\theta}}}{1+e^{-x_i^T\boldsymbol{\theta}}}\right)\right]\right] \\
&= \sum_{i=1}^{n}\left[\log\left(\frac{1}{1+e^{x_i^T\boldsymbol{\theta}}}\right) + y_i\left[\log\left(\frac{1}{1+e^{-x_i^T\boldsymbol{\theta}}}\right)\right]\right] \\
&= \sum_{i=1}^{n}[-\log(1+e^{x_i^T\boldsymbol{\theta}}) + y_i \cdot x_i^T\boldsymbol{\theta}]
\end{aligned} \tag{A.2}$$

A.2 Log-Likelihood First Derivative

For the derivation of the log likelihood taking the gradient with respect to $\boldsymbol{\theta}$ yields

$$\begin{aligned}
\frac{\partial \log(L(D|\boldsymbol{\theta}))}{\partial \boldsymbol{\theta}} &= \frac{\partial}{\partial \boldsymbol{\theta}} \sum_{i=1}^{n} [-\log(1 + e^{x_i^T \boldsymbol{\theta}}) + y_i \cdot x_i^T \boldsymbol{\theta}] \\
&= \sum_{i=1}^{n} \left[- \frac{e^{x_i^T \boldsymbol{\theta}}}{1 + e^{x_i^T \boldsymbol{\theta}}} x_i + y_i x_i \right] \\
&= \sum_{i=1}^{n} \left[x_i \left[y_i - \frac{1}{1 + e^{-x_i^T \boldsymbol{\theta}}} \right] \right] \\
&= \sum_{i=1}^{n} (x_i [y_i - p_i]) \qquad \text{(A.3)}
\end{aligned}$$

A.3 Log-Likelihood Second Derivative

For the second derivative it is required to compute the gradient of the above expression.

$$\begin{aligned}
\frac{\partial^2 \log(L(D|\boldsymbol{\theta}))}{\partial^2 \boldsymbol{\theta}} &= \frac{\partial}{\partial \boldsymbol{\theta}} \sum_{i=1}^{n} (x_i [y_i - p_i]) \\
&= - \sum_{i=1}^{n} \left(x_i \frac{\partial}{\partial \boldsymbol{\theta}} p_i \right) \\
&= - \sum_{i=1}^{n} x_i^T x_i p_i [1 - p_i] \qquad \text{(A.4)}
\end{aligned}$$

Appendix B

B.1 Influence of Likelihood Multiplier

In this section, the distribution of samples violating the rejection principle is derived. Let u be a uniform random variable with support $[0, 1]$, c be the likelihood multiplier and p_A the probability of $\boldsymbol{\theta}$ being accepted via the rejection principle. As outlined in Chapter 3, $\boldsymbol{\theta}$ and u are only returned in the event of step 2 of the rejection sampling algorithm being satisfied. By Bayes theorem let the posterior distribution of a sample being accepted by the rejection algorithm be given by

$$P(\boldsymbol{\theta}, u|cL(D|\boldsymbol{\theta}) > u) = \frac{Q(\boldsymbol{\theta})I[cL(D|\boldsymbol{\theta}) > u]}{p_A} \tag{B.1}$$

Marginalising to isolate $\boldsymbol{\theta}$ yields

$$P(\boldsymbol{\theta}|cL(D|\boldsymbol{\theta}) > u) = \frac{Q(\boldsymbol{\theta}) \int_0^1 I[cL(D|\boldsymbol{\theta}) > u]du}{p_A} = \frac{Q(\boldsymbol{\theta})cL(D|\boldsymbol{\theta})}{p_A} \tag{B.2}$$

since $\int_0^1 I(cL(D|\boldsymbol{\theta}) > u)du = cL(D|\boldsymbol{\theta})$ when $cL(D|\boldsymbol{\theta}) \leq 1$ for all $\boldsymbol{\theta}$. In the event of the rejection principle being satisfied, the generated $\boldsymbol{\theta}$ is distributed according to the target posterior distribution. Consider a set A which contains all samples violating the rejection principle for a given c.

$$A = \{\boldsymbol{\theta} \in \mathbb{R}^d : cL(D|\boldsymbol{\theta}) > 1\} \tag{B.3}$$

In the case of $\boldsymbol{\theta} \in A$, for any $\boldsymbol{\theta}$ the sample will be accepted and the indicator function will equal 1. Meaning that the distribution of $\boldsymbol{\theta}$ is given by

$$P(\boldsymbol{\theta}|cL(D|\boldsymbol{\theta}) > u) = \frac{Q(\boldsymbol{\theta}) \int_0^1 I[cL(D|\boldsymbol{\theta}) > u]du}{p_A} = \frac{Q(\boldsymbol{\theta})}{p_A} \propto Q(\boldsymbol{\theta}) \tag{B.4}$$

Therefore, for the case of a generated $\boldsymbol{\theta}$ violating the rejection principle, the distribution of the resulting sample is proportional to the prior. This highlights the influence of choosing c correctly. A value of c greater than the reciprocal of the maximum of the likelihood function causes the distribution of $\boldsymbol{\theta}$ to be proportional to the prior. While choosing an appropriate value of c ensures the rejection principle is satisfied, meaning samples are being drawn from the correct distribution. Therefore a prudent choice of likelihood multiplier is required.

B.2 nBUS Characteristic Trends Derivations

The characteristic trends discussed in Chapter 4, may act as an indication that the minimum required intermediate level has been generated by $nBUS$. The following presents derivations of the trends.

B.2.1 Probability of Failure

Firstly, consider the probability of failure p_F. Examining the functional behaviour of $\ln(p_F)$ reveals a transition which occurs for $b > b_{min}$. Let p_F be expressed in terms of the model evidence P_D and the failure threshold b.

$$p_F = e^{-b} P_D \qquad b > b_{min} \tag{B.5}$$

The introduction of the natural logarithm yields

$$\ln(p_F) = \ln(P_D) - b \tag{B.6}$$

Substituting the expression for P_D into the above

$$\begin{aligned} \ln(p_F) &= \ln\left(e^{-b} \int Q(\boldsymbol{\theta}) L(D|\boldsymbol{\theta}) d\boldsymbol{\theta}\right) + b - b \\ &= \ln(e^{-b}) + \ln\left(\int Q(\boldsymbol{\theta}) L(D|\boldsymbol{\theta}) d\boldsymbol{\theta}\right) \end{aligned} \tag{B.7}$$

Taking the gradient of $\ln(p_F)$ with respect to b reveals

$$\frac{\partial \ln(p_F)}{\partial b} = -1 \tag{B.8}$$

Therefore, for any b such that $b > b_{min}$ the log of the probability of failure exhibits a trend with a slope of -1.

B.2.2 Model Evidence

From Chapter 4, it was noted that $\ln(P_D)$ goes from a linearly increasing function as b increases through a transition stage to remaining constant equal to $\ln(P_D)$ once the samples are distributed according to the posterior. For the sake of simplicity let $J(b)$ represent $\ln(P_D)$. Firstly, in the case of an intermediate threshold not

surpassing the minimum required level, let the gradient of the log evidence for $b < b_{min}$ be given by

$$J(b) = b + \ln(p_F), \qquad b < b_{min} \tag{B.9}$$

At the left tail of the CCDF (Complementary CDF) the $p_F \approx 1$. Resulting in $\ln(p_F) = 0 + b$. Taking the gradient of $J(b)$ with respect to b

$$\frac{\partial J(b)}{\partial b} = 1 \tag{B.10}$$

For the case of $b > b_{min}$

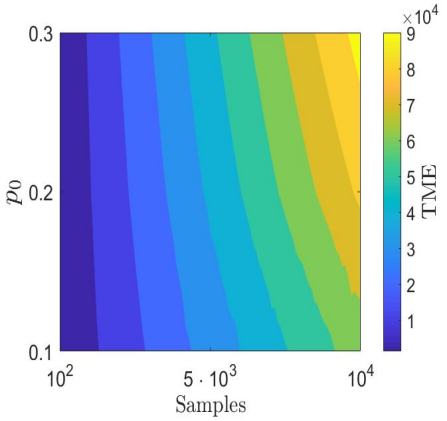
$$J(b) = b + \ln\left(e^{-b} \int Q(\boldsymbol{\theta}) L(D|\boldsymbol{\theta}) d\boldsymbol{\theta}\right) \tag{B.11}$$

