Synthesis course
31910
- - - - -
UTokyo Research Student
Multimodal Optimization
- - - - -
Department of Electrical Engineering
- - - - -

Francis Nils Hamilton, s160959

01/04/2018 – 23/08/2018

.....
Author
Mr. Francis Nils Hamilton
s160959
Danmarks Tekniske Universitet

.....
Signature of student

Date: .....

Contents

1 Introduction

I first came to Japan in 2017 as an exchange student. I instantly fell in love with the country and wanted to carry on living and studying here. During my masters I wanted to expose myself more to the world of machine learning as I believe it will play an important role in my future studies and career. Coincidently the laboratory I was part of during my exchange was just starting an AI group. So they agreed to keep me as a research student and to work under Professor Takoyaki Osa. I would later come to find out my work at UTokyo in the area of machine learning would allow myself to become an intern for a well respected Japanese Research group called, RIKEN-AIP.

1.1 Scope of the Research Project

The idea myself and Prof. Osa came up with regards to how the research and DTU synthesis course should be approached was to not only include work of systematic design but to also learn more broad concepts in terms of machine learning. Thus my experience as an University of Tokyo research student will be extensive. The general outline of the report could be viewed as two related parts namely, the amalgamation of my understanding of machine learning concepts and the steps and results from the implementation of my research.

1.2 Research Problem Definition and Background

The Sugita Mitsubishi Laboratory often performs research for companies which either sponsor the funding for laboratory or are affiliated in someway. Unfortunately due to confidentiality agreements, the name and a lot of information regarding this project can not be displayed in my report. As such a car manufacturing company, herein referred to as the stage name kurumakaisha, approached the laboratory as they would like to start integrating machine learning into the systems they manufacturer. As of now the method of which they change gears is controlled by an oil pressure signal sent to the gearbox. It is highly non-linear complicated function as such they have approximated it in parameters in the order of around 30. Over the last couple of years the company has developed an accurate Amesim model in which they can adjust the parameters, and thus the input signal, until they achieve their desired results. These parameter values are then stored in a look up table to be used in the real time system. For example when a gear change of first to second gear is demanded the code looks up the parameters values needed to generate the appropriate oil pressure input signal.

The company would like to improve upon this type of setup by instead using a control structure which has learnt, through interaction with the system and model, which parameter values to use depending on the context i.e gear change requirements, motor speed etc. Such a setup is in the area of reinforcement learning (RL), specifically hierarchical reinforcement learning (HRL). Prof Osa recently published a paper in which this type of HRL problem is addressed [21]. However, the company is reluctant to immediately switch to this type of control architecture and so they would like to see an intermediate step of finding multiple solutions to the optimization of the parameters for fixed contexts. This would be similar to the current setup of a look up table with the benefit of multiple solutions generated from an algorithm rather than from human tuning via looking at graph outputs. As such the research topic stated in the lab and Prof. Osa's website is such:

OPTIMIZING CONTROL PARAMETERS FOR DRIVING SYSTEMS
Many driving systems still require heuristic parameter tuning for control, and it is desired to automate

this parameter tuning process. The difficulty of this problem is that there often exist multiple solutions that give high returns. Since a designed reward function is often not optimal in practice, the solution that gives the highest return may not be the optimal solution. For this reason, it is often necessary to check the multiple candidates of the solution and see which one is the best in the actual system. To address this issue, we are developing a hierarchical gradient-free optimization method that can find multiple modes of the return function. This framework is capable of proposing multiple solutions to users and allowing them to choose the one that satisfies their requirements.

1.3 Project Outline

As I was fairly new to the area of machine learning and completely new to RL. Prof. Osa recommended that before I proceed with starting to solve the project‘s problem, I should learn to be familiar with a lot of machine learning concepts. It was this recommendation that gave me the opportunity to learn enough so as to be able to qualify for the RIKEN-AIP internship program. As the end goal of this collaboration between kurumakaisha and my laboratory was to implement a variant of Prof. Osa‘s paper on Hierarchical Policy Search [21]. I was tasked to learn the necessary concepts to understand the paper.

The report first starts on with a brief literature review in the area of multimodal optimization, it then follows with a section dedicated to the various machine learning concepts learnt and used during the project. Afterwards the section of the design synthesis to address the project's problem is outlined and finally a conclusion is presented.

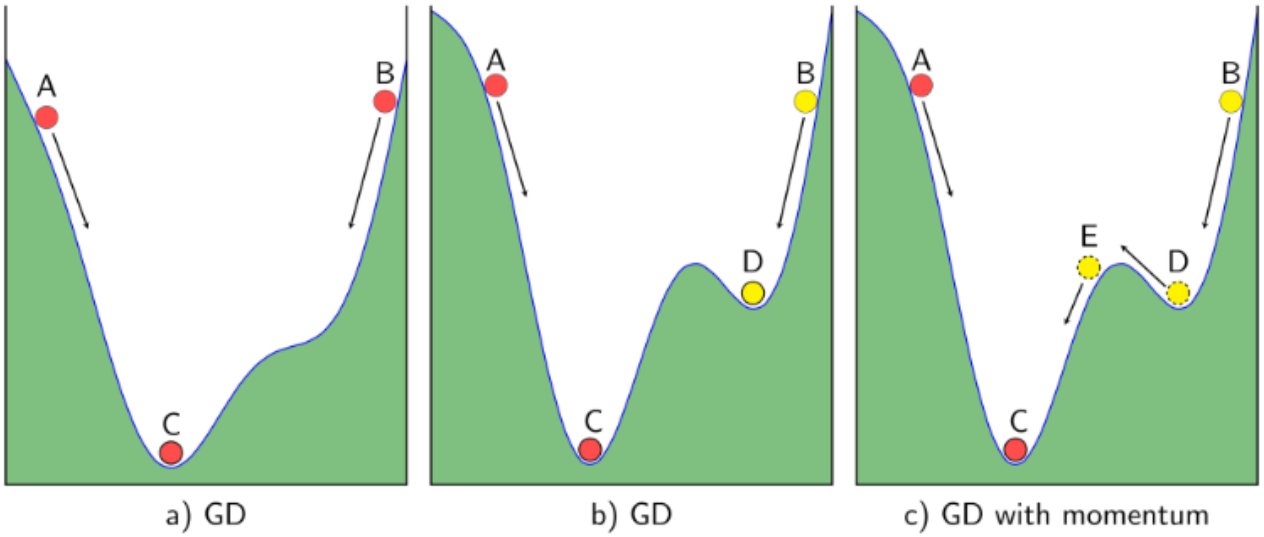

Figure 2.1: Gradient Decent with momentum term

2 Literature Review

Multi modal optimization goes way back to the 1980s and 1990s whereby it is assumed that the function to optimize has multiple local and global optima. During this time gradient descent or hill climbing methods were the norm. Often hill climbing methods suffer from being trapped in local optima. Many methods were developed to circumvent this downfall such as including a momentum term to help push the particle through the local minimum, figure 2.1. However, all the modifications of hill climbing still do not guarantee a hard convergence to a global optima in the face of a multi-modal function. A simple idea to solve this would be start the hill descending algorithm at different points in the exploration space, run the algorithm and compare solutions. This idea forms the basis for what is known as Multi-start methods, please see [20] for an overview of these algorithms. The most well known multi-start algorithm is Greedy Randomized Adaptive Search Procedures, GRASP [14].

Although Multi-start hill descent variations are appealing there has been a lot more interest into bio-mimicry algorithms such as genetic, evolutionary, swarm etc, see [26]. The types of algorithms have not only the benefit of stronger global optima convergence but also being able to handle discontinuous functions.

In terms of optimization multimodal means finding the global optima among multiple local and global optima. However, in the context of hierarchical reinforcement learning, it means to be able to finding multiple policies relating to the modes of the reward function [28] [29]. In general, policy search algorithms will average over the modes of the function or settle into a local optima [11]. In most applications the reward function is multi-modal and thus recently there is a lot of interest into hierarchical RL [7] [21].

3 Machine Learning Concepts

3.1 Reinforcement Learning

Reinforcement learning has its roots in behavioral psychology whereby an agent operates in an unknown world where it receives rewards based on its actions. The goal is for the agent to learn high reward yielding actions. In general the basic RL structure is modeled as a Markov Decision Process (MDPs) [29]. I invite the reader to refer to what could be called the origins of RL by Sutton and Barton et al. [28].

3.1.1 Policy Search

Essentially RL can be divided into two groups of model based and model free. Model based tries not only to learn the optimal control strategy, or policy, but also tries to learn the model of the system. Whereas model free is only concerned in generating actions which lead to higher rewards. There are both step based and episodic versions of both however in this discussion only episodic or trajectory based model free RL will be used. I refer the reader to an incredibly insightful book on Policy Search in RL [11].

There is a distinction between policy exploration and evaluation. The former being how the policy generates new unexplored rewards and the latter being how the policy updates itself based on the new data. In general a policy will relate states/contexts to actions. Often it is much more practically and mathematically sound to use features of contexts with linear Gaussian weights. A very simply yet powerful approach for an exploration policy in this case is to model the exploration policy as an upper-level zero mean Gaussian policy. Whereby the covariance of the upper-level determines the variation of parameters of the action policy and thus allows them to be perturbed so as to explore the parameter space, see figure 3.1. There are many methods to evaluate and update the policy given new data, they include the common gradient ascent, inference based ML and information-theorietic approaches. The type of policy update used in this paper is of the inference based, specifically, Reward Weighted Regression (RWR) [22].

Essentially RWR works by fitting the ML estimate of the data which is weighted by the reward value of each of the data points. In our case there is no context and as such given samples $D = (\theta, R)_i$ the idea is to fit the Maximum Likelihood of a Gaussian Policy with a mean and covariance to the data weighted by the reward data. A common way to calculate the weight of the reward is through the exponential. As such we can define the Gaussian policy as:

$$\mu = \frac{\sum_{i=1}^{N} W_i * \theta_i}{\sum_{i=1}^{N} W_i} \qquad \sigma = \frac{\sum_{i=1}^{N} W_i * (\theta_i - \mu) * (\theta_i - \mu)^T}{Z}$$

Where θ are the parameters and W_i is the weight of the associated sample and Z is given by:

$$Z = \frac{(\sum_{i=1}^{N} W_i)^2 - \sum_{i=1}^{N} (W_i)^2}{\sum_{i=1}^{N} W_i}$$

3.1.2 Gaussian Process

In simple regression the goal is to find the best fit of the data to a parametric model such as a straight line. Simple Bayesian regression is a probabilistic approach which finds the distribution over parameters as new data is added. In contrast a Gaussian Process (GP) [25] is a non-parametric approach which finds

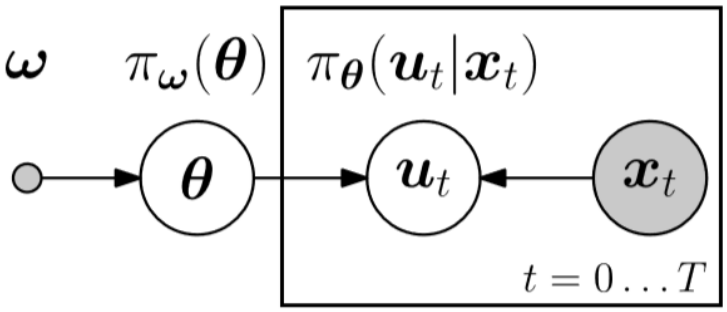

Figure 3.1: Gaussian Process [11]

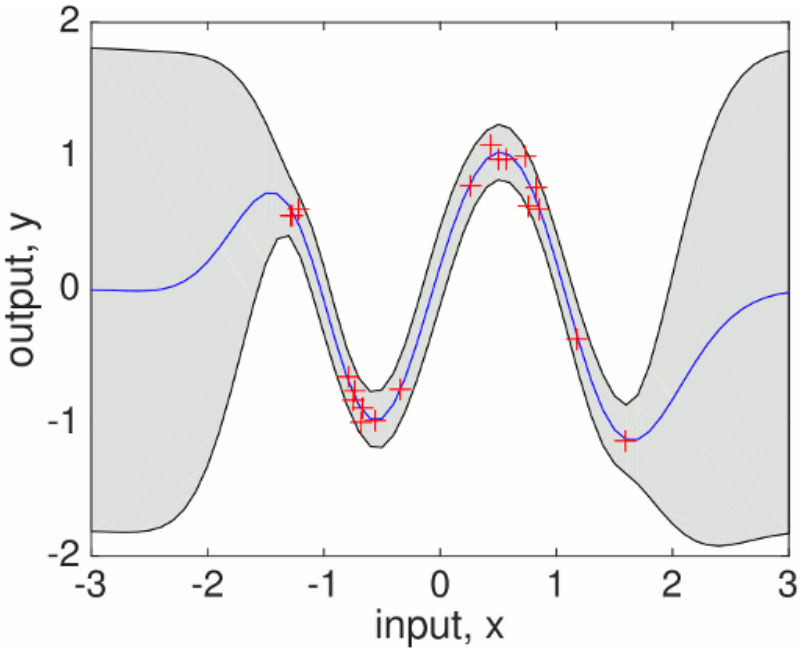

Figure 3.2: Gaussian Process [25]

a distribution over functions which are consistent with the data.

