Estimating Parameters of Truncated Distributions using Likelihood-Free Simulation-Based Methods.

Daniel Gardner

Supervised by Mathieu Gerber
Level M
40 Credit Points

October 23, 2023

Acknowledgement of Sources

For all ideas taken from other sources (books, articles, internet), the source of the ideas is mentioned in the main text and fully referenced at the end of the report.

All material which is quoted essentially word-for-word from other sources is given in quotation marks and referenced.

Pictures and diagrams copied from the internet or other sources are labelled with a reference to the web page or book, article etc.

Signed _____

Date _____

Abstract

In this paper we consider methods of statistical inference where the likelihood function of a model is not directly tractable but still able to be sampled from. We look at cases where this density is intractable due to having a normalising constant that cannot be analytically calculated due to being integrated over a complex truncated domain. We first investigate two well known non-likelihood estimation methods: indirect inference and approximate Bayesian computation (ABC), then go on to introduce the maximum mean discrepancy (MMD) as a distance to use for parameter estimation.

We then go on to look at ways of simulating a truncated normal distribution given different truncation domains, and several different ways of optimising the MMD distance function, which we will show contains a lot of deterministic noise. Finally we end by analysing the MMD estimator's performance in different scenarios, and comparing it to the two initial non-likelihood estimation methods in the context of truncated domains. We conclude that the MMD estimator is more effective than conventional methods such as basic ABC in these contexts.

Contents

1 Introduction **4**

2 Likelihood-free Estimation Methods **7**
2.1 Indirect Inference 7
2.1.1 Choice of Auxiliary Model 9
2.1.2 α-Stable example 10
2.2 Approximate Bayesian Computation 11
2.2.1 Choice of Summary Statistic 16
2.2.2 α-Stable example 18
2.3 Combining Indirect Inference and ABC 18

3 Maximum Mean Discrepancy **21**
3.1 Reproducing Kernel Hilbert Spaces 21
3.1.1 RKHS Properties 21
3.2 Motivation 23
3.3 Characteristic Kernels 25
3.3.1 Motivating Example 25
3.3.2 Characterisation 26
3.4 Parameter Estimation 28
3.4.1 α-Stable Example 30

4 Truncated Domains **34**
4.1 Truncated Normal Distribution 35
4.1.1 Rejection Sampling 35
4.1.2 Gibbs Sampling 41
4.2 Applying the MMD 43
4.2.1 Shape of Truncation Region 45
4.2.2 MMD Algorithm 45
4.2.3 Example 49

5 Noisy Optimisation **50**
5.1 Direct Search and Simplex Methods 52
5.1.1 Pattern Search 52
5.1.2 Nelder-Mead Simplex Methods 54
5.1.3 Accounting for Noise 56
5.2 Bayesian Optimisation 59
5.2.1 Gaussian Process 59
5.2.2 Acquisition Functions 60
5.2.3 Modifying EI for Noisy Functions 62
5.3 Application to the MMD 64
5.3.1 Higher Dimensions 69
5.4 Discussion 72

6 Evaluation **73**
6.1 MMD vs ABC 73
6.1.1 Location of Truncation Region 74
6.1.2 Shape of Truncation Region 75
6.2 MMD within ABC 76

7 Conclusion **77**
7.1 Further Reading 80

1 Introduction

Consider the general case of statistical inference where we have observed data $y \in \mathcal{Y} \subseteq \mathbb{R}^n$, which we know comes from a distribution $f(.|\theta)$ where the defining parameter $\theta \in \Theta \subseteq \mathbb{R}^d$ is unknown. In most cases, we can define the log-likelihood to be $l(\theta|y) = \log(f(y|\theta))$ and compute the maximum likelihood estimator as:

$$\hat{\theta}_{mle} := \arg\max_{\theta \in \Theta} l(\theta|y)$$

Normally we have a **normalisation constant** associated with the proposal distribution which makes sure the function is equal to one when integrated over its support i.e. $\int_{\mathbb{R}^n} f(y|\theta)dy = 1$. However problems begin to arise when we restrict the support to a truncated, finite domain $C \subset \mathbb{R}^n$. Depending on the complexity of the truncation set, we can have situations where the integral $\int_C f(y|\theta)dy$ cannot be calculated, meaning the density function of our truncated distribution,

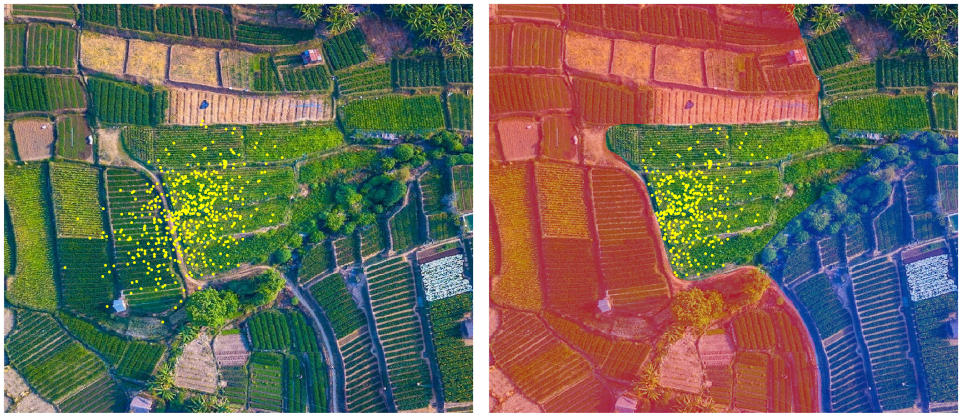
$$f(y|\theta, C) := \frac{f(y|\theta)}{\int_C f(y|\theta)dy}$$

Is intractable, meaning we are not able to derive a gradient for the optimisation used in maximum likelihood estimation. This set C is can be geometrically very complex due to being a very irregular shape, such as a geographical border.

For motivation, consider this example. In figure 1, we have a fictional farm with a mole infestation problem[1]. In one small region of the farm, we can see the locations of molehill sites denoted by the yellow dots, which are known to follow a bivariate normal distribution $f(.|\mu, \Sigma) \sim N_2(\mu, \Sigma)$. We wish to find the parameters (μ, Σ) of this distribution. Normally this problem is trivial to solve, and there exist many ways to estimate these parameters, most commonly using the maximum likelihood estimator. However, consider the case where the farms highlighted in red and blue are owned by private farmland, who are uncooperative in sharing their data. In this case, we have lost a chunk of potential information, and whilst we can assume that the true distribution of moles does not follow administrative boundaries, our observed data is now truncated as such. Potentially we could still perform standard maximum likelihood estimation on the observed data without assuming truncation, but then we risk calculating misleading estimates, e.g. an estimated mean that would be further to the right than the true mean.

With no analytical expression for the likelihood, we are hence restricted to methods which only use repeated simulation using the distribution. For the rest of the paper, we will only consider cases where the truncation boundary

[1] Photo by Tony Wu from Pexels: https://www.pexels.com/photo/aerial-view-of-green-grass-field-7564340/



(a) Satellite view (Non-truncated) (b) Satellite view (Truncated)

Figure 1: Satellite view of our fictional farmland, with molehill sites shown in yellow. The true, non-truncated density is shown in (a), and (b) shows the private farmland as highlighted in red and blue. Here the only molehills shown are the ones we observe.

is complex enough that the we cannot compute the normalisation constant, but we are still able to simulate data from the truncated distribution for any value of $\theta \in \Theta$.

In Section 2, we will first look into well known simulation based methods for estimating parameter values, both in the Bayesian and frequentist sense: approximate Bayesian computation (ABC), and indirect inference. Here we will introduce the running example of Sections 2 and 3: the α-*stable distribution*, and use it to demonstrate some recurring properties of simulation-based parameter estimation, such as defining a distance metric.

In Section 3, we will introduce the main simulation-based method used in this paper: **Maximum Mean Discrepancy** (MMD). We will start by looking into reproducing kernel Hilbert spaces, the theory that motivates the MMD as a distance between distributions. We will look at the benefits of using this distance for simulation based inference, and how to use it in practice on the α-stable distribution.

In Section 4 we will then look into theory around truncated Gaussian distributions, namely how to simulate from them in the case of simple truncation

boundaries, and then how we can extend this for distributions in more complex truncation boundaries. We will look at how the shape and position of the truncation region affects data simulation and the effectiveness of the MMD. We conclude that the MMD distance contains deterministic noise which will cause problems with optimisation.

In Section 5, we will address the problem of how best to optimise the MMD distance. We will first explain why conventional methods of optimisation are inappropriate in this case, and go on to introduce two non-differentiable optimisation methods: direct search and Bayesian optimisation. Once defined, we will investigate how to modify these methods to work best in the case of noisy data, and go on to apply both to the MMD distance.

Finally in Section 6 we explore the effectiveness of the MMD estimator in different scenarios, such as the effect of both the location and shape of the truncation region. We then reintroduce the ABC method as an alternative method, and attempt to apply it to the problem of truncated domains. We conclude by comparing overall the performance of the MMD estimator with the performance of this other non-likelihood estimation method, as well as extensions to the MMD estimator.

2 Likelihood-free Estimation Methods

We first look at two potential methods for estimating parameters of a model without using the direct form of the likelihood, *indirect inference* and *approximate Bayesian computation*. These two methods are similar in the sense that they both define some measure of distance that we wish to minimise with respect to a parameter θ. For both of these, as with any method used in this paper, we will assume that although the likelihood is intractable, we can still simulate generated data from the model.

2.1 Indirect Inference

The first such likelihood-free method is *indirect inference*, first proposed by Gourieroux et al. (1993)[1]. The motivation behind this method is that for an initial model $f(.|\theta)$ and parameter θ, if the likelihood (or log-likelihood) $l(\theta|.)$ is intractable, then we aim to *replace* this likelihood with an alternative one that *is* tractable.

Formally, we introduce an **auxiliary model** $\tilde{f}(.|\beta)$ with auxiliary parameter β and auxiliary likelihood $\tilde{l}(\beta|.)$. The auxiliary model is chosen to best approximate the true distribution, and the choice of model will be crucial in the effectiveness of indirect inference.

The distance defined in this case is a *distance between parameters*, namely an estimated parameter from the observed data and an estimated parameter from simulated data. We begin by using maximum likelihood estimation to estimate $\hat{\beta}$ using the observed data y, i.e.

$$\hat{\beta} = \arg\min_{\beta} \tilde{l}(\beta|y)$$

Then for a given value of θ, we can generate simulated data $\tilde{y}(\theta) \sim f(.|\theta)$ using our true distribution's generator. For each simulation, we again use the data to optimise our auxiliary model and its parameter to get an estimate

$$\tilde{\beta}(\theta) = \arg\min_{\beta} \tilde{l}(\beta|\tilde{y}(\theta))$$

And repeating this simulation S times for improved accuracy, we can define our simulated estimated parameter $\tilde{\beta}_S(\theta)$ as:

$$\tilde{\beta}_S(\theta) = \frac{1}{S}\sum_{s=1}^{S} \tilde{\beta}_s(\theta)$$

We then define our **indirect inference estimator** $\hat{\theta}_{II}$ as the initial parameter that minimises the distance between $\hat{\beta}$ and $\tilde{\beta}_S(\theta)$, with the distance defined as:

$$D_{II}(\theta, \Omega) = [\hat{\beta} - \tilde{\beta}_S(\theta)]^T \Omega [\hat{\beta} - \tilde{\beta}_S(\theta)]$$

Where Ω is some symmetric, positive definite matrix. Hence $\hat{\theta}_{II} = \arg\min_{\theta\in\Theta} D_{II}(\theta, \Omega)$[2]. A general algorithm for indirect inference is given in algorithm 1.

Alternatively, we could define the distance using the *score* of the auxiliary model:

$$D^*_{II}(\theta, \Sigma) = \left[\frac{d\tilde{l}}{d\hat{\beta}} - \frac{d\tilde{l}}{d\tilde{\beta}_S(\theta)}\right]^T \Sigma \left[\frac{d\tilde{l}}{d\hat{\beta}} - \frac{d\tilde{l}}{d\tilde{\beta}_S(\theta)}\right]$$

Where again Σ is a symmetric positive definite matrix , and our estimator is defined as $\hat{\theta}^*_{II} = \arg\min_{\theta\in\Theta} D^*_{II}(\theta, \Sigma)$[3].

It is worth noting as well that for $\hat{\theta}_{II}$ to be identifiable, we need an auxiliary model with a parameter such that $\dim(\beta) \geq \dim(\theta)$. Moreover if our initial and auxiliary parameter are of equal dimension, then asymptotically,

Algorithm 1 Indirect Inference

Let $\{\theta_1, ..., \theta_N\}$ be the values of θ we consider in the algorithm.
Calculate $\hat{\beta} = \arg\min_\beta \tilde{l}(\beta|y)$.
for i in $\{1, ..., N\}$ **do**
  for s in $\{1, ..., S\}$ **do**
    Simulate data $\tilde{y}(\theta_i) \sim f(.|\theta_i)$
    Calculate $\tilde{\beta}_s(\theta_i) = \arg\min_\beta \tilde{l}(\beta|\tilde{y}(\theta_i))$
  Calculate $\tilde{\beta}_S(\theta_i) = \frac{1}{S}\sum_{s=1}^{S} \tilde{\beta}_s(\theta_i)$
  Calculate $D_{II}(\theta, \Omega) = [\hat{\beta} - \tilde{\beta}_S(\theta)]^T \Omega [\hat{\beta} - \tilde{\beta}_S(\theta)]$
Choose $\hat{\theta}_{II} = \arg\min_{\theta \in \{\theta_1, ..., \theta_N\}} D_{II}(\theta, \Omega)$

i $\hat{\theta}_{II}$ becomes independent of Ω

ii $\hat{\theta}^*_{II}$ becomes independent of Σ

iii $\hat{\theta}_{II} = \hat{\theta}^*_{II}$, i.e. our estimate calculated using the score is equal to our original estimate.

2.1.1 Choice of Auxiliary Model

Evidently the only 'parameter' that needs to be chosen in this method is the choice of auxiliary model and auxiliary parameter. Whilst there is no general gold-standard auxiliary model to use in any case, several papers have put forward metrics to maximise the efficiency of the II estimation given the auxiliary parameter.

One school of thought is to focus on *moment matching.* Here the literature refers to the score function $d\tilde{l}(\beta; .)/d\beta$ and calls the auxiliary model the 'score generator'. Here the score generator is defined using the method of moments approach, and whilst an auxiliary score function that encompasses the true score function would be ideal, Gallant and Tauchen (1996)[4] show that the efficiency of estimating $\hat{\theta}$ can be maximised when the auxiliary score function *asymptotically spans* the true score.

Also, for a good finite-sample performance of the indirect inference estimator, it is beneficial to conserve the number of moments used in the moment matching. Some authors suggest minimising the Akaike information

criterion (AIC) in this case:

$$AIC = 2K - 2\log\left(\frac{d\tilde{l}(\beta;.)}{d\beta}\right)$$

Where K is the number of matched moments used in the score function. Similarly we can also use the Bayesian information criterion (BIC):

$$BIC = K\log(n) - 2\log\left(\frac{d\tilde{l}(\beta;.)}{d\beta}\right)$$

Where n is the number of observations. However Frazier and Renault (2019)[5] disagree with the use of AIC and BIC. They argue that the need for the auxiliary model to be well-specified is unnecessary, and the fact that asymptotic efficiency can be achieved even with a span of the true score negates the need for model selection criterion.

2.1.2 α-Stable example

To illustrate this method, we will use the example of the **α-stable distribution**, which has 4 parameters:

- $\alpha \in (0, 2]$ the key stability parameter,
- $\beta \in [-1, 1]$ the skewness parameter,
- $\gamma \in (0, \infty)$ the scale parameter,
- $\delta \in (-\infty, \infty)$ the location parameter.

The α-stable distribution is a general family of distributions, defined by the following property: Consider a variable X, and two realisations of X, labelled X_1' and X_2', and four constants a, b, c, d such that $a, b, c > 0$. X is said to have a **stable distribution** if $aX_1' + bX_2'$ has the same distribution as $cX + d$ for any a, b, c, d[6]. This family contains many well-known special cases, such as the Gaussian distribution when $\alpha = 2$, and the Lévy distribution when $\beta = 0$. Whilst there are ways of defining the characteristic function for this distribution, it has no way to express the density analytically. This works well for us as the distribution has no tractable density function, yet we are able to simulate from it in R. We will continue to use this as a running example throughout the first half of this paper.

We start by considering an 'observed' dataset of 5000 datapoints, which in reality are randomly generated data from an α-stable distribution with $\alpha = 1.8, \beta = 0.7, \gamma = 1, \delta = 0$.

As an auxiliary model $\tilde{f}(.|\beta)$, an appropriate one in this case would be using a normal distribution to estimate the data with auxiliary parameter $\beta = (\mu, \sigma^2)$. However one property of the α-stable distribution is that there does not exist a second moment for $\alpha < 2$, and no first moment for $\alpha < 1$. [7] Therefore assuming $1 < \alpha < 2$, it only makes sense to match the first moments and use $\beta = \mu$ as our auxiliary parameter. Recall then that we must have that $\dim(\beta) \geq \dim(\theta)$ for $\hat{\theta}_{II}$ to be identifiable, so in this case we can only estimate $\theta = \alpha$ itself.

Therefore assuming that β, γ, δ are known, we define our true distribution's generator as $f(.|\alpha, 0.7, 1, 0) = f(.|\alpha)$ and the auxiliary distribution as $\tilde{f}(.|\mu)$. Then we generate N potential values of α uniformly from $U(1, 2)$, labelled $\alpha_1, ..., \alpha_N$. For each one, we simulate S datapoints $\tilde{y}(\alpha_i) \sim f(.|\alpha_i)$, then find our maximum likelihood estimate $\tilde{\mu}_S(\alpha_i) = \arg\max_{\mu \in \mathbb{R}} \tilde{l}(\mu|\tilde{y}(\alpha_i))$. We then calculate the distance between our auxiliary parameter from the simulated data $\tilde{\mu}_S(\alpha_i)$ and our auxiliary parameter from the observed data $\hat{\mu}$.

$$\hat{\alpha}_{II} = \arg \min_{\alpha \in \{\alpha_1, ..., \alpha_N\}} [\hat{\mu} - \tilde{\mu}_S(\alpha_i)]^T \Omega [\hat{\mu} - \tilde{\mu}_S(\alpha_i)]$$

And as we can see from figure 2, using N=1000, S=10000, and $\Omega = I$, the log of this distance is indeed minimised by the true value of $\alpha = 1.8$

2.2 Approximate Bayesian Computation

Similar to the previous method, this is also used in the case where we do not have an explicit form of the likelihood to work with, but we do have a way of generating data from it. Once again we define a measure of distance that we wish to minimise, except in this case we look at the distance between observed and simulated *data*, rather than observed and simulated *parameters*. Also now we are working in the Bayesian case, so our output will be in the form of a posterior distribution of θ, rather than a point estimate of θ itself.

As before, we begin with our initial model $f(.|\theta)$ with intractable likelihood $l(\theta|.)$ and parameter θ. First we assign a prior distribution π to θ which captures any prior assumptions about the model. Then as in the case of indirect inference, we choose a value of θ (This time taken as $\theta \sim \pi(.)$)

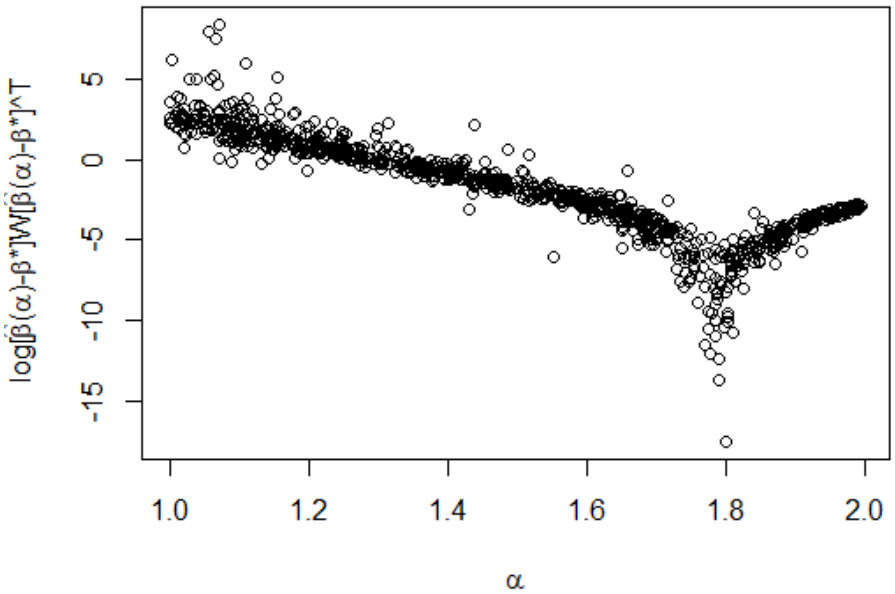


Figure 2: Uniformly random α between 1 and 2 vs. the distance between auxiliary parameters. Clearly this distance is minimised around $\alpha = 1.8$

and generate simulated data $\tilde{y}(\theta) \sim f(.|\theta)$.

We then define some distance D between the observed data y and the simulated data $\tilde{y}(\theta)$, and only accept this proposed θ if the distance falls below a defined tolerance value ϵ. Alternatively, and most commonly in practice, we also apply *summary statistics* S to the data to reduce dimensionality whilst still retaining information.

For data $y \in \mathbb{R}^n$, common summary statistics include the mean $\bar{y}$, the variance $Var(y)$, order statistics:

$$y_{(1)} \ \cdots, \ \ y_{(n)}$$

Or empirical percentiles of the data:

$$P_0, \quad P_{25}, \quad P_{50}, \quad P_{75}, \quad P_{100}$$

Where $P_i = \frac{i(n+1)}{100}$ is the estimated percentage of data found before before i. For multidimensional data, we can apply these summary statistics to each variable, or use dimensionality-reduction methods to combine variables.

For all of these, we are choosing to change the original dimension n of our data to the dimension m of our summary statistic. In the case of the mean and variance, this statistic is one-dimensional by default, although for the order statistics and percentiles we are given a choice of how many dimensions to use.

The simplest way of achieving this is using rejection sampling. We start with our intractable likelihood $f(.|\theta)$ we can sample from , a prior distribution $\pi(\theta)$ on θ, and our distance function D.