Computing the gradient of $J(b)$ with respect to b results in

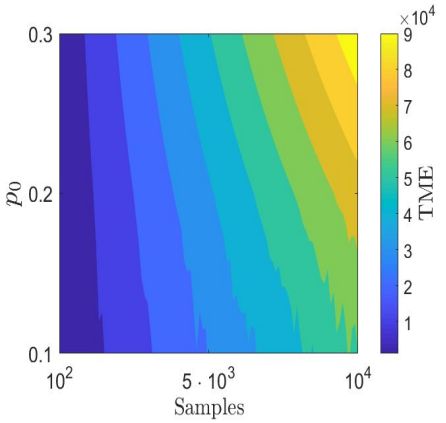
$$\begin{aligned} \frac{\partial J(b)}{\partial b} &= 1 + \frac{\partial}{\partial b}\ln(e^{-b}) + \frac{\partial}{\partial b}\ln\left(\int Q(\boldsymbol{\theta}) L(D|\boldsymbol{\theta}) d\boldsymbol{\theta}\right) \\ &= 1 + \frac{\partial}{\partial b}(-b) + 0 = 0, \qquad b > b_{min} \end{aligned} \tag{B.12}$$

Therefore, for any $b > b_{min}$, the slope of the log model evidence is 0 as the generated samples are now distributed according to the posterior while the log model evidence remains constant.

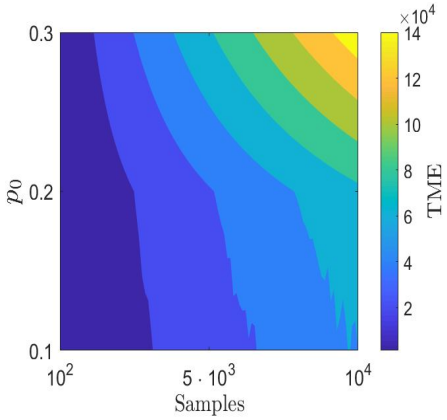
B.3 Computational Expense of Numerical Examples



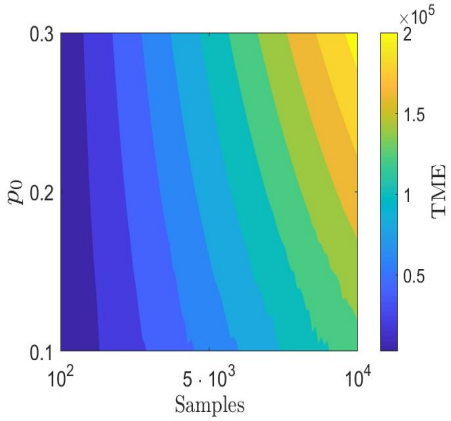
(a) Number of TME for $aBUS$.



(b) Number of TME for $nBUS_4$.

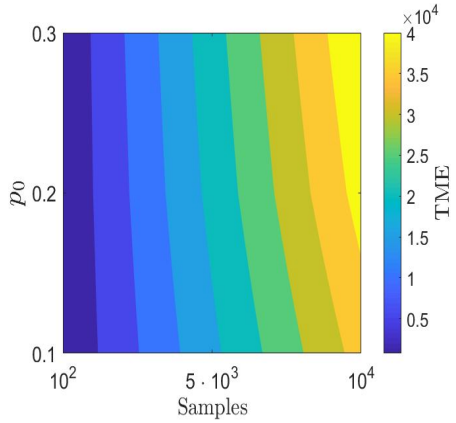


(c) Number of TME for $nBUS_6$.

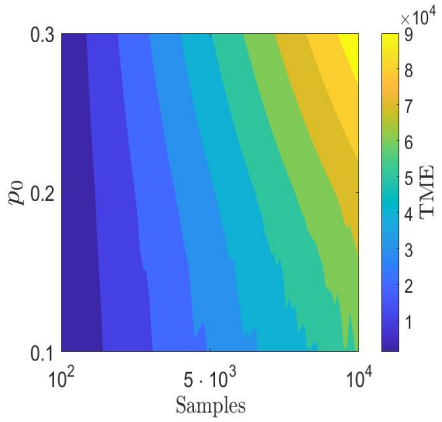


(d) Number of TME for $nBUS_8$.

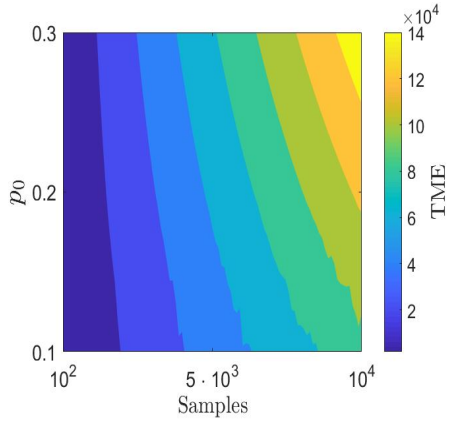
Figure B.1: Illustration of the required TME for $aBUS$, $nBUS_4$, $nBUS_6$ and $nBUS_8$ for drawing posterior samples in example 2.



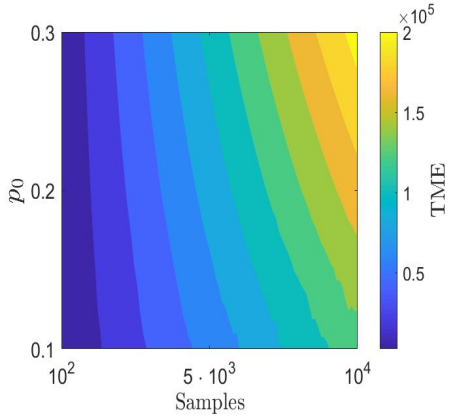
(a) Number of TME for $aBUS$.



(b) Number of TME for $nBUS_4$.

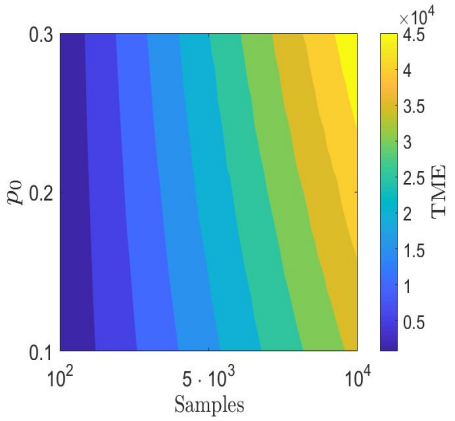


(c) Number of TME for $nBUS_6$.

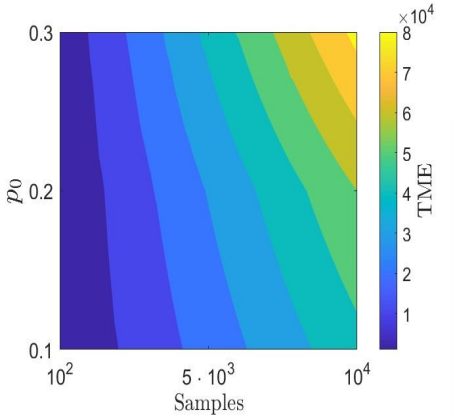


(d) Number of TME for $nBUS_8$.

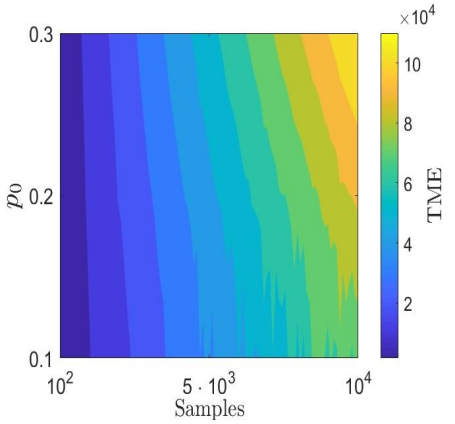
Figure B.2: Illustration of the required TME for $aBUS$, $nBUS_4$, $nBUS_6$ and $nBUS_8$ for drawing posterior samples in example 3.