A prior is defined over functions which is thus transformed to a posterior over functions once new data is seen. A GP assumes that the values of the functions $p(f(x_1), f(x_2), ...f(x_n))$ are jointly Gaussian and thus are defined by a mean μ and covariance σ. The covariance is given by the kernal, most often the squared exponential, function which essentially relates how close two points are then it assumes their associated evaluated value are close.

If we have a set of points and their associated value we can fit the GP with the appropriate prior to these points and since there is the assumption of jointly Gaussian it is possible to define the conditional probability of unseen new points' values.

3.1.3 Importance Sampling

Importance Sampling is an approximate solution whereby if an pdf is intractable or infeasible to sample from, we can introduce a different pdf in which we can readily sample from. The two pdfs are related through what is called an importance weight.
If a function on X exists as $f(X)$ the unbiased expectation on $f(X)$ under a pdf of p(X) through Monte-Carlo is:

$$E_p\, f(X) \approx \frac{1}{N} \sum_{i=1}^{N} f(X_i)$$

The actual expectation is:

$$E_p\, f(X) = \int f(x)p(x)dx \tag{1}$$

Whereby the subscript p indicates under which pdf the samples were generated to be evaluated through $f(X)$. If we would like to sample under a different distribution $q(x)$ of our choosing we can modify the expectation 1 as such.

$$E_q\, f(X) = \int g(x)q(x)dx \qquad where \quad g(x) = f(x)\frac{p(x)}{q(x)}$$

One can see that the unbiased expectation of $f(x)$ under $q(x)$ is given as:

$$E_q\, f(X) \approx \frac{1}{N} \sum_{i=1}^{N} f(X_i)W_i \qquad where \quad W_i = \frac{p(x_i)}{q(x_i)}$$

The term W_i is known as the importance weight.

3.2 Clustering

Clustering is the organization of unlabeled data into similarity groups called clusters. A cluster is a collection of data items which are "similar" between them, and "dissimilar" to data items in other clusters. The most common types of clustering techniques are:

- Centroid based: K-means clustering
  Uses a distance based metric to separate the data into k clusters.
- Hierarchical based: Agglomerative and Divisive, HCA.
  Essentially defines how the hierarchical tree is built either starting with all samples belonging to their own cluster and building up to a final set of clusters or starting with samples belonging to one cluster and proceeding to split the clusters up until reaching a final set, respectively.
- Latent variable models: Mixture Models, EM, VBEM
  The use of latent variables stems from the task of supervised learning whereby the some of the data's labels are incomplete. As such latent variable models assume that the data's labels are incomplete however they were generated by distribution often mixture model, i.e Gaussian/Dirichlet mixture. The task is to match each sample in the data to the distribution in the mixture model which generated it. As such the task is how to best match the data to the mixture model through its parameters this is the exact case of probabilistic inference, i.e. Maximum Likelihood. The most common way to solve this is to use the Expectation Maximization method. The problems of EM are solved through using a variational Bayesian approach, know as the Variational Bayes EM.

3.2.1 Maximum Likelihood and Expectation Maximization

The following explanation will use Gaussians in the mixture model, the reader is referred to Chris Bishops's book for further reading [4]. Basic Bayesian inference is about fitting unknown parameters θ of a model that generated the data x.

$$\theta_{ML} = argmax \quad p(x;\theta) \tag{2}$$

In almost all case the direct evaluation of the likelihood is intractable, and so here in lies the power of latent variables. If the joint distribution between the observed data and unknown latent variables is known, one can easily marginal likelihood.

$$p(x;\theta) = \int p(x,z;\theta)dz = \int p(x|z;\theta)p(z;\theta)dz \tag{3}$$

Through using Baye's theorem the posterior of the latent variables can be calculated as:

$$p(z|x;\theta) = \frac{p(x|z;\theta)p(z;\theta)}{p(x;\theta)} \tag{4}$$

Often the integral in equation 3 cannot be evaluated easily and so it forms a lot of the theory in Bayesian Inference. Generally it is easiest to work with the logarithm of the likelihood rather than the likelihood, since likelihoods, being products of the probabilities of many data points, tend to be very small. Likelihoods multiply, log likelihoods add.
The following is derived from [4], it is shown the log likelihood is defined as:

$$ln\, p(x;\theta) = F(q,\theta) + KL(q||p) \tag{5}$$

where

$$F(q,\theta) = \int q(z) ln\, (\frac{p(x,z;\theta)}{q(z)})dz \tag{6}$$

and

$$KL(q||p) = -\int q(z) ln\, (\frac{p(z|x;\theta)}{q(z)})dz \tag{7}$$

Commonly equation 6 is know as the evidence lower bound (ELBO) meaning that $p(x;\theta) \geq F(q,\theta)$ and so equation forms the lower bound of the log likelihood. The EM algorithm is thus a two step procedure which maximizes the ELBO and thus the log likelihood as well. Namely, the Expectation step is when the parameters are fixed and the ELBO is maximized with respect to q(z) i.e. $F(q,\theta^{old})$. It is easily seen this is when the Kullback-Leibler[1] divergence is zero which corresponds to $q(z) = p(z|x;\theta^{old})$. In the Maximization step the distribution over the latent variables $p(z)$ is held fixed and the ELBO is maximized with respect to the parameters θ. Another way to look at it is to substitute $q(z) = p(z|x;\theta^{old})$ into the ELBO and then to maximize this ELBO with respect to the parameters.

$$F(q,\theta) = \int p(z|x;\theta^{old}) ln\, p(x,z:\theta)dz - \int p(z|x;\theta^{old}) ln\, p(z|x;\theta^{old}) = Q(\theta,\theta^{old}) + constant \tag{8}$$

Whereby $Q(\theta,\theta^{old}))$ is recursively maximized. In most non-trivial problems knowing the $p(z|x;\theta^{old})$ is intractable and so for interesting problems often an approximation is needed, which brings the next section variational Bayes EM.

[1]The Kullback-Leibler divergence is a measure between two probability distributions

3.2.2 VBEM: Variational Bayes Expectation Maximization

If any form of $q(z)$ is assumed that means the KL divergence is zero when $q(z)$ perfectly matches the posteriori distribution $p(z|x)$. However in a lot of cases this is intractable and so by considering a restricted family of distributions the KL divergence is minimized. For Bayesian Inference a particular form that has been met with great success is that of factorized distributions whereby $q(z)$ can be factorized. If is assumed that the latent variables are independent of the rest of the latent variables given x. This is known as the mean field approximation.

$$q(z) = \prod_{i=1}^{M} q_i(z_i) \tag{9}$$

Using the mean field approximation, equation 9, and the evidence lower bound, equation 6.

$$F(q,\theta) = \int q(z) ln\,(\frac{p(x,z;\theta)}{q(z)})dz$$

$$F(q,\theta) = \int \prod_i q(z_i) ln\, p(x,z;\theta)dz - \sum_i \int q(z_i) ln\, q(z_i)dz_i$$

$$F(q,\theta) = \int q(z_j) \int (\prod_{i\neq j} q(z_i) ln\, p(x,z;\theta)) \prod_{i\neq j} dz_i dz_j - \int q(z_j) ln\, q(z_j)dz_j) - \sum_{i\neq j} \int q(z_i) ln\, q(z_i)dz_i$$

$$F(q,\theta) = \int q(z_j) \ln\big(\frac{\exp(\langle \ln p(x,z;\theta)\rangle_{i\neq j})}{q(z_j)}\big)dz_j \quad - \sum_{i\neq j} \int q(z_i) \ln q(z_i)dz_i$$

$$F(q,\theta) = \int q(z_j) \ln\big(\frac{\tilde{p}_{i\neq j}}{q(z_j)}\big)dz_j + H(z_{i\neq j}) + c$$

$$F(q,\theta) = -KL(q_j||\tilde{p}_{i\neq j}) + H(z_{i\neq j}) + c \tag{10}$$

Where $\langle\cdot\rangle_i$ is the expectation over the latent variable z_i and $H()$ is the entropy. The constant c is added due to the fact that $\exp(\langle \ln p(x,z;\theta)\rangle_{i\neq j})$ is not a proper pdf. One can seen that since the KL-divergence is always positive the ELBO is maximized when equals zero as such:

$$q(z_j) = \tilde{p}_{i\neq j} = \frac{1}{Z} \exp\langle \ln p(x,z;\theta)\rangle_{i\neq j} \tag{11}$$

It says that the log of the optimal solution for factor q_j is obtained simply by considering the log of the joint distribution over all hidden and visible variables and then taking the expectation with respect to all of the other factors q_i for $i \neq j$ [4].

3.2.3 Relation to Gaussian Mixture Models

The main assumption here is to assume the data came from one of j probability distribution functions. Thus $P(Z = k) = \pi_j$ and $P(X = x|Z = j) = p_j(x)$ and so the joint pdf is defined as:

$$p(X,Z) = p(X|Z)p(Z) \tag{12}$$

Through marginalizing, equation 12, one easily obtains the common mixture model:

$$p(X = x) = \sum_{j=1}^{M} p(X = x|Z = j)p(Z = j) = \sum_{j=1}^{M} \pi_j p_j(x) \tag{13}$$

Such as in the case of GMMs the pdf of each component is $p_j(x) = \mathcal{N}(x; \mu_j, \sigma_j)$. Using Bayes' theorem the probability of a sample belonging to a cluster is given by:

$$P(j|x) = \frac{p(x|Z=j)p(Z=j)}{p(x)} = \frac{\pi_j \mathcal{N}(x|\mu_j, \sigma_j)}{\sum_{j=1}^{M} \pi_j \mathcal{N}(x|\mu_j, \sigma_j)} \tag{14}$$

Equation 14 is known as the responsibility of component j having generated sample x. In terms of EM the parameters to be estimated are $\theta = [\pi_j, \mu_j, \sigma_j] \quad for \quad j = 1, ..., M$. The appealing aspect of mixture models is that it is possible to compute the exact posterior and thus EM is feasible. The drawbacks are that the covariances might become singular and thus the log likelihood will go to infinity. This occurs when the algorithm assigns only one sample to one cluster through through the responsibilities. As well it does not automatically determine the optimal number of components. To overcome this a Bayesian approach is used whereby the parameters are considered variables with priors. As such the joint pdf is:

$$p(x) = \sum_{j=1}^{M} \pi_j \mathcal{N}(x; \mu_j, T_j) \tag{15}$$

whereby $\boldsymbol{\pi} = \pi_j$ are the mixing coefficients (weights) $\boldsymbol{\mu} = \mu_j$ are the means and $\boldsymbol{T} = T_j$ are the precision matrices(inverse Covariance). The prior imposed over the weights is a Dirichlet with parameters α_j. In general, to caputre correlation between the data, a Wishart prior is imposed over the precision variable and a Normal prior of the means given the pricision [2]. This leads to the Gaussian-Wishart distribution with governing parameters: $[v, V, \mu_0, \beta_0]$. The first two being the degrees of freedom and scale matrix of the Wishart distribution respectively.
This fully Bayesian approach whereby hyper parameters are set means no parameters need to be estimated only the hidden variables' , $[Z, \pi, \mu, T]$, posterior $p([Z, \pi, \mu, T], x)$ must be computed. Again through Bayes' theorem the posterior probability is given as [27]:

$$p(\pi, \mu, T|X; \alpha_0, \beta_0, V_0, v_o) = \frac{p(X|pi, \mu, T)p(\pi; \alpha_0)p(\mu, T; \beta_0, V_0, v_o)}{p(X; \alpha_0, \beta_0, V_0, v_0}$$

As the exact solution to this is intractable an approximation can be found through the variational mean field assumption defined in the previous section. As such the approximation to the posterior is:

$$q(Z, \pi, \mu, T) = q_Z(Z)q_\pi(\pi)q_{\mu T}(\mu, T)$$

The derivations of the update equations for the latent variables is shown in [4]. It simply follows the standard procedure outlined at the end of the last section, whereby the latent variable q_j is obtained by considering the log of the distribution over all hidden and visible variables and then taking the expectation with respect to the others. The parameter updates $[r_{jn}, \lambda_j, m_j, n_j, U_j]$ are also shown in [4].