Algorithm 2 ABC Rejection Sampling algorithm

```
for i in 1 : N do
    Generate a proposal θ' from the prior π(.)
    Simulate data ỹ from the generator f(.|θ')
    if D(S(ỹ), S(y)|ε) ≤ ε then
        set θ_i = θ'
```

It can be shown that the resulting samples $\{\theta_1, ..., \theta_M\}$ (M being the number of accepted values) are samples from the joint distribution:

$$\pi_\epsilon(\theta, \tilde{y}|y) = \frac{\pi(\theta) f(\tilde{y}|\theta) \mathbb{I}(\tilde{y} \in A)}{\int_A \int_\Theta \pi(\theta) f(\tilde{y}|\theta) d\tilde{y} d\theta}$$

Where $\mathbb{I}$ is the indicator function, Θ is the support of θ, and A is the acceptance region defined by D,y, and ϵ. Then given a suitable choice of tolerance value, this can produce an approximation to the posterior distribution of θ.[8]

$$\pi_\epsilon(\theta|y) = \int_A \pi_\epsilon(\theta, \tilde{y}|y) d\tilde{y} \approx \pi(\theta|y)$$

Clearly this algorithm depends a lot on the choice of tolerance value. if ϵ is too small, then we risk not having many accepted values M out of N iterations, and we will end up with a less accurate posterior estimate. However if ϵ is too large, we accept too many proposal values and lose information about the distribution of θ.

Whilst rejection sampling ABC is generally the simplest option, it gives heavy influence to our choice of prior $\pi(\theta)$, so we can avoid this by using more complex sampling methods such as ABC-SMC and ABC-MCMC.

Algorithm 3 Likelihood-free MCMC sampling

```
Generate an initial value of (θ_0, ỹ_0) using Algorithm 2
for i in 1 : N do
    Generate proposal θ' from Markovian kernel q(.|θ_{i-1})
    Simulate data ỹ' from the generator f(.|θ')
    Simulate random U ~ U[0,1]
    if U ≤ π(θ')q(θ_{i-1}|θ') / π(θ_{i-1})q(θ'|θ_{i-1}) AND D(S(ỹ'), S(y)|ε) ≤ ε then
        Set (θ_i, ỹ_i) = (θ', ỹ')
    else
        Set (θ_i, ỹ_i) = (θ_{i-1}, ỹ_{i-1})
Return (θ_0, ..., θ_N)
```

Here the acceptance probability,

$$\alpha(\theta, \theta') = \frac{\pi(\theta') q(\theta|\theta')}{\pi(\theta) q(\theta'|\theta)}$$

does not require the likelihood so we are still able to use it in this case. We also now need to pick a Markovian kernel to generate θ as well as a prior

distribution for it, although this will allow the algorithm to search through values of θ much more efficiently.

When choosing a measurement of distance, there are many choices to choose from. Two popular ones are the Gaussian distance

$$D_G(x, y|\epsilon) = \exp\left(- \frac{||x - y||^2}{2\epsilon^2} \right)$$

Where $||.||$ is the Euclidean norm, or the Laplace distance using the absolute distance.

$$D_L(x, y) = \exp\big(- |x - y|\big)$$

(Note that data may need to be first sorted before calculating this distance in some cases.)

As for the choice of ϵ, there is no accepted standard choice. In general a lower value is better, although too low and you risk the sampler becoming inefficient. If you are looking at a constant value of ϵ, you can calibrate it using posterior predictive checks after simulation, and Bayesian p-values for different values of ϵ.[9]

Other improvements include replacing the distance metric with a kernel density, or introducing *regression weightings*. This is an improvement suggested by Beaumont, Zhang and Balding (2002)[10]. The idea here is that up till now in the algorithm, once a proposed $(\theta', \tilde{y}')$ gives a small enough distance $D(y, \tilde{y}'|\epsilon)$ such that it is accepted, we lose the information given by the value of that distance. I.e. two values of θ where one gives a large distance and one gives a small distance (both less than ϵ however) are treated equally.

Their paper proposes a change to this by giving weights to accepted parameter values based off their distance. They introduce a standard linear regression model for the parameter estimates:

$$\theta_i = \alpha + (S(y) - S(\tilde{y}'))^T \beta + \xi_i$$

Where α and β are coefficients to be estimated via least squares estimation, and ξ are uncorrelated random variables with zero mean and equal variance. Note that in practice, the linearity assumptions required for this method

would generally be unreasonable across the whole model, but may be able to implement locally around y. Hence we would instead use *local* linear regression in the region around y.

2.2.1 Choice of Summary Statistic

Note: This section is inspired by a paper I have previously written on ABC: 'Portfolio B' as part of the 'Data Science Toolbox' unit.

However, the most key 'parameter' choice within ABC is the choice of *summary statistic*. In general, $S(.)$ is a function mapping our n-dimensional data into a m-dimensional summary statistic, which can be useful to reduce the dimensionality of the data we are working with whilst still retaining enough information for inference. Even if m is equal to or larger than n, it can still prove useful in making our ABC method more efficient as we will go on to show.

However this now introduces the problem of how to select such a statistic. There is no 'one size fits all' choice of $S(.)$ which will be appropriate for all models. Both the calculation of the statistic, and more importantly, its dimension m are generally subjective choices made by the user depending on the model and parameters.

Prangle and Fearnhead (2011)[11] propose a method for dealing with the so called 'curse of dimensionality'. Consider the variance of the ABC-MCMC method, which they showed to be $Var_{ABC}(\theta|s)/M$, where $M = N\int \hat{\pi}(\theta|s)\pi(\theta)d\theta$ is our number of accepted proposal values and $s = S(y)$. They also exchange the distance function $D(x, y)$ for a kernel $k(z)$, where $z = x - y$. Hence the *Monte Carlo error* is going to increase inversely proportional to the average acceptance probability $\int \hat{\pi}(\theta|s)\pi(\theta)d\theta$, which gives rise to the following Lemma:

Lemma: Assume either:

i The marginal distribution $\pi(s)$ is continuous at s and that kernel $k(.)$ has finite support.

ii $\pi(s)$ is continuously differentiable, $|\delta\pi(s)/\delta s_i|$ is bounded above for all i, and $\int |x_i|k(x)dx$ is bounded for all i.

Then as $\epsilon \to 0$,

$$\int \hat{\pi}(\theta|s)\pi(\theta)d\theta = \pi(s)\epsilon^m + o(\epsilon^m)$$

This is the basis for the curse of dimensionality. Whilst we want the dimension of our summary statistic m to be large enough to capture enough information, we don't want to increase the size of ϵ^m too much to minimise the Monte Carlo error. It is important to strike a balance between information and accuracy when choosing dimension, and so ideally we would want a low-dimensional summary statistic that is still relatively informative about θ.

Prangle and Fearnhead (2011) propose that this ideal summary statistic is in fact the *true posterior expectation* of θ. They show this using a quadratic loss function between a true θ^* and an estimated $\hat{\theta}$, using $p \times p$ positive-definite matrix A:

$$L(\theta^*, \hat{\theta}; A) = (\theta^* - \hat{\theta})^T A(\theta^* - \hat{\theta})$$

Then by using this as a metric to minimise when implementing ABC, they provide a theorem that shows the optimal summary statistic in this case is $S(y) = \mathbb{E}[\theta|y]$. *Note that a different loss function would have emitted a different optimal statistic, e.g. using the absolute loss function gives the posterior median instead.*

Alternatively, Prangle (2015)[12] introduces many alternative methods for summary statistic selection. Out of these, two main categories are *subset selection* and *projection*, both of which require simulation from the model to create training data.

Subset selection works by first proposing candidate summary statistics $S = (S_1, .., S_n)$, and then runs ABC many times to try and find the most informative subset. This has the advantage of interpretability if the candidate statistics have intuitive interpretations, although the computational cost of rerunning ABC is a major drawback, especially if the dimension n is large. *Projection methods* begin with k data features $z_1, ..., z_k$, and then attempt to find a low-dimension projection using simulated training data. This method has a far lower computational cost and has the advantage of being adaptable to almost any problem.

2.2.2 α-Stable example

We now return to the α-stable distribution from Section 2.1.2 to see if ABC methods are similarly effective. Using the same data as before, we initially assume γ and δ are known and define the generator as $f(.|\alpha, \beta)$ with priors $\pi_\alpha \sim U(1,2)$, $\pi_\beta \sim U(0,1)$ and the distance function as simply the Euclidean norm:

$$D(x, y) = ||x - y||_2$$

Then for N iterations, we generate $\tilde{\alpha} \sim \pi_\alpha, \tilde{\beta} \sim \pi_\beta$ and simulate M datapoints $\tilde{y} \sim f(.|\tilde{\alpha}, \tilde{\beta})$. We calculate the distance $D = D(y, \tilde{y})$ for each iteration, and sorting by D, we only accept the smallest p percentage of values $(\tilde{\alpha}, \tilde{\beta})$. So using N=10000,M=5000 and p=0.01, we can plot the approximate densities of our accepted values to get the posteriors as shown in figure 3, which do seem to converge around the true values of $\alpha = 1.8$ and $\beta = 0.7$.

We can also try to use this method in the case where we don't assume to know any of the parameters. We first generate new non-trivial 'observed' data y from an α-stable distribution with parameters $\alpha = 1.8, \beta = 0.7, \gamma = 1.2$, and $\delta = 0.5$, and define priors $\pi_\alpha \sim U(1,2), \pi_\beta \sim U(0,1), \pi_\gamma \sim U(0,5)$, and $\pi_\delta \sim N(1,2)$. Even with N=50000, M=5000 and p=0.001 shown in figure 4, the method is very effective in the case of γ and δ, but falls short in the case of α and β, both of which have significantly high posterior variance.

2.3 Combining Indirect Inference and ABC

One connection between the two methods is that we can actually implement indirect inference *inside* ABC, by using it to estimate a summary statistic.

Thus for a problem that we wish to approach using ABC, we first consider using indirect inference. We then use the heuristics as described in section 2.1.1 to propose an appropriate auxiliary model with auxiliary parameter β. Then defining this as:

$$\hat{\beta}(y) = \arg\min_{\beta} \tilde{l}(\beta; y)$$

We can define our summary statistic within ABC to be $S(y) = \hat{\beta}(y)$. I.e. for observed or simulated data y, we apply the auxiliary model to the data and

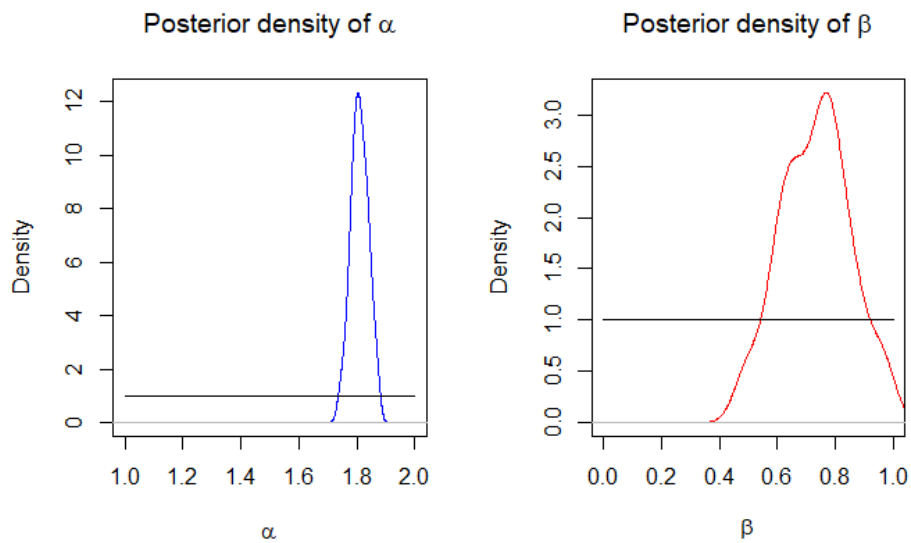


Figure 3: Prior vs. Posterior distribution estimates for α and β. The uniform priors are shown in black and the posteriors shown in red and blue. Whilst the estimate of α seems to perform better, both show signs of converging to a tight posterior around the true value

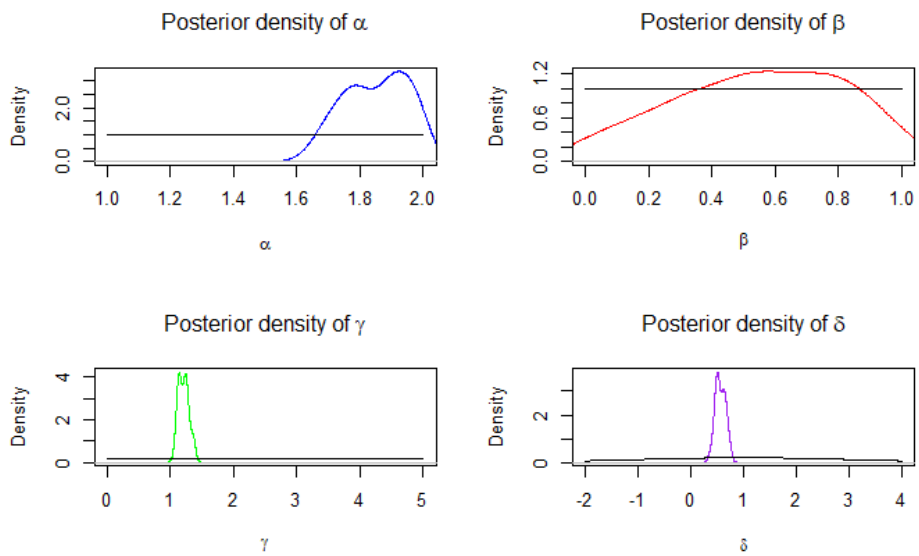


Figure 4: Prior vs. Posterior estimates for all 4 parameters $\alpha, \beta, \gamma, \delta$. Again prior distributions are shown in black and posterior estimates are shown in colour. Here the new estimates for γ and δ seem to converge very well, whilst the ones for α and β aren't as nice in the previous case where γ and δ were presumed known.

use the likelihood $\tilde{l}(\beta; y)$ to estimate β, then use this value as our summary statistic of y.[13]

3 Maximum Mean Discrepancy

We now look at another non-likelihood estimation method in more detail: *maximum mean discrepancy* (MMD). Similarly to both indirect inference and approximate Bayesian computation, maximum mean discrepancy uses simulation from our intractable distribution to try and find a choice of parameter θ that minimises a distance D. Most of this chapter will be dedicated to how we define this distance.

3.1 Reproducing Kernel Hilbert Spaces

We first introduce the idea of a *reproducing kernel* which defines a unique Hilbert space of functions[14]. This space, whilst complex and not so well understood, has many important properties that are essential to the definition and derivation of MMD. We will begin by setting out many key ideas about this space, and how to define one.

3.1.1 RKHS Properties

Definition 3.1: Let $\mathcal{X}$ be a non-empty set. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a **kernel** if there exists an $\mathbb{R}$-Hilbert space and a map $\phi : \mathcal{X} \to \mathcal{H}$ such that $\forall x, x' \in \mathcal{X}$,

$$k(x, x') := \langle \phi(x), \phi(x') \rangle_{\mathcal{H}}$$

Where $\langle ., . \rangle$ is the inner product defined on $\mathcal{H}$. Note that this does not impose any restrictions on $\mathcal{X}$ itself. In fact, we do not even need an inner product defined on $\mathcal{X}$, only $\mathcal{H}$. There are many useful results about the construction of kernels using other kernels, such as the sum of kernels being a kernel, and the product of kernels also being kernels. Two examples are the **Exponential kernel**

$$k(x, x') = \exp(\langle x, x' \rangle)$$

and the **Gaussian kernel**

$$k(x, x') = \exp\left(- \frac{||x - x'||^2}{\sigma^2} \right)$$

Definition 3.2: A symmetric function $f : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is **positive definite** if $\forall n \geq 1$,$\forall (a_1, ..., a_n) \in \mathbb{R}^n$,$\forall (x_1, ..., x_n) \in \mathcal{X}^n$,

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j f(x_i, x_j) \geq 0$$

This leads into an important property of kernels; that they all are positive definite by definition. In fact, taking k and ϕ as defined in Definition 3.1, we can simply show

$$\begin{aligned}
\sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j k(x_i, x_j) &= \sum_{i=1}^{n} \sum_{j=1}^{n} \langle a_i \phi(x_i), a_i \phi(x_j) \rangle_{\mathcal{H}} \\
&= \left\langle \sum_{i=1}^{n} a_i \phi(x_i), \sum_{j=1}^{n} a_j \phi(x_j) \right\rangle_{\mathcal{H}} \left\| \sum_{i=1}^{n} a_i \phi(x_i) \right\|_{\mathcal{H}}^2 \\
&\geq 0
\end{aligned}$$

This leads us to the most important property of all: the **reproducing property** of RKHS kernels. Formally, this is defined as follows:

Definition 3.3: (RKHS) Let $\mathcal{H}$ be a Hilbert space of functions defined on a set $\mathcal{X} \subset \mathbb{R}$. Then a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is said to be a *Reproducing Kernel* of $\mathcal{H}$, and $\mathcal{H}$ hence a *Reproducing Kernel Hilbert Space*, if,

1. $\forall x \in \mathcal{X}, k(., x) \in \mathcal{H}$

2. $\forall x \in \mathcal{X}, \forall f \in \mathcal{H}, \langle f, k(., x) \rangle_{\mathcal{H}} = f(x)$

With the second property being the *reproducing property*. Equivalently, we can also define a reproducing kernel Hilbert space as such[15]:

Alternative Definition:(RKHS) $\mathcal{H}$ is a *Reproducing Kernel Hilbert Space* if $\forall x \in \mathcal{X}$, there exists some positive constant $C \in \mathbb{R}$ such that,

$$|f(x)| \leq C||f||_{\mathcal{H}}$$

where the norm of f is either given by the inner product in $\mathcal{H}$, or the *supremum norm*: $||f|| = \sup_{x \in \mathcal{X}} |f(x)|$. This then leads to two useful results of which we will use when defining the MMD. Firstly, we have that for any $x, y \in \mathcal{X}$,

$$k(x, y) = \langle k(., x), k(y, .) \rangle_{\mathcal{H}}$$

i.e. the inner product of two kernels in the RKHS is also a kernel in the RKHS. We also have that for any function $f \in \mathcal{H}$, we can represent f as a linear sum of kernels:

$$f(.) = \sum_{i=1}^{n} a_i k(x_i, .)$$

Which is one of the main important properties that makes the RKHS such a useful space of functions.

3.2 Motivation

The maximum mean discrepancy stems from the initial definition of an *integral probability metric*, a general way of defining distance between two probability distributions $\mathbb{P}$ and $\mathbb{Q}$.

$$\gamma_{\mathcal{F}}(\mathbb{P}, \mathbb{Q}) := \sup_{f \in \mathcal{F}} \left| \int_M f \, d\mathbb{P} - \int_M f \, d\mathbb{Q} \right|$$

Where M is our metric space and $\mathcal{F}$ a class of all real-valued bounded measurable functions on M. By substituting $M = \mathbb{R}^n$ and $\mathcal{F} = \{f : ||f||_{\mathcal{H}_k} \leq 1\}$ where $\mathcal{H}_k$ is our reproducing kernel Hilbert space with kernel k, this leads to our first definition of the maximum mean discrepancy.[16]

$$
\begin{aligned}
\sup_{\{f:||f||_{\mathcal{H}_k}\leq 1\}} \left| \int_{\mathbb{R}^d} f \, d\mathbb{P} - \int_{\mathbb{R}^d} f \, d\mathbb{Q} \right| &= \sup_{\{f:||f||_{\mathcal{H}_k}\leq 1\}} |\mathbb{E}_{X\sim\mathbb{P}}[f(X)] - \mathbb{E}_{Y\sim\mathbb{Q}}[f(Y)]| \\
&= \sup_{\{f:||f||_{\mathcal{H}_k}\leq 1\}} \left| \mathbb{E}_{X\sim\mathbb{P}}[\langle k(X,.), f\rangle_{\mathcal{H}_k}] - \mathbb{E}_{Y\sim\mathbb{Q}}[\langle k(Y,.), f\rangle_{\mathcal{H}_k}] \right| \\
&= \sup_{\{f:||f||_{\mathcal{H}_k}\leq 1\}} \left| \langle \mu_{\mathbb{P}}, f\rangle_{\mathcal{H}_k} - \langle \mu_{\mathbb{Q}}, f\rangle_{\mathcal{H}_k} \right| \\
&= \sup_{\{f:||f||_{\mathcal{H}_k}\leq 1\}} \left| \langle \mu_{\mathbb{P}} - \mu_{\mathbb{Q}}, f\rangle_{\mathcal{H}_k} \right|
\end{aligned}
$$

By basic properties of inner products, where $\mu_{\mathbb{P}}$ is the *kernel mean embedding*:

$$\mu_{\mathbb{P}} = \mathbb{E}_{X\sim\mathbb{P}}[k(X,.)]$$

Then since,

$$|\langle \mu_{\mathbb{P}} - \mu_{\mathbb{Q}}, f\rangle_{\mathcal{H}_k}| \leq ||\mu_{\mathbb{P}} - \mu_{\mathbb{Q}}||_{\mathcal{H}_k}||f||_{\mathcal{H}_k}$$

By the Cauchy-Schwarz inequality[17], and,

$$\langle \mu_{\mathbb{P}} - \mu_{\mathbb{Q}}, \mu_{\mathbb{P}} - \mu_{\mathbb{Q}}\rangle_{\mathcal{H}_k} = ||\mu_{\mathbb{P}} - \mu_{\mathbb{Q}}||^2_{\mathcal{H}_k}$$

We have that:

$$\sup_{\{f:||f||_{\mathcal{H}_k}\leq 1\}} \left| \langle \mu_{\mathbb{P}} - \mu_{\mathbb{Q}}, f\rangle_{\mathcal{H}_k} \right| = ||\mu_{\mathbb{P}} - \mu_{\mathbb{Q}}||_{\mathcal{H}_k}$$

This gives us our first definition of the MMD.[18]

Definition 3.4: (Maximum Mean Discrepancy) The Maximum Mean Discrepancy is the distance $D_k(\mathbb{P}, \mathbb{Q})$ between two probability distributions with kernel k.

$$D_k(\mathbb{P}, \mathbb{Q}) = ||\mathbb{E}_{X\sim\mathbb{P}}[k(X,.)] - \mathbb{E}_{Y\sim\mathbb{Q}}[k(Y,.)]||_{\mathcal{H}_k}$$

where $\mathcal{H}_k$ is the reproducing kernel Hilbert space defined by kernel k, and

X, Y variables distributed according to $\mathbb{P}, \mathbb{Q}$ respectively.