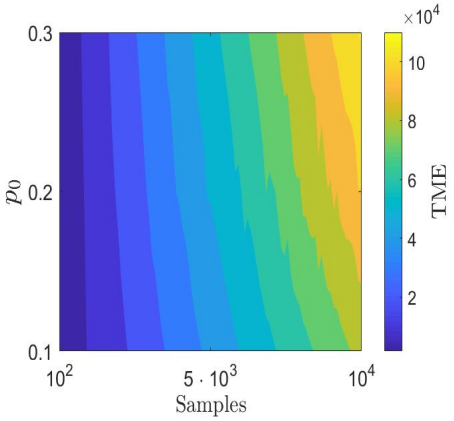
(a) Number of TME for $aBUS$.



(b) Number of TME for $nBUS_4$.



(c) Number of TME for $nBUS_6$.



(d) Number of TME for $nBUS_8$.

Figure B.3: Illustration of the required TME for $aBUS$, $nBUS_4$, $nBUS_6$ and $nBUS_8$ for drawing posterior samples in example 4.

Appendix C

C.1 Proof of Corollary 1

Let N denote the number of generated samples, p_0 the level probability and m the number of intermediate levels. Let RHS denote the right hand side of an expression. The following illustrates that for any $p_0 < 0.5$, the beta PDF of p_F becomes increasingly skewed for an increasing number of intermediate levels. Let $P_F^+ \sim Be(\alpha, \beta)$ with $\alpha > 1$ and $\beta > 1$. Then for any $p_0 < 0.5$, $\beta > \alpha$. Consider the expressions for $\beta > \alpha$ to be

$$\frac{(1-(\frac{p_0 \cdot N+1}{N+2})^m) \cdot (1-(\frac{p_0 \cdot N+2}{N+3})^m)}{(\frac{p_0 \cdot N+2}{N+3})^m - (\frac{p_0 \cdot N+1}{N+2})^m} > \frac{(\frac{p_0 \cdot N+1}{N+2})^m \cdot (1-(\frac{p_0 \cdot N+2}{N+3})^m)}{(\frac{p_0 \cdot N+2}{N+3})^m - (\frac{p_0 \cdot N+1}{N+2})^m} \tag{C.1}$$

Rearranging these expressions allows for them to be simplified to

$$1-(\frac{p_0 N+1}{N+2})^m > (\frac{p_0 N+1}{N+2})^m \tag{C.2}$$

Regarding the RHS, given $p_0 \in [0,1]$ the expression is less than 1 for all N. Consequently, satisfying this inequality is equivalent to showing

$$0.5 > (\frac{p_0 N+1}{N+2})^m \tag{C.3}$$

Given the upper bound of the RHS is 1, with respect to m the following holds:

$$(\frac{p_0 N+1}{N+2})^1 > (\frac{p_0 N+1}{N+2})^2 > ... > (\frac{p_0 N+1}{N+2})^m \tag{C.4}$$

Therefore it is enough to show that the RHS is less than 0.5 in the case of $m = 1$.

$$0.5 > \frac{p_0 N+1}{N+2} \tag{C.5}$$

Rearranging these expressions produces

$$0.5N + 1 > p_0 N + 1 \tag{C.6}$$

which holds for any $p_0 < 0.5$. Therefore for any $p_0 < 0.5$, the beta PDF becomes increasingly skewed as the intermediate levels of BUS_+ progress.

C.2 Derivations of CDF and PDF of a Log Transformed Beta Distribution

As the characteristic trend for the model evidence is in log space, transformations of the beta PDF and CDF are required. Firstly, consider the case of the CDF. Let $Y = \ln(X)$, where $X \sim B(\alpha, \beta)$ and ln the natural logarithm. The CDF is expressed as

$$F_Y(y) = P(Y < y) = P(\ln(X) < y) = P(X < e^y) \tag{C.7}$$

From the fundamental theorem of calculus, integrating the PDF of a random variable produces the subsequent CDF.

$$F_Y(y) = \int_0^{e^y} \frac{1}{B(\alpha, \beta)} X^{(\alpha-1)}(1 - X)^{(\beta-1)} dy \tag{C.8}$$

Inversely, the PDF may be derived by differentiating the CDF. Let $g^{-1}(y) = e^y$ and $|\cdot|$ denote the absolute value. The transformed PDF is given by

$$f_Y(y) = f_X(g^{-1}(y)) \mid \frac{\partial g^{-1}}{\partial y} \mid \tag{C.9}$$

Replacing the arguments of the standard beta PDF with e^y and taking the derivative of $g^{-1}(y)$ results in the following log transformation

$$\begin{aligned} f_Y(y) &= \frac{1}{B(\alpha, \beta)} e^{y(\alpha-1)}(1 - e^y)^{(\beta-1)} e^y \\ &= \frac{e^{\alpha y}(1 - e^y)^{(\beta-1)}}{B(\alpha, \beta)} \end{aligned} \tag{C.10}$$

C.3 MAP Derivation of Log Transformation

Having identified in Chapter 5 that the MAP (mode) of the post processor PDF is equivalent to the p_F estimate in standard SuS, the derivation of the MAP of the log transformed PDF is required. Consider $z = y + b$, where $b \in \mathbb{R}$. Using the above derivations, for $z \geq b$

$$f_Z(z) = \frac{e^{\alpha(z-b)}(1 - e^{(z-b)})^{(\beta-1)}}{B(\alpha, \beta)} \tag{C.11}$$

Letting $L(z)$ represent the numerator while omitting the normalizing constant and introducing the natural logarithm yields

$$\ln(L(z)) = \alpha(z - b) + (\beta - 1)\ln(1 - e^{(z-b)}) \tag{C.12}$$

The MAP occurs at the maximum point of the PDF i.e. where the slope is 0. Consequently, differentiating $L(z)$ with respect to z yields

$$\begin{aligned}
\frac{\partial \ln(L(z))}{z} &= \alpha + \frac{(\beta - 1) \cdot -e^{(z-b)}}{(1 - e^{(z-b)})} \\
&= \frac{\alpha - e^{(z-b)}\alpha - e^{(z-b)}\beta + e^{(z-b)}}{(1 - e^{(z-b)})} \\
&= \frac{\alpha - e^{(z-b)}(\alpha + \beta - 1)}{(1 - e^{(z-b)})}
\end{aligned} \tag{C.13}$$

This quantity represents the MAP if and only if it equals 0

$$\alpha - e^{(z-b)}(\alpha + \beta - 1) = 0 \tag{C.14}$$

Simplifying and expressing in terms of z allows for the MAP of the log transformed beta PDF to be defined.

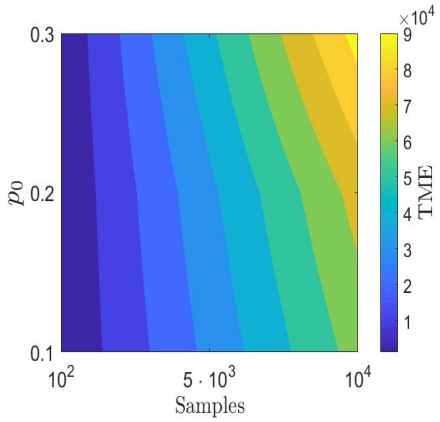
$$-e^{(z-b)}(\alpha + \beta - 1) = -\alpha$$

$$\implies e^{(z-b)} = \frac{\alpha}{(\alpha + \beta - 1)}$$

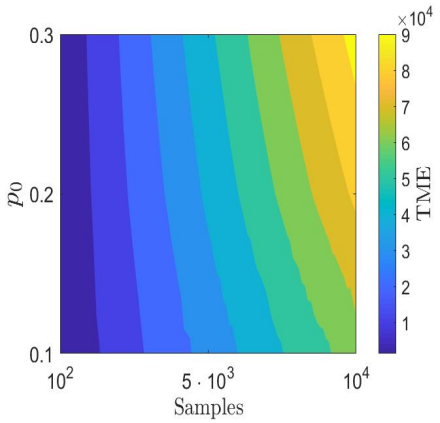
$$\implies z = \ln\left(\frac{\alpha}{(\alpha+\beta-1)}\right) + b \tag{C.15}$$

In practice, b represents the intermediate threshold of BUS_+.