3.2.4 Extensions

The full variational Bayes' EM is a powerful and common method for clustering of data. However, like any algorithm, it has problems and is sensitive to the initialization of the prior parameters such as the Dirichlet, α. As such (Chris Bishp et al.)[6] addressed this issue by not imposing a prior over the mixing coefficients but rather left it as a parameter. The interesting property of this algorithm is that there is a strong tendency to eliminate redundant components through setting the mixing coefficients to zero thus provides a good basis for model selection. The priors imposed over the means and covariances still prevent the draw backs of traditional EM, i.e. not letting the ELBO go to infinity [6]. Figure 3.4, illustrates the

$$q_Z(\mathbf{Z}) = \prod_{n=1}^{N}\prod_{j=1}^{M} r_{jn}^{z_{jn}}$$
$$q_\pi(\boldsymbol{\pi}) = \mathrm{Dir}(\boldsymbol{\pi}|\{\lambda_j\})$$
$$q_{\mu T}(\boldsymbol{\mu}, \mathbf{T}) = \prod_{j=1}^{M} q_\mu(\boldsymbol{\mu}_j|\mathbf{T}_j) q_T(\mathbf{T}_j)$$
$$q_\mu(\boldsymbol{\mu}_j|\mathbf{T}) = \prod_{j=1}^{M} N(\boldsymbol{\mu}_j; \mathbf{m}_j, \beta_j \mathbf{T}_j)$$
$$q_T(\mathbf{T}) = \prod_{j=1}^{M} W(\mathbf{T}_j; \eta_j, \mathbf{U}_j)$$

Figure 3.3: Approximate densities of latent variables [2]

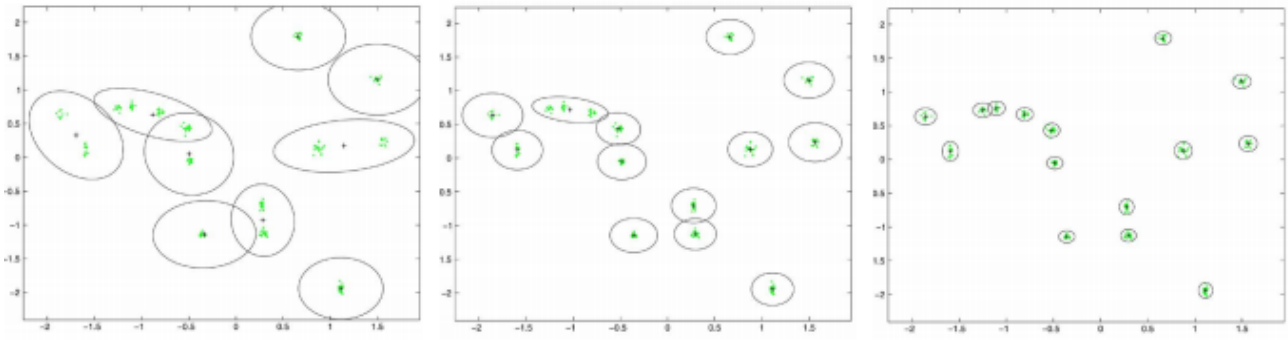

Figure 3.4: Sensitivity of the (Chris Bishp et al.) algorithm [5] to scale matrix V

algorithm's sensitivity to the scale matrix V of the Wishart prior imposed over the precision matrix. To address this an incremental method was developed in [5] whereby the data is initialing clustered via the (Chris Bishp et al.) method and then subsequently the data is restricted to a cluster where a local precision prior can be specified.

3.3 Optimization

Optimization is the area of mathematics concerned with finding the optima of a given function. In this section I will only cover some algorithms I have encountered which are gradient free. Gradient free methods all share one common them in which they iteratively sample the function in a randomized fashion and through these samples converge to optima. As such there are many methods left out of this section such as simulated annealing [30], tabu search [16] etc, CMA-ES [18], evolutionary algorithms such Particle Swam Optimization PSO.

3.3.1 Cross-Entropy Method

The Cross-Entropy, CE, method was developed by Rubinstein in 1999, whereby he realized a simple cross entropy modification of a previous paper on rare event simulation allowing for combinatorial optimization [9]. The basic thought process behind the math, is:

1. Generate samples according to the sample scheme
2. Update the sampling scheme to produce "better" samples next generation.

If $X = (X_1, X_2, ..., X_n)$ is a random vector taking values in the space of $\mathcal{X}$ and $S(X)$ is a real value function on $\mathcal{X}$. The probability of $S(X)$ being greater than a real number γ under $f(\cdot; u)$ is:

$$\ell = P_u(S(X) \geq \gamma) = E_u I_{S(X)\geq\gamma} \approx \frac{1}{N}\sum_{i=1}^{N} I_{S(X)\geq\gamma} \tag{16}$$

Whereby, equation 16, is the expectation of the indicator function. A straight forward way to evaluate equation 16 is through the crude Monte-Carlo approach however, for rare events this requires a large simulation effort. Thus an alternative is through importance sampling, see the section on Importance sampling. Sampling through a different density g on $\mathcal{X}$ we can write:

$$\ell = \frac{1}{N}\sum_{i=1}^{N} I_{S(X)\geq\gamma}\frac{f(X_i; u)}{g(X_i} \tag{17}$$

It is known that the best way to estimate ℓ is to use the change of measure of density [9].

$$g^*(x) := \frac{I_{S(X)\geq\gamma} f(x; u)}{\ell} \tag{18}$$

As this would mean the estimator in equation 17 would have an exact measurement of ℓ with only one sample and zero variance. However this is obviously not possible as you would have to know ℓ before you have calculated it. As well as the fact that $g(x)$ is generally chosen to be a family of pdfs $f(\cdot; v)$. This v is defined as the reference/titling parameter and is chosen such that the difference between the densities $g^*(x)$ and $f(\cdot; v)$ is minimal. The measure of difference used between densities is often the Kullback-Leibler divergence. Which in this case, is defined as:

$$\mathcal{D}(g^*, f(\cdot; v)) = \mathbb{E}_g \, ln \, \frac{g^*(X)}{f(X; v))} = \int g^*(x) ln \, g^*(x) - \int g^*(x) ln \, f(x; v)) \tag{19}$$

As can be seen from eq.19 minimizing the KL-divergence is equivalent to choosing v such that $\int g^*(x) ln\, f(x;v))$ is maximized:

$$\max_v \int g^*(x) ln\, f(x;v)) \tag{20}$$

Subsisting eq.18 and eq.16 into eq.20. Using a change of measure[2] $g(X) = f(X;w)$ we can write:

$$\underset{v}{D} = \max_v \mathbb{E}_w\, I_{S(X)\geq\gamma} \boldsymbol{W}(X;u;w) ln\, f(X;v) \qquad where \qquad \boldsymbol{W}(x;u,w) = \frac{f(x;u)}{f(x;w)} \tag{21}$$

An unbiased estimate of the expectation under u results with:

$$\max_v \mathcal{D}(v) = \max_v \frac{1}{N} \sum_{i=1}^{N} I_{S(X_i)\geq\gamma} \boldsymbol{W}(X_i;u;w) ln\, f(X_i;v)$$

Since the only term which depends on v is $ln\, f(X;v)$ differentiating with respect to v and setting to zero results with:

$$\frac{1}{N} \sum_{i=1}^{N} I_{S(X)\geq\gamma} \boldsymbol{W}(X_i;u;w)\, \nabla\, f(X_i;v) = 0 \tag{22}$$

If the family of classes is chosen to be in the natural exponential family the solution of eq.22 can be computed analytically. An issue with the use of the LR estimator is that the indicator function must have "enough" samples in which some are 1. As such a way to approach this is not fix γ but allow it to be changed. Iteratively updating the indicator value will allow the indicator function to have a sufficient amount of samples.

In terms of combinatorial optimization this is equivalent to increasing the cut off point of sample values i.e. indicator value. As such the algorithm is outlined nicely in [9] whereby the performance cut off point (indicator value γ) is increased whilst updating the titling parameter v until the performance value converges to an optimal value. However, since the initial parameter u does not matter as in combinatorial optimization one only searches for the distribution around the optima and not relating it back to the original density the parameter W is set to 1. So essentially all the math amalgamates to a simple algorithm in which one samples the space $\mathcal{X}$ with a simple distribution such as a Normal, then updates the parameters of the distribution through Maximum Likelihood with a subset of the samples related to the best performing samples.

The CE method is extremely similar to the classical Evolutionary algorithm, however its founding is in mathematical proof rather than bio-mimicry. The extensions of the CE method known as Fully Adaptive CE method (FACE) allow the CE method to be more robust and less function evaluation expensive through letting the performance quantile and number of samples to be adaptive.

3.3.2 Differential Evolution

Differential Evolution's basic idea is to move the population towards better regions in the search space whilst not converging too quickly but exploring the space sufficiently. Similarly to traditional EAs it consists of an initialization, mutation, crossover and selection phase. However, they differ in how they construct the mutant vector by adding the weighted difference between two individuals to a third. I refer the reader to [8] for an in depth study and [15] for a good broad understanding. The two control factors of the basic DE algorithm are the scaling parameter, **F**, and crossover rate, **Cr**. They control the weight

[2]The change of measure refers to the density in which samples were drawn from the chosen Importance Sampling density, see Importance Sampling Section

of the added difference between two individuals in the mutation step and probability of how many components will be included into the trial vector in the Crossover step, respectively. As it is fairly simple without relying heavily on mathematical proofs I will leave it to the reader to understand the simple steps.

Niching refers to how nature maintains a diverse population and in genetic algorithms it refers to how one can maintain multiple solutions during the run. The most common types of niching are *crowding* [10] and *fitness sharing* [17]. Often implemented these approaches within genetic algorithms requires parameter tuning which is often quite problem specific. Studies into the dynamics of DE revealed its natural ability to cluster around optima after a few iterations [13]. Through this understanding, a simple niching adaptation of classical DE was proposed whereby the individual chosen in the mutation step was nearest neighbor of the target individual, known as DE/nrand. It was further improved upon in [12] whereby the population size is adaptive. The main advantage of this approach is that it only modifies the classical DE algorithm with a few lines of simple code yet it is a powerful niching method as it does not rely on tunning a niching parameter.