Whilst this is the main definition of the MMD, it is much more useful to work with the **squared** MMD, as it has a much better form due to the properties of the RKHS.

$$\begin{aligned}
D_k^2(\mathbb{P}, \mathbb{Q}) &= ||\mu_{\mathbb{P}} - \mu_{\mathbb{Q}}||_{\mathcal{H}_k}^2 \\
&= \langle \mu_{\mathbb{P}}, \mu_{\mathbb{P}} \rangle_{\mathcal{H}_k} + \langle \mu_{\mathbb{Q}}, \mu_{\mathbb{Q}} \rangle_{\mathcal{H}_k} - 2 \langle \mu_{\mathbb{P}}, \mu_{\mathbb{Q}} \rangle_{\mathcal{H}_k} \\
&= \mathbb{E}_{X,X' \sim \mathbb{P}}[\langle k(X, .), k(., X') \rangle_{\mathcal{H}_k}] + \mathbb{E}_{Y,Y' \sim \mathbb{Q}}[\langle k(Y, .), k(., Y') \rangle_{\mathcal{H}_k}] - 2\mathbb{E}_{X \sim \mathbb{P}, Y \sim \mathbb{Q}}[\langle k(X, .), k(., Y) \\
&= \mathbb{E}_{X,X' \sim \mathbb{P}}[k(X, X')] + \mathbb{E}_{Y,Y' \sim \mathbb{Q}}[k(Y, Y')] - 2\mathbb{E}_{X \sim \mathbb{P}, Y \sim \mathbb{Q}}[k(X, Y)]
\end{aligned}$$

Then by replacing our mean embedding $\mu_{\mathbb{P}}$ with its empirical estimate $\hat{\mu} = \frac{1}{m} \sum_{i=1}^{m} k(x_i, .)$ based off sample data $\mathbf{x}$, we get the empirical estimator of the MMD:

$$\hat{D}_k^2(\mathbb{P}, \mathbb{Q}) = \frac{1}{m^2} \sum_{i,j=1}^{m} k(x_i, x_j) + \frac{1}{n^2} \sum_{i,j=1}^{n} k(y_i, y_j) - \frac{2}{mn} \sum_{i,j=1}^{m,n} k(x_i, y_j)$$

where the x_i's and y_i's are observed realisations from distributions $\mathbb{P}$ and $\mathbb{Q}$, with m and n the number of these observations respectively.

3.3 Characteristic Kernels

Clearly the way we define the MMD is very reliant on our choice of k and $\mathcal{H}_k$. For this distance to be a proper metric between probability distributions, we need to have that $D_k(\mathbb{P}, \mathbb{Q}) = 0 \implies \mathbb{P} = \mathbb{Q}$. Whilst our use of the norm means that $D_k(\mathbb{P}, \mathbb{Q}) = 0 \implies \mu_{\mathbb{P}} = \mu_{\mathbb{Q}}$, this does not necessarily mean that $\mathbb{P} = \mathbb{Q}$ for all kernels k.

For this we require the notion of a **characteristic kernel**, which is a kernel which makes the map $\mathbb{P} \to \mu_{\mathbb{P}}$ *injective*. i.e, if $\mu_{\mathbb{P}} = \mu_{\mathbb{Q}}$, then we do have that $\mathbb{P} = \mathbb{Q}$ as desired. However it can be difficult to check for this injectivity in practice.

3.3.1 Motivating Example

To understand which kind of kernels can be used as opposed to others, consider two kernels $k_1(x, y) = e^{xy}$ and $k_2 = (xy + c)^2$, and two probability

distributions $\mathbb{P} = \mathcal{N}(0,1)$ and $\mathbb{Q} = U(-\sqrt{3}, \sqrt{3})$.

First consider k_1. Here we have $k_1(X,t) = e^{Xt}$ (t arbitrary). If we perform a Taylor expansion on this kernel with respect to X, we get,

$$k_1(X,t) = 1 + Xt + \frac{t^2}{2!}X^2 + \frac{t^3}{3!}X^3 + ...$$

Then taking the expectation of this,

$$\begin{aligned}\mathbb{E}_{X\sim\mathbb{P}}[k_1(X,t)] &= 1 + t\mathbb{E}_{X\sim\mathbb{P}}[X] + \frac{t^2}{2!}\mathbb{E}_{X\sim\mathbb{P}}[X^2] + \frac{t^3}{3!}\mathbb{E}_{X\sim\mathbb{P}}[X^3] + ... \\ \implies \mu_{\mathbb{P}} &= 1 + t\mathbb{E}_{X\sim\mathbb{P}}[X] + \frac{t^2}{2!}\mathbb{E}_{X\sim\mathbb{P}}[X^2] + \frac{t^3}{3!}\mathbb{E}_{X\sim\mathbb{P}}[X^3] + ...\end{aligned}$$

Hence in this case, the kernel mean embedding contains information for an infinite amount of moments, so we can uniquely determine a distribution $\mathbb{P}$ from $\mu_{\mathbb{P}}$. However this is not the case for k_2. Here the Taylor expansion is finite, and following the same steps as before we have that,

$$\mu_{\mathbb{P}} = c^2 + 2tc\mathbb{E}_{X\sim\mathbb{P}}[X] + t^2\mathbb{E}_{X\sim\mathbb{P}}[X^2]$$

Now the embedding is only made up of the first and second moment, which is a problem since in the example we are considering, $\mathbb{P}$ and $\mathbb{Q}$ have equal first and second moments, whilst being wildly different distributions. Hence our mean embedding is unable to distinguish the two and means the MMD defined by it cannot be used as a proper metric.[19]

Remark: By considering the most basic kernel $k(x,y) = xy$, then the MMD looks similar to the indirect inference approach used in section 2.1.2. There we essentially looked at comparing the first moments of two distributions and using the distance between them as our distance between distributions. Now using characteristic kernels, we can expand on this to look at the differences between all moments and define a better distance.

3.3.2 Characterisation

The problem is now to be able to say which kernels, or classes of kernels, are in fact characteristic. Whilst there exist several criteria to guarantee

a kernel is indeed characteristic, most require assumptions upon the kernel that can be difficult to check in practice. Sriperumbudur et al. (2008)[20] proposes several theorems for kernels that are *translation invariant*, i.e. kernels k such that $k(x, y) = \phi(x - y)$ where ϕ is some bounded, continuous, $\mathbb{R}$-valued positive definite function on $\mathbb{R}^n$. We introduce a key theorem and corollary from the paper here, although the proof is beyond the scope of this paper.

Theorem 3.1: Let $\mathcal{F}$ be a unit ball in an RKHS $\mathcal{H}_k$ defined by kernel k on $\mathbb{R}^n$. Suppose ϕ is translation-invariant, and that ϕ can be expressed as a Fourier transform of a finite non-negative Borel measure Λ on $\mathbb{R}^n$:

$$\phi(x) = \int_{\mathbb{R}^n} \exp(-ix^T) d\Lambda(\omega) \quad \forall x \in \mathbb{R}^n$$

Then k is a *characteristic kernel* for $\mathcal{P}$, the family of all probability distributions defined on $\mathbb{R}^d$, if and only if the support of Λ is $\mathbb{R}^n$.

The conditions for this theorem are somewhat inconvenient to verify for a given kernel, but fortunately the theorem emits a more practical corollary:

Corollary 3.1: Let $\mathcal{F}$ be a unit ball in an RKHS $\mathcal{H}_k$ defined by a kernel k on $\mathbb{R}^n$. Suppose ϕ is translation invariant and the support of ϕ is *compact*. Then k is a characteristic kernel for $\mathcal{P}$.

Checking the compactness of the support of ϕ is evidently much more efficient than verifying whether the support of the Borel measure is exactly $\mathbb{R}^n$. Especially since in most cases we will consider, the support of ϕ can be directly calculated from the support of the observed data or model parameters, which are known.

Alternatively, Sriperumbudur et al. (2009)[21] propose a criterion for defining characteristic kernels without the requirement of being translation invariant: instead being *strictly positive definite*. Note that this is not referring to the common definition of strict p.d., i.e. $\forall n \geq 1, \forall (a_1, ..., a_n) \in \mathbb{R}^n, \forall (x_1, ..., x_n) \in \mathcal{X}^n$,

$$\sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j k(x_i, x_j) > 0$$

But instead defining strictly postive definiteness using Borel measures:

Definition 3.5: Let T be some topological space. A measurable and bounded kernel k is said to be strictly positive definite if and only if,

$$\int_T \int_T k(x, y) d\Lambda(x) d\Lambda(y) > 0$$

For all finite, non-zero signed Borel measures Λ defined on T. This then leads to the key theorem given in the paper:

Theorem 3.2: If k is strictly positive definite on a space T as in definition 3.5, then k is a characteristic kernel for $\mathcal{P}$.

Again the proof of this result is too complex to include here, although conveniently, it can be shown that for $T = \mathbb{R}^n$, both the Gaussian and Exponential kernel are strictly positive definite in this case. This implies that both kernels are characteristic for any family of probability distributions defined on $\mathbb{R}^n$, which covers all distributions we will be considering in this paper. Even better, Sriperumbudur et al. also shows that both the Gaussian and Matérn kernel defined on $\mathbb{R}^n$ yield consistent estimates of the MMD distance $D_k^2(\mathbb{P}, \mathbb{Q})$ when using the empirical estimator $\hat{D}_k^2(\mathbb{P}, \mathbb{Q})$. (Both these kernels will be used later in the paper).

3.4 Parameter Estimation

Now that we have a well-defined distance between two probability distributions using a characteristic kernel, we can use this distance for parameter estimation in a similar way to the previous non-likelihood estimation methods. Consider again that we have observed data y which we believe to be from a distribution $f(.|\theta)$ (Denoted f_θ.) with unknown parameter θ. We define our **MMD estimator** to be,

$$\hat{\theta} = \arg \inf_{\theta \in \Theta} D_k^2(f_\theta, \hat{f}_n)$$

Where $\hat{f}_n$ is the empirical distribution based off the observed data:

$$\hat{f}_n = \frac{1}{n} \sum_{i=1}^{n} \delta_{y_i}$$

By plugging f_θ and $\hat{f}_n$ into our formula for D_k^2, we can derive a much more

convenient form of the estimator.[22]

$$D_k^2(f_\theta, \hat{f}_n) = \mathbb{E}_{Y,Y'\sim f_\theta}[k(Y,Y')] + \mathbb{E}_{X,X'\sim \hat{f}_n}[k(X,X')] - 2\mathbb{E}_{Y\sim f_\theta, X\sim \hat{f}_n}[k(X,Y)]$$

$$\implies D_k^2(f_\theta, \hat{f}_n) = \mathbb{E}_{Y,Y'\sim f_\theta}[k(Y,Y')] + \mathbb{E}\left[k\left(\frac{1}{n}\sum_{i=1}^n \delta_{y_i}, \frac{1}{n}\sum_{j=1}^n \delta_{y_j}\right)\right] - 2\mathbb{E}_{Y\sim f_\theta}\left[k\left(\frac{1}{n}\sum_{i=1}^n \delta_{y_i}, Y\right)\right]$$

$$\implies D_k^2(f_\theta, \hat{f}_n) = \mathbb{E}_{Y,Y'\sim f_\theta}[k(Y,Y')] + \frac{1}{n^2}\sum_{i=1}^n\sum_{j=1}^n k(y_i,y_j) - \frac{2}{n}\sum_{i=1}^n \mathbb{E}_{Y\sim f_\theta}[k(Y,y_i)]$$

Then since the second term is not in terms of θ, we can ignore it when trying to minimise D_k^2 with respect to θ and define:

$$\hat{\theta} = \arg\min_{\theta\in\Theta}\left(\mathbb{E}_{Y,Y'\sim f_\theta}[k(Y,Y')] - \frac{2}{n}\sum_{i=1}^n \mathbb{E}_{Y\sim f_\theta}[k(Y,y_i)]\right)$$

(Note: by removing the second term, we may encounter seemingly negative values of the MMD distance later in the paper. This is acceptable however as we are only interested in minimising the right hand side of this equation, and not its actual value.) The problem now is how to minimise the right hand side of this equation with respect to θ. First, we replace the simulated expectation with its estimate:

$$\mathbb{E}_{Y,Y'\sim f_\theta}[k(Y,Y')] \to \frac{1}{M}\sum_{i=1}^M k(\tilde{Y}_i, \tilde{Y}_i')$$

Where M is the number of independent simulated values of $\tilde{Y}, \tilde{Y}'$ from the distribution f_θ. This leads us to the empirical estimate of the MMD estimator, which we will use for the following example:

$$\hat{\theta} = \arg\min_{\theta\in\Theta}\left(\frac{1}{M}\sum_{i=1}^M k(\tilde{Y}_i, \tilde{Y}_i') - \frac{2}{nM}\sum_{j=1}^n\sum_{i=1}^M k(\tilde{Y}_i, y_j)\right)$$

Where the y_i's are our observed realisations from the distribution. Now we consider how to transform this equation into that we can numerically optimise with respect to θ.

Francois-Xavier et. al.[23] suggest using a method of stochastic gradient descent, although this requires being able to differentiate our likelihood, which

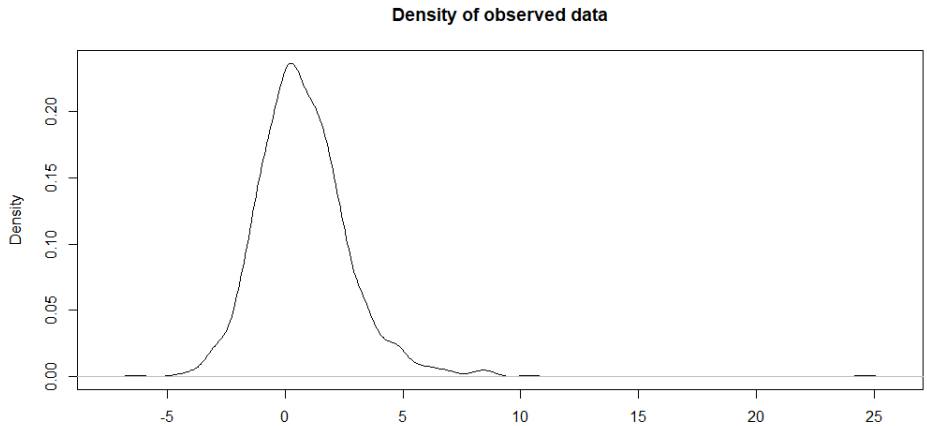


Figure 5: Density of the 1000 observed values. We wish to estimate the parameter θ that generated this distribution. (Note the observed values have one outlier at 25.)

is not possible for us in this case due to the likelihood being intractable. Instead, we look at the way in which we simulate the variables $\tilde{Y}, \tilde{Y}'$. If the formula for these simulations emits a tractable form in terms of θ, we can use this in the place of $\tilde{Y}$ and $\tilde{Y}'$ in the equation above to get an MMD function purely in terms of θ which we can optimise using R.

3.4.1 α-Stable Example

We now return to the α-stable example we visited in section 2.2.2. Here we have 1000 'observed' values from an α-stable distribution with true parameters $(1.8, 0.7, 1.2, 0.5)$, giving a target distribution as seen in figure 5.

Now whilst the α-stable distribution has no analytical expression for its density, it does have a tractable formula for simulating variables which we will use here.[6]

This algorithm generates realisations from an α-stable distribution with parameters $\theta = (\alpha, \beta, \gamma, \delta)$ by simulating only uniform and exponential variables U, W. We define this simulator by $g(U, W, \theta)$.

So by first generating N lots of variables U, U', W and W', we can set these

Algorithm 4 Simulation of N α-stable variables

for i in $1:N$ **do**
  Simulate $U \sim U[\frac{\pi}{2}, \frac{\pi}{2}]$ and $W \sim Exp(1)$
  Set $\zeta = -\beta \tan \frac{\pi\alpha}{2}$
  Set $\xi = \frac{1}{\alpha} \arctan -\zeta$
  Calculate $\tilde{Y}_i' = (1+\zeta^2)^{\frac{1}{2\alpha}} \frac{\sin \alpha(U+\xi)}{(\cos U)^{\frac{1}{\alpha}}} \left(\frac{\cos U - \alpha(U+\xi)}{W}\right)^{\frac{1-\alpha}{\alpha}}$
  Set $\tilde{Y}_i = \gamma \tilde{Y}_i' + \delta$

effectively as 'constants', then replace $\tilde{Y}, \tilde{Y}'$ by $g(U, W, \theta), g(U', W', \theta)$ in our MMD estimator. It is important to keep these randomly generated variables the same across all parameter estimates, so to keep the optimisation purely set on θ. Hence we now have a function in terms of just θ, which we can then optimise using R:

$$M(\theta) = \frac{1}{M} \sum_{i=1}^{M} k\Big(g(U_i, W_i, \theta), g(U_i', W_i', \theta)\Big) - \frac{2}{nM} \sum_{j=1}^{n} \sum_{i=1}^{M} k\Big(g(U_i, W_i, \theta), y_j\Big)$$

We choose our kernel to be Gaussian as it is one of the simplest and most popular characteristic kernels. This does however bring in to question the choice of σ within the kernel,

$$k(x, x') = \exp\left(-\frac{||x - x'||^2}{\sigma^2}\right)$$

Which we will look at by repeating the estimation for various values of σ. Then by assuming that we are in the case of $\alpha \in (1,2), \beta \in (0,1), \gamma \in [0,2], \delta \in [0,2]$, we can use R's inbuilt `optim` function with an initial value of $(1.5, 0.5, 1.1, 0.1)$ and the "L-BFGS-B" method.

```
theta.init<-c(1.5,0.5,1.1,0.1)
optim(theta.init,MMDist,method="L-BFGS-B",
lower=c(1,0,0,0),upper=c(2,1,2,2))
```

I repeated the simulated estimation for six values of σ: $(20, 5, 1, 0.5, 0.2, 0.01)$. Firstly, we look at the parameter estimates for each one. In figure 6 I have plotted the estimates of $\alpha, \beta, \gamma, \delta$ for each value of σ. From these there is

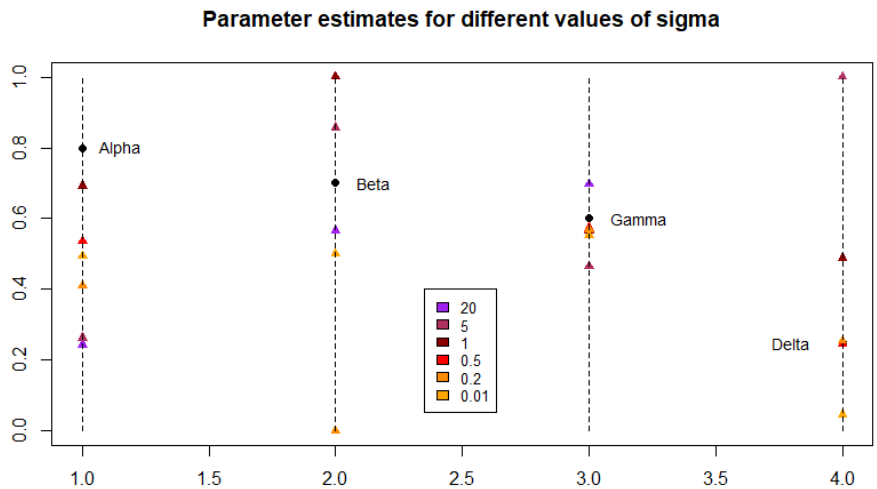


Figure 6: Parameter estimates of $\alpha, \beta, \gamma, \delta$ for each of our six values of σ. Here I have scaled each estimate to be between 0 and 1 for sake of comparing prediction accuracy between parameters.

no clear trend in σ affecting the result. For γ and δ, the lower values of σ seem to be clustered closer around the true values, although for β there are also high values of σ around the true value, and for α none of the estimates seem to get near at all.

However the MMD estimator was designed to minimise the distance between *distributions*, and we can see this in figure 7. Here for each value of sigma, I have used its respective estimate $\hat{\theta}$ to generate 1000 realisations from an α-stable distribution with parameter $\hat{\theta}$. From this I can plot an estimated density alongside the true density to see how close they were. By doing this, we can see that although the actual parameter estimates may have been inaccurate, the distributions they produce do fit the observed data very well, especially as σ decreases. From the look of figure 7, the optimal value of σ seems somewhere between 0.1 and 1.

It is interesting that in this case we can get a parameter estimate that is fairly far from the true value, but still fits the data well enough that we might have mistaken it to be correct had we not known the true value. It

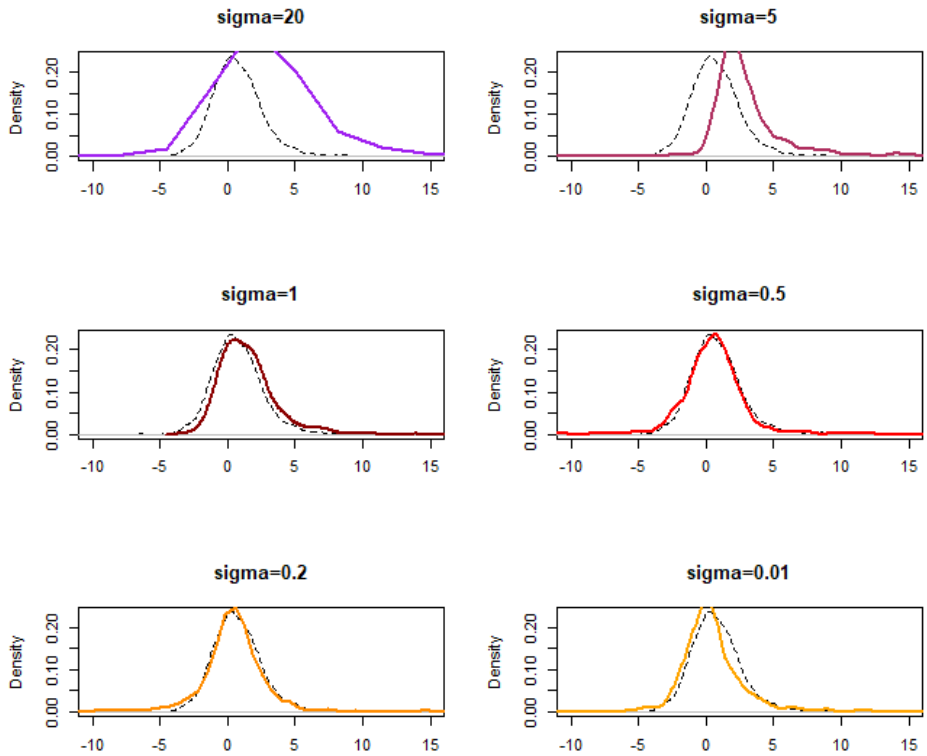


Figure 7: Estimated densities for each value of σ. Here $\sigma = 20$ and $\sigma = 5$ seem to have been affected by the outlier and have too much mass to the right of the true mean, whereas $\sigma \leq 1$ and below fit the data well.