C.4 Computational Expense of Numerical Examples

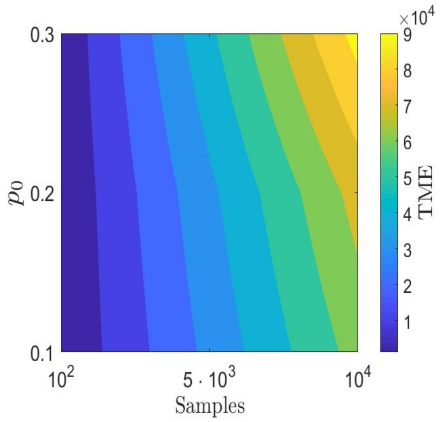


(a) Number of TME for BUS_+.

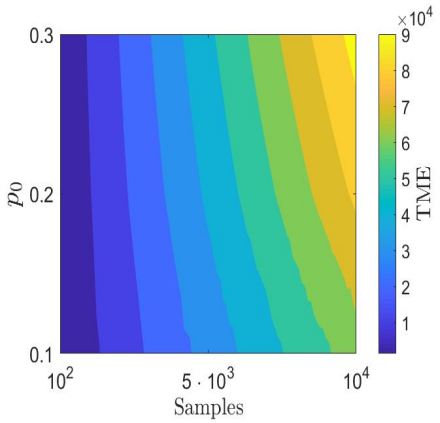


(b) Number of TME for $nBUS_D$.

Figure C.1: Illustration of the required TME for BUS_+ and $nBUS_D$ for drawing posterior samples in example 1.

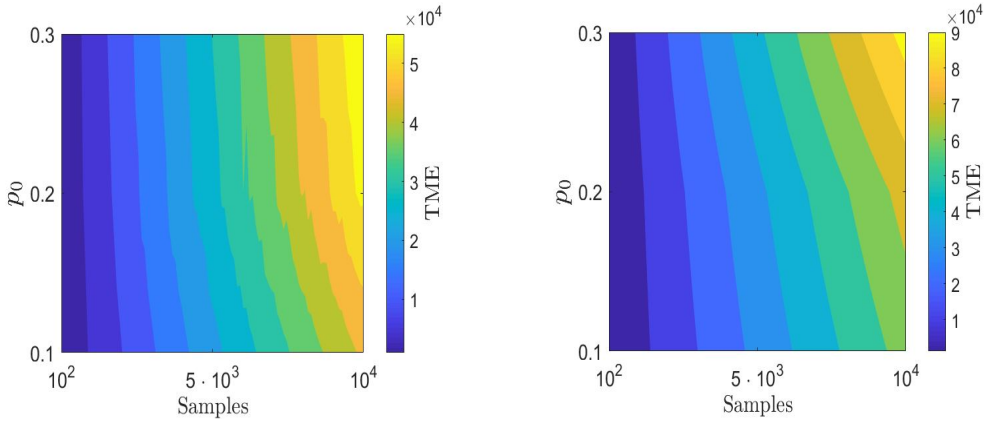


(a) Number of TME for BUS_+.



(b) Number of TME for $nBUS_D$.

Figure C.2: Illustration of the required TME for BUS_+ and $nBUS_D$ for drawing posterior samples in example 2.



(a) Number of TME for BUS_+. (b) Number of TME for $nBUS_D$.

Figure C.3: Illustration of the required TME for BUS_+ and $nBUS_D$ for drawing posterior samples in example 3.

Bibliography

[1] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac, S. García, L. Sánchez, and F. Herrera. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. *Journal of Multiple-Valued Logic and Soft Computing*, 17:255–287, 2011.

[2] E. Alpaydin. *Introduction to machine learning*. MIT Press, 2004.

[3] G. Altekar, S. Dwarkadas, J. P. Huelsenbeck, and F. Ronquist. Parallel Metropolis coupled Markov chain Monte Carlo for Bayesian phylogenetic inference. *Bioinformatics*, 20(3):407–415, feb 2004.

[4] M. Anderberg. Cluster Analysis for Applications,. Technical report, Defense Technical Information Center, 1973.

[5] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan. An Introduction to MCMC for Machine Learning. *Machine Learning*, 50(1/2):5–43, 2003.

[6] C. Andrieu and É. Moulines. On the ergodicity properties of some adaptive MCMC algorithms. *The Annals of Applied Probability*, 16(3):1462–1505, aug 2006.

[7] P. Angelikopoulos, C. Papadimitriou, and P. Koumoutsakos. X-TMCMC: Adaptive kriging for Bayesian inverse modeling. *Computer Methods in Applied Mechanics and Engineering*, 289:409–428, jun 2015.

[8] Y. F. Atchadé, G. O. Roberts, and J. S. Rosenthal. Towards optimal scaling of metropolis-coupled Markov chain Monte Carlo. *Statistics and Computing*, 21(4):555–568, oct 2011.

[9] Y. F. Atchadé and J. S. Rosenthal. On Adaptive Markov Chain Monte Carlo Algorithms.

[10] S. Au and E. Patelli. Rare Event Simulation in Finite-Infinite Dimensional Space. *Reliability Engineering and System Safety*, 148:66–77, 2015.

[11] S. K. Au and J. Beck. Estimation of Small Probabilities in High Dimensions by Subset Simulation. *Probabilistic Engineering Mechanics*, 16(4):263–277, 2001.

[12] S. K. Au and Y. Wang. *Engineering risk assessment and design with subset simulation*. Wiley, 2014.

[13] D. Bamber. The area above the ordinal dominance graph and the area below the receiver operating characteristic graph. *Journal of Mathematical Psychology*, 12(4):387–415, nov 1975.

[14] R. Bardenet, A. Doucet, and C. Holmes. An Adaptive Subsampling Approach for MCMC Inference in Large Datasets. In *International Conference of Machine Learning*, 2014.

[15] R. Bardenet, A. Doucet, and C. C. Holmes. On Markov chain Monte Carlo methods for tall data. *Journal of Machine Learning Research*, 18:1–43, 2017.

[16] A. Basudhar and S. Missoum. Reliability assessment using probabilistic support vector machines. *International Journal of Reliability and Safety*, 7(2):156, 2013.

[17] A. Basudhar, S. Missoum, and A. Harrison Sanchez. Limit state function identification using Support Vector Machines for discontinuous responses and disjoint failure domains. *Probabilistic Engineering Mechanics*, 23(1):1–11, jan 2008.

[18] D. Bates and D. Watts. *Nonlinear regression analysis and its applications*. Wiley, 1988.

[19] J. Beck. Bayesian System Identification and the Bayesian Ockham Razor. In *Eurodyn 2014 proceedings of the 9th International Conference on Structural Dynamics, Porto, Portugal, 30 June - 2 July 2014*, pages 185–192. Faculty of Engineering], 2014.

[20] J. Beck and S. Au. Bayesian Updating of Structural Models and Reliability using Markov Chain Monte Carlo Simulation. *Journal of Engineering Mechanics*, 128(4):380–391, 2002.

[21] J. L. Beck and K. Yuen. Model Selection Using Response Measurements: Bayesian Probabilistic Approach. *Journal of Engineering Mechanics*, 130(2):192–203, 2004.

[22] J. L. Beck and K. M. Zuev. Asymptotically Independent Markov Sampling: a new MCMC scheme for Bayesian Inference. oct 2011.

[23] J. Bect, L. Li, and E. Vazquez. Bayesian Subset Simulation *. *arXiv*, 2017.

[24] D. R. Bellhouse. The Reverend Thomas Bayes, FRS: A Biography to Celebrate the Tercentenary of His Birth. *Statistical Science*, 19(1):3–43, 2004.

[25] R. Bellman. *Dynamic programming*. Dover Publications, 2003.

[26] Y. Bengio, R. Ducharme, P. Vincent, C. Jauvin, J. U. Ca, J. Kandola, T. Hofmann, T. Poggio, and J. Shawe-Taylor. A Neural Probabilistic Language Model. Technical report, 2003.

[27] J. Bennett and S. Lanning. The Netflix Prize. Technical report, Netflix, 2007.

[28] J. O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer New York, 1985.

[29] J. O. Berger. The case for objective Bayesian analysis. *Bayesian Analysis*, 1(3):385–402, sep 2006.