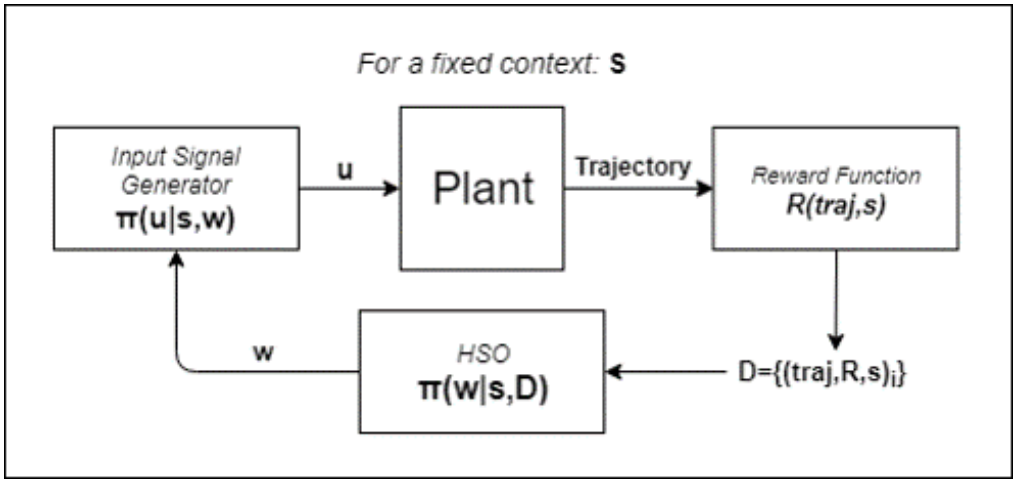

Figure 4.1: Problem Definition Block Diagram

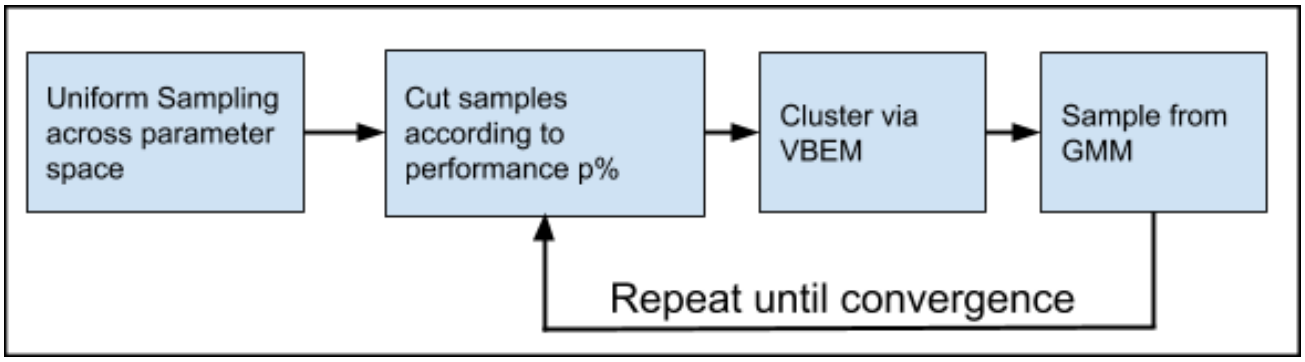

Figure 4.2: Stage one algorithm flow

4 Design Synthesis

The following section outlines the synthesis design steps taken in order to solve the project's problem, see section 1.2. Firstly, the preliminary work performed prior to the design is presented, following that the design phase is outlined. At each stage of the design the types of algorithms developed or concepts used are evaluated, based on the evaluation new steps are taken to improve the design. The stages are presented in discrete steps to allow the synthesis of the final algorithm design easy to explain. The overall idea is illustrated in the block diagram, figure 4.1. As the Amesim model was not yet available until the middle of August the development of the algorithm design had to be on test functions. As such I started with simple functions to understand more the concepts I was using and worked towards a more complicated design.

In general the design synthesis is based on the algorithm developed in [21] for hierarchical RL except with a fixed context. In RL the context refers to the properties which define the system and do not change between each iteration. Examples of context are starting states etc.

4.1 Preliminary Work

In order to show the company the applications of using gradient free optimization a simple application of applying the CE method to Proportional Integral controller tuning was presented to them. The reward function will be based on the time response of the system, such as overshoot, rising time, settling time, oscillations etc. The Simulink model is shown in figure 4.3. To illustrate how constructing the reward

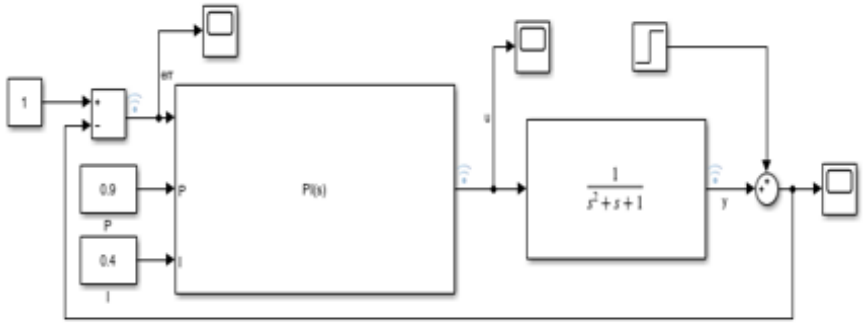

Figure 4.3: Basic Simulink Plant Model

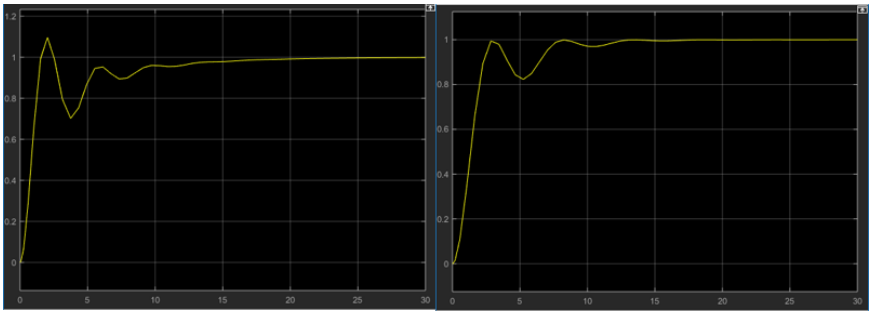

Figure 4.4: Step response of two different optimized parameters

signal influences how the optimal parameters are learned. The difference between the two step responses in figure 4.4 is how the reward function penalizes the overshoot.

4.2 Stage One

The initial idea was to use the Variational Bayes Expectation Maximization(VBEM) to cluster samples into a Gaussian Mixture Model (GMM) then iteratively cut samples and re-cluster according to some performance value. It is an approach similar to the Cross Entropy (CE)[9] method however for a mixture of multivariate Gaussians instead of just one. The flow diagram is shown in figure 4.2 and the simple 2D reward function used is shown in figure 4.5.

Initially, just like in the CE method, only a percentage of the total samples will used in the VBEM algorithm to fit the GMM that will in turn generate the next set of samples. The simulation results of applying this method to the simple reward function is shown in figure 4.6(Alg.1). The obvious problem with this approach is that just like the CE method it finds only one optima. It is likely that since it explores more of the parameter space through using a GMM the chance it convergences to the global optima and not local is probably higher.

In order to maintain multiple optima or modes of the reward function, the algorithm should cut the individual clusters‘ samples based on their associated reward value. This is essentially the same as running the CE method for different areas of the sample space whereby these areas are determined by each clusters’ mean and covariance. The simulation results of applying this proposed idea is shown in figure 4.6(Alg.2). As can be seen the simulation video highlights the significant downfall of this basic idea. The algorithm is extremely sensitive to the initialization of the sample space which is a uniform pdf across the parameter space.

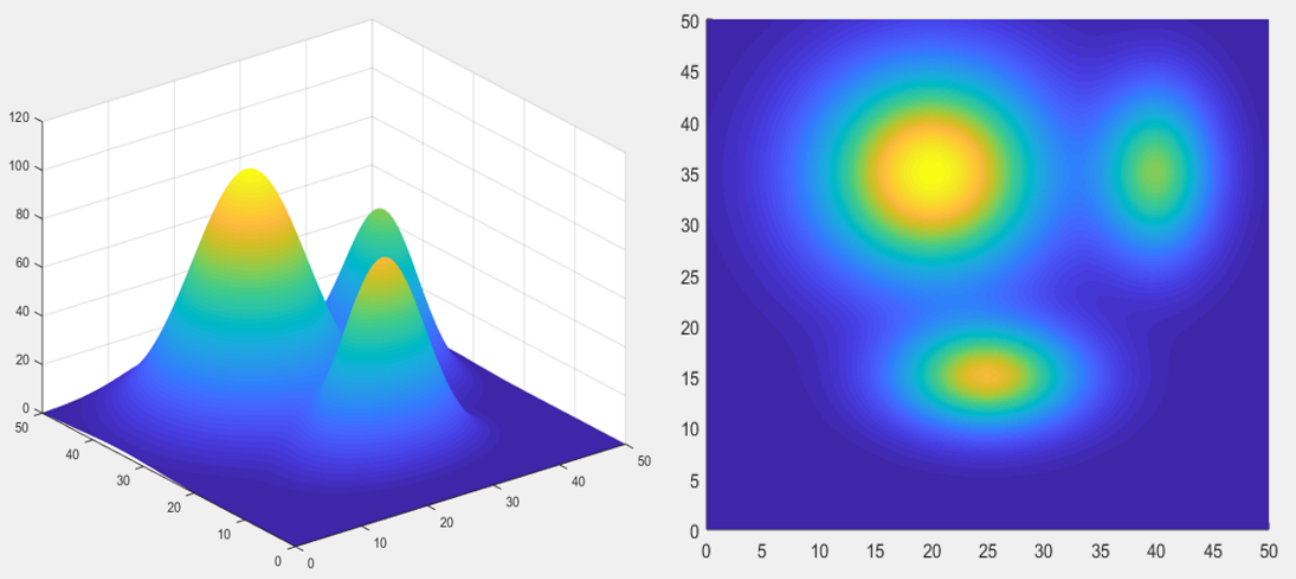

Figure 4.5: 2D Reward Function

4.2.1 Remarks

This section forms the basis of the thought process behind performing multi modal optimization in the reinforcement learning context. Even though it is simply presented it highlights important characteristics that need to addressed. As in RL there are two important attributes, namely, exploration vs exploitation. Stage ones' algorithms 1 and 2 illustrate those points respectively. Algorithm 1 first explores the parameter space and the converges to an optima, where as algorithm 2 exploited the local information provided by the initial clustering via VBEM. The initial clustering via VBEM in Alg.2 also highlights a very important point of the difference between classical unsupervised learning and unsupervised learning in the RL. In both cases the algorithm does not know the true labels or cluster assignments but it tries to find the best fit, Maximum likelihood see section 3.1, of the data to labels. However, in our case of RL we have the value of the reward for each sample in the data. So the clustering algorithm being used, VBEM, should somehow incorporate this knowledge into the assignment of the cluster labels in order to achieve the goal of finding the modes of the reward function. The above problems are addressed in Stage Two.

4.3 Stage Two

A way to incorporate the reward values of the samples would be to weight each sample by its associated reward. Whereby a sample with a higher reward should influence the fitting of the GMM model more than a sample with a lower reward. This can be achieved in VBEM through weighting the responsibility matrix, see section 3.1.3, relative to each samples reward value. Since samples are being generated from pdf we can use importance sampling/weights, see section 3.3, to achieve this [21]. Using importance weights (IW) also decreases the amount of samples needed. In order to see the effect of using IW to fit a Gaussian to samples with an associated reward value, see figure 4.9. As you can see the samples with higher rewards influence the fitting of the Gaussian more and thus cluster better represents the mode of the reward function.