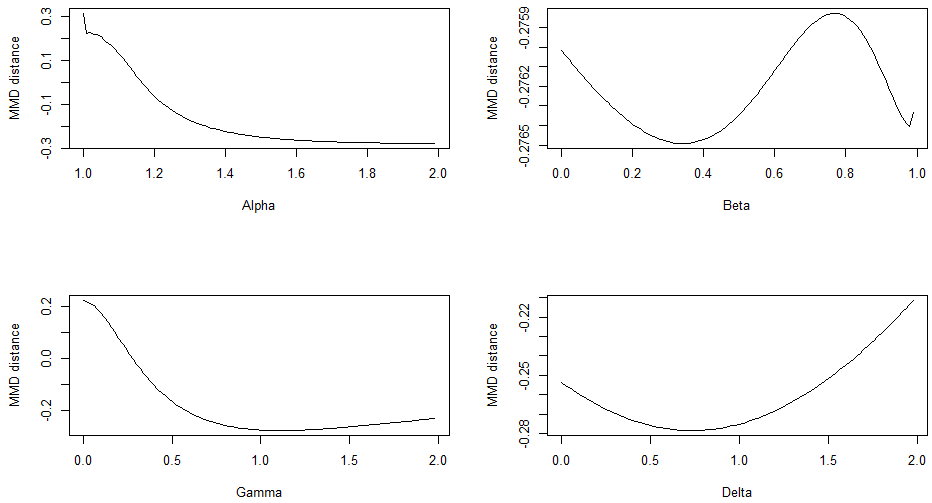


Figure 8: Possible parameters vs. MMD distance for $\alpha, \beta, \gamma, \delta$. In each case, I have set all others parameters to their true value so the MMD is effectively now working in one dimension. (Note: I have used a value of $\sigma = 2$.)

could be that for this example, our MMD function $M(\theta)$ defined in section 3.4 has multiple minima or is not even convex. Out of interest, we can check how this function might be minimised by setting each parameter bar one in turn to the true value, and plot the MMD estimate for one parameter at a time. We can then see in figure 8 that our MMD function seems to work well for γ and δ. It appears convex and is indeed minimised close to the true values of 1.2 and 0.5 respectively. α does not appear to have a minimum and is just strictly decreasing, and β is strangely maximised at the true value instead of minimised. This performance is reminiscent of the our results in section 2.2.2.

4 Truncated Domains

We now consider the case which we will try to apply the method of MMD estimation to: *truncated domains*. These are cases where our data exists within a complicated, finite domain C. This gives rise to the likelihood in these cases being intractable, since when computing the probability density function:

$$f(y|\theta, C) := \frac{f(y|\theta)}{\int_C f(y|\theta)dy}$$

The normalising constant $\int_C f(y|\theta)dy$ is very complicated or potentially impossible to calculate if our subset C is complex.

Recall the motivating example from the introduction, as shown in figure 1. Here we have a boundary complex enough that our likelihood becomes intractable. although not complex enough that we cannot parameterise it to estimate the indicator function $\mathbb{I}(x \in C)$. (We will attempt this later in the chapter). We know this observed data was generated by a bivariate normal distribution with unknown parameters (μ^*, Σ^*) and then truncated by a known truncation boundary C. So to apply the MMD, we need to first define a way of simulating variables $(\tilde{Y}, \tilde{Y}') \sim N_2(\mu, \Sigma)$ subject to $\tilde{Y}, \tilde{Y}' \in C$ for any μ, Σ, C. Hence in this chapter we will be looking at the most efficient methods of simulation given relative values of μ, Σ, and C.

4.1 Truncated Normal Distribution

To first simplify our problem significantly, a first step is to restrict our search of distributions by looking at the **truncated normal distribution**, effectively drawing a rectangle around our domain and sampling $(\tilde{Y}_1, \tilde{Y}_2) \sim N_2(\mu, \Sigma)$ with the added constraint that $a_i < \tilde{Y}_i < b_i$ for $i \in \{1, 2\}$. This allows us to have a much larger acceptance rate when we use rejection sampling later for our complex domain.

There exist many ways of sampling from the truncated normal distribution, most being in the form of a rejection sampler or a Gibbs sampler. We will look at both here, although it would be best for the MMD estimator to emit a simulator in an analytical form, similarly to how we constructed the method in section 3.4.1.

4.1.1 Rejection Sampling

There are many different methods for rejection sampling to generate truncated normal variables from regular normal variables. For now we consider the univariate case where we have a variable $X \sim N(\mu, \sigma^2)$ restricted to $X \in [a, b]$. In the case where the acceptance region is large and contains a lot of the data, the problem is fairly trivial, and we would use **normal**

rejection sampling: simulate $X \sim N(\mu, \sigma^2)$ and accept if $X \in [a, b]$. This works well if the mean is well contained in the acceptance region.

Other more robust methods include **half-normal rejection sampling**: simulate $X \sim N(\mu, \sigma^2)$ and accept if $|X| \in [a, b]$, and **Uniform rejection sampling**: simulate $X \sim U[a, b], u \sim U[0, 1]$, and accept if

$$u < \frac{F_Y(X|\mu, \sigma^2)}{F_Y(x^*|\mu, \sigma^2)}$$

Where $F_Y(y|\mu, \sigma^2)$ is the cumulative normal density for a variable $Y \sim N(\mu, \sigma^2)$ and $x^* = \arg\max_{x \in [a,b]} F_Y(x; \mu, \sigma^2)$.

Geweke (1998)[24] gives an algorithm of when to use each method depending on the size of the acceptance region relative to the distribution. For the case of a standard normal distribution in one dimension, truncated to the set $[a, b]$ (a and b both finite), they propose the algorithm shown in algorithm 5 for choosing a rejection sampling method.

Algorithm 5 Algorithm for choice of rejection sampling method

For known μ and σ^2, let $\phi(x) = f(.|\mu, \sigma^2)$ be the density function of a $N(\mu, \sigma^2)$ distribution.
if $0 \in [a, b]$ **then**

  if $\phi(a) \leq 0.15$ OR $\phi(b) \leq 0.15$ **then**

    use **normal rejection sampling**
  else
    use **uniform rejection sampling**
else if $a > 0$ **then**

  if $\frac{\phi(a)}{\phi(b)} \leq 2.18$ **then**

    use **uniform rejection sampling**
  else if $\frac{\phi(a)}{\phi(b)} \leq 0.15$ AND $a < 0.725$ **then**

    use **half-normal rejection sampling**

The case of $b < 0$ is symmetric to the case where $a > 0$, so we do not include it here. However the case still remains of when our acceptance region is only a small subset of the support of our probability density, most often one of the tails. In this case it would be where the truncation region is $[a, +\infty)]$ and $\phi(a)$ is close to zero. In general, it would be the case where the mean of the distribution is significantly outside the acceptance region. Evidently the issue here is that regular accept/reject algorithms are going to have a very small acceptance probability, leading to an excessively long run time when trying to simulate new data.

Geweke (1998) proposes a rejection sampling method that relies upon the truncated normal distribution resembling an exponential distribution as $a \to \infty$ in the truncation region $[a, +\infty)$. In this case we now simulate a random exponential random variable Z from kernel $\exp(-aZ)$ for $Z \geq a$. We then accept Z with acceptance probability $\exp(-\frac{1}{2}(Z-a)^2)$. The idea of this method is it increases the acceptance probability relative to a standard rejection sampling method.

Alternatively, Maatouk and Bay (2016)[25] suggest an improvement to the Von Neumann rejection method using the mode, which similarly increases the acceptance rate of random samples. First we define the Von Neumann rejection method in the multivariate case.

Von Neumann: Consider a convex truncation set $C \subset \mathbb{R}^n$, with f and g two truncated densities defined only on C such that $f(x) < kg(x) \forall x$. Then the following algorithm generates a variable $X \sim f(.)$:

1. Generate a random variable $X \sim g(.)$
2. Generate a random uniform variable $U \sim U[0, 1]$
3. If $kg(X)U < f(X)$, accept X. Otherwise repeat steps 1 and 2.

So now we need to find a suitable proposal density g that bounds f above, where f is our desired bivariate normal restricted to C, i.e.

$$f(x|\mu, \Sigma, C) = \frac{1}{2\pi|\Sigma|} \exp\left(\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right) \mathbb{I}\{x \in C\}$$

First, we assume $\mu = 0$ without loss of generality and calculate the mode x^*

of f, i.e. $x^* = \arg\max_{x\in\mathbb{R}^n} f(x|0,\Sigma,C)$. This can be simplified down to,

$$x^* = \arg\min_{x\in C} \frac{1}{2}x^T\Sigma x$$

Then by setting our proposal density to f with mean shifted to the mode $\mu^* = x^*$, we have that this bounds f above as desired in the Von Neumann algorithm, with the smallest value of k being $k^* = \exp(-\frac{1}{2}(\mu^*)^T\Sigma^{-1}\mu^*)$

Proof: Suppose we have $f(x|0,\Sigma,C) < kg(x|\mu^*,\Sigma,C)$. Then the smallest value of k is given by,

$$k^* = \arg\max_{x\in C} \exp\left(\frac{1}{2}(\mu^*)^T\Sigma^{-1}\mu^* - x^T\Sigma^{-1}\mu^*\right)$$

Then the angle between the gradient vector $\Sigma^{-1}\mu^*$ of the function $\frac{1}{2}x^T\Sigma^{-1}x$ at the mode μ^*, and the vector $(x-\mu^*)$ is acute since our truncation set C is convex. Hence $(x-\mu^*)\Sigma^{-1}\mu^*$ is non negative for all $x \in C$, which gives $k^* = \exp(\frac{1}{2}(\mu^*)^T\Sigma^{-1}\mu^*)$, which is positive. $\square$

This leads us to the 'rejection sampling from the mode' (RSM) algorithm, defined similarly to the Von Neumann algorithm.

RSM algorithm: Consider a convex truncation set $C \subset \mathbb{R}^n$ with $f(x;0,\Sigma,C)$ denoting a zero-mean truncated normal density. First calculate $\mu^* = \arg\min_{x\in C} \frac{1}{2}x^T\Sigma^{-1}x$ and set the proposal density to $g(x) = f(x;\mu^*,\Sigma,C)$. Then,

1. Generate a random variable $X \sim g(.)$
2. Generate a random uniform variable $U \sim U[0,1]$
3. If $k^*g(X)U < f(X)$, where $k^* = \exp(\frac{1}{2}(\mu^*)^T\Sigma^{-1}\mu^*)$, accept X. Otherwise repeat steps 1 and 2.

And for simplicity, we can also simplify Step 3 to become:

$$k^*g(X)U < f(X)$$
$$\implies \exp\left(-\frac{1}{2}(\mu^*)^T\Sigma^{-1}\mu^*\right)\exp\left(\frac{1}{2}x^T\Sigma^{-1}x\right)U < \exp\left(\frac{1}{2}(x-\mu^*)^T\Sigma^{-1}(x-\mu^*)\right)$$
$$\implies U < \exp\left(\frac{1}{2}\mu^*)^T\Sigma^{-1}\mu^*\right)\exp\left(-\frac{1}{2}x^T\Sigma^{-1}x\right)\exp\left(\frac{1}{2}(x-\mu^*)^T\Sigma^{-1}(x-\mu^*)\right)$$
$$\implies U < \exp\left((\mu^*)^T\Sigma^{-1}\mu^* - X^T\Sigma^{-1}\mu *\right)$$

Note that in practice, we can either compute μ^* analytically, or use the 'optim' function in R once again as in section 3.4.1. As for generating $X \sim g(.)$, this forces us to use a basic rejection sampling algorithm, although since the mean is now captured in the truncation region, we should have a much higher acceptance probability than before.

In fact, we can show this by considering a simple example of a bivariate normal distribution with zero-mean and Σ such that $\sigma_{11} = 0.5$, $\sigma_{12} = \sigma_{21} = 0$, and $\sigma_{22} = 0.05$. We will use a square truncation region C as a sliding window as shown in figure 9, which we will move further away from the origin. We contrast the RSM method with the normal 'crude' method of simply simulating $X \sim N(0, \Sigma)$ and accepting if $X \in C$. For each window, we will use 10,000 iterations of each method and see how many variables are successfully generated to estimate the acceptance probability.

The results are shown in table 1: Clearly in the case where the truncation region is close to the mean, the two methods seem to work about the same. However once the region approaches the tail end of the distribution, we can see that the RSM method is far more efficient, and hence can be used for generating truncated normals with much less computational time.

% of data in C	Acc. Prob. of RSM	Acc. Prob. of normal method
49	47.6	48.0
35	35.9	33.0
20	27.7	19.6
10	22.3	9.8
4	18.7	4.5
2	15.8	1.8

Table 1: Results of experiment generating truncated normal distributions using the *rejecting sampling using the mode* method and the *normal rejection sampling* method.

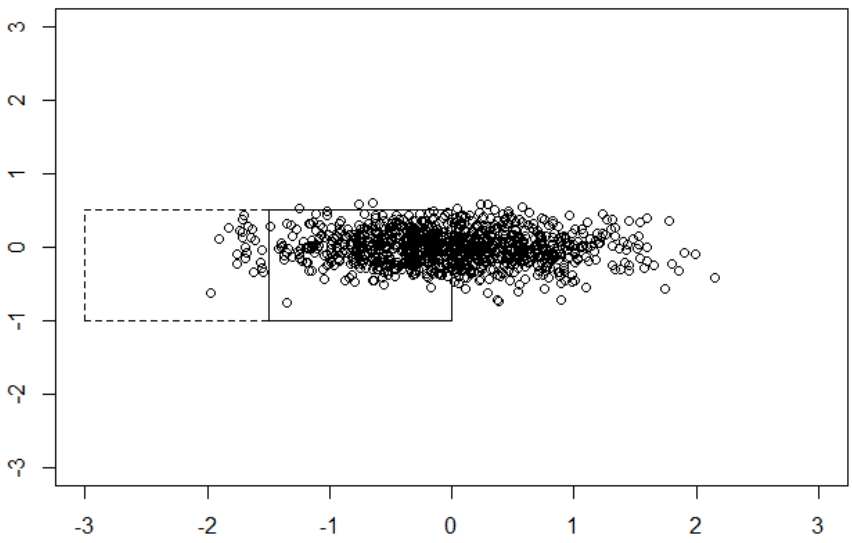


Figure 9: 1000 datapoints from the bivariate $N(0, \Sigma)$ distribution. Here our truncation region C is a sliding window starting at the black square (48% of the data) and moving left to the dotted square (2% of the data).

4.1.2 Gibbs Sampling

Alternatively, we can try and simulate these truncated normal variables in a way which does not require any rejection sampling. One way of doing this is by using a Gibbs sampler in two dimensions using transformed normal variables, as proposed by Kotecha and Djuric (1999) [26] . In this method, we first need to define a way to simulate truncated normal variables in **one** dimension without using rejection sampling.

Simulating a Truncated Normal: Suppose we have a variable $Y \sim N(\mu, \sigma^2)$ truncated on a region $C = [a, b]$. Define $F_X(x)$ to be the non-truncated, cumulative density of the normal, i.e. $F_X(x) = \Phi((x - \mu)/\sigma)$. Then to sample y, we use this formula:

1. Sample $x \sim N(\mu, \sigma^2)$
2. Map x to y by the transformation

$$y = F_X^{-1}\big(F_X(x)(F_X(b) - F_X(a)) + F_X(a)\big)$$

Note that now when $x = -\infty$, $y = a$, and when $x = \infty, y = b$. Also since $F_X(x)$ is a cumulative distribution function, it is monotonically increasing, so our transformation from x to y is also monotonically increasing. Hence our transformation is well defined and will be one-to-one.

So now that we have a formula for generating truncated normals in one dimension, we need to extend this to define a Gibbs sampler for **bivariate** truncated normal distributions. Firstly, let $f_{X,Y}(x, y)$ denote the joint probability density of $X, Y \sim N_2(\mu, \Sigma)$, and $f_{X|Y}(x|y)$ and $f_{Y|X}(y|x)$ the one-dimensional conditional densities of $X|Y$ and $Y|X$.

Gibbs Sampler in Two Dimensions: Begin by setting an initial value of (x_0, y_0), where x_0, y_0 are contained in our truncation region $C = [a, b] \times [c, d]$. Then for $t = 1, ..., M$:

1. Sample $x_t \sim f_{X|Y}(.|y_{t-1})$
2. Sample $y_t \sim f_{Y|X}(.|x_t)$
3. Return (x_t, y_t)

This will generate M samples of our joint density $f_{X,Y}(x,y)$. So in the case where this density is a bivariate normal distribution $N_2(\mu,\Sigma)$:

$$f_{X,Y}(x,y) = \frac{1}{2\pi\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right)$$

Where $\mathbf{x} = (x,y)^T$, $\mu = (\mu_1,\mu_2)$, and $\Sigma = \left(\begin{smallmatrix}\sigma_{11} & \sigma_{12}\\ \sigma_{12} & \sigma_{22}\end{smallmatrix}\right)$ we can derive the conditional density:

$$\begin{aligned}
f_{X|Y}(x|y) &= \frac{f_{X,Y}(x,y)}{f_Y(y)} \\
&\propto \frac{1}{2\pi\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right) \\
&\propto \exp\left(-\frac{1}{2}(x-\mu_1, y-\mu_2)\frac{1}{\sigma_{11}\sigma_{22}-\sigma_{12}^2}\begin{pmatrix}\sigma_{22} & -\sigma_{12}\\ \sigma_{-12} & \sigma_{11}\end{pmatrix}\begin{pmatrix}x-\mu_1\\ y-\mu_2\end{pmatrix}\right) \\
&\propto \exp\left(-\frac{1}{2(\sigma_{11}\sigma_{22}-\sigma_{12}^2)}\left[\sigma_{22}(x-\mu_1)^2 - 2\sigma_{12}(x-\mu_1)(y-\mu_2) + \sigma_{11}(y-\mu_2)^2\right]\right) \\
&\propto \exp\left(-\frac{1}{2(\sigma_{11}\sigma_{22}-\sigma_{12}^2)}\left[\sigma_{22}(x^2-2x\mu_1+\mu_1^2) - 2\sigma_{12}(xy - x\mu_2 - y\mu_1 + \mu_1\mu_2)\right]\right) \\
&\propto \exp\left(-\frac{1}{2(\sigma_{11}\sigma_{22}-\sigma_{12}^2)}\left[x^2(\sigma_{22}) - x(2\mu_1\sigma_{22} + 2\sigma_{12}y - 2\sigma_{12}\mu_2)\right]\right) \\
&\propto \exp\left(-\frac{\sigma_{22}}{2(\sigma_{11}\sigma_{22}-\sigma_{12}^2)}\left[\left(x - \frac{\mu_1\sigma_{22}+\sigma_{12}y-\sigma_{12}\mu_2}{\sigma_{22}}\right)^2\right]\right)
\end{aligned}$$

Hence we have that $X|Y \sim N(\mu^*, \sigma^{*2})$, where

$$\mu^* = \mu_1 + \frac{\sigma_{12}}{\sigma_{22}}(y-\mu_2), \qquad \sigma^{*2} = \sigma_{11} - \frac{\sigma_{12}^2}{\sigma_{22}}$$

Note that if X and Y are independent, i.e. $\sigma_{12} = 0$, then this reduces to simply using the marginal density for X, and in this case it is unnecessary to use a Gibbs sampling approach. By using a near-identical derivation, we can also get the other conditional density for $Y|X$. Now we are ready to define our Gibbs sampler for truncated normals.

Gibbs Sampler for Truncated Normals: Begin by setting an initial value of (x_0, y_0), where x_0, y_0 are in our truncation region $C = [a,b]\times[c,d]$. Then for $t = 1, ..., M$:

1. Sample $x_t' \sim N(\mu_x^*, \sigma_x^{*2})$, where $\mu_x^* = \mu_1 + \frac{\sigma_{12}}{\sigma_{22}}(y_{t-1}-\mu_2)$ and $\sigma_x^{2*} = \sigma_{11} - \frac{\sigma_{12}^2}{\sigma_{22}}$

2. Map x'_t to x_t by $x_t = F_X^{-1}\big(F_X(x'_t)(F_X(b) - F_X(a)) + F_X(a)\big)$, where $F_X(x)$ is the cumulative density function of a $N(\mu_x^*, \sigma_x^{2*})$ distribution.

3. Sample $y'_t \sim N(\mu_y^*, \sigma_y^{*2})$, where $\mu_y^* = \mu_2 + \frac{\sigma_{12}}{\sigma_{11}}(x_t - \mu_1)$ and $\sigma_y^{2*} = \sigma_{22} - \frac{\sigma_{12}^2}{\sigma_{11}}$

4. Map y'_t to y_t by $y_t = F_Y^{-1}\big(F_Y(y'_t)(F_Y(d) - F_Y(c)) + F_Y(c)\big)$, where $F_Y(y)$ is the cumulative density function of a $N(\mu_y^*, \sigma_y^{*2})$ distribution.

5. Return (x_t, y_t)

We can test this using a bivariate normal with $\mu = (-0.5, -0.1)$ and $\Sigma = \left(\begin{smallmatrix} 0.1 & 0.05 \\ 0.05 & 0.15 \end{smallmatrix}\right)$ with truncation boundary $C = (-\infty, -0.5] \times (-\infty, 0]$, i.e. the bottom left quadrant of the data. We originally had 500 datapoints, and by sampling about a quarter of this using the Gibbs-sampling method, we get a truncated sample as shown in figure 10 that seems to follow the non-truncated distribution quite well.

4.2 Applying the MMD

Now that we have multiple methods of simulating truncated normal distributions, we are able to begin applying the maximum mean discrepancy method to the problem. The idea here is whatever the shape of our truncation region, we will aim to draw the smallest possible rectangle around the region and use truncated normals to limit our simulation to this area. For this then, it would probably be better to use the Gibbs sampling approach. Firstly, using a rejection sampling method to generate the truncated normals would be somewhat unnecessary as we will then be using rejection sampling methods again to generate samples within the truncation region anyway. This would effectively be using rejection sampling twice in succession, so it would be more efficient to simply use rejection sampling applied to the truncation region.

