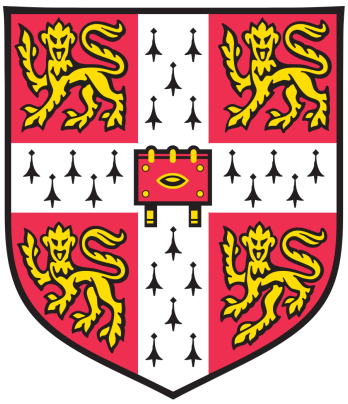
Fashion Products Identification Using Bayesian Latent Variable Models



Areeb Ur Rehman Siddique
Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of Master of Philosophy in Machine Learning, Speech and Language Technology

Darwin College

16 August 2018

Declaration

I, Areeb Ur Rehman Siddique of Darwin College, being a candidate for the MPhil in Machine Learning, Speech and Language Technology, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose.

This thesis contains 9,399 words.

Areeb Ur Rehman Siddique
16 August 2018

Acknowledgements

I would like to sincerely thank my supervisor Dr. Javier González from Amazon Research Cambridge for his deft guidance throughout this project and for giving me very valuable feedback at various stages.

Abstract

Bayesian optimization does not work well in high dimensional spaces. We implement an approach called "Latent Variable Bayesian Optimization (LVBO)" to solve this problem for a particular use case: finding the closest match for an image of an item of clothing. The images we consider are high dimensional and the closest match for them cannot be found through standard Bayesian optimization.

LVBO involves finding a low dimensional "latent" space corresponding to the "image" space and then carrying out Bayesian optimization in the latent space. Since probabilistic dimensionality reduction methods are employed to find the latent space, every point in the latent space has an uncertainty associated with it. We explore two methods to propagate this uncertainty to the Bayesian optimization: by using a Bayesian GPLVM as the surrogate model instead of the standard Gaussian process, and by replacing Euclidean distance in the surrogate model kernel with Hellinger or Geodesic distance. The latter method is our main novel contribution.

We found that LVBO works quite well for our particular use case. Although propagating uncertainty in the optimization through the Bayesian GPLVM surrogate model did not yield any better results than the standard Gaussian process surrogate model, Hellinger and Geodesic distances were found to outperform the standard Euclidean distance.

Contents

1 Introduction **1**

2 Background **3**

2.1 Gaussian Processes 3

2.1.1 GP Regression Model 4

2.1.2 Learning Hyperparameters of GP Regression Model 5

2.2 Dimensionality Reduction Methods 6

2.2.1 Gaussian Process Latent Variable Model 8

2.2.2 Bayesian GPLVM 11

2.2.3 Variational Autoencoder 12

2.3 Bayesian Optimization 14

3 Latent Variable Bayesian Optimization **18**

3.1 Specific Use Case 19

3.2 LVBO Metamodels 20

3.3 Dimensionality Reduction Model 21

3.4 Surrogate Model 21

3.4.1 Hellinger Distance 22

3.4.2 Geodesic Distance 23

3.5 LVBO Algorithm 24

4 Experiments **26**

4.1 Selecting a Dimensionality Reduction Model 26

4.2 Experiment Parameters 29

4.3 LVBO vs Bayesian Optimization 30

4.4 Effect of Surrogate Model on LVBO 32

4.5 Effect of Surrogate Model Kernel Distance on LVBO 33

4.6 Effect of No. of Iterations and Initialization on LVBO 36

5 Conclusion and Further Work **38**

References **39**

A Code **41**

B Dimensionality Reduction Model Parameters **41**

C Detailed LVBO Algorithm **42**

List of Figures

1 Example image. 1

2 GP regression example. 6

3 Graphical model of GP regression. 6

4 Examples of images in the Fashion MNIST dataset. 7

5 Graphical model of the GPLVM. 11

6 Graphical model of the VAE. 14

7 Bayesian optimization on the 1D Forrester function. 17

8 Graphical model for LVBO. 20

9 Latent space for VAE. 27

10 Latent space for Bayesian GPLVM. 27

11 Closest matches generated for test image using LVBO and BO. 30

12 Comparison of LVBO and BO convergence. 31

13 Evolution of closest matching image