Addressing the idea of exploration vs exploitation, ideally an algorithm should first sufficiently explore the parameter space and then exploit that information to hone in on multiple optima. Since we are only really interested in optima that are close in value to the global optima by a percentage of around $p = 15\%$. The proposed algorithm, Alg.3, will consist of two parts namely an exploration and exploitation phase. In the exploration phase will be similar to Alg.1 however the cutting performance is related to a percentage

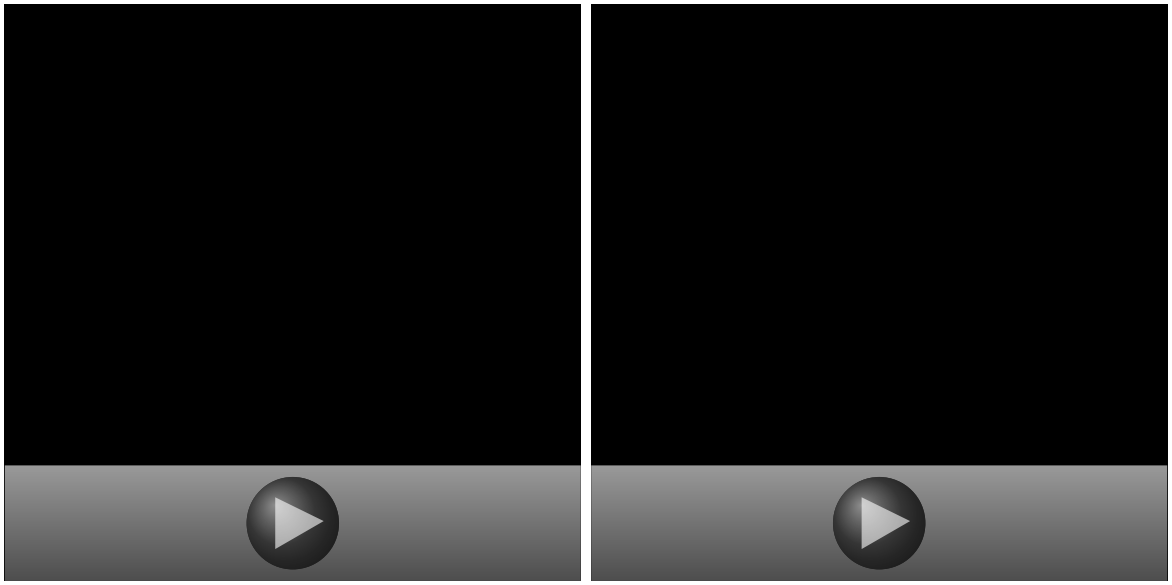

Figure 4.6: Simulation (left) Alg.1 cutting samples according the total
(right) Alg.2 cutting samples according to individual clusters

of the maximum reward value of the data and not a percentage of the whole reward set. The exploitation phase is the same as alg.2.

Algorithm 3 Stage Two Algorithm

procedure SIMULATIONPARAMETERS($N_{ini}, ClusterNum, Priors, ...$)
    Initialization of parameter space ▷ Uniform pdf
    for ExplorationCount **do** ▷ User set parameter of exploration
        Cluster via VBEM with IW ▷ Priors need to be set
        Cut %p of Rmax of Samples ▷ Similar to CE but with Rmax cutting condition
    while Covariance Matrices are >0.02 **do** ▷ Ensures convergence to optima
        Cluster via VBEM with IW ▷ Different priors need to be set
        Cut Clusters according to CE ▷ Alg.2
    return Cluster means as optima

The syntax of the algorithm is shown in alg.3. As can be seen the algorithm combines alg.1 and alg.2 in way which allows both exploration and exploitation after sufficient exploration has occurred. The results of applying the algorithm to the reward function 4.7 can be seen in figure 4.8. The reward function used in stage two is significantly more complex than stage one. Alg.3 handles the multi-modality of the function by first exploring the space and then essentially applies the CE method to the individual clusters until that local optima is reached.

4.3.1 Remarks

There are many issues with this algorithm but the main problem is the exploration side of it. The exploration side of it, is a variant of the CE method used in alg.1 but with the cutting performance being related to the maximum reward in the sample set rather than a fix percentage. The main point of the exploration part is to identify the basins of attraction. The basins of attraction being the local area in the

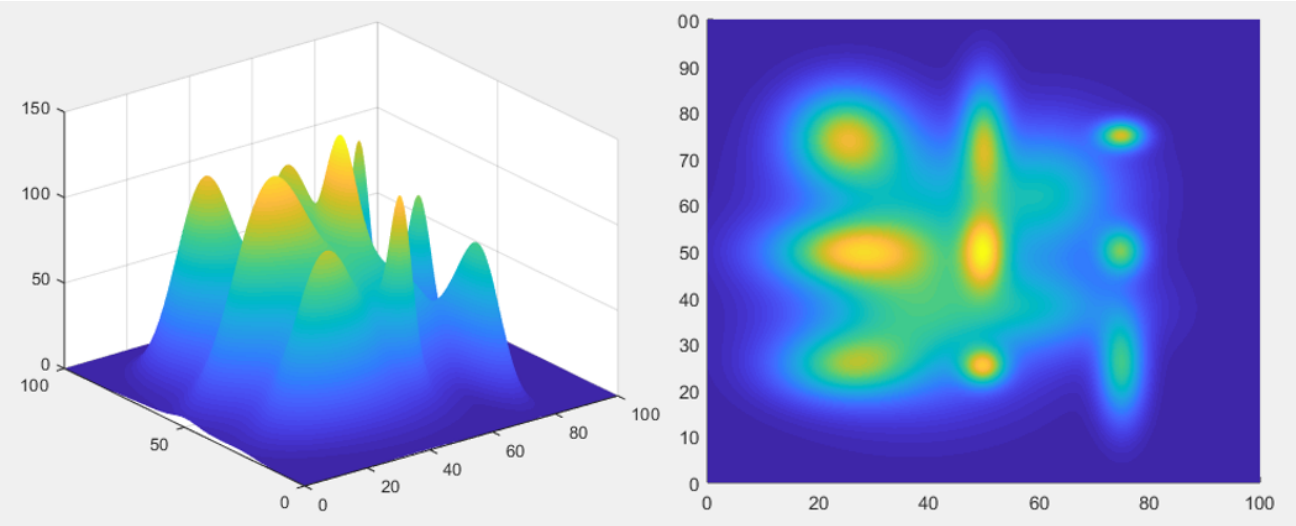

Figure 4.7: Stage Two test bed function

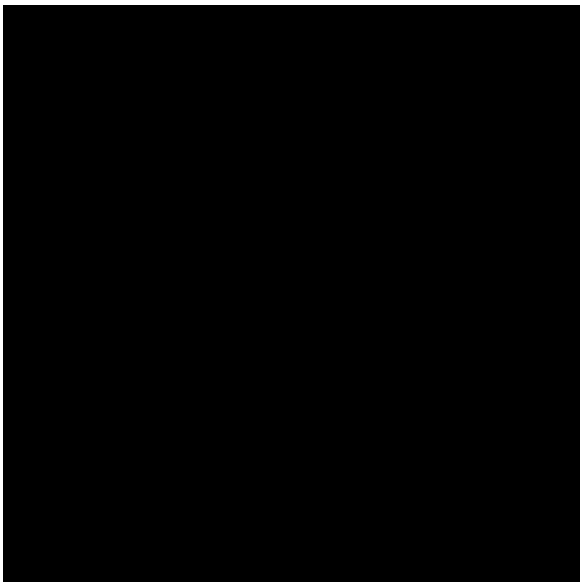

Figure 4.8: Simulation of alg.3 on test stage two test function 4.7

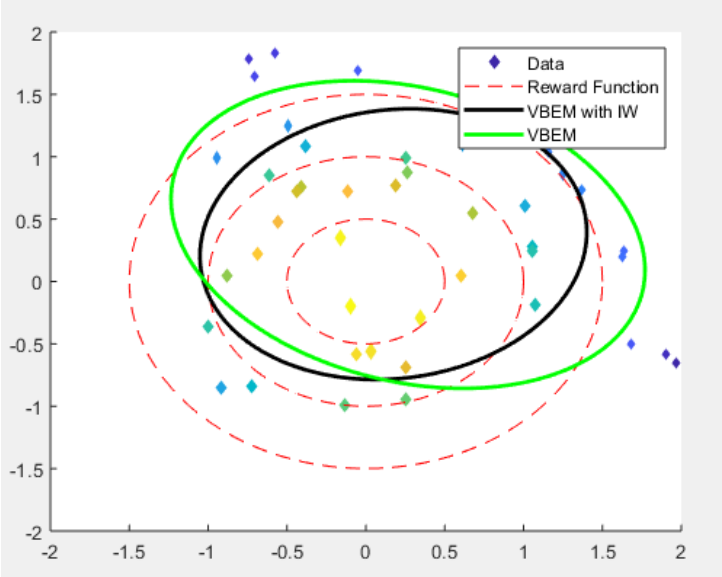

Figure 4.9: Difference between using importance weights

parameter space whereby a optima is located. I refer the reader to an excellent overview of multi modality optimization [24]. Once the basins of attraction have been found it is just a matter of applying a local search algorithm to hone in on the optima values. The exploration phase in alg.3 is not able to capture these basins of attraction adequately. A different exploration idea is outlined in Stage Three.

4.4 Stage Three

In interesting approach [23], in the field of evolution computation, is one where the authors first perform exploration of the parameter space via a niching variant of DE (DE_{nrand}), see section 3.3.2, and then cluster via k-Windows and subsequently optimize, with either PSO or DE, the local cluster information to find the optima of each cluster.

Applying an optimization algorithm to local cluster information whereby the cluster should be related to the basin of attractions of the different optima is not a new concept. In [24] the algorithm they suggest won the Congress on Evolutionary Computation (CEC) 2013 Competition. The idea is extremely simply whereby they initially uniformly sample across the parameter space and then cluster via the Nearest Best Neighbors approach (NBC). NBC is essentially K-means clustering but it takes into account the reward value of each of the samples into the clustering. They assume that NBC has sufficiently captured the relevant basins of attraction. The optimization algorithm they use after clustering is CMA-ES.

The issue with this type of approach lies in how the parameter space is initialized and first clustered. In [24] they note that the initialization/exploration of the parameter space is crucial and could be improved. As such this design phase is improving upon the exploration part of alg.3 whereby DE_{nrand} is used to perform basin of attraction search. Subsequently the "basins" are clustered through VBEM with importance weights and the each of the local cluster information is optimized through the CE method which is analogous to alg.2 in our case. The pseudo algorithm is outlined in alg.4. To highlight the difference between how modes of the reward function are captured with $DE_{nrand/1}$ and normal DE, please see figures 4.10 and 4.11. Both algorithms are run for the same number of iterations with similar parameters. As can be seen, $DE_{nrand/1}$ niches the reward function far better than DE as it is able to capture all the modes rather than just two.

Algorithm 4 Stage Three Algorithm

procedure SIMULATIONPARAMETERS($Population_{N_{ini}}, F, CR, ClusterNum, Priors, ...$)
  Initialization DE population ▷ Uniform pdf from N_ini
  for ExplorationCount **do** ▷ User set parameter of exploration
    DE_nrand/1 ▷ Use parameters F,CR,..
  Cluster via VBEM with IW ▷ Priors need to be set
  while Covariance Matrices are >0.02 **do** ▷ Ensures convergence to optima
    Cut Clusters according to CE ▷ Alg.2
  return Cluster means as optima

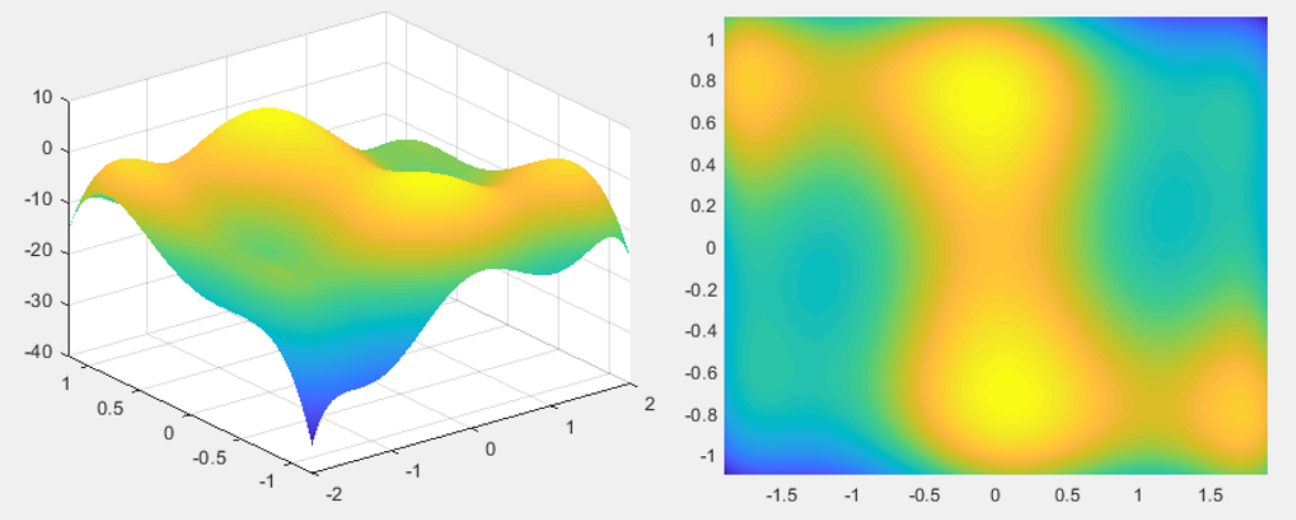

Figure 4.10: Stage Three Reward: Six Hump Camel from CEC2013 [19]

The Six Hump Camel Reward function, in 4.10, have two global and two local optima. Alg.4 was applied to this reward function for 20 simulations and the results of the amount of optima found are shown in figure 4.13. One instance of the results, in figure 4.13, are shown in figure 4.12. Whereby the left plot shows the clustering and the right the optima found. It clearly shows the algorithms inadequacy to locate the four optima consistently.