Secondly, by using the Gibbs sampler, we could replace the sampling from normals in steps (1) and (3) by sampling from uniform variables, i.e. $x'_t = F_X^{-1}(U)$, where $U \sim U(0,1)$ and $F_X(x)$ is the cumulative density of a $N(\mu_x^*, \sigma_x^{*2})$ distribution. This would allow us to modify the method in a similar way as the α-stable distribution in section 3.4.1, where we would generate the uniform variables first, then set them to be constant throughout. Then the Gibbs sampler would effectively become deterministic, and

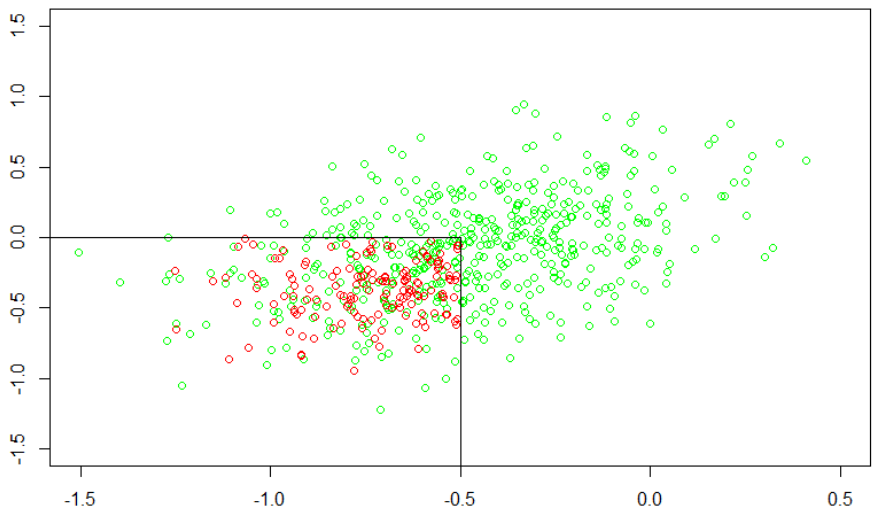


Figure 10: 500 data points as shown in green which follow a bivariate normal distribution. We consider only the bottom left quadrant as a truncation boundary, with our generated Gibbs-samples shown in red. By keeping the number of samples close to the number of observed points in the quadrant, we get a sample that seems to follow the distribution well.

we would eliminate the randomness that would come from the rejection sampling method.

The main downside of the Gibbs sampler however is that it is an estimate of the truncated normal distribution rather than an actual true sample, and whilst a large sample would approach the true distribution, it will never truly be distributed as such. This could potentially affect the accuracy of the estimator, although with a high enough amount of samples generated, this should not be too much of an issue.

4.2.1 Shape of Truncation Region

One issue however is going to be how the shape of the truncation region effects the model. Our current method relies on being able to draw a rectangle around the region and limit our search to this shape, in theory increasing the acceptance probability. However as we can see in figure 11, each of these three shapes acts very differently. Shape A acts well for our method (73.1 % accepted). Shape B is problematic at first (23.6 % accepted), although this can be solved by rotating our data space and re-drawing a surrounding rectangle (64.0 % accepted). Shape C however presents a problem as there is no way to draw a rectangle around it without still having a worryingly low acceptance probability (45.6 % accepted). *(These probabilities were calculated using an image colour summarizer - http://mkweb.bcgsc.ca/color-summarizer/ to estimate the fraction of space the shape takes up out of the rectangle.)*

We will investigate the effect of these differences in acceptance probability on the effectiveness of the MMD estimator later on in the paper.

4.2.2 MMD Algorithm

We are now ready to develop the algorithm for implementing the MMD for a general truncation region C. Recall the empirical MMD estimator from section 3.4:

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \left(\frac{1}{M} \sum_{i=1}^{M} k(\tilde{Y}_i, \tilde{Y}_i') - \frac{2}{nM} \sum_{j=1}^{n} \sum_{i=1}^{M} k(\tilde{Y}_i, y_j) \right)$$

Where y is our observed data and $\tilde{Y} \sim f_\theta$ is our simulated data. So for n observed data points $y_1, ..., y_n$ and a chosen choice of kernel $k(y, y')$, we now

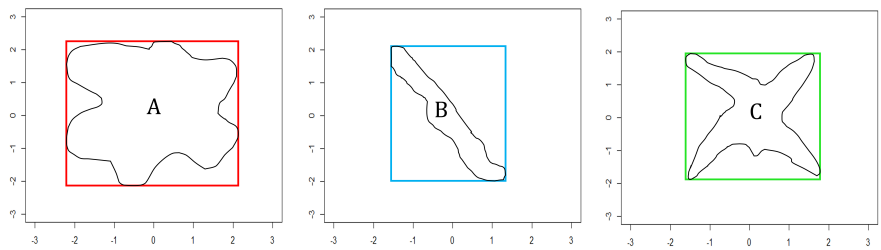


Figure 11: Acceptance probabilites for three different shaped truncation regions. By restricting simulation to the coloured rectangles, shape A has an acceptance probability of 73.1%. Similarly shape C has acc. prob. 45.6%, and shape B has acc. prob. 23.6%, although by rotating the data space and drawing another rectangle we can increase this to 64.0%.

need a method for generating two independent samples $\tilde{Y}, \tilde{Y}'$ of truncated normal variables of length M. Note also that now Y is two-dimensional, so we need to change our Gaussian kernel to,

$$k(x, y) = \exp\left(- \frac{||x - y||^2}{\sigma^2} \right)$$

Now we will combine our Gibbs sampler with rejection sampling. Due to the introduction of rejection sampling, we would also like to put an upper limit on how long we will let the rejection sampling algorithm run before 'giving up'. For instance, trying to generate normal variables with means far outside the truncation region would have an incredibly low acceptance probability. Luckily this should not be an issue as these extreme cases would be for extreme parameter values that we wish to avoid anyway. In instances where our data generation takes too long, we can simply return an 'NA' value and have the MMD return ∞. For a general set C, we will thus use algorithm 6.

Evidently this requires being able to know whether our Gibbs proposal value is in C, so we need to be able to parameterise C in some way. If so however, this algorithm will generate M random realisations of a bivariate normal distribution truncated on any general set C, and will terminate if it takes K or more simulations to achieve this.

Algorithm 6 Simulation of M bivariate normals truncated on a general set C

```
Set Upper-Limit = K, Desired No. of Values = M
Set Initial value = (x̃_0, ỹ'_0)
Create blank list for accepted values A.
Set Count = 0, Accept.Count = 0
for i in 1 : K do
    Generate proposal value (x', y') from (x̃_i, ỹ_i) using the Gibbs sampler
for truncated normals in section 4.1.2.
    if (x', y') ∈ C then
        Add (x', y') to the list of accepted values A.
        Set (x̃_{i+1}, ỹ_{i+1}) = (x', y')
        Increment Accept.Count by 1
    if (x', y') ∉ C then
        Set (x̃_{i+1}, ỹ_{i+1}) = (x̃_i, ỹ_i)
    Increment Count by 1
    if Accept.Count = M then
        Return list of accepted values A
    if Count = K then
        Return 'NA'
```

So as with the α-stable example, we will define our MMD function $M(\theta) = M(\theta|y)$ that we will try to minimise with respect to θ. Denote the output of Algorithm 6 by $g(U_1, U_2, \theta)$, (i.e. $g_i(U_1, U_2, \theta) = (x_i, y_i)$) where U_1, U_2 are the pre-generated $U(0,1)$ random variables used in Gibbs sampling to generate (x', y'). Then $M(\theta)$ is defined as follows:

$$M(\theta) = \frac{1}{M}\sum_{i=1}^{M} k(g_i(U_1, U_2, \theta), g_i(U_1', U_2', \theta)) - \frac{2}{nM}\sum_{j=1}^{n}\sum_{i=1}^{M} k(g_i(U_1, U_2, \theta), y_j)$$

If both $g(U_1, U_2, \theta) \neq$ 'NA' and $g(U_1', U_2', \theta) \neq$ 'NA'. Else, we have that $M(\theta) = \infty$. This way we are effectively putting a 'barrier' around our possible parameter values. Our optimiser cannot end up converging to an extreme parameter, as once our value of θ gets extreme enough that simulating from it is not possible, the MMD simply returns a value of ∞ and any method we use to minimise it will avoid this value of θ.

We can see this in figure 12, where we have a simple $N(\mu, \Sigma)$ distribution ($\mu = (0,0), \Sigma = \left(\begin{smallmatrix} 0.5 & 0 \\ 0 & 0.05 \end{smallmatrix}\right)$) with a circular truncation region centred at (-0.3,0.3) with radius 0.7. Keeping Σ at its true value, we can calculate the MMD for various values of μ over the space using $M = 800, K = 1300$. We can see that although in theory μ could be anywhere in $\mathbb{R}^2$, the MMD returns ∞ for anywhere shown in red. Hence our search for the minimal value of μ is in practice restricted to the blue area. *(Note: the area is not symmetric due to exponential transforms of Σ defined in my code.)* Similarly, we can imagine that for μ outside the truncation region, an extremely small value of σ_{11} or σ_{22} would not return simulations as not enough would be accepted.

So to summarise, our truncated MMD distance function $M(\theta) = M(\mu, \Sigma)$ works as follows:

1. Input parameters $\theta = (\mu, \Sigma)$, observed data y and truncation region C. Choose kernel k with tolerance value σ. Choose number of simulations M and maximum number of attempted simulations K. Define rectangular truncation region $[a_1, b_1] \times [a_2, b_2]$. Pre-generate a suitable amount of uniform random variables $U \sim Unif(0,1)$.

2. Set the pre-generated uniform variables as constants and use these to generate the $X_t' \sim N(\mu_x^*, \sigma_x^{*2})$ and $Y_t' \sim N(\mu_y^*, \sigma_y^{*2})$ for the Gibbs sampler algorithm within Algorithm 6.

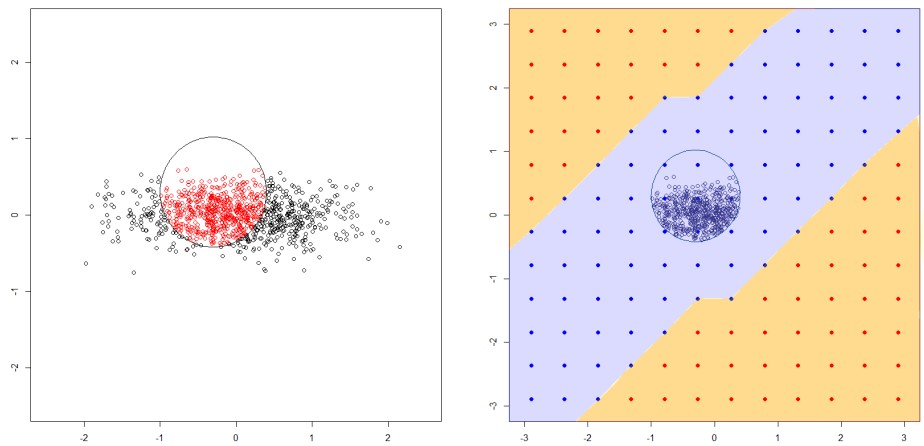


Figure 12: Left: Non-truncated bivariate normal density with truncated observed data shown in red. Right: Gridsearch for values of $\mu \in [-3, 3] \times [-3, 3]$. Only values of μ in the blue region give a finite value of the MMD.

3. Run Algorithm 6 and output M simulations of $(\tilde{Y}, \tilde{Y}')$, where $\tilde{Y}, \tilde{Y}' \sim N_2(\mu, \Sigma)$ subject to $\tilde{Y}, \tilde{Y}' \in C$.

4. Input these simulations into the MMD distance $M(\theta)$ and return the result.

4.2.3 Example

We will now attempt to implement our MMD on our motivating example introduced at the beginning of this paper. Recall we had data distributed as bivariate normal over a farmland, where we only have access to the truncation region defined by the non highlighted area. First of all we need to define the truncation region. We will be required to parameterise this shape, so we fit a fairly simple polygon to the region, where we can calculate whether a point is in the region by dividing the polygon into trapezoids and checking each case. (See figure 13.)

Then once again, we will draw a rectangle around this region, using coordinates $(-1.1, -0.833), (1, -0.833), (1, 0.9), (-1.1, 0.9)$, which gives a rejection sampling acceptance probability of 70.9%. However now that we are in a position to implement the MMD estimator, we encounter a big problem.

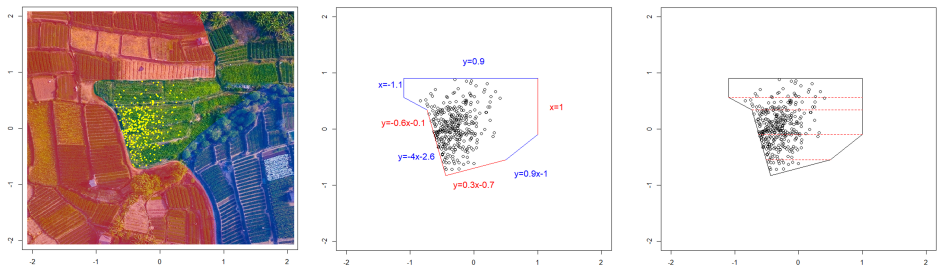


Figure 13: Left: Truncated observed farm data from the motivating example. Middle: seven-sided polygon of truncated region defined by linear equations. Right: Method for defining the set enclosed in the shape. We check the y-value of any point and then check whether it falls in the corresponding trapezoid.

Although we can simulate bivariate normal variables almost as deterministic within the rectangle using the Gibbs sampling method, we are now forced to introduce noise into the MMD by using rejection sampling.

Recall in figure 8 in section 3.4.1 where we investigated the shape of the MMD in one dimension by only changing one parameter, and we saw that whilst the MMD was not always convex, it was smooth in every case. Repeating that analysis now for this example, i.e. keeping all true values constant and only changing the value of μ_1, we get a plot of the MMD as seen in figure 14. Clearly now although the shape of the overall graph does seem convex and minimised around the true value of -0.5, it is no longer smooth. Due to this we will be unable to use our current strategy of simply optimising using R. We could simply search across all possible parameter values and pick a minimiser, although this would be incredibly inefficient, so it is the goal of the next section to find a robust method to minimise this noisy MMD.

5 Noisy Optimisation

This chapter is devoted to investigating various methods of optimisation and assessing which are most appropriate to the problem of minimising the MMD distance. Going forward, we will refer to our MMD distance function for a set of parameters θ given observed data y as $M(\theta; y)$, or simply $M(\theta)$

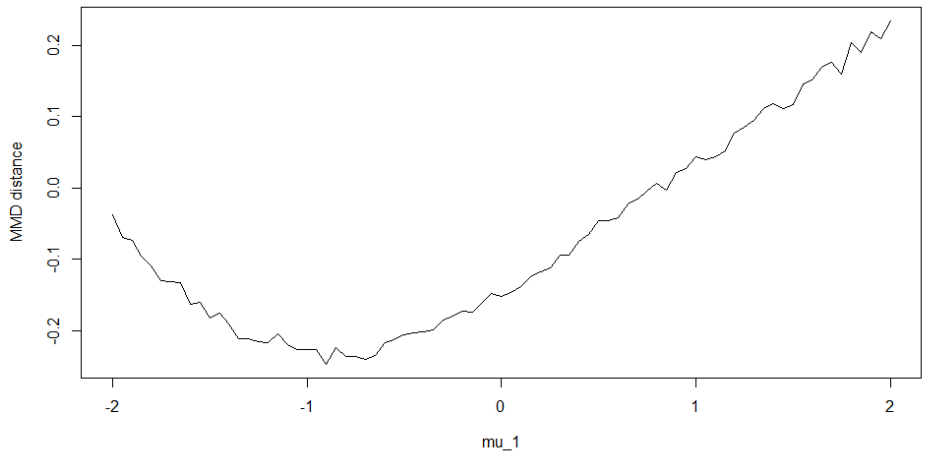


Figure 14: Our MMD distance for our farm example for values of μ_1 between -2 and 2. The true value of μ_1 is -0.5.

for short. This function has two key-properties that we will need to consider when choosing a method of optimisation.

Firstly, due to the random nature of the rejection sampling algorithm within $M(\theta)$, the function contains a lot of noise (As seen in figure 14). Since this noise depends both on the variance of the truncated normal distribution and the shape of the truncation region, its distribution is unknown. The best we can hope for is that the noise is bounded by some constant, and in some methods we will assume that it is iid, although this may not necessarily be the case.

Secondly, again due to the random rejection sampling and also the complexity of the function, a derivative of $M(\theta)$ does not exist. Hence even any derivative-free method attempting to approximate a derivative would be inappropriate in this case, as there is no derivative to approximate. Instead we desire an optimization method that assumes $M(\theta)$ is *non-differentiable*.

Hence when considering optimization methods, descent methods such as Newton's method and BFGS are unavailable to us. Ideally we require a method that can cope with $M(\theta)$ being both **non-differentiable and**

noisy. Also, treating the MMD as a 'black-box' function, we only have access to the parameter inputs θ and their function evaluations $M(\theta)$, so we need an optimization method which only requires only this information. Finally, for a more accurate estimator, it is beneficial for us to use a larger number of simulations within the MMD distance. However this has the effect of making the $M(\theta)$ more expensive to compute, so for the optimisation methods studied here, it will be more advantageous to use a method which uses the least function evaluations.

5.1 Direct Search and Simplex Methods

The first such method is *direct search*. Whilst this method is not necessarily designed for functions with noise, it is appropriate for the case where we only have access to function evaluations. Hence we will define the original form of direct search initially, and then go on to look at ways to adapt the method in the case of noisy functions.

The defining quality of direct search methods, other than the fact they do not compute or approximate derivatives, is the fact that they do not even use the numerical value of the function evaluation. Instead, the method only uses the relative ranks of the function values to estimate the direction towards the minimum[27]. Informally, all direct search methods generally follow these steps.

1. Begin with initial point $\theta_0 \in \Theta$.
2. At each iteration, use other points $\theta'_1, ..., \theta'_k$ and the relative ranking of their function values $M(\theta_i), M(\theta'_1), ..., M(\theta'_k)$ to determine an optimal direction vector d
3. Determine a step length $\triangle$ to move along d
4. Set the new point to be $\theta_{i+1} = \theta_i + \triangle d$
5. Check to see if some convergence test has been satisfied. If so, stop the algorithm. If not, repeat steps 2-4.

5.1.1 Pattern Search

The most basic of such algorithms is pattern search, or *compass search*. The key idea within pattern search is that at every iteration k, the algorithm

generates **trial points** to search for the direction of minimisation. Generally these points are all at an equal step size $\triangle_k$ from current point θ_k. The directions of search are vectors d_k belonging to a *positive spanning set* $\mathcal{D}$[28]. Common choices of $\mathcal{D}$ include the positive and negative orthonormal bases $\mathcal{D} = \{\mathbf{e}_1, ..., \mathbf{e}_n, -\mathbf{e}_1, ..., -\mathbf{e}_n\}$ (which leads to compass search) or $n+1$ vectors forming a minimal positive basis with uniform angles.

Once we have our trial points $\{\theta'_k = \theta_k + \triangle d_k, d_k \in \mathcal{D}\}$ and function evaluations $\{M(\theta'_k), d \in \mathcal{D}\}$, we can decide whether to perform a *contraction* or *acceptance*. If our optimal trial point $\theta'_{min} = \arg\min_\theta\{M(\theta'_k), d \in \mathcal{D}\}$ is smaller than the current point θ_k, then we will accept this as our new iteration θ_{k+1}. However if no trial point is smaller than the current point θ_k, then we can assume that our step size $\triangle_k$ was too large, and we decrease it to contract the search region. A full algorithm can be seen in algorithm 7[29].

Algorithm 7 Pattern search algorithm

```
Set initial value θ_0 and initial step size △_0
Set tolerance value τ
for k = 1,2,... do
    Calculate trial points {θ'_k = θ_k + △d_k, d_k ∈ D}
    Calculate smallest trial point θ'_min = arg min_θ{M(θ'_k), d ∈ D}
    if θ'_min < θ_k then
        (Acceptance)
        Set θ_{k+1} = θ'_min
        Set △_{k+1} = △_k
    if θ'_min > θ_k then
        (Contraction)
        Set θ_{k+1} = θ_k
        Set △_{k+1} = 1/2 △_k
    if △_{k+1} < τ then
        End algorithm
```

Some algorithms use a measure of *sufficient decrease* to determine whether to contract or not instead of simply comparing the relative values of θ_k and θ'_{min}, although this then takes into account more than just the relative ranking of function values. To illustrate this, an example using compass search (i.e. $\mathcal{D} = \{\mathbf{e}_1, \mathbf{e}_2, -\mathbf{e}_1, -\mathbf{e}_2\}$) to find the maximum of a bivariate normal is

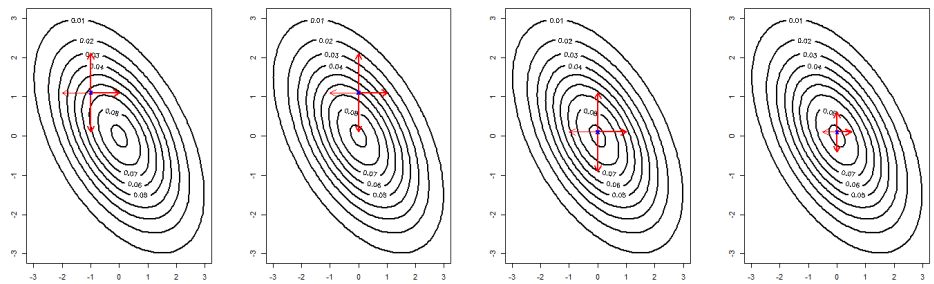


Figure 15: Four iterations of the compass search algorithm. The first graph shows the initial point in blue and trial points indicated by the red arrows. The first two steps are an acceptance, and the final graph shows a contraction.

shown in figure 15.

5.1.2 Nelder-Mead Simplex Methods

The other, more popular direct search method is the *Nelder-Mead* algorithm, which is a type of *simplex* method. Similar to pattern search, simplex methods use the ranking of a few function evaluations to calculate a best direction of minimisation. However instead of trial points, we now introduce the notion of a simplex.

Definition 5.1: A **simplex** is the convex hull $S \subset \mathbb{R}^n$ of $n+1$ points[30].

Hence we now consider the $n-dimensional$ polygon that our points create rather than just the points themselves, and how the ranking of function values at these points affects the geometry of this polygon. This leads to the Nelder-Mead algorithm[31], as described below.

1. Begin with $n+1$ relatively extreme points (vertices) to form a simplex, and evaluate the function at each vertex to get initial values.

2. At each iteration, we identify the vertices which give the highest, second-highest, and lowest function values. We denote these vertices by θ_{high}, θ_{2nd}, and θ_{low}, with their respective function values being M_{high}, M_{2nd}, and M_{low}.

3. **Reflection:** Calculate the *centroid* θ_{cent} of all vertices other than θ_{high}, and generate a new vertex θ_{ref} by reflecting θ_{high} through θ_{cent}, i.e.

$$\theta_{ref} = 2\theta_{cent} - \theta_{high}$$

If $M_{low} \leq M_{ref} \leq M_{2nd}$, then θ_{ref} replaces θ_{high} and we return to step 2 for another iteration with the new simplex.