[30] J. O. Berger and R. L. Wolpert. *Institute of Mathematical Statistics LECTURE NOTES-MONOGRAPH SERIES The Likelihood Principle (Second Edition)*. Institute of Mathematical Statistics, 1988.

[31] J. Bergstra and Y. Bengio. Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[32] W. Betz, I. Papaioannou, J. Beck, and D. Straub. Bayesian inference with Subset Simulation: Strategies and improvements. *Computer Methods in Applied Mechanics and Engineering*, 331(1):72–93, 2018.

[33] W. Betz, I. Papaioannou, and D. Straub. Adaptive variant of the BUS approach to Bayesian Updating. In *9th International Conference on Structural Dynamics, EURODYN,Porto,Portugal*, 2014.

[34] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006.

[35] D. Blei, A. Kucukelbir, and J. McAuliffe. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518):859–877, apr 2017.

[36] J. Bootkrajang and A. Kabán. Multi-class Classification in the Presence of Labelling Errors. *In Proceedings of European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pages 345–350, 2011.

[37] J. Bootkrajang and A. Kabán. Label-Noise Robust Logistic Regression and Its Applications. *Machine Learning and Knowledge Discovery in Databases*, Springer B:143–158, 2012.

[38] A. Borri and E. Speranzini. Structural reliability analysis using a standard deterministic finite element code. *Structural Safety*, 4:361–382, 1997.

[39] J.-M. Bourinet. Rare-event probability estimation with adaptive support vector regression surrogates. *Reliability Engineering & System Safety*, 150:210–221, jun 2016.

[40] J.-M. Bourinet, F. Deheeger, and M. Lemaire. Assessing small failure probabilities by combined subset simulation and Support Vector Machines. *Structural Safety*, 33(6):343–353, sep 2011.

[41] C. Bouveyron and S. Girard. Robust Supervised Classification with Mixture Models: Learning from Data and Uncertain Labels. *Pattern Recognition*, 42(11):306–313, 2009.

[42] L. Breiman. *Classification and regression trees*. Chapman & Hall, 1993.

[43] N. E. Breslow and R. Holubkov. Maximum Likelihood Estimation of Logistic Regression Parameters Under Two-Phase, Outcome-Dependent Sampling. *Journal of the Royal Statistical Society*, 59(2):447–461, 1997.

[44] P. G. Byrnes and F. A. DiazDelaO. Bayesian Updating for Probabilistic Classification using Reliability Methods. *Eccomas Proceedia UNCECOMP*, pages 339–349, 2017.

[45] P. G. Byrnes and F. A. DiazDelaO. Reliability Based Bayesian Inference for Probabilistic Classification: An Overview of Sampling Schemes. *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, 34:250–263, 2017.

[46] B. Calderhead. A general construction for parallelizing Metropolis-Hastings algorithms. *Proceedings of the National Academy of Sciences of the United States of America*, 111(49):17408–17413, dec 2014.

[47] C. Calì and M. Longobardi. Some mathematical properties of the ROC curve and their applications. *Research of Matematica*, 64(2):391–402, 2015.

[48] B. Carpenter, A. Gelman, M. D. Hoffman, D. Lee, B. Goodrich, M. Betancourt, M. Brubaker, J. Guo, P. Li, and A. Riddell. Stan : A Probabilistic Programming Language. *Journal of Statistical Software*, 76(1):1–32, jan 2017.

[49] G. C. Cawley and N. L. C. Talbot. Preventing Over-Fitting during Model Selection via Bayesian Regularisation of the Hyper-Parameters. *Journal of Machine Learning Research*, 8(Apr):841–861, 2007.

[50] S. Chatterjee and A. Hadi. *Regression analysis by example*. Wiley, 2012.

[51] D. Chauveau and P. Vandekerkhove. Improving Convergence of the Hastings-Metropolis Algorithm with an Adaptive Proposal.

[52] T. Chen, E. Fox, and C. Guestrin. Stochastic gradient hamiltonian monte carlo. *In Proceedings of the 31st International Conference on Machine Learning*, pages 1683–1691, 2014.

[53] J. Ching and J.-S. Wang. Application of the transitional Markov chain Monte Carlo algorithm to probabilistic site characterization. *Engineering Geology*, 203:151–167, mar 2016.

[54] S. Ching and Y. Chen. Transitional Markov Chain Monte Carlo Method for Bayesian Model Updating, Model Class Selection, and Model Averaging. *Journal of Engineering Mechanics*, 133(7):816–832, 2007.

[55] H. A. Chipman, E. I. George, and R. E. Mcculloch. BART: Bayesian Additive Regression Trees. Technical report, 2008.

[56] D. Cireşan, U. Meier, J. Masci, and J. Schmidhuber. Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32:333–338, aug 2012.

[57] D. Cook, P. Dixon, W. M. Duckworth, M. S. Kaiser, K. Koehler, W. Q. Meeker, and W. R. Stephenson. Binary Response and Logistic Regression Analysis. Technical report, Iowa State, 2001.

[58] C. Cortes. Support-Vector Networks. *Machine learning*, 20(3):273–297, 1995.

[59] F. Cui and M. Ghosn. Implementation of machine learning techniques into the Subset Simulation method. *Structural Safety*, 79:12–25, 2019.

[60] Y. Cun and H. Fröhlich. Prognostic gene signatures for patient stratification in breast cancer - accuracy, stability and interpretability of gene selection approaches using prior knowledge on protein-protein interactions. *BMC Bioinformatics*, 13(1):69, dec 2012.

[61] S. A. Czepiel. Maximum Likelihood Estimation of Logistic Regression Models: Theory and Implementation. Technical report, Direct Partners, 2011.

[62] T. Dasu and T. Johnson. *Exploratory Data Mining and Data Cleaning*. Wiley Series in Probability and Statistics. John Wiley & Sons, Inc., Hoboken, NJ, USA, may 2003.

[63] P. Del Moral, A. Doucet, and A. Jasra. Sequential Monte Carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, jun 2006.

[64] J.-F. Delmas and B. Jourdain. Does Waste Recycling Really Improve the Multi-Proposal Metropolis–Hastings algorithm? an Analysis Based on Control Variates. *Journal of Applied Probability*, 46(4):938–959, dec 2009.

[65] J. Deng, W. Dong, R. Socher, L. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[66] A. Der Kiureghian and P. Liu. Structural Reliability under Incomplete Probability Information. *Journal of Engineering Mechanics*, 112(1):85–104, jan 1986.

[67] F. A. DiazDelaO, A. Garbuno-Indigo, S. Au, and I. Yoshida. Bayesian Updating and Model Class Selection with Subset Simulation. *Computer Methods in Applied Mechanics and Engineering*, 317:1102–1221, 2017.

[68] N. Draper and H. Smith. *Applied Regression Analysis*. John Wiley & Sons, 1998.

[69] D. Dua and C. Graff. UCI Machine Learning Repository. *arXiv*, 2019.

[70] S. Duane, A. Kennedy, B. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, 1987.

[71] D. J. Earl and M. W. Deem. Parallel tempering: Theory, applications, and new perspectives. *Physical Chemistry Chemical Physics*, 7(23):3910, nov 2005.

[72] B. Echard, N. Gayton, and M. Lemaire. AK-MCS: An active learning reliability method combining Kriging and Monte Carlo Simulation. *Structural Safety*, 33:145–154, 2011.

[73] A. W. Edwards. *Likelihood*. Johns Hopkins University Press, 1992.

[74] R. Ellsworth, J. Hooke, B. Love, D. Ellsworth, and C. Shriver. Molecular Changes in Primary Breast Tumors and the Nottingham Histologic Score. *Pathology & Oncology Research*, 15(4):541–547, dec 2009.

[75] A. Erraqabi, M. Valko, A. Carpentier, and O.-A. Maillard. Pliable rejection sampling. *ICML 2016*, jun 2016.

[76] D. Y. Fan. The distribution of the product of independent beta variables. *Communications in Statistics - Theory and Methods*, 20(12):4043–4052, 1991.

[77] B. Frenay and M. Verleysen. Classification in the Presence of Label Noise: A Survey. *IEEE Transactions on Neural Networks and Learning Systems*, 25(5):845–869, 2014.