VBEM is rather sensitive to the priors imposed on it especially, the Dirichlet alpha prior, see Section 3.2.4. Using importance weights greatly improves the ability of VBEM to cluster around the modes of the reward function however it is still fairly sensitive. This is illustrated in figure 4.14. The data being clustered is the result of having applied the $DE_{nrand/1}$ algorithm for 10 iterations to a reward function with 4 exponential peaks. The data is clustered with different values of alpha and as can be seen only when $\alpha = 13$ does the clustering capture the four modes/peaks.

As such I meticulously researched Chris Bishop's paper, CB, [5] which a Variational Bayes EM is applied without the use of a Dirichlet prior. The paper's update equations are similar to that of the classical VBEM [4] except the weights over the clusters is calculated in the "Expectation" step of the EM algorithm. Since it has no Dirichlet prior of the weights, one must initialize the clusters using K-means where K is the maximum number of clusters. Essentially once this is done the CB algorithm eliminates clusters formed by K-means that could be better explained by fewer clusters. There is an extension of the CB algorithm whereby they then developed an incremental method of define the clusters after applying the CB algorithm to the data, see section 3.2.4. The number of "basins"/modes captured via the CB clustering algorithm for 20 simulations of applying the $DE_{nrand/1}$ and then clustering, is shown in figure 4.15.

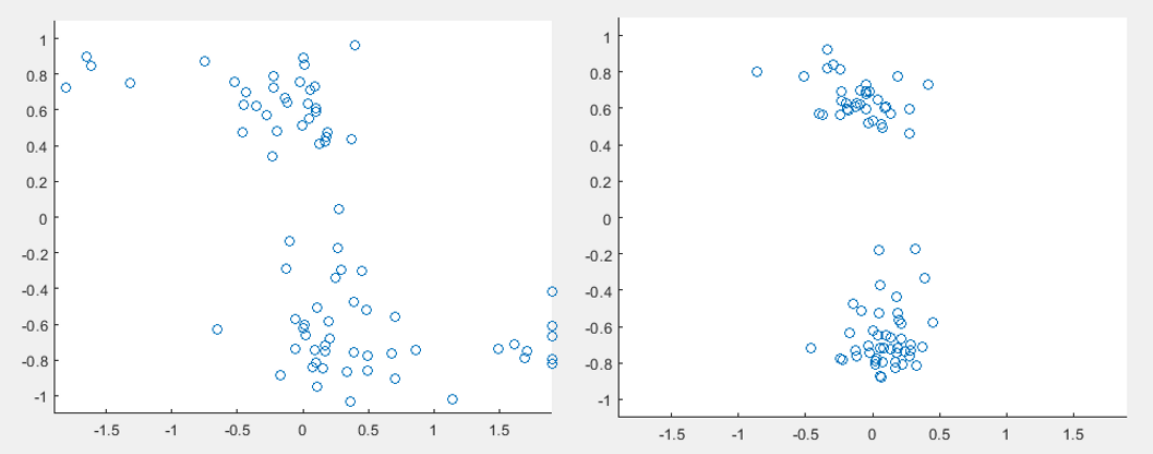

Figure 4.11: $DE_{nrand/1}$ (left) compared to DE (Right) on Six Hump Camel Reward, figure 4.10

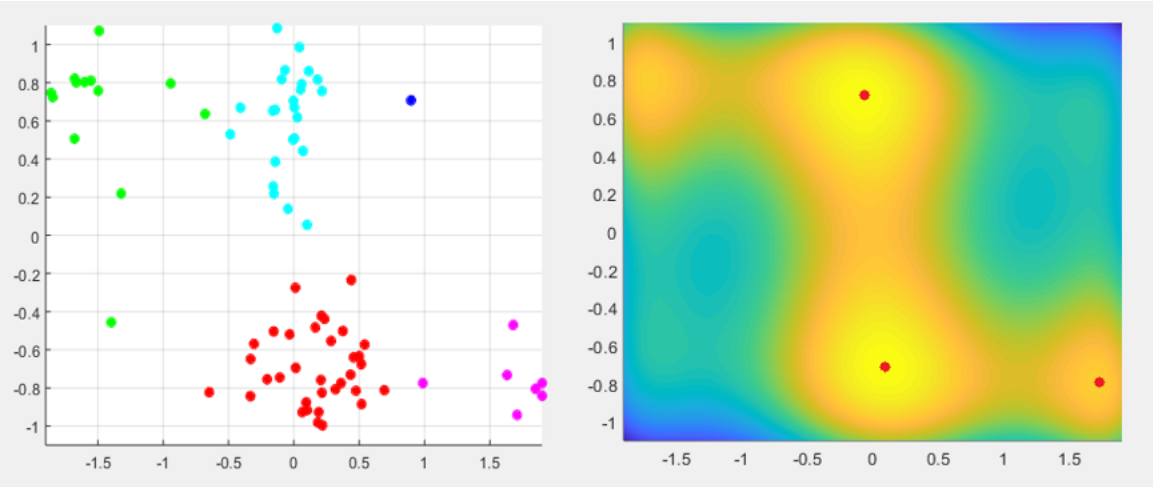

Figure 4.12: Results of alg.4 to six hump reward 4.10. (Left) Clustering after $DE_{nrand/1}$ (Right) Location of optima found, in red

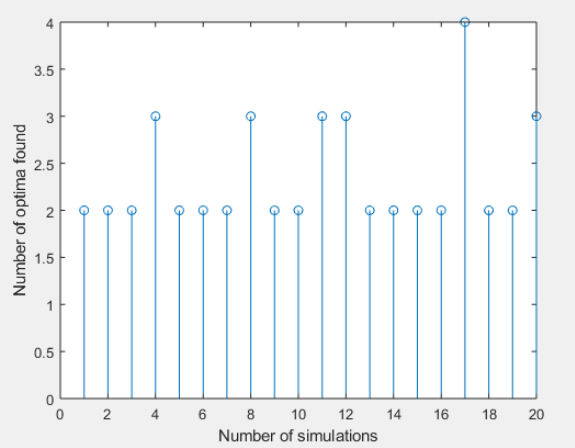

Figure 4.13: Simulation results of the number of optima found through applying alg.4 to the Six Hump Camel Reward Function

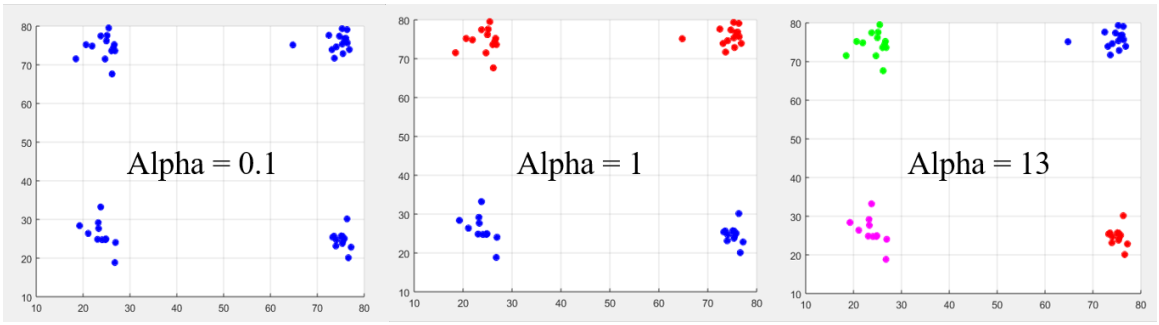

Figure 4.14: Clustering results of different Dirichlet alpha priors

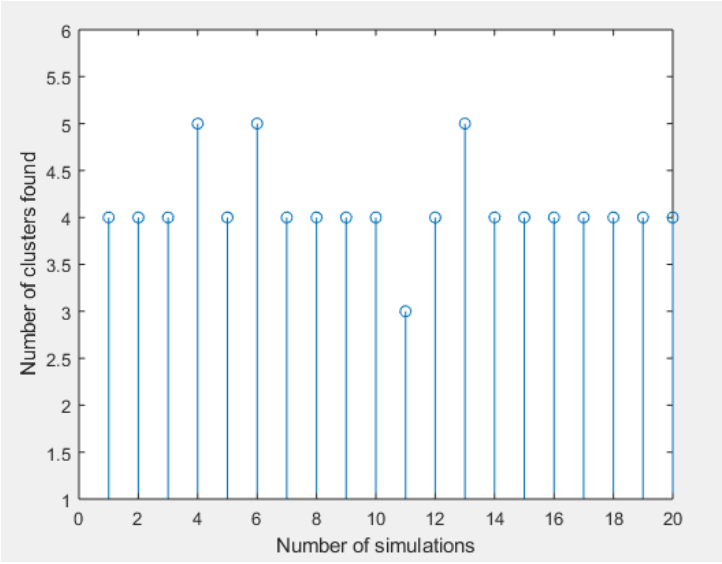

Figure 4.15: Number of modes captured via the CB algorithm[5]

4.4.1 Remarks

As can be seen in figure 4.13 the alg.4 performs adequately well, meaning it is consistently able to capture the global optima however it really struggles to locate the two local optima. The reason is do with the optimization algorithm used to hone in on the optima corresponding to each local clusters' samples. Even though the CE method is a powerful derivative free optimizer it is more considered a global optimizer. Meaning, in order for it to find the optima, the optima's point must be adequately contained inside the parameter interval used to initialize the CE method. In our case the parameter interval is each clusters samples, so if these samples did not sufficiently encompass the optima's point the CE method struggles to locate it. This is not often a problem as normally the whole parameter space is used as the interval and the CE method only tried to find the global optima. However, in alg.4's case often the local clusters data is not sufficient for the CE method to locate the optima point and thus converges to a sub optima value and terminates. And so, it is required to investigate a new optimization method to deal with this problem, this addressed in the next stage.

Regarding using the niching version of DE , DE_{nrand}, it obviously points to a good candidate as a substitute for initializing the parameter space with a uniform pdf. Not only is backed by literature, [23] [13], it also performed well in my on simulations on test functions. However, its relation to how much it to improves the overall algorithm, in terms of function evaluations etc, still needs to be investigated.

Finally regarding classical VBEM [4] vs the CB [5] method, clearly from figure 4.15, it can be seen the CB method performs very well. The variation of the number of modes clustered is more to do with the variation of the DE_{nrand}, however again it shows how sometimes the modes are not captured adequately enough for the CE method to perform well. In relation to incorporating this into the next stage's algorithm, I decided against it due to deadline constraints I would not have time to incorporate importance weights and thus it was advised by Prof. Osa to keep using VBEM with importance weights and if time allows it to try incorporate them into the CB method. It is a matter of adjusting the responsibility matrix relative to the data point's reward value and the probability value of how the data point was sampled.

4.5 Final Stage

This section outlines the synthesis of the final design which is to be test on the model provided by Kurumakaisha. In this section the algorithm will be tested on known reward functions and the following will be the practical implementation.