4. **Expansion:** If $M_{ref} < M_{low}$, then the reflection is expanded to find potentially better improvement in the same direction.

$$\theta_{exp} = 2\theta_{ref} - \theta_{cent}$$

If $M_{exp} < M_{low}$, then θ_{exp} replaces θ_{high}, otherwise, we replace θ_{high} with θ_{ref} as in step 3, and we return to step 2 for another iteration.

5. **Contraction:** If the reflected vertex is the worst of our new simplex, we contract the simplex. If $M_{ref} \leq M_{high}$, then θ_{ref} replaces θ_{high} and M_{ref} replaces M_{high}. Then,

$$\theta_{cont} = 0.5\theta_{high} + 0.5\theta_{cent}$$

If $M_{cont} \leq M_{high}$, then contraction is accepted and we proceded with the next iteration.

6. **Shrink:** If contraction is not accepted, then the entire simplex halves in size, with every vertex halving in distance relative to θ_{low}:

$$\theta_i = 0.5\theta_i + 0.5\theta_{low}$$

The algorithm then returns to step 2 for another iteration.

As for stopping criterion, we can impose multiple. Such as having the standard deviations of the function values fall below a certain tolerance, i.e.

$$\sigma_S^2 = \sum_{i=1}^{n+1} \frac{(M(\theta_i) - \bar{M})^2}{n+1}$$

Where $\bar{M} = \sum M(\theta_i)/(n+1)$. Alternatively, we can end the algorithm whenever the actual size of the simplex itself becomes sufficiently small, or when we have hit a predetermined maximum number of iterations.

One advantage of the Nelder-Mead method over pattern search is that we

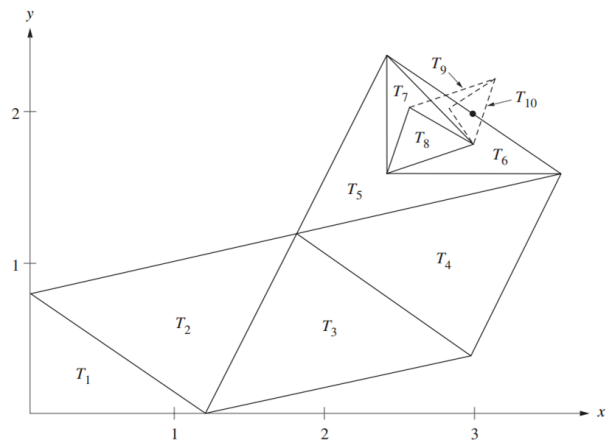


Figure 16: The sequence of triangles (T_i) for the first ten iterations of the Nelder-Mead method. The true minimum point of the function is $(3, 2)$ as shown as a black dot.

only need to compute one function evaluation each iteration, whereas in pattern search we computed $N_k = |\mathcal{D}|$ evaluations every iteration when generating trial points. Now since Nelder-Mead has memory of $n - 1$ past iterations at each iteration, we can run the function less times and achieve a similar result.

The Nelder-Mead method is best thought in the two-dimensional case, where the simplex becomes a triangle of three vertices in $\mathbb{R}^2$. Here our best, second-best and worst vertices are our only three vertices, and the geometric relations as described in the above algorithm are better visualised, as in figure 16. This is an example taken from Mathews and Fink (2004)[32], where the algorithm is trying to find the minimum of $f(x, y) = x^2 - 4x + y^2 - y - xy$, starting with initial simplex $(0, 0), (1.2, 0), (0, 0.8)$. After 33 iterations, the lowest function value was -6.99999998 at point $(2.99996456, 1.99983839)$.

5.1.3 Accounting for Noise

Whilst both methods are successful in that neither use the function's derivative or approximate a derivative, they do not take into account the possibility of noise within the function. For a low amount of noise however, the Nelder-Mead algorithm is attractive as it only uses the rank of the function values. Hence any perturbation within the actual function values not large enough

to affect the ranked values will not have any affect on the method.

This reasoning breaks down however once the algorithm approaches convergence. If we imagine our noisy function M to be broken down into a deterministic and stochastic component: $M = f + \epsilon$, then as our simplex shrinks in size close to the minimum, the differences in $f(\theta)$ at each point will be relatively similar, whereas the iid noise ϵ will remain relatively large. Hence close to the minimum, the observed function evaluations $M(\theta)$ are dominated by the stochastic component, leading to the possibility of rankings being changed due to random noise.

This runs the risk of leading the algorithm in the wrong direction, or prematurely terminating the algorithm before an optimal solution is reached. Barton and Ivey (1996)[31] also showed that in the limit as noise dominates the function, the probability of performing a contraction, and performing a shrink given a contraction rapidly increases. As well as causing the algorithm to potentially get 'stuck', running more shrink steps in the algorithm requires the most function evaluations, and hence increase the overall cost of the method.

Barton and Ivey then go on to propose modifications to the Nelder-Mead algorithm that aim to either correct the bias towards shrink and contract steps, or limit the impact of these steps.

Modification 1: (S9) The first change (Shrink90%) is not to change the probability of a shrink step, but to lessen the change in the size of the simplex. This is achieved by shrinking the simplex by 10% rather than 50%. The idea is that this delays premature algorithm termination at the cost of more function evaluations.

Modification 2: (RS) The second (ReSample) is to re-evaluate the lowest valued point after a shrink step. For a deterministic function, we only evaluate n new points during a shrink step, keeping the lowest the same. Once we are in the case where the function is dominated by random noise however, this 'lowest value' is most likely caused by the random component. Hence we are then less likely to perform an expansion step, as for this to happen would require two very low random realisations in a row. Hence by reevaluating all $n + 1$ points, we begin the next iteration with iid noise components and the bias towards contraction and shrinking is reduced.

Alternatively, Chang (2012)[33] suggests replacing the function value with an average function value at the cost of more function evaluations. His updated Nelder-Mead algorithm is as follows:

1. Let $\hat{M}_k(\theta) = \sum_{i=1}^{N_k} M(\theta)/N_k$ be the average function value of N_k function evaluations at step k.
2. At each iteration, generate a new vertex θ' using the Nelder-Mead algorithm.
3. Take N_k function evaluations at the new vertex, and $N_k - N_{k-1}$ evaluations from all other vertices.
4. Calculate $\hat{M}_k(\theta')$ for the new vertex θ' and update $\hat{M}_k$ for all other vertices. Remove the vertex with highest average function value.
5. If converge criterion has been satisfied, terminate the algorithm. If not, increment k by 1 and return to step 2 for the next iteration.

Chang (2012) gives a theorem showing that for the algorithm to converge, we require that

$$\sum_{k=1}^{\infty} \alpha^{N_k} \qquad \forall \alpha \in (0,1)$$

So a typical sample size N_k is given by $\lceil \sqrt{k} \rceil$ (where $\lceil . \rceil$ is the ceiling function), and taking the level of noise into account, the paper suggests using a signal to noise ratio of at least three, i.e. $|g(\theta)\sqrt{N_k}/\sigma(\theta)| \geq 3$. This gives a better value of N_k:

$$N_k' = \max\left\{ \lceil \sqrt{k} \rceil, 9\left(\frac{\sigma(\theta)}{f(\theta)}\right)^2 \right\}$$

Where $\sigma(\theta)$ is the standard deviation of the noise at point θ, and $f(\theta)$ the 'true' value of the function at θ. Both of these quantities are unavailable to us in practice, so we would have to replace them with estimates.

Note: Both of these modifications assume that the noise is stochastic, whereas the noise within the MMD distance is deterministic. Therefore we cannot 'resample' function values at the same point to get multiple distinct values. Instead, for a point θ, we could look at points closely surrounding *θ and use the function evaluations at these points to generate the average function value $\hat{M}(\theta)_k$*

5.2 Bayesian Optimisation

Bayesian optimisation is a method specifically designed for the problem of high-cost function evaluation, such as the case of the MMD distance. Just as with simplex methods, it also only assumes access to points in the parameter space θ and the function $M(\theta)$ evaluated at those points. The general framework of the method is to fit an auxiliary model to initial training data, most commonly a *Gaussian process (GP)* which captures prior belief about the model, and then suggest a new candidate point and update the GP based off the function value at that point. Whilst the ways of defining the Gaussian process are all relatively similar, there are many different ways of generating a new candidate point, all of which depend on which *acquisition function* we use. These are functions which use information from the current GP posterior distribution to propose a new candidate point at each step.

Once again, we will first define Bayesian Optimisation in the noiseless case, and then go on to investigate methods of changing the algorithm to work well in a noisy setting, most of which involve modifications to the acquisition function.

5.2.1 Gaussian Process

A Gaussian process is a stochastic process fitting a multivariate Normal distribution to any finite set of points in $\mathbb{R}^d$[34]. This essentially works as our prior distribution, defined by a mean function μ_0 and positive definite kernel k. Then for initial points $\theta_1, .., \theta_n$, these define a mean vector $m_i = \mu_0(\theta_i)$ and a covariance matrix $K_{i,j} = k(\theta_i, \theta_j)$, giving the GP prior as

$$M|\theta_1, ..., \theta_n \sim N(m, K)$$

Then given the set of points and evaluations $\mathcal{D}_n = \{(\theta_i, M_i\}_{i=1}^n$, the posterior predictive distribution of our new candidate point θ' can be derived as:

$$M'(\theta')|\mathcal{D}_n \sim N(\mu_n(\theta'), \sigma_n^2(\theta))$$

With posterior mean and variance defined as,

$$\mu_n(\theta') = \mu_0(\theta') + k(\theta')^T(K + \sigma^2 I)^{-1}(M - m)$$
$$\sigma_n^2(\theta') = k(\theta', \theta') - k(\theta')^T(K + \sigma^2 I)^{-1}k(\theta')$$

Where $K = k(\theta_{1:n}, \theta_{1:n})$ is a square matrix of the covariances of all previous points, and $k(\theta') = k(\theta', \theta_{1:n})$ is a vector of the covariances between θ' and

all previous points.

This of course requires a choice of kernel, each of which come with their own set of hyper parameters to optimise. A popular kernel to use for covariance matrices is the Matérn kernel, used by both Wagner and Ginsbourger (2013)[35] and Diessner et al. (2022)[34].

$$k_\nu(x,y) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{||x-y||}{\rho} \right)^\nu B_\nu \left(\sqrt{2\nu} \frac{||x-y||}{\rho} \right)$$

Where Γ is the Gamma function and B_ν is the modified Bessel function of the second kind of order ν[36]. The popularity of the Matérn kernel is due to its flexibility, with the parameter ν controlling its smoothness. Note that for $\nu = 0.5$ and $\nu = \infty$, the Matérn kernel reduces to the Exponential and Gaussian kernel respectively, although in practice a fractional power of ν is beneficial.

However even with a chosen ν, we still need to estimate the *process variance* σ^2 and the *range* ρ. Fortunately this is easy to compute as we are modelling the predictive distribution as a multivariate normal, so we can use maximum likelihood estimation by maximizing the log-likelihood with respect to kernel hyperparameters,

$$l(\theta'|\theta_{1:n}) = \frac{1}{2}\theta'^T K^{-1}\theta' - \frac{1}{2}\log|K| - \frac{n}{2}\log(2\pi)$$

Where $K_{i,j} = k_\nu(\theta_i, \theta_j)$.

We can also add a 'nugget term' κ to the covariance matrix to take some noise into account, giving a matrix $K_{i,j} = k_\nu(\theta_i, \theta_j) + \kappa I$.

5.2.2 Acquisition Functions

Acquisition functions, also known as infill functions, are our means of generating new candidate points. These functions use current knowledge about the posterior distribution, and carefully weight exploration versus exploitation of the search space to find the candidate point that is likely to give the most information. Formally, given a proposal point θ and its corresponding function value $M = M(\theta)$, the general form of an acquisition function is,

$$\alpha(\theta; \mathcal{D}_n) = \mathbb{E}_{M|\theta}[U(\theta, M)]$$

Where $U(\theta, M)$ is the *utility function*[37]. We will briefly consider two main classes of acquisition functions: improvement-based policies and optimistic policies.

Improvement-Based Policies: These functions favor points that are more likely to improve upon the current best value. An early example of this would be *probability of improvement (PI)*, which measures the probability that a new point θ improves upon a current best. So for a current best function value M^*, the utility function is defined as $U_{PI}(\theta, M) = \mathbb{I}(M(\theta) > M^*)$. Then since the posterior distribution of M is Gaussian, the acquisition function is,

$$\alpha_{PI}(\theta; \mathcal{D}_n) = \mathbb{E}_{M|\theta}[\mathbb{I}(M(\theta) > M^*)] = \mathbb{P}(M(\theta) > M^*) = \Phi\left(\frac{\mu_n(\theta) - M^*}{\sigma_n(\theta)}\right)$$

Where Φ is the standard normal cumulative distribution. Then the acquisition function is maximised to find the next candidate point.

Similarly the *expected improvement (EI)* uses a similar heuristic, except it also takes the amount of improvement into account, i.e. $U_{EI}(\theta, M) = (M(\theta) - M^*)\mathbb{I}(M(\theta) > M^*)$, giving,

$$\alpha_{EI}(\theta; \mathcal{D}_n) = (\mu_n(\theta) - M^*)\Phi(\frac{\mu_n(\theta) - M^*}{\sigma_n(\theta)}) + \sigma_n(\theta)\phi(\frac{\mu_n(\theta) - M^*}{\sigma_n(\theta)})$$

Where ϕ is the standard normal probability density function.

Optimistic Policies: These policies use the principle of using 'best-case' bounds on probabilities according to our model. In the case of the most common function: the *upper confidence bound (UCB)*, it assumes the variance of the Gaussian process to be true. These functions contain a parameter β_n that must be tuned by the user in order to set the desired level of exploration vs. exploitation.

$$\alpha_{UCB}(\theta; \mathcal{D}_n) = \mu_n(\theta) + \sqrt{\beta_n}\sigma_n(\theta)$$

The value of β_n can either be set to a constant, or can vary with n as in the GP-UCB algorithm.

Alternatively, instead of an analytical acquisition function, we can use Monte-Carlo sampling acting as an approximation. Some other methods such as

entropy search (ES) and *predictive entropy search (PES)* can be computationally expensive, so Monte-Carlo can bypass this. We can even apply it to the functions discussed previously, by reparameterising them as so:

$$\alpha_{MC-PI}(\theta; \mathcal{D}_n) = \sigma\left(\frac{\max(\mu_n(\theta) + Lz) - M*}{\tau}\right)$$

$$\alpha_{MC-EI}(\theta; \mathcal{D}_n) = \max(0, \max(\mu_n(\theta) + Lz) - M*)$$

$$\alpha_{MC-UCB}(\theta; \mathcal{D}_n) = \max(\mu_n(\theta) + \sqrt{\beta_n \pi/2}|Lz|)$$

Where $z \sim N_d(0, 1)$, L is the lower triangular matrix of the Cholesky decomposition of the covariance matrix K (i.e $K = LL^T$), and σ is the softmax function with 'temperature' parameter τ.

This leads us to the general form of the Bayesian optimisation algorithm, shown in algorithm 8.

Algorithm 8 Generalised Bayesian Optimisation Algorithm

Begin with initial data θ_0 and initial function values $\mathbf{M}_0$.
for $n \leq N$ **do**
  update the mean and variance μ_n and σ_n as in section 5.2.1.
  if MCMC **then**
    Generate M samples $z \sim N_d(0, 1)$
    Find new candidate point $\theta' = \arg\max_{\theta \in \Theta} \frac{1}{M} \alpha_{MC}(\theta; z, \mathcal{D}_{n-1})$
  else
    Find new candidate point $\theta' = \arg\max_{\theta \in \Theta} \alpha(\theta; \mathcal{D}_{n-1})$
  Calculate $M' = M(\theta')$
  Set $\theta_n = \theta'$, $M_n = M'$.

5.2.3 Modifying EI for Noisy Functions

Out of these available acquisition functions, we will focus on the expected improvement (EI), and look at three modifications suggested by Wagner and Ginsbourger (2013)[35] and Letham et al. (2018)[38] to account for noisy observations.

In this section we once again assume our model to contain a 'true' deterministic component, and a random component, i.e. $M(\theta) = f(\theta) + \epsilon$. Then the

observations M we make are considered to be noisy observations of the 'true' function f. Following this, the first problem with EI then is the assumption that the current lowest function value M^* does in fact correspond to the lowest value of f. In a noisy case, this risks severely underestimating the true function value if M^* is dominated by an extreme random component.

Instead then, we replace M^* with a 'target' T, leading to the method *expected improvement with plugin (EI-T)*. A sensible form for T is to minimise the model rather than the noisy observed function values, so we define $T = \min(\mu_n(\theta'))$, and swapping M^* for T gives an almost identical form of the EI:

$$\alpha_{EI-T}(\theta; \mathcal{D}_n) = (\mu_n(\theta) - T)\Phi(\frac{\mu_n(\theta) - T}{\sigma_n(\theta)}) + \sigma_n(\theta)\phi(\frac{\mu_n(\theta) - T}{\sigma_n(\theta)})$$

However this still assumes that the future observation will be deterministic, so a better choice of T is the one used in *augmented expected improvement (AEI)*. Here we use $T = \mu_n(\theta^{**})$, where θ^{**} is the *effective best solution* obtained by minimising $\mu_n + \alpha\sigma_n$ over all current observations. We also introduce a term penalising points whose GP variance σ_n^2 is small relative to the noise level τ.

$$\alpha_{AEI}(\theta; \mathcal{D}_n) = \alpha_{EI-T}(\theta; \mathcal{D}_n) \times \left(1 - \frac{\tau}{\sqrt{\sigma_n^2(\theta) + \tau^2}}\right)$$

When $\tau = 0$ this reduces back to EI-T, and as $\tau \to \infty$, the penalty term decreases to 0. The justification of adding such a term is penalise small predictive variances with the hope of increases exploration.

Alternatively, Letham et al. (2018) proposes *noisy expected improvement (NEI)*. Recall the posterior distribution of the proposed point $M_n|\mathcal{D}_n \sim N(\mu_n, \sigma_n^2)$. Then the noisy expected improvement is defined as:

$$\alpha_{NEI}(\theta|\mathcal{D}_n) = \inf_M \alpha_{EI}(\theta|M_n)p(M_n|\mathcal{D}_n)dM_n$$

Whilst this acquisition function has no analytical expression, we can approximate it using *quasi-Newton methods.* For this we use the approximation:

$$\int_{[0,1]^d} f(u)du \approx \frac{1}{N}\sum_{k=1}^{N} f(t_k)$$

Where t_k are chosen from a uniform distribution on $[0,1]^d$. Then we transform the support of the integral to $[0,1]^d$ using the following proposition.

Proposition: Let $\phi(\theta|\mu, \Sigma)$ be a multivariate normal density function and decompose Σ by the Cholesky decomposition to get $\Sigma = LL^T$. Then,

$$\int_{\mathbb{R}^d} f(y)\phi(\theta|\mu, \Sigma)dy = \int_{[0,1]^d} f(L\Phi^{-1}(u) + \mu)du$$

Then by setting $\tilde{M}_n(u) = L\Phi^{-1}(u) + \mu_n$, we have,

$$\alpha_{NEI}(\theta|\mathcal{D}_n) = \int_{[0,1]^n} \alpha_{EI}(\theta|\tilde{M}_n(u))du \approx \frac{1}{N}\sum_{k=1}^{N} \alpha_{EI}(\theta|\tilde{M}_n(t_k))$$

5.3 Application to the MMD

In this section, we will performing a series of numerical experiments to assess the effectiveness of both direct search and Bayesian optimisation methods. Initially, we will consider the truncated farm problem as described in the introduction and section 4.2.3, and shown in figure 13. For the sake of visualisation, we will assume Σ known, and only attempt to estimate the mean of the truncated bivariate Gaussian for now, as to be able to plot the optimisation in two dimensions. In this case, $\theta = (\mu_1, \mu_2)$, and we wish to find the point $\hat{\theta} \in \mathbb{R}^2$ that minimises the MMD distance $M(\theta) : \mathbb{R}^2 \to \mathbb{R}$.

For each experiment, I will also be reporting the time it took to run the optimisation in minutes. This was done by storing `Sys.time()` before and after the code was run and calculating the difference. All experiments were done on a laptop using an 11th Gen Intel Core i5-1135G7 2.40GHz processor with 16.0 Gb of RAM.

Experiment 1: Nelder-Mead Firstly, we will consider the Nelder-Mead method as defined in section 5.1.2. Using the `nm_optim()` function from the R package `optimization`[2], I performed Nelder-Mead optimisation using an initial point of $\theta_0 = (0, 0)$ to build the initial simplex around, a tolerance of 0.001, and a maximum number of 30 iterations. I then repeated the experiment for three different values of M, the sample size used in the MMD: $M = 100, M = 500, M = 1000$.

```
nelder_mead<-optim_nm(MMD_Distance,k=2,start=c(0,0),
trace=TRUE,tol=0.001,exit=30)
```

[2]https://github.com/kaihusmann/optimization

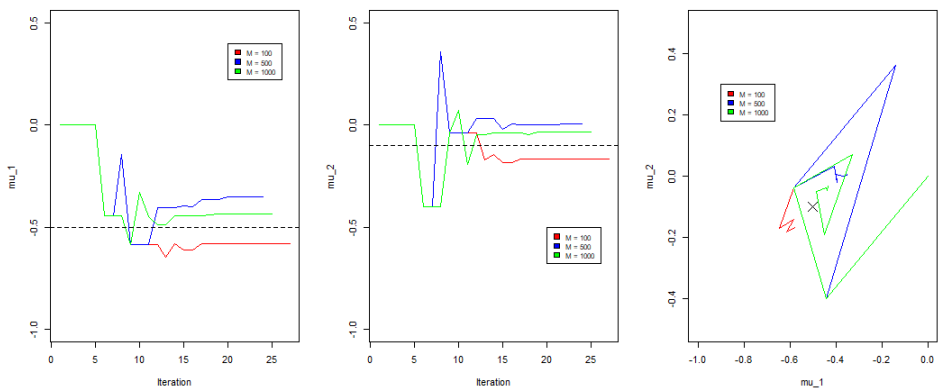


Figure 17: Results of experiment 1: Nelder-Mead.

The results of the experiment can be seen in figure 17. Here we have the results for $M = 100$ (1.15 mins), $M = 500$ (4.60 mins), and $M = 1000$ (10.75 mins) overlayed, labelled as red, blue, and green respectively. In the first two graphs, we can see the NM estimate of μ_1 and μ_2 vs number of iterations, with the true values of $\mu^* = (-0.5, -0.1)$ shown as a dotted black line. The trajectory of the three optimisations can be seen in the third graph, with all three beginning at (0,0) and the true value shown as a black 'x'. *Note: Some of the optimisation paths overlap.*

As expected, in the case of unmodified Nelder-Mead, we run into the problem of premature termination. Due to the noise of the MMD, the method gets 'stuck' at a close but non-optimal value of θ. This effect does not seem to be as severe as number of MMD simulations increase (And the overall variance of the MMD decreases), but still leads to non-optimality in this noisy scenario.