[78] D. Gamberger, N. Lavrav, and S. Dzeroski. Noise elimination in inductive concept learning: a case study in medical diagnosis. *In Proceedings of 7th International Workshop Algorithmic Learning Theory*, Sydney:199–212, 1996.

[79] D. Gamerman and H. F. Lopes. *Markov chain Monte Carlo : stochastic simulation for Bayesian inference.* Taylor & Francis, 2006.

[80] P. H. Garthwaite, J. B. Kadane, and A. O'hagan. Statistical Methods for Eliciting Probability Distributions. *Journal of the American Statistical Association*, 100(470):680–701, 2005.

[81] A. Gelman, J. Carlin, S. Stern, and D. Rubin. *Bayesian Data Analysis.* Taylor and Francis, Florida, 2014.

[82] A. Gelman, G. Roberts, and W. Gilks. Efficient Metropolis Jumping Rules. *Bayesian Statistics*, 5:599–607, 1996.

[83] A. Gelman and A. Rubin. Inference from Iterative Simulation Using Multiple Sequences. *Statistical Science*, 7(4):457–472, 1992.

[84] S. Geman and D. Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, nov 1984.

[85] C. J. Geyer. Markov Chain Monte Carlo Maximum Likelihood. 1991.

[86] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markov chain Monte Carlo in practice*. Chapman & Hall, 1996.

[87] W. R. Gilks, G. O. Roberts, and E. I. George. Adaptive Direction Sampling. *The Statistician*, 43(1):179, 1994.

[88] W. R. Gilks, G. O. Roberts, and S. K. Sahu. Adaptive Markov Chain Monte Carlo through Regeneration. *Journal of the American Statistical Association*, 93(443):1045–1054, sep 1998.

[89] W. R. Gilks and P. Wild. Adaptive Rejection Sampling for Gibbs Sampling. *Applied Statistics*, 41(2):337, 1992.

[90] D. Giovanis, I. Papaioannou, D. Straub, and V. Papadopoulos. Bayesian updating with subset simulation using artificial neural networks. *Computer Methods in Applied Mechanics and Engineering*, 319:124–145, 2017.

[91] M. Girolami. Bayesian inference for differential equations. *Theoretical Computer Science*, 408(1):4–16, nov 2008.

[92] I. J. Good. The Bayesian Influence, or How to Sweep Subjectivism under the Carpet. In *Foundations of Probability Theory, Statistical Inference, and Statistical Theories of Science*, pages 125–174. Springer Netherlands, Dordrecht, 1976.

[93] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*.

[94] D. M. Green and J. A. Swets. *Signal detection theory and psychophysics*. Peninsula Pub, 1988.

[95] P. J. Green. Reversible jump Markov chain monte carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732, dec 1995.

[96] P. L. Green and K. Worden. Bayesian and Markov chain Monte Carlo methods for identifying nonlinear systems in the presence of uncertainty. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 373(2051):20140405, sep 2015.

[97] H. Haario, M. Laine, A. Mira, and E. Saksman. DRAM: Efficient adaptive MCMC. *Statistics and Computing*, 16(4):339–354, dec 2006.

[98] H. Haario, E. Saksman, and J. Tamminen. Adaptive proposal distribution for random walk Metropolis algorithm. *Computational Statistics*, 14(3):375, 1999.

[99] H. Haario, E. Saksman, and J. Tamminen. An Adaptive Metropolis Algorithm. *Bernoulli*, 7(2):223, apr 2001.

[100] H. Haario, E. Saksman, and J. Tamminen. Componentwise adaptation for high dimensional MCMC. *Computational Statistics*, 20(2):265–273, jun 2005.

[101] A. Hajek. Interpretations of Probability. *Stanford encyclopedia of philosophy.*, 1(1), 2002.

[102] J. A. Hanley and B.Mcneil. The Meaning and Use of the Area Under a Receiver Operating Characteristic (ROC) Curve. *Radiology*, 143(1):29–36, 1982.

[103] J. A. Hanley and B. J. McNeil. A method of comparing the areas under receiver operating characteristic curves derived from the same cases. *Radiology*, 148(3):839–43, sep 1983.

[104] W. Hardle. *Applied nonparametric regression.* Cambridge University Press, 1990.

[105] W. Hastings. Monte Carlo Sampling Methods Using Markov Chains and Their Applications. *Biometrika*, 7(1):97–109, 1970.

[106] M. D. Hoffman and A. Gelman. The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15:1593–1623, 2014.

[107] D. Hosmer, S. Lemeshow, and R. X. Sturdivant. *Applied logistic regression.* Wiley-Interscience, 2013.

[108] P. Hristov, F. DiazDelaO, U. Farooq, and K. Kubiak. Adaptive Gaussian process emulators for efficient reliability analysis. *Applied Mathematical Modelling*, 71:138–151, jul 2019.

[109] C.-W. Hsu, C.-C. Chang, and C.-J. Lin. A Practical Guide to Support Vector Classification. *arXiv*, 2003.

[110] J. E. Hurtado. An examination of methods for approximating implicit limit state functions from the viewpoint of statistical learning theory. *Structural Safety*, 26(3):271–293, jul 2004.

[111] J. E. Hurtado. *Structural reliability : statistical learning perspectives.* Springer, 2004.

[112] J. E. Hurtado. Filtered importance sampling with support vector margin: A powerful method for structural reliability analysis. *Structural Safety*, 29(1):2–15, jan 2007.

[113] P. Jeatrakul, K. Wong, and C. Fung. Data cleaning for classification using mislcassification analysis. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 14(3):297–302, 2010.

[114] H. Jeffreys. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 186(1007):453–461, sep 1946.

[115] H. Jeffreys. *Theory of probability.* Clarendon Press, 1998.

[116] R. I. Jennrich and P. F. Sampson. Newton-Raphson and Related Algorithms for Maximum Likelihood Variance Component Estimation. *Technometrics*, 18(1):11, feb 1976.

[117] R. E. Kass, B. P. Carlin, A. Gelman, and R. M. Neal. Markov Chain Monte Carlo in Practice: A Roundtable Discussion. *The American Statistician*, 52(2):93, may 1998.

[118] R. E. Kass and A. E. Raftery. Bayes Factors. *Journal of the American Statistical Association*, 90(430):773, jun 1995.

[119] L. Kaufman and P. Rousseeuw. *Finding groups in data : an introduction to cluster analysis*. Wiley, 1990.

[120] S. Keerthi and C. Lin. Asymptotic Behaviors of Support Vector Machines with Gaussian Kernel. *Neural Computation*, 15(7):1667–1689, jul 2003.

[121] M. C. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(3):425–464, aug 2001.

[122] M. Khoshgoftaar T. and Rebours P. Generating Multiple Noise Elimination Filters with the Ensemble-Partitioning Filter. *In Proceedings of IEEE International Conference on Information Reuse and Information for Data Science*, Las Vegas:369–375, 2014.

[123] J. Kim. Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. *Computational Statistics & Data Analysis*, 53(11):3735–3745, sep 2009.

[124] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science (New York, N.Y.)*, 220(4598):671–80, may 1983.

[125] D. A. Kofke. On the acceptance probability of replica-exchange Monte Carlo trials. *The Journal of Chemical Physics*, 117(15):6911–6914, oct 2002.

[126] A. Kolmogoroff. *Grundbegriffe der Wahrscheinlichkeitsrechnung*. Springer Berlin Heidelberg, 1933.

[127] A. Kone and D. A. Kofke. Selection of temperature intervals for parallel-tempering simulations. *The Journal of Chemical Physics*, 122(20):206101, may 2005.

[128] A. Korattikara, Y. Chen, and M. Welling. Austerity in MCMC Land: Cutting the Metropolis-Hastings Budget. In *International Conference of Machine Learning*, 2014.

[129] S. C. Kou, Q. Zhou, and W. H. Wong. Equi-energy sampler with applications in statistical inference and statistical mechanics. *The Annals of Statistics*, 34(4):1581–1619, aug 2006.

[130] B. Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016.

[131] M. Kuderer, S. Gulati, and W. Burgard. Learning driving styles for autonomous vehicles from demonstration. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2641–2646. IEEE, may 2015.