In order to address the issue of the optimizing algorithm used not being able to converge to the local cluster's associated optima. An algorithm which stems from more the field of RL is proposed, it is called Reward Weighted Regression (RWR) [22], see section 3.1.1. It is designed for policy search however in our case there is not context and so it reduces to essentially just a ML fitting of a weighted Gaussian. Through weighting the ML fitting by the reward values it allows the policy or cluster to move towards values which lead to higher rewards. This essentially means that if the clusters values do not sufficiently encompass the optima it will still allow it to move towards the optima point and converge. The weighting W_i will be the exponential which is "normalized" through the maximum value of the reward in the cluster or policy set, it is given as:

$$W_i = exp(10 * \frac{RewardSet - Rmax}{Rmax - Rmin})$$

Regarding the clustering via VBEM with IW, in order to deal with high dimensional non Gaussian data, the Laplacian Eigenmapping [3] is used. Also with regards to RWR in high dimensional space I chose to

rather use the diagonal covariance update version, which is given by:

$$\sigma^h = \frac{\sum_{i=1}^{N} W_i(\theta_i^h - \mu^h)^2}{Z}$$

In order to balance exploration vs exploitation, the method of upper confidence bounds for RL [1] is employed. It works by balancing the expected reward value of a policy and the confidence of how accurate that estimate is. In order to achieve mean and variance of the estimated reward we must approximate the reward function somehow, possible implementations could be Neural Nets, Auto-encoders but the simplest would be to use a Gaussian Process (GP)[25], see section 3.1.2. In terms of hierarchical RL, during the training process a context is chosen and based on that context each policy produces an action, the reward of these actions are then predicted with the GP approximate. The mean and variance of each policies' expected reward is evaluated in the UCB function below, and the policy with the largest UCB value is chosen to be used to sample the real model. The parameter α can be used to increase the exploration. In our case there is no context but since each policy is Gaussian the mean of each policies' will be the chosen parameter value and then the UCB process choses the policy based on the same criteria of exploration vs exploitation.

$$o^* = GPmean_h + \alpha * GPsigma_h$$

The pseudo code of the algorithm can be seen in alg.5. The "runtime" of the algorithm is governed by three values namely, how many times the data is clustered (ClusterNum), how often the policies are updated (UpdateNum) so as to converge to the optima, and how many samples are drawn from each chosen policy (SampleNum) via the UCB acquisition function. The value of O_{max} sets the maximum of how many clusters the data can be clustered into. The algorithm was implemented on two test functions in 2D, figure 4.16 shows the maximum reward found on a simple function. Whereas figure 4.17 illustrates the algorithm being applied to the more complicated function used in Stage 2, figure 4.7.

Algorithm 5 Stage Four Algorithm

Input: Max Clusters, O_{max}, Max times of clustering, updating the cluster policies,
Initialization Parameter Space ▷ Uniform pdf from N_ini or from $DE_{nrand}/1$
for ClusterNum **do** ▷ User set parameter of clustering
    Compute Importance Weights of each sample
    Label samples via VBEM with IW
    for UpdateNum **do**
        for each o up till O_{max} **do**
            Train the oth policy via RWR
        Select option o^* from UCB acquisition function
        for SampleNum **do**
            Sample using the o^*th policy and add to the o^*th policy sample set
        Train GP model to approximate the return function
Return: All clusters' means as optima

As can be seen, in figure 4.17, 5 out of the 9 optima are located. Which is much better than in stage two, and it is more to do with the relative values of each of the optima. It only captures optima that are relatively close to the global optima value. Since the reward function used in figure 4.16 is very simple it captures all 4 optima very quickly.

However, in order to really test the algorithms ability the dimensionality and number of modes must

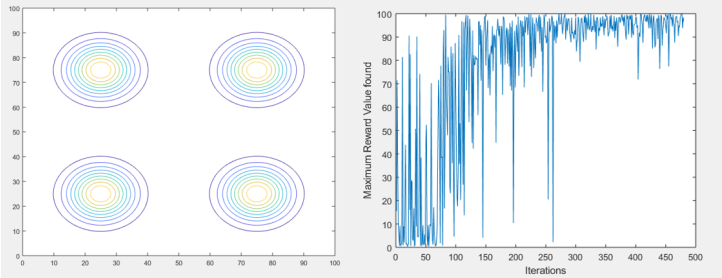

Figure 4.16: Results of applying alg.5 to simple 4 mode reward function (Left). Reward Values found by the algorithm (Right)

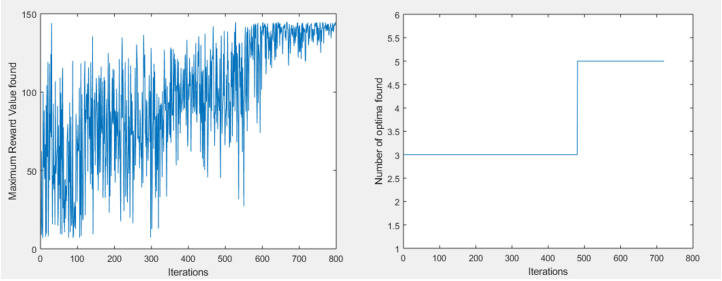

Figure 4.17: (Left) Maximum Reward Value found, (Right) number of optima located

be increased. The proposed reward function, see figure 4.18, is of dimensionality 8 and is extremely multimodal. In order to assess the functions modality, I applied the $DE_{nrand}/1$ the results are shown in figure 4.19. The population size was 800 and run for 5000 iterations. Running such an intense simulation through DE_{nrand} will capture the complicated function's modality and as such the figure clearly shows a convergence to at least thirteen optima of varying value.

As mentioned in the previous stage, the efficacy of using the DE_{nrand} as a pre-exploration of the parameter space instead of just using a uniform pdf needs to be addressed. From here on the former will be referred to as the Niching test vs the latter uniform test. As such two simulations on the reward function, figure 4.18, will be performed. In order to compare the two methods, the simulation must be the same in terms of parameters and function evaluations. As such the initialization of the parameter space in both methods is limited to 3000 function evaluations. In the case of the Niching test since DE_{nrand} uses $(i+1)*n$ function evaluations where $i = Iterations$ and $n = PopulationSize$, so the population size was set to 750 and was run for 3 iterations. This means the data at the end of 3 iterations of DE_{nrand} will be 750 points versus the 3000 data points obtained via uniformly sampling the parameters space. The figures 4.20 and 4.21 illustrate the difference of their simulations respectively.

```
R = 100*exp(-1*(((x1-20)/10).^2+((x2-50)/10).^2))...
    +80*exp(-1*(((x2-30)/14).^2+((x3-40)/15).^2))...
    +40*exp(-1*(((x3-40)/12).^2+((x4-30)/11).^2))...
    +75*exp(-1*(((x4-50)/20).^2+((x5-20)/10).^2))...
    +30*exp(-1*(((x5-50)/16).^2+((x6-20)/12).^2))...
    +80*exp(-1*(((x6-40)/15).^2+((x7-30)/15).^2))...
    +90*exp(-1*(((x7-30)/12).^2+((x8-40)/13).^2))...
    +100*exp(-1*(((x8-20)/14).^2+((x1-50)/12).^2));
```

Figure 4.18: 8 Dimensional Reward Function

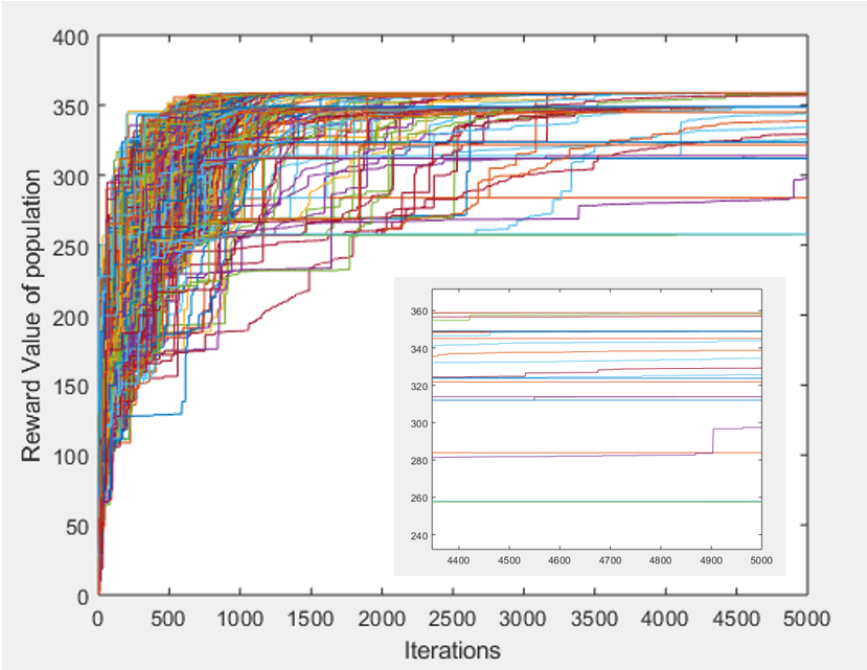

Figure 4.19: Individual's reward values

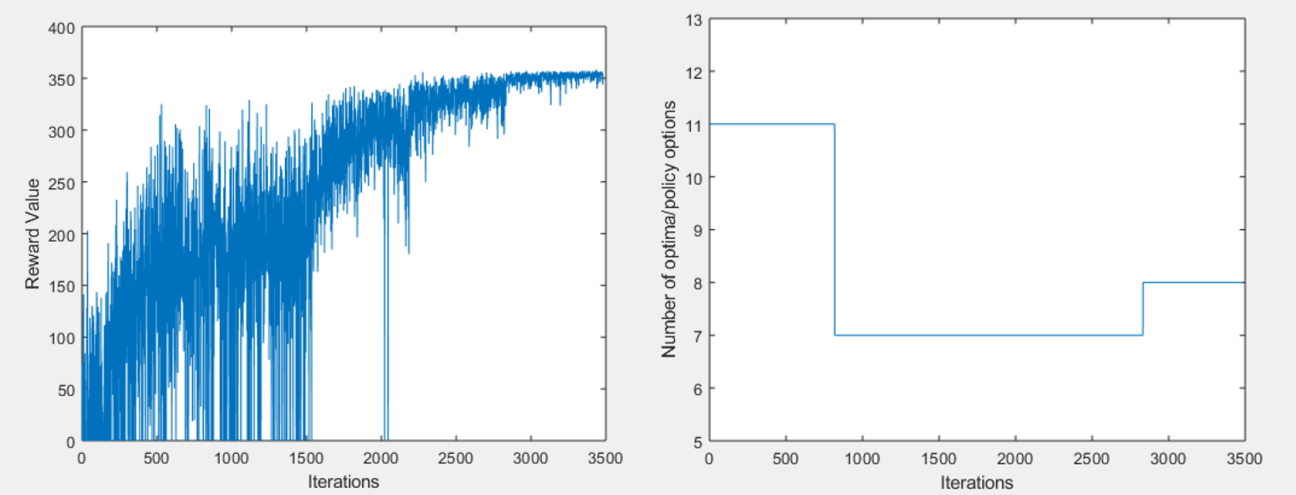

Figure 4.20: Using $DE/nrand$: (Left) Maximum Reward Value found, (Right) number of optima located