Experiment 2: Nelder-Mead S9 To remedy this then, let us implement the first Nelder-Mead modification introduced in section 5.1.3. Here the only parameter we change will be the size of the shrunk simplex, or in our case, the value of δ within the `nm_optim()` function, i.e.

```
nelder_mead_S9 <- optim_nm(MMD_Distance,k=2,start=c(0,0),
trace=TRUE,tol=0.001,exit=30, delta = 0.9)
```

Once again we can see a similar set of results in figure 18 in the same format as before. Here $M = 100, M = 500, M = 1000$ took 1.02 mins, 4.21 mins,

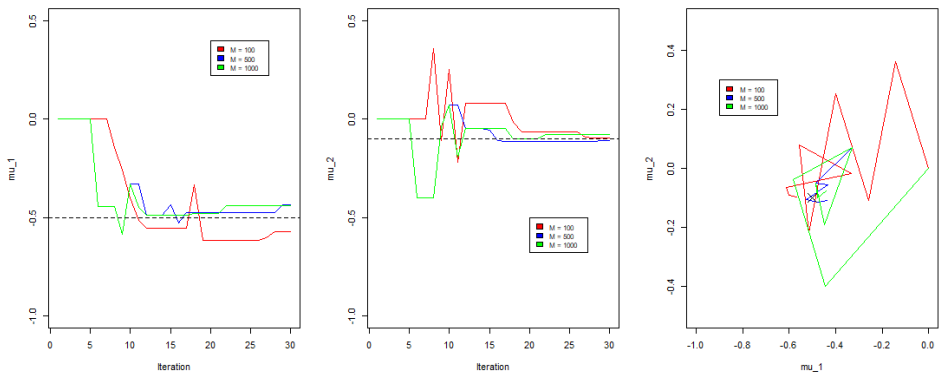


Figure 18: Results of experiment 2: Nelder-Mead S9.

and 7.45 mins respectively. Immediately we can see an improved performance in comparison to the unmodified NM method. By not prematurely shrinking the simplex around a non-optimal value, the algorithm was able to explore the region around the optimal value more and converge more accurately. Once again as well we see an improved performance with increased simulations, although this time the difference between $M = 500$ and $M = 1000$ seems more negligible.

Experiment 3: Nelder-Mead Average Function Next, we attempt to implement the method proposed by Chang (2012), where we replace the function M with its average value when sampled N_k times: $\hat{M}_k = \sum_{i=1}^{N_k} M/N_k$. Unfortunately, the `optim_nm` function has no way of changing the optimised function per iteration, so in this case we will take the number of repeated samples to be a constant $N_k = N$.

However as mentioned in section 5.1.3, we are unable to 'resample' our function and get distinct values as M is noisy but not stochastic. Instead we define our average value function to be:

$$\hat{M}(\theta) = \sum_{i=1}^{N} \frac{M(\theta_i')}{N}$$

Where $\theta_i' = (\theta_{i,1}', \theta_{i,2}')$, and $\theta_{i,1}' \sim U[\theta_1 - \delta, \theta_1 + \delta], \theta_{i,2}' \sim U[\theta_2 - \delta, \theta_2 + \delta]$. I.e. we uniformly sample points in a small square of length 2δ surrounding θ, then take the average function value of those points.

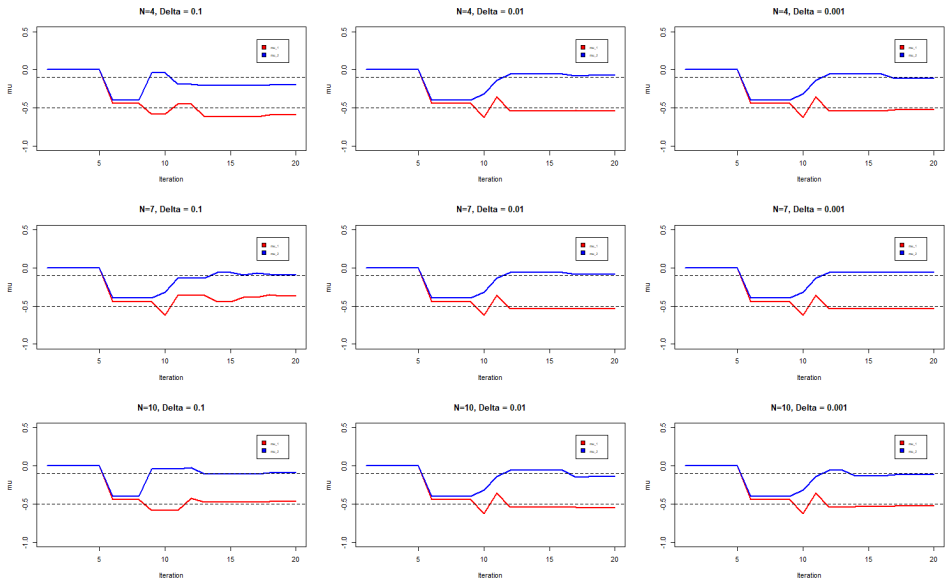


Figure 19: Results of experiment 3: Nelder-Mead Average Function.

Now we have introduced two parameters N and δ that will potentially affect the effectiveness of the Nelder-Mead method with this average function modification. For the experiment therefore, we repeat the NM algorithm over $N = 4, N = 7, N = 10$ and $\delta = 0.001, \delta = 0.01, \delta = 0.1$, using $M = 200$ for each. The results can be seen in figure 19, where each graph contains the convergence of μ_1 (shown in red), and μ_2 (shown in blue).

Between all nine of these optimisations, we can see that $\delta = 0.001$ and $\delta = 0.01$ behave similarly well, whereas $\delta = 0.1$ performs relatively poorly. Then within each δ, changing the value of N does not actually seem to prove that important, with $N = 4$ acting seemingly just as well as $N = 10$. Moreover, all the optimisations converged better than the unmodified NM, and at least on par if not better than the S9 modification.

Between values of δ, the times taken to run the optimisation were very similar. On average though, the times for $N = 4, N = 7, N = 10$ were 3.68 mins, 6.18 mins, and 9.34 mins respectively.

Experiment 4: Bayesian Optimisation. Next we consider the Bayesian

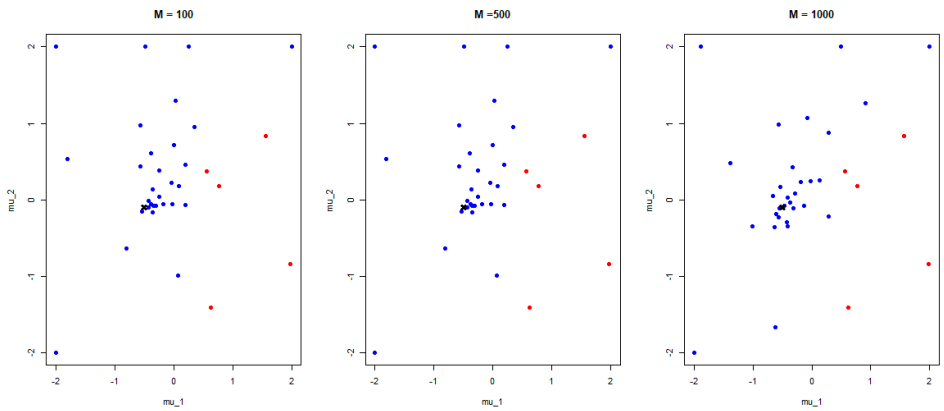


Figure 20: Results of experiment 4: Bayesian Optimisation.

optimisation algorithm. For this we will use the `BayesianOptimization()` function included in the `rBayesianOptimization`[3] R package. This function required search bounds on θ, so I used [-2,2] for both μ_1 and μ_2. Unfortunately the package is unreliable when using the EI acquisition function, and only contains three inbuilt functions: EI, UCB, and PI. Hence we are unable to implement the modified EI acquisition functions as described in section 5.2.3. Therefore, the following section uses a default UCB acquisition function and a Matérn kernel with $\nu = 5/2$. Similar to the Nelder-Mead experiments, I ran the optimisation using five initial points and 30 iterations, then repeated the experiment for $M = 100, M = 500, M = 1000$ (*Note: Bayesian optimisation automatically maximises the function, so in this case we use the negative MMD*):

```
bayes_opt_ucb <- BayesianOptimization(FUN = -MMD_Distance,
bounds = list(par1 = c(-2,2), par2 = c(-2,2)),
init_grid_dt = NULL, init_points = 5, n_iter = 30,
acq = "ucb",kernel = list(type = "matern", nu = 5/2))
```

The results can be seen in figure 20, where I have plotted the five initial points in red and the next 30 iterations in blue. All three optimisations seem to generate a distribution centering around the true value, although $M = 100$ and $M = 500$ seem to contain more bias than $M = 1000$.

We can also replace M with $\hat{M}$ as defined previously. We repeated the same

[3]https://github.com/yanyachen/rBayesianOptimization

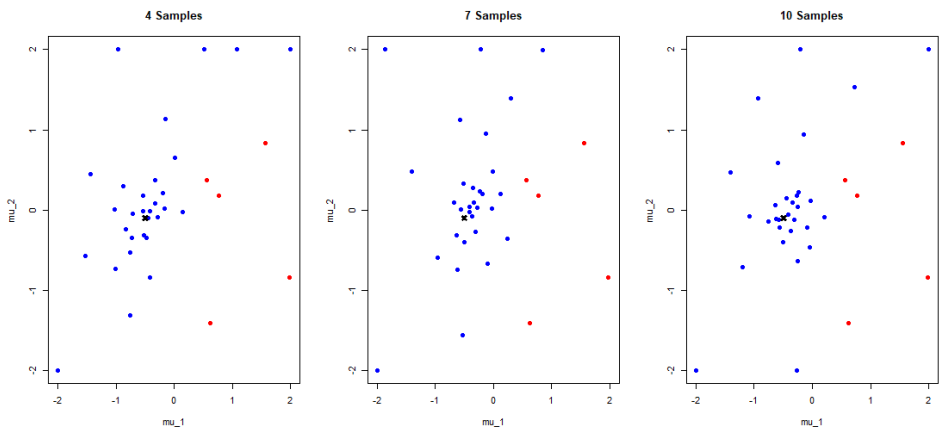


Figure 21: Results of experiment 4: Bayesian Optimisation with Average Function.

Bayesian optimisation, using $\delta = 0.001$ and $M = 200$, then varying N over $N = 4, N = 7, N = 10$. The results of this are seen in figure 21, where the bias of the unmodified method is seemingly corrected in all three models, and the variance of the posterior distribution seems to decrease as N increases.

For the unmodified model, the times to run were 2.80 mins, 2.82 mins and 4.68 mins for $M = 100, M = 500, M = 1000$ respectively, and for the average model, the times to run were 3.83 mins, 5.30 mins, and 6.07 mins for $N = 4, N = 7, N = 10$ respectively. I also ran the same average function optimisation algorithm for 80 iterations instead of 30 using $N = 10$, which took 39.57 mins to run, suggesting the runtime does not scale linearly with respect to number of iterations.

However in all cases, Bayesian optimisation took less time to run than the Nelder-Mead method, implying that the former method used less function evaluations for the same number of iterations.

5.3.1 Higher Dimensions

Now that have we attempted these optimisation methods in practice and visualised them using a simple two-dimensional case, we introduce the higher-dimensional case that we wish to work with.

For this section then, we again consider the farm truncation example as described in section 4.2.3. The optimisation problem is now defined as finding the point $\hat{\theta} \in \mathbb{R}^5$ that minimises the MMD distance $M(\theta) : \mathbb{R}^5 \to \mathbb{R}$. Here we define $\theta = (\mu_1, \mu_2, \sigma_1, \sigma_2, \rho)$, where the parameters modelling the truncated normal distribution ($Y \sim N_2(\mu, \Sigma)$ s.t. $Y \in C$) are:

$$\mu = (\mu_1, \mu_2) \qquad\qquad \Sigma = \begin{pmatrix} \sigma_1 & \rho\sqrt{\sigma_1\sigma_2} \\ \rho\sqrt{\sigma_1\sigma_2} & \sigma_2 \end{pmatrix}$$

Adding the extra parameters of the variances σ_1, σ_2 and the correlation ρ incurs two problems for our current method. Firstly, we now have to account for *constraints* on the optimisation since we must have that $\sigma_1 > 0, \sigma_2 > 0$, and $-1 < \rho < 1$. The `BayesianOptimization()` function can account for this since we can input the constraints in the `bounds` parameter, although the Nelder-Mead function has no such option.

To keep the parameter search within the constraints therefore, we apply an exponential transformation $\sigma_1' = \exp(\sigma_1), \sigma_2' = \exp(\sigma_2)$, and a logistic transformation to ρ, i.e.

$$\rho' = 2(\text{logit}(\rho)) - 1 = \frac{e^\rho - 1}{e^\rho + 1}$$

Then we perform the optimization on these transformed parameters. The second issue is that intuitively, we can expect that working under the assumption of Σ unknown will increase the overall uncertainty, and thus increase the noise within the MMD function $M(\theta)$. To counteract this, we increase the simulation size to $M = 2000$, and repeat each function evaluation using $N = 10$ in $\hat{M}(\theta)$, at the cost of greatly increased computation time.

Since each optimization experiment now takes over an hour, I only performed five variations of the various optimization algorithms. (*Note: For these examples I ran the code using a Kaggle notebook on https://www.kaggle.com/. Here the CPU Kaggle uses is 2.20 GHz, so code may take slightly longer to run than before.*) In all cases, I used $\delta = 0.001$. For the Bayesian optimization, I used the same search bounds in both experiments: $-1 < \mu_1, \mu_2 < 1$, $0.05 < \sigma_1, \sigma_2 < 0.55$, $0.05 < \rho < 0.95$. For the Nelder-Mead experiments, I used the same initial point in all cases: $(0, 0, 0.14, 0.14, 0.6)$. The results of

	BO-1	BO-2	NM-1	NM-S9-1	NM-S9-2
M	2000	2000	2000	1500	1000
N	5	10	10	10	10
#Init. Vals.	10	10	None	None	None
#Iterations	30	30	30	40	40
$\hat{\mu}_1$	-0.782	-0.445	-0.5	-0.753	-0.877
$\hat{\mu}_2$	-0.573	0.104	-0.07	-1.548	-1.048
$\hat{\sigma}_1$	0.550	0.200	0.159	26.42	18.64
$\hat{\sigma}_2$	0.550	0.05	0.393	27.514	19.796
$\hat{\rho}$	0.932	0.05	0.317	4.06×10^{-4}	6.64×10^{-3}
Time (hrs)	1.648	2.102	2.32	2.213	1.453

Table 2: Comparisons of optimisation results for various optimisation methods.

this section can be seen in table 2.

For the Bayesian Optimisation, I re-ran virtually the same code as before with $N = 5$ (BO-1) and $N = 10$ (BO-2). Similarly for Nelder-Mead, I tried both the regular method (NM-1) and the S9 method with the 90% shrunk simplex. In this case I increased the iterations due to the chance of convergence being lower, so I decreased M to $M = 1500$ (NM-S9-1) and $M = 1000$ (NM-S9-2) to compensate.

Recall the true value of θ for the farm example is $\theta^* = (-0.5, -0.1, 0.1, 0.15, 0.4)$. From table 2, see can see that NM-1 massively outperforms all other methods. Both the Bayesian methods gave estimates somewhat close to their true values, but both simply set some parameters to their upper or lower bound. Using the S9 modification for Nelder-Mead in NM-S9-1 and NM-S9-2 was wholly unsuccessful. In fact, the two methods took 37 iterations and 39 iterations respectively to even return a point other than the initial value.

However, even when the parameter estimates seem inaccurate in the case of the first three models, we encounter the same effect as in section 3.4.1. Consider the estimate in NM-1. Here the estimates of μ_1 and μ_2 are very accurate, but the estimates of σ_1 and σ_2 are higher than the true value. If we look at the bivariate normal distribution produced by these estimated parameters in figure 22, it seems that the vertical variance (i.e. $\hat{\sigma}_2^2$) is much higher than the true distribution. However when truncated, the distribution

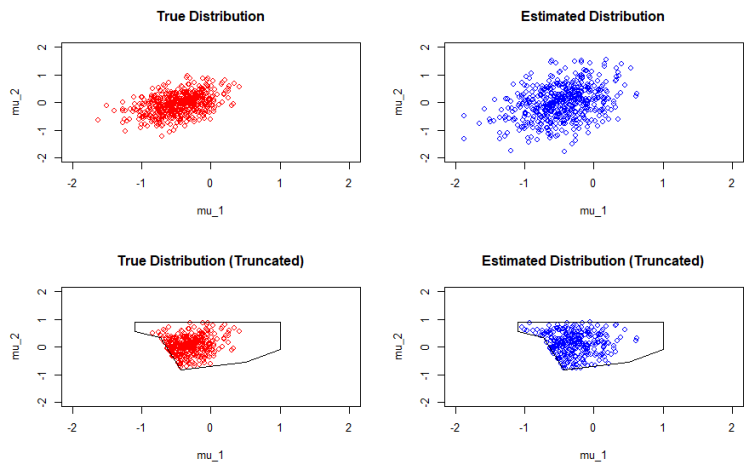


Figure 22: Observed data (truncated and Non-truncated) shown in red, and simulated data shown in blue using the parameter estimates for the NM-1 optimisation shown in table 2.

matches the observed data quite well. It could be due to the shape of this truncation region, the excess variance is 'cut off' by the top of the shape, and the resulting data gives a good fit such that the MMD distance is low.

5.4 Discussion

The main advantage of the Nelder-Mead is its simplicity, this being the main reason for its popularity remaining high today even with more advanced methods available. Whilst more effective methods such as BFGS exist in the case where a derivative can be calculated or estimated, direct search methods are still useful for non-differentiable optimisation.[29]

One main drawback of direct search methods is the time they take to run, although one could argue that this time is saved by not having to formulate a more complex optimisation algorithm. The main issue however is that direct search methods do not scale particularly well with n, and work best for only a few variables. For us this should not affect the optimisation too much as we are only working with $n = 2$ variables in the case where we assume to know the covariance, and $n = 5$ variables in the case where we do not.

Bayesian Optimisation is evidently a far more complex method, although this invites the added challenge of choosing kernels and estimating hyperparameters for the optimisation process itself as well as the parameters of the function. The main advantage here is the use of less function evaluations whilst retaining a similar amount of information, which we shown empirically in the experiments above.

In the case of assuming Σ known, both methods performed well, with Bayesian optimisation slightly outperforming Nelder-Mead due to giving a similarly accurate estimate in about half the time. However once we included the estimation of Σ, Nelder-Mead was able to give a better estimate at the cost of significantly longer computation time. Therefore for cases involving an unknown Σ, we would use a Nelder-Mead optimisation with large M, N and number of iterations to reduce noise within the MMD as much as possible. However for the sake of visualisation, we will return to the case where Σ is known for the next chapter, and use Bayesian optimisation for minimising the MMD distance.

6 Evaluation

In this section we test the effectiveness of the MMD estimator in a few different scenarios, to investigate where it works best and what conditions can cause the method to break down. As a comparison, we reintroduce the ABC method from section 2.2 as a means of parameter estimation. For every example, we will apply both MMD estimation and ABC to compare the relative performance of the two methods.

We will consider two main factors that we hypothesise will have the most effect on the performance of the MMD estimator: The amount of data contained in the truncation region, and the shape of the truncation region..

6.1 MMD vs ABC

To compare our MMD method so far to ABC methods: the common alternative, let us reintroduce the idea of *indirect inference within ABC* (ABC-II) from section 2.3. Here we will use a rejection sampling algorithm for ABC with a vague Gaussian prior and a distance function defined by the same

Gaussian kernel we have been using for MMD so far, i.e

$$D(x, y|\sigma) = \exp\left(-\frac{||x-y||^2}{\sigma^2}\right)$$

Then as for our summary statistic, we introduce an auxiliary likelihood to calculate an auxiliary parameter. A natural choice in this scenario is to use $f_Z \sim N_2(\mu_Z, \Sigma_Z)$, where f_Z is the *non-truncated* normal distribution of the data. I.e. we calculate the bivariate density of the data under the assumption that it has not been truncated. Under this assumption, f_Z has a tractable likelihood, and using maximum likelihood estimation, we can define our summary statistic to be the maximum likelihood estimate $(\hat{\mu}_Z, \hat{\Sigma}_Z)$ of f_Z given data y:

$$S(y) := (\hat{\mu}_Z(y), \hat{\Sigma}_Z(y)) = \arg\min l(\mu_Z, \Sigma_Z|y)$$

Where $l(.|y)$ is the log-likelihood, i.e. $l(.|y) = \log(f_Z(y|.))$.

6.1.1 Location of Truncation Region

For this chapter we will consider the example as seen in sections 4.1.1 and 4.2.2. The first potential limiting factor of the effectiveness of the MMD estimator is the location of the truncation region relative to the mass of the true data. In other words, what percentage of the true data do we observe within the observed data. We can expect that for a truncation region capturing a lot of information, the MMD will perform well, whereas for a region only containing a tail of a distribution, problems may arise.

To investigate this, we propose a sliding window on the distribution, as in section 4.1.1. In this example however, we consider a circular truncation region of radius 0.7, and three locations. The regions can be seen in figure 23, with shapes A, B and C capturing 55.9%, 15.8%, and 2.5% of the data respectively.

For the MMD estimator, we used Bayesian optimisation with $M = 500, N = 5, \delta = 0.001$, search bounds $(-2, 2) \times (-2, 2)$, a UCB acquisition function, and a Matérn kernel with $\nu = 2.5$. For a starting value, we used the mean of the observed data in each case. The first 60 iterations are plotted in figure 25. Alongside, we used an ABC-II method with $M' = 500$ simulations for each iteration, a tolerance of $\epsilon = 0.1$, and enough iterations such that > 100

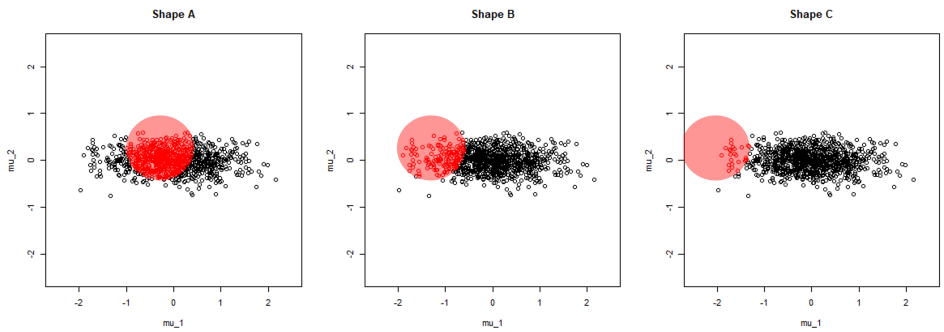


Figure 23: Shapes A, B, and C: Sliding window of truncation region.

accepted values were generated in each case. As a prior distribution, we used a bivariate normal with standard deviation 0.25, centered at the mean of the observed data.