[132] M. Kuss and C. E. Rasmussen. Assessing Approximate Inference for Binary Gaussian Process Classification. *Journal of Machine Learning Research*, 6(Oct):1679–1704, 2005.

[133] P. Laplace. *A Philosophical Essay on Probabilities*. Dover Publications Inc, New York, 1814.

[134] N. D. Lawrence and B. Schölkopf. Estimating a Kernel Fisher Discriminant in the Presence of Label Noise. *In Proceedings of the 18th International Conference on Machine Learning*, San Franci:306–313, 2001.

[135] N. D. Lawrence, M. Seeger, and R. Herbich. Fast sparse Gaussian process methods: the informative vector machine. In *Proceedings of the 15th International Conference on Neural Information Processing Systems*, pages 625–632, 2002.

[136] R. Levy. Probabilistic Models in the Study of Language. Technical report, University of California San Diego, 2012.

[137] S. Li and W. K. Liu. Meshfree and particle methods and their applications. *Applied Mechanics Reviews*, 55(1):1–34, jan 2002.

[138] A. Liaw and M. Wiener. Classification and regression by randomForest, 2007.

[139] L. Lin, K. F. Liu, and J. Sloan. A noisy Monte Carlo algorithm. *Physical Review D*, 61(7):074505, mar 2000.

[140] J. S. Liu, F. Liang, and W. H. Wong. The Multiple-Try Method and Local Optimization in Metropolis Sampling. *Journal of the American Statistical Association*, 95(449):121–134, mar 2000.

[141] D. Lunn, D. Spiegelhalter, A. Thomas, and N. Best. The BUGS project: Evolution, critique and future directions. *Statistics in Medicine*, 28(25):3049–3067, nov 2009.

[142] D. J. C. MacKay. *Information theory, inference, and learning algorithms*. Cambridge University Press, 2003.

[143] D. Malakoff. Bayes offers a 'new' way to make sense of numbers. *Science (New York, N.Y.)*, 286(5444):1460–4, nov 1999.

[144] M. Manica, J. Cadow, R. Mathis, and M. Rodríguez Martínez. PIMKL: Pathway-Induced Multiple Kernel Learning. *npj Systems Biology and Applications*, 5(1):8, dec 2019.

[145] E. Marinari and G. Parisi. Simulated Tempering: A New Monte Carlo Scheme. *Europhysics Letters (EPL)*, 19(6):451–458, jul 1992.

[146] L. Martino and J. Míguez. A generalization of the adaptive rejection sampling algorithm. *Statistics and Computing*, 21(4):633–647, oct 2011.

[147] I. McGraw, R. Prabhavalkar, R. Alvarez, M. Arenas, K. Rao, D. Rybach, O. Alsharif, H. Sak, A. Gruenstein, F. Beaufays, and C. Parada. Personalized speech recognition on mobile devices. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5955–5959. IEEE, mar 2016.

[148] S. W. Menard. *Applied logistic regression analysis*. Sage Publications, 2002.

[149] F. Miao and M. Ghosn. Modified subset simulation method for reliability analysis of structural systems. *Structural Safety*, 33(4-5):251–260, jul 2011.

[150] B. Miasojedow, E. Moulines, and M. Vihola. An Adaptive Parallel Tempering Algorithm. *Journal of Computational and Graphical Statistics*, 22(3):649–664, jul 2013.

[151] T. P. Minka. Expectation Propagation for approximate Bayesian inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence (UAI2001)*, 2001.

[152] L. Miranda A., P. Garcia L., C. Carvalho A., and C. Lorena A. Use of Classification Algorithms in Noise Detection and Elimination. *In Proceedings of 4th International Conference Hybrid Artificial Intelligence Systems*, Spain:417–424, 2009.

[153] T. M. Mitchell. *Machine Learning*. McGraw-Hill Book Company Inc., 1997.

[154] J. Mockus, V. Tiesis, and A. Zilinskas. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2:117–129, 1978.

[155] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. MIT Press, 2012.

[156] A. M. Mood. *Introduction to the Theory of Statistics*. McGraw-Hill Book Company Inc., 1950.

[157] K. P. Murphy. *Machine learning : a probabilistic perspective*. MIT Press, 2012.

[158] K. P. Murphy. *Machine learning : a probabilistic perspective*. MIT Press, 2012.

[159] P. Mykland, L. Tierney, and B. Yu. Regeneration in Markov Chain Samplers. *Journal of the American Statistical Association*, 90(429):233, mar 1995.

[160] I. J. Myung. Tutorial on maximum likelihood estimation. *Journal of Mathematical Psychology*, 47:90–100, 2003.

[161] R. M. Neal. *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. Springer New York, New York, NY, 1996.

[162] R. M. Neal. Annealed importance sampling. *Statistics and Computing*, 1(11):125–139, 2001.

[163] R. M. Neal. Slice sampling. *The Annals of Statistics*, 31(3):705–767, jun 2003.

[164] W. Neiswanger, C. Wang, and E. P. Xing. Asymptotically Exact, Embarrassingly Parallel MCMC. In *Proceedings of the conference on Uncertainty in Artificial INtelligence*, 2014.

[165] H. Nickisch and C. E. Rasmussen. Approximations for Binary Gaussian Process Classification. *Journal of Machine Learning Research*, 9(Oct):2035–2078, 2008.

[166] N.Metropolis, A.W.Rosenbluth, M.N.Rosenbluth, A.H.Teller, and E.Teller. Equations of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[167] O. Ore. Pascal and the Invention of Probability Theory. *The American Mathematical Monthly*, 67(5):409–419, may 1960.

[168] J. Pacheco, S. Casado, and L. Núñez. A variable selection method based on Tabu search for logistic regression models. *European Journal of Operational Research*, 199(2):506–511, dec 2009.

[169] Q. Pan and D. Dias. An efficient reliability method combining adaptive Support Vector Machine and Monte Carlo Simulation. *Structural Safety*, 67:85–95, jul 2017.

[170] V. Papadopoulos, D. G. Giovanis, N. D. Lagaros, and M. Papadrakakis. Accelerated subset simulation with neural networks for reliability analysis. *Computer Methods in Applied Mechanics and Engineering*, 223-224:70–80, jun 2012.

[171] M. Papadrakakis, V. Papadopoulos, and N. D. Lagaros. Structural reliability analyis of elastic-plastic structures using neural networks and Monte Carlo simulation. *Computer Methods in Applied Mechanics and Engineering*, 136(1-2):145–163, sep 1996.

[172] I. Papaioannou, W. Betz, K. Zwirglmaier, and D. Straub. MCMC algorithms for Subset Simulation. *Probabilistic Engineering Mechanics*, 41:89–103, 2015.

[173] P. Pati, S. Andani, M. Palhares Viana, M. Gabrani, P. Wild, J. H. Ruschoff, and M. Pediaditis. Deep positive-unlabeled learning for region of interest localization in breast tissue images. In M. N. Gurcan and J. E. Tomaszewski, editors, *Medical Imaging 2018: Digital Pathology*, page 3. SPIE, mar 2018.

[174] C. S. Peirce. Notes on the Doctrine of Chances. In *Dispositions*, pages 237–245. Springer, Dordrecht, 1978.

[175] J. C. Platt. Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. Technical report, Microsoft Research, 1998.

[176] J. C. Platt. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. *Advances in Large Margin Classifiers*, pages 61—-74, 1999.

[177] M. T. Plummer. JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling, 2003.

[178] K. R. Popper. The Propensity Interpretation of the Calculus of Probability, and the Quantum Theory. *The Colston Papers*, 9:65–70, 1957.

[179] P.Pati, S.Andani, M.Pediaditis, M.P.Viana, J.H.Ruschoff, P.Wild, and M.Gabrani. Deep positive-unlabeled learning for region of interest localization in breast tissue images. In *Medical Imaging 2018: Digital Pathology*, volume 10581, page 1058107, 2018.

[180] S. J. Press. *Subjective and objective Bayesian statistics : principles, models, and applications*. Wiley-Interscience, 2003.

[181] N. V. Queipo, R. T. Haftka, W. Shyy, T. Goel, R. Vaidyanathan, and P. Kevin Tucker. Surrogate-based analysis and optimization. *Progress in Aerospace Sciences*, 41(1):1–28, jan 2005.