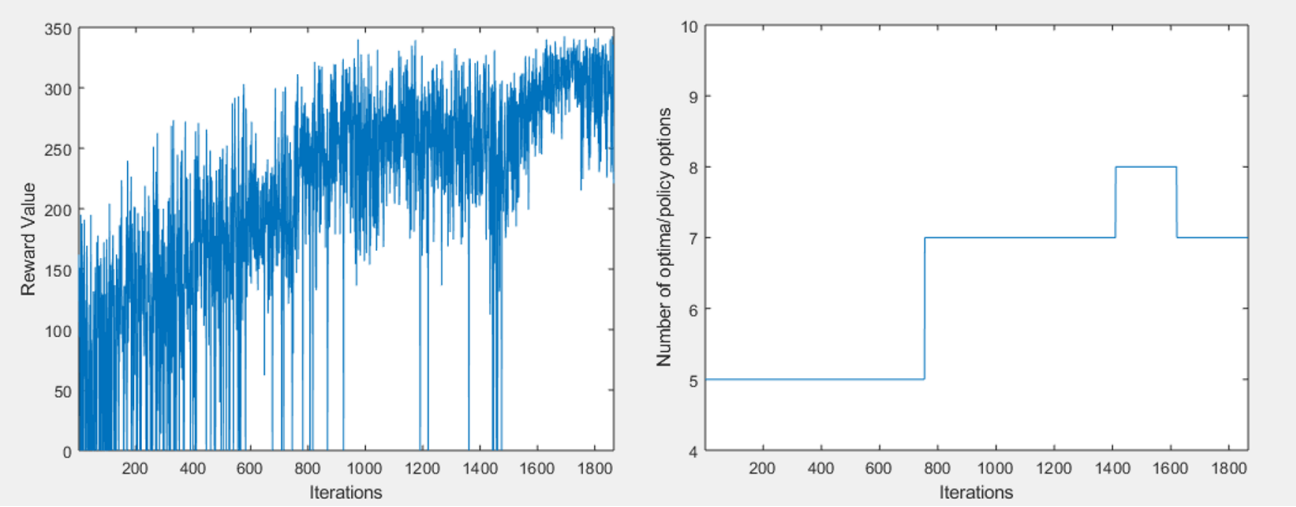

Figure 4.21: Uniform pdf: (Left) Maximum Reward Value found, (Right) number of optima located

4.6 Implementation

The Amesim model of the system to be optimized was only provided half way through August which left only around 8 working days for myself to test the alg.5 on the model. Firstly the Amesim model had to be converted such that it could be used in Simulink. In order to do that you have to set up Amesim using a C++ compiler from Microsoft Visual Studio. The document outlining this process is was lengthy and was provided by Siemens Japan, and thus was in Japanese as well. Once this was setup, Simulink could simulate the Amesim model with the parameters provided by the algorithm. As was stated in the project outline, it is not possible to provide a lot of details regarding the model however I will describe the results and where possible provide figures.

Essentially the output data, from the Amesim model, is time stamped data which corresponds to wheel rotational speeds, toques etc. The people at Kurumakaisha would input parameters which governs the oil pressure signal sent to the gearbox and look at the plots to evaluate the parameter human tuning. They evaluated the quality of the output signals based on certain points of interest which corresponded to maximum rotational speeds, jerk values etc. It is similar to PID tuning whereby the qualities looked at from a graph are overshoot rise time etc. They have values of each of the outputs which are desired and so the reward signal will be a sum of squared errors from the nominal/desired values. Since the sum of squared errors is a positive cost value from optimal control, to convert into a reward, the value is made negative and put into an exponential. Through using an exponential reward function of the returned reward values it allows reward values with higher values to be more important and based on the maximum entropy principle the exponential is best [21].

All the parameters have constraint bounds where they can only be on a certain interval due to mechanical properties etc. In order to deal with this there are many methods. The most common way is to just not allow the policies to generate parameter values outside the bounds and just retry until an acceptable value is drawn. However, a slightly more refined way to approach these bounds would be to penalize them heavily in the reward they generate. Meaning if the policy provides parameter values which are outside the bounds, the parameter values are not simulated but the returned reward is zero. Since we are using an exponential function of the returned reward value this is the smallest reward possible. This will not cost any more CPU time as no extra simulations of the model are needed but it would also allow the assumption of the policy being Gaussian to remain and allow policies to converge to optima points on the bounds if they so exist.

The hardward used to perform the simulation on the model is a my work laptop which has a Duel Core i7-6500U CPU @ 2.59GHz. A single function evaluation of the Amesim model through Simulink which is called via *sim()* function takes approximately $\approx 1s$ however, if simulations are run consecutively, the time is increased drastically. This most likely has to do with laptop overheating, not enough RAM etc. Like I mentioned before the delivery of the Amesim model was late and unfortunately the model they provided had some errors as they had to adjust it in a more understandable manner so I could link it to Simulink, however in that process some errors occurred and so I could only control 3 input parameters. They also provided the constraint bounds and the values they had been tuned by hand as a reference for how well my algorithm performed.

Unfortunately due to simulation time coupled with time constraints for the project's time line only the uniform pdf sampling for parameter space initialization was tested on the Amesim model. Five different types of reward functions of the output signals was designed and tested. The variation for the reward values was based on weights of the negative sum of squared errors of the output signals. The relative percentage increase compared to that of the provided human tuned parameters is shown in figure 4.22. As can be seen alg.5 consistently finds at least one optima which performs better in terms of the reward

Figure 4.22: Performance increase using different reward functions

function used. As the parameter space was only of size 3D and it was a error based reward function the problem was not extremely multimodal. However, on average alg.5 returned at least 3 different optima.

5 Conclusion

As can be seen in figures 4.16 and 4.17, alg.5 performs well in the 2D cases. However, in the case of an 8D highly multimodal function, figures 4.20 and 4.21, it does not capture all the optima seen by running the intense investigative simulation via $DE/nrand$, figure 4.19. The difference in simulation iterations between the two initializations methods is due again to time constraints of finishing the paper. Using a very large initial sample space (3000) makes the training of the GP extremely slow and thus the prediction of the reward value in the sample step slow as well. Thus in order to run the simulation test I had to decrease the amount of samples taken between policy updates in the uniform pdf test, figure 4.21. However, you can see it is tending to converge to the highest possible reward value and I believe it would reach it within the next 800 samples. The amount of optima found by both tests is relatively the same however the Niching initialization found an additional one. The main point to highlight between the two is that of simulation time. In order for the parameter space to be sufficiently initialized using a uniform pdf the value must be quite high and increases exponentially due to the curse of dimensionality. Thus even 3000 initial samples is not enough to capture all the basins of attraction. Coupled with if the initial sample size is increased it makes the training and utilization of the GP very slow and thus the simulation time is drastically increased. As such this shows the potential efficacy of using a niching initialization, such as $DE/rand$ of the parameter space instead. In the simulation comparison test in the Final Stage, the simulation parameters where similar however the uniform pdf took significantly longer to complete. As such if the comparison test was on simulation time not function evaluations the population size of the Niching initialization could be increased which would capture more basins of attraction for the algorithm to hone in on and thus find more optima points. Of course, if the processing power was increased this would not make such a difference.

As has been explained throughout the report, the ability for the optimization algorithm or policy up-

date to seek out optima is crucial. Regarding the CE method, it is a powerful optimization algorithm with simplistic yet mathematical founded proofs however, it lacks the ability to be a good local optimizer. As such I think the implementation of RWR was a good choice. It is extremely simple in its update equations and it has the ability to move towards an optima point located outside its initialization, too a certain degree. Since the project dealt with non-contextual policy/optimization another choice could be the CMA-ES [18]. Unlike the CE method it weights its samples by the reward, in a similar fashion to RWR as well. Through the use of updating the covariance matrix in different ways it also alleviates the problem of needing a large amount of samples to fit the Gaussian, as in the case of the CE method.

Regarding the implementation, due to time constraints and the model being delivered late the implementation was disappointing. Interfacing the complicated Amesim model with Matlab took quite a long time however, it is now setup up for future tests. Although I could not have run as many tests as I would of liked to. I still showed the efficacy of using my algorithm for the company Kuramakaisha. As can be seen, in figure 4.22, alg.5 consistently managed to find at least one optima that was more optimal than parameters tuned by the company. Although it is more optimal in the sense of the reward function used to evaluate the output, however through testing different types of reward weightings it shows the ability for the algorithm to always perform better than human tuning of the parameters.

6 Recommendations and Summery

Prof. Osa will continue to test and most likely implement a variate of my algorithm for the whole Amesim model. Moving forward, I would recommend possibly not using a GP but a Neural Network instead. Although it needs significantly more data it handles higher dimensionality much faster. The validity of using a evolutionary niching algorithm as an initialization of the parameter space still needs to be investigated more. My opinion is that it will help however it most likely will be problem specific as it relies on the function evaluation time to be small.

The work I have performed over the last six months has allowed myself to learn an incredible amount of information regarding the machine learning world. It has also allowed me the opportunity to write my thesis through the Japanese government research group called RIKEN-AIP.

References

[1] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[2] Thomas Bayes. The variational approximation for bayesian inference. 2008.

[3] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[4] Chris Bishop, Christopher M Bishop, et al. *Neural networks for pattern recognition*. Oxford university press, 1995.

[5] Constantinos Constantinopoulos and Aristidis Likas. Unsupervised learning of gaussian mixtures based on variational component splitting. *IEEE Transactions on Neural Networks*, 18(3):745–755, 2007.

[6] Adrian Corduneanu and Christopher M Bishop. Variational bayesian model selection for mixture distributions. In *Artificial intelligence and Statistics*, volume 2001, pages 27–34. Morgan Kaufmann Waltham, MA, 2001.

[7] Christian Daniel, Gerhard Neumann, and Jan Peters. Hierarchical relative entropy policy search. In *Artificial Intelligence and Statistics*, pages 273–281, 2012.

[8] Swagatam Das and Ponnuthurai Nagaratnam Suganthan. Differential evolution: a survey of the state-of-the-art. *IEEE transactions on evolutionary computation*, 15(1):4–31, 2011.

[9] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

[10] Kenneth Alan De Jong. Analysis of the behavior of a class of genetic adaptive systems. 1975.

[11] Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

[12] Michael G Epitropakis, Xiaodong Li, and Edmund K Burke. A dynamic archive niching differential evolution algorithm for multimodal optimization. In *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pages 79–86. IEEE, 2013.

[13] Michael G Epitropakis, Dimitris K Tasoulis, Nicos G Pavlidis, Vassilis P Plagianakos, and Michael N Vrahatis. Enhancing differential evolution utilizing proximity-based mutation operators. *IEEE Transactions on Evolutionary Computation*, 15(1):99–119, 2011.

[14] Thomas A Feo and Mauricio GC Resende. Greedy randomized adaptive search procedures. *Journal of global optimization*, 6(2):109–133, 1995.

[15] Kelly Fleetwood. An introduction to differential evolution. In *Proceedings of Mathematics and Statistics of Complex Systems (MASCOS) One Day Symposium, 26th November, Brisbane, Australia*, pages 785–791, 2004.

[16] Fred Glover and Manuel Laguna. Tabu search. In *Handbook of combinatorial optimization*, pages 2093–2229. Springer, 1998.

[17] David E Goldberg, Jon Richardson, et al. Genetic algorithms with sharing for multimodal function optimization. In *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, pages 41–49. Hillsdale, NJ: Lawrence Erlbaum, 1987.

[18] Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.

[19] Xiaodong Li, Andries Engelbrecht, and Michael G Epitropakis. Benchmark functions for cec'2013 special session and competition on niching methods for multimodal function optimization. *RMIT University, Evolutionary Computation and Machine Learning Group, Australia, Tech. Rep*, 2013.

[20] Rafael Martí, Jose A Lozano, Alexander Mendiburu, and Leticia Hernando. Multi-start methods. *Handbook of Heuristics*, pages 1–21, 2016.

[21] Takayuki Osa and Masashi Sugiyama. Hierarchical policy search via return-weighted density estimation. *arXiv preprint arXiv:1711.10173*, 2017.

[22] Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pages 745–750. ACM, 2007.

[23] Vassilis P Plagianakos. Unsupervised clustering and multi-optima evolutionary search. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 2383–2390. IEEE, 2014.

[24] Mike Preuss. *Multimodal optimization by means of evolutionary algorithms*. Springer, 2015.

[25] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced lectures on machine learning*, pages 63–71. Springer, 2004.

[26] Mandavilli Srinivas and Lalit M Patnaik. Genetic algorithms: A survey. *computer*, 27(6):17–26, 1994.

[27] Masashi Sugiyama. *Introduction to statistical machine learning*. Morgan Kaufmann, 2015.

[28] Richard S Sutton and Andrew G Barto. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[29] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[30] Peter JM Van Laarhoven and Emile HL Aarts. Simulated annealing. In *Simulated annealing: Theory and applications*, pages 7–15. Springer, 1987.