The results of each estimation can be seen in figure 25. Here we have the observed data and truncation region highlighted in the background, with the Bayesian posterior samples of $\theta = \mu$ highlighted in blue. The true value μ^*, is shown in red, as a circle, with the posterior estimate $\hat{\mu}$ shown as a triangle.

Immediately we can see that the MMD estimator performs very well in all cases, seemingly regardless of the amount of observed data. However it is worth noting that although the mean of all iterations is close to the true value, the actual 'best value' the optimisation returns (i.e. the point θ that gave the lowest MMD distance) was not always as accurate. As for the ABC-II method, it gives a similar estimate for Shape A: the case where we are given the most observed data. However once the truncation approaches the tail end of the data, whilst the posterior estimate does indeed move away from the prior and closer to the true value, it is not as accurate as the MMD estimate.

6.1.2 Shape of Truncation Region

Another factor to consider is the actual *shape* of the truncation region. We hypothesise that this is one of the most important factors in determining the performance of the MMD estimator, although in this chapter we will only consider two extreme cases. So far, every region we have encountered has been relatively convex and simple. To test the MMD estimator in a

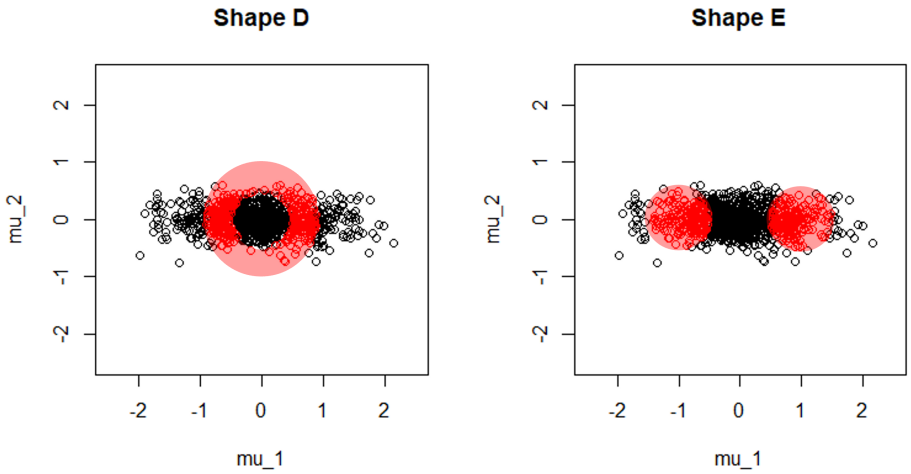


Figure 24: Shapes D and E: Non-convex and disjoint truncation regions.

more extreme scenario therefore, we introduce a *non-convex* region (Shape D), and a *disjoint* region (Shape E). Both contain over 30% of the data.

From the results of figure 25, we can see that these problematic shape cause major difficulties for the ABC-II method. In both cases the method cannot handle two separated clusters of data, and its posterior samples are biased towards the cluster with more observed data. This leads to a biased posterior estimate in Shape D, and a wildly biased estimate in Shape E. Fortunately, the MMD estimator again seems relatively unaffected by both cases, with only Shape E causing some bias to the left cluster.

6.2 MMD within ABC

As an alternative to choosing between ABC and MMD as a simulation-based estimation method, Park et al. (2015)[39] propose combining the two methods to use the MMD distance *within* ABC. This is done by replacing the distance metric $D(y, \tilde{y})$ and summary statistic with the MMD distance. The paper uses the same Gaussian distance as us, except replaces the Euclidean distance $||y - \tilde{y}||^2$ with the MMD distance $||\mu_{f_y} - \mu_{f_{\tilde{y}}}||^2_{\mathcal{H}_k}$, where $f_y, f_{\tilde{y}}$ are the distributions of the observed data y and simulated data $\tilde{y} \sim f(.|\theta)$

respectively.

$$D_\kappa(y, \tilde{y}) = \exp\left(-\frac{||\mu_{f_y} - \mu_{f_{\tilde{y}}}||^2_{\mathcal{H}_k}}{\kappa}\right)$$

Note that now we have two tolerance parameters to either estimate or tune manually: σ for the kernel k that defines the MMD distance, and κ that defines this distance $D_\kappa(y, \tilde{y})$ used for ABC.

To see if this method improves upon either MMD estimation or ABC using indirect inference, we will briefly apply it to all examples seen previously. Here we again use $M = 500, N = 5, \delta = 0.001$ for the MMD distance, then set $\kappa = 0.1$, and run for 500 iterations. As for the ABC tolerance, we manually tune it in each case to generate about 100-200 samples.

The results are shown alongside the previous results in figure 25. For shapes A,B, and C, the posterior estimates are similar to that of the ABC-II estimates, although the posterior distribution is more consistent. For shape D, the bias seen in the ABC-II method is no longer present, and for shape E, we see an interesting effect where the samples surround the two clusters, and yet the average of these samples is very close to the true value.

7 Conclusion

To summarise, this paper has introduced the maximum mean discrepancy as a metric for simulation-based estimation. We have considered the use of this estimation method in the scenario of a truncated normal distribution, where the truncation boundary C is complex but also able to be parameterised. We have seen the need for this method to be able to simulate normal data truncated to this region, and the various ways of doing so. We have then investigated the problem of minimising this distance with respect to a parameter θ, and the difficulty associated with it. Finally we have shown the performance of the MMD estimator in relation to the ABC estimator, and how the two methods can be combined to become more effective.

By testing the MMD estimator on various truncated distributions, we can conclude that using the MMD as a distance of *distributions* is more robust than methods such as ABC-II which use a distance of *datapoints*, when it comes to the problem of truncated domains. Whilst both the direct MMD method and the ABC-MMD method take a lot longer to run than regular

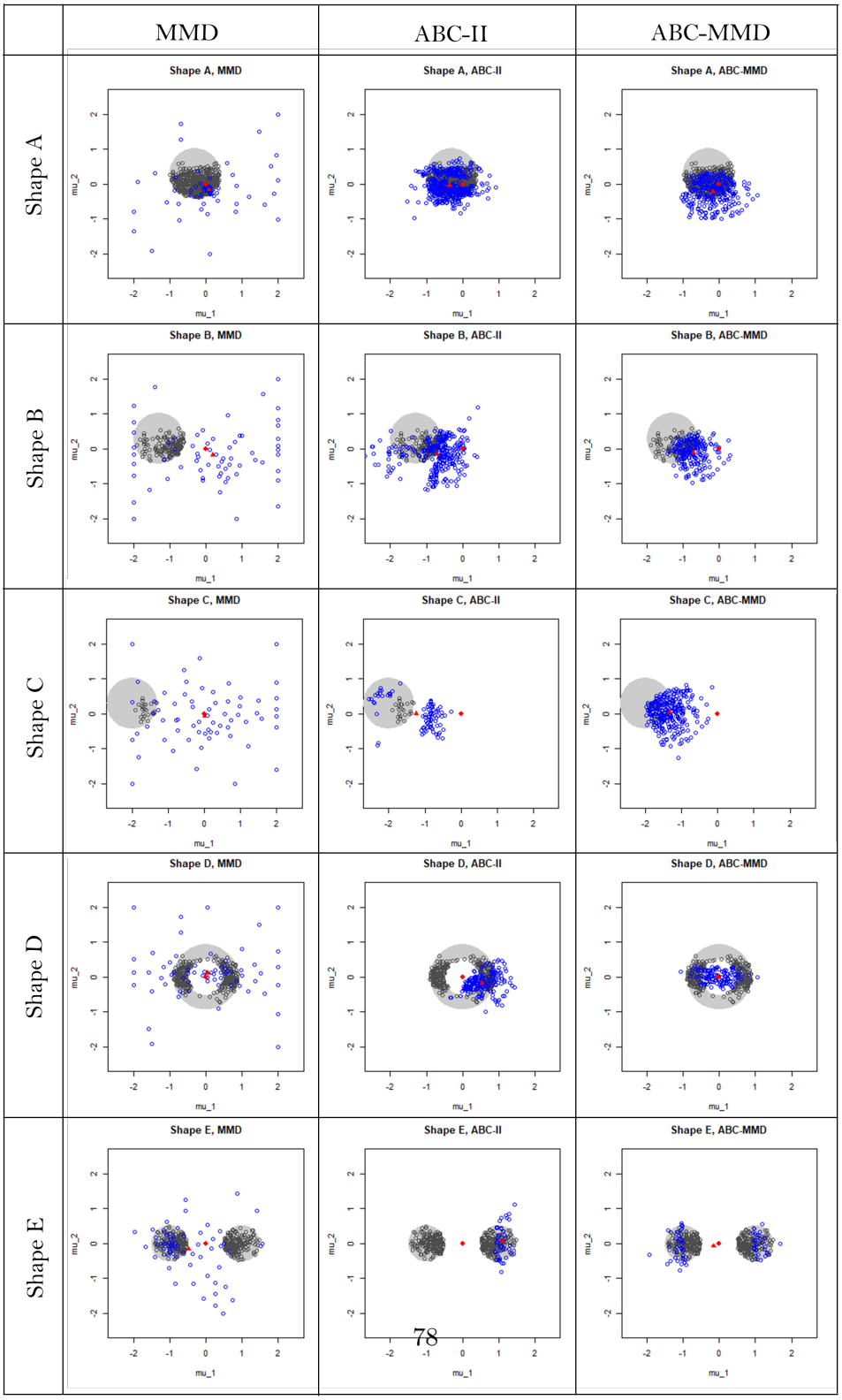


Figure 25: Results for all shapes and methods.

ABC, the samples they give are much more consistent. Similarly to how the MMD estimator was not affected by the outlier in the α-stable example, we have seen that the MMD estimator is an effective tool at parameter estimation even when the observed data is a somewhat distorted version of the 'true' data.

As for the best way to implement the MMD, we have seen in the evaluation chapter that both the MMD and ABC-MMD method give good posterior estimates. As said before however, we must be careful in the sense that the 'best value' of the MMD optimisation is not always accurate to the true value, and neither necessarily are all of the samples. If we were to pick one random posterior sample generated by either method, especially in the case of shape E, we could get a potentially very inaccurate parameter estimate. We therefore recommend methods of implementing the MMD which generate posterior *distributions* through many samples, such as the Bayesian methods shown in this paper.

As for the overall effectiveness of the MMD, we have seen that in multiple examples, the MMD performs very well at finding parameters that match the *distribution* of the observed data, but not always the *true parameters* of the observed data. There could be several reasons behind this; in the α-stable example we saw this effect in section 3.4.1 and hypothesised that the relationship between the parameters $(\alpha, \beta, \gamma, \delta)$ and their resulting distribution may not be injective in this case. (I.e. Two different sets of parameters could generate very similar observed data). For the example in section 5.3.1, we showed that the reason in this case may be the shape of the truncation region in relation to the amount of data it contains.

Another issue is computation time. In situations where the number of simulations M is high, the cost of computing the MMD distance is large. Also if the size of the *observed data* is large, this also massively increases computation time. This is obviously a disadvantage if we have to consider a trade-off between amount of observed data we use versus computation time. Evidently to use the MMD distance effectively in application to truncated problem where M will be large, we need a method which uses very little evaluations, such as Bayesian optimisation or ABC-MMD with few iterations.

7.1 Further Reading

For future work, there exist many modifications or improvements to the methods we have provided in this paper. One obvious improvement would be to simply increase the number of simulations M to the point where the estimation algorithm takes several hours to run, and see if the performance is improved. This could be applied to the situation in 5.3.1 where Σ is unknown, which I decided to avoid for most of this paper as the computation time was costly and the method's performance was inconsistent. Also whilst ABC-MMD appears to be somewhat effective already, I only implemented a simple ABC rejection algorithm. There exist many improvements to this which could be added in future, such as implementing an ABC-MCMC algorithm with MMD or adding regression weightings as discussed in section 2.2.

We also saw performance decrease in the case where the truncation region only captures the tail end of the true distribution. To improve upon this, we could apply the simulation algorithms specifically designed for this case, such as the 'RSM' algorithm as described in section 4.1.1.

Finally, one major extension to this method could be *transforming the truncation region* by transforming the data space. We have seen in sections 4.2.1 and 6.1.2 that the shape of the truncation region has a great impact on performance, mainly when the size of C relative to the rectangle we draw around it is small. To limit this, we could firstly attempt to implement a method of simulating truncated normal variables inside an ellipsoid, as proposed by Ellis and Maitra (2007)[40]. They give a method for simulating truncated normals *outside* an arbitrary ellipsoid, but this methodology could potentially be applied to the opposite case.

In terms of transformations however, we very briefly mentioned in section 4.2.1 the possibility of rotating the data space to draw a better fitting rectangle around the truncation region (See shape B in figure 11). In practice this could be done by applying a rotation matrix to the data[41],

$$R = \begin{pmatrix} \cos(\omega) & -\sin(\omega) \\ \sin(\omega) & \cos(\omega) \end{pmatrix}$$

Where ω is the angle of rotation. If the truncation shape were able to be parameterised, we could apply this transformation directly too. Or alternatively, if using an ellipsoid instead of a rectangle were successful, we

could attempt to 'bulge' the truncation set outward to better fit an ellipsoid. This could potentially be achieved using *polar coordinates*, i.e. setting $(x, y) \to (r, \omega)$, with $x = r\cos(\omega)$, $y = r\sin(\omega)$. Then using a transformation such as $T : r \to ar^b$, it may be possible to 'bulge' the data space in order to essentially 'push' the shape of the truncation region closer to an ellipsoid.

These however are just hypothesised improvements to our method so far, and to expand on them any further would be the responsibility of a future paper.

References

[1] C. Gourieroux, A. Monfort, and E. Renault, "Indirect inference," *Journal of Applied Econometrics*, vol. 8, pp. 85–118, 11 1993. [Online]. Available: https://www.jstor.org/stable/2285076

[2] T. Li, "Indirect inference in structural econometric models," *Journal of Econometrics*, vol. 157, pp. 123–128, 10 2009.

[3] C. Gouriéroux and A. Monfort, "Indirect Inference," in *Simulation-based Econometric Methods*. Oxford University Press, 01 1997. [Online]. Available: https://doi.org/10.1093/0198774753.003.0004

[4] A. R. Gallant and G. Tauchen, "Which moments to match?" *Econometric Theory*, vol. 12, no. 4, p. 657–681, 1996.

[5] D. T. Frazier and E. Renault, "Indirect inference: Which moments to match?" *Econometrics*, vol. 7, no. 1, 2019. [Online]. Available: https://www.mdpi.com/2225-1146/7/1/14

[6] A. Misiorek and R. Weron, "Heavy-tailed distributions in var calculations," *Handbook of Computational Statistics*, p. 1025–1059, December 2011. [Online]. Available: http://prac.im.pwr.edu.pl/ hugo/RePEc/wuu/wpaper/HSC$_1$0$_0$5.*pdf*

[7] B. Mandelbrot, "The pareto-lévy law and the distribution of income," *International Economic Review*, vol. 1, no. 2, pp. 79–106, 1960. [Online]. Available: http://www.jstor.org/stable/2525289

[8] J.-M. Marin, P. Pudlo, C. P. Robert, and R. J. Ryder, "Approximate bayesian computational methods," *Stat Comput*, vol. 22, pp. 1167–1180, Nov. 2012.

[9] O. A. Martin, R. Kumar, and J. Lao, *Bayesian Modeling and Computation in Python*, Boca Raton, Dec. 2021.

[10] M. A. Beaumont, W. Zhang, and D. J. Balding, "Approximate Bayesian Computation in Population Genetics," *Genetics*, vol. 162, no. 4, pp. 2025–2035, 12 2002. [Online]. Available: https://doi.org/10.1093/genetics/162.4.2025

[11] P. Fearnhead and D. Prangle, "Constructing summary statistics for approximate bayesian computation: semi-automatic approximate bayesian computation," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 74, no. 3, pp. 419–474, 2012. [Online]. Available: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2011.01010.x

[12] D. Prangle, "Summary statistics in approximate bayesian computation," Dec. 2015. [Online]. Available: https://arxiv.org/abs/1512.05633

[13] C. C. Drovandi, A. N. Pettitt, and M. J. Faddy, "Approximate bayesian computation using indirect inference," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 60, no. 3, pp. 317–337, 2011. [Online]. Available: http://www.jstor.org/stable/41262278

[14] A. Gretton, "Introduction to rkhs, and some simple kernel algorithms," October 2019. [Online]. Available: https://www.gatsby.ucl.ac.uk/ gretton/coursefiles/lecture4$_{i}ntroToRKHS.pdf$

[15] K. Muandet, K. Fukumizu, B. Sriperumbudur, and B. Schölkopf, "Kernel mean embedding of distributions: A review and beyond," *Foundations and Trends in Machine Learning*, vol. 10, no. 1-2, pp. 1–141, May 2016. [Online]. Available: https://arxiv.org/abs/1605.09522

[16] B. Sriperumbudur, K. Fukumizu, A. Gretton, B. Schölkopf, and G. Lanckriet, "On integral probability metrics, ϕ-divergences and binary classification," January 2009. [Online]. Available: https://arxiv.org/pdf/0901.2698.pdf

[17] W. Ford, *Numerical Linear Algebra with Applications*. Academic Press, 2014. [Online]. Available: https://www.sciencedirect.com/book/9780123944351/numerical-linear-algebra-with-applicationsbook-info

[18] A. Gretton, K. Borgwardt, M. Rasch, B. Scholkopf, and A. Smola, "A kernel method for the two-sample problem," *Technical Report No. 157*, April 2008. [Online]. Available: http://www.gatsby.ucl.ac.uk/ gretton/papers/GreBorRasSchetal08.pdf

[19] P. Chao, "Characteristic kernel," March 2014. [Online]. Available: https://www.stat.purdue.edu/ panc/research/dr/talks/Characteristic$_Kernel.pdf$

[20] B. K. Sriperumbudur, A. Gretton, K. Fukumizu, B. Schölkopf, and G. R. G. Lanckriet, "Hilbert space embeddings and metrics on probability measures," 2010.

[21] B. Sriperumbudur, K. Fukumizu, A. Gretton, G. Lanckriet, and B. Schölkopf, "Kernel choice and classifiability for rkhs embeddings of probability distributions." *Advances in Neural Information Processing Systems 22 - Proceedings of the 2009 Conference*, pp. 1750–1758, 01 2009.

[22] B.-E. Chérief-Abdellatif and P. Alquier, "Finite sample properties of parametric MMD estimation: Robustness to misspecification and dependence," *Bernoulli*, vol. 28, no. 1, pp. 181 – 213, 2022. [Online]. Available: https://doi.org/10.3150/21-BEJ1338

[23] F.-X. Briol, A. Barp, A. B. Duncan, and M. Girolami, "Statistical inference for generative models with maximum mean discrepancy," 2019. [Online]. Available: https://arxiv.org/abs/1906.05944

[24] J. Geweke, "Efficient simulation from the multivariate normal and student-t distributions subject to linear constraints and the evaluation of constraint probabilities," *Comput. Sci. Statist.*, vol. 23, 03 1998.

[25] H. Maatouk and X. Bay, "A new rejection sampling method for truncated multivariate gaussian random variables restricted to convex sets," *Monte Carlo and Quasi-Monte Carlo Methods*, vol. 163, pp. 521–530, 2016.

[26] J. Kotecha and P. Djuric, "Gibbs sampling approach for generation of truncated multivariate gaussian random variables," vol. 3, pp. 1757–1760 vol.3, 1999.

[27] R. Lewis, V. Torczon, and M. Trosset, "Direct search methods: Then and now," *Journal of Computational and Applied Mathematics*, vol. 124, 06 2000.

[28] S. Anagnostidis, A. Lucchi, and Y. Diouane, "Direct-search for a class of stochastic min-max problems," 2021.

[29] T. Kolda, R. Lewis, and V. Torczon, "T.g. kolda, r.m. lewis, v. torczon: Optimization by direct search: New perspectives on some classical and modern methods. siam review 45, 385-482," *SIAM Review*, vol. 45, pp. 385–482, 09 2003.

[30] D. M. Bortz and C. T. Kelley, *The Simplex Gradient and Noisy Optimization Problems*. Boston, MA: Birkhäuser Boston, 1998, pp. 77–90. [Online]. Available: https://doi.org/10.1007/978-1-4612-1780-0_5

[31] R. R. Barton and J. S. Ivey, "Nelder-mead simplex modifications for simulation optimization," *Management Science*, vol. 42, no. 7, pp. 954–973, 1996. [Online]. Available: http://www.jstor.org/stable/2634361

[32] J. H. Mathews and K. K. Fink, *Numerical Methods Using Matlab*, 4th ed. Pearson Prentice Hall, 2004.

[33] K.-H. Chang, "Stochastic nelder–mead simplex method – a new globally convergent direct search method for simulation optimization," *European Journal of Operational Research*, vol. 220, no. 3, pp. 684–694, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221712001609

[34] M. Diessner, J. O'Connor, A. Wynn, S. Laizet, Y. Guan, K. Wilson, and R. D. Whalley, "Investigating bayesian optimization for expensive-to-evaluate black box functions: Application in fluid dynamics," *Frontiers in Applied Mathematics and Statistics*, vol. 8, 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fams.2022.1076296

[35] V. Picheny, T. Wagner, and D. Ginsbourger, "A benchmark of kriging-based infill criteria for noisy optimization," *Structural and Multidisciplinary Optimization*, vol. 48, 09 2013.

[36] M. Genton, "Classes of kernels for machine learning: A statistics perspective." *Journal of Machine Learning Research*, vol. 2, pp. 299–312, 01 2001.

[37] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas, "Taking the human out of the loop: A review of bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2016.

[38] B. Letham, B. Karrer, G. Ottoni, and E. Bakshy, "Constrained bayesian optimization with noisy experiments," 2018.

[39] M. Park, W. Jitkrittum, and D. Sejdinovic, "K2-abc: Approximate bayesian computation with kernel embeddings," 2015.

[40] N. Ellis and R. Maitra, "Multivariate gaussian simulation outside arbitrary ellipsoids," *Journal of Computational and Graphical Statistics*, vol. 16, no. 3, pp. 692–708, 2007. [Online]. Available: http://www.jstor.org/stable/27594266

[41] E. Weisstein, "Rotation matrix." [Online]. Available: https://mathworld.wolfram.com/RotationMatrix.html