[182] E. Rahm and H. Hai Do. Data Cleaning:Problems and Current Approaches. *Bulletin of the Technical Committee on Data Engineering*, 23(4):1—-42, 2000.

[183] C. Rasmussen. Gaussian Processes in Machine Learning. In *Gaussian Processes in Machine Learning*, pages 63–71. Springer, Berlin, Heidelberg, 2004.

[184] H. Robbins and S. Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400–407, sep 1951.

[185] C. P. Robert and G. Casella. *Monte Carlo Statistical Methods*. Springer New York, 2004.

[186] C. P. Robert, V. Elvira, N. Tawn, and C. Wu. Accelerating MCMC algorithms. *Wiley Interdisciplinary Reviews: Computational Statistics*, 10(5):e1435, sep 2018.

[187] G. O. Roberts, A. Gelman, and W. R. Gilks. Weak convergence and optimal scaling of random walk Metropolis algorithms. *The Annals of Applied Probability*, 7(1):110–120, feb 1997.

[188] G. O. Roberts and J. S. Rosenthal. Optimal scaling for various Metropolis-Hastings algorithms. *Statistical Science*, 16(4):351–367, nov 2001.

[189] C. M. Rocco and J. A. Moreno. Fast Monte Carlo reliability evaluation using support vector machine. *Reliability Engineering & System Safety*, 76(3):237–243, jun 2002.

[190] M. Rosenblatt. Remarks on a Multivariate Transformation. *The Annals of Mathematical Statistics*, 23(3):470–472, sep 1952.

[191] J. Rosenthal. Optimizing and Adapting the Metropolis Algorithm. In *Statistics in Action*. Chapman and Hall/CRC, mar 2014.

[192] S. K. Sahu and A. A. Zhigljavsky. Self-regenerative Markov chain Monte Carlo with adaptation. *Bernoulli*, 9(3):395–422, jun 2003.

[193] A. L. Samuel. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, 3(3):210–229, jul 1959.

[194] J. Sanchez del Rio, D. Moctezuma, C. Conde, I. Martin de Diego, and E. Cabello. Automated border control e-gates and facial recognition systems. *Computers & Security*, 62:49–72, sep 2016.

[195] Sang-Bum Kim, Kyoung-Soo Han, Hae-Chang Rim, and Sung Hyon Myaeng. Some Effective Techniques for Naive Bayes Text Classification. *IEEE Transactions on Knowledge and Data Engineering*, 18(11):1457–1466, nov 2006.

[196] A. Santoso, K. Phoon, and S. Quek. Modified Metropolis–Hastings algorithm with reduced chain correlation for efficient subset simulation. *Probabilistic Engineering Mechanics*, 26(2):331–341, apr 2011.

[197] J. A. Scales and L. Tenorio. Tutorial Prior information and uncertainty in inverse problems. Technical Report 2.

[198] S. L. Scott, A. W. Blocker, F. V. Bonassi, H. A. Chipman, E. I. George, and R. E. McCulloch. Bayes and big data: the consensus Monte Carlo algorithm. *International Journal of Management Science and Engineering Management*, 11(2):78–88, apr 2016.

[199] G. A. F. Seber and A. J. Lee. *Linear regression analysis*. Wiley-Interscience, 2003.

[200] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning : from theory to algorithms*. University Cambridge Press, 2014.

[201] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis*. Cambridge University Press, 2004.

[202] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, oct 2017.

[203] D. S. Sivia and J. J. Skilling. *Data analysis : a Bayesian tutorial*. Oxford University Press, 2006.

[204] J. Skilling. Nested sampling for general Bayesian computation. *Bayesian Analysis*, 1(4):833–859, dec 2006.

[205] R. Smith. *Uncertainty quantification : theory, implementation, and applications.*

[206] J. Snoek, H. Larochelle, and R. Adams. Practical Bayesian optimization of machine learning algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, pages 2951–2959, 2012.

[207] D. F. Specht. Probabilistic neural networks. *Neural Networks*, 3(1):109–118, jan 1990.

[208] D. Straub and I. Papaioannou. Bayesian Updating with Structural Reliability Methods. *Journal of Engineering Mechanics*, 141(3):455–461, 2015.

[209] R. Sutton and A. Barto. *Reinforcement learning : an introduction.* MIT Press, 1998.

[210] R. H. Swendsen and J.-S. Wang. Replica Monte Carlo Simulation of Spin-Glasses. *Physical Review Letters*, 57(21):2607–2609, nov 1986.

[211] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to data mining.* Pearson Addison Wesley, 2005.

[212] J. Tang and A. K. Gupta. On the distribution of the product of independent beta random variables. *Statistics & Probability Letters*, 2(3):165–168, 1984.

[213] A. I. Taylor. Autosomal trisomy syndromes: a detailed study of 27 cases of Edwards' syndrome and 27 cases of Patau's syndrome. *Journal of medical genetics*, 5(3):227–52, sep 1968.

[214] L. Tierney. Markov Chains for Exploring Posterior Distributions. *The Annals of Statistics*, 22(4):1701–1728, dec 1994.

[215] L. Tierney and J. B. Kadane. Accurate Approximations for Posterior Moments and Marginal Densities. *Journal of the American Statistical Society*, 81(393):82–86, 1986.

[216] M. E. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of machine learning research*, 1(Jun):211–244, 2001.

[217] J. W. Tukey and S. Wilks. Approximation of the Distribution of the Product of Beta Variables by a Single Beta Variable. *The Annals of Mathematical Statistics*, 17(3):318–324, 1946.

[218] C. Van Der Malsburg. Frank Rosenblatt: Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. In *Brain Theory*, pages 245–248. Springer Berlin Heidelberg, Berlin, Heidelberg, 1986.

[219] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer New York, 1995.

[220] J. A. Vrugt, C. ter Braak, C. Diks, B. A. Robinson, J. M. Hyman, and D. Higdon. Accelerating Markov Chain Monte Carlo Simulation by Differential Evolution with Self-Adaptive Randomized Subspace Sampling. *International Journal of Nonlinear Sciences and Numerical Simulation*, 10(3), jan 2009.

[221] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers. ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases. In *Computer Vision and Pattern Recognition*, pages 3462–3471, may 2017.

[222] G. M. Weiss. Mining with Rarity: A Unifying Framework. *SIGKDD Explor. Newsl.*, 6(1), 2004.

[223] M. Welling and Y. Teh. Bayesian learning via stochastic gradient langevin dynamics. *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 681–688, 2011.

[224] J. F. Wendt, J. D. Anderson, and Von Karman Institute for Fluid Dynamics. *Computational fluid dynamics : an introduction*. Springer, 2008.

[225] D. Wilks. Cluster Analysis. *International Geophysics*, 100:603–616, jan 2011.

[226] D. Williamson and M. Goldstein. Posterior Belief Assessment: Extracting Meaningful Subjective Judgements from Bayesian Analyses with Complex Statistical Models. *Bayesian Analysis*, 10(4):877–908, 2015.

[227] R. Xu and I. Wunsch D. Survey of Clustering Algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[228] R. Yang and J. O. Berger. A Catalog of Noninformative Priors. Technical report, Duke University, 1998.

[229] C. Yin and A. Kareem. Computation of failure probability via hierarchical clustering. *Structural Safety*, 61:67–77, jul 2016.

[230] K. Yuen. *Bayesian Methods for Structural Dynamics and Civil Engineering*. John Wiley & Sons, 2010.

[231] R. Zhang, C. Li, J. Zhang, C. Chen, and A. G. Wilson. Cyclical Stochastic Gradient MCMC for Bayesian Deep Learning. *arXiv*, 2019.

[232] X. Zhu. Semi-Supervised Learning Literature Survey. Technical report, Univeristy of Wisconsin-Madison, 2006.

[233] K. Zuev. Subset Simulation Method for Rare Event Estimation: An Introduction. Technical report, ZuevSuSIntro.

[234] K. Zuev. Statistical Inference. *SSRN Electronic Journal*, feb 2018.

[235] K. Zuev, J. Beck, S. K. Au, and L. S. Katafygiotis. Bayesian post processor and other enhancements of Subset Simulation for estimating failure probabilities in high dimensions. *Computers and Structures*, 92-93:283–296, 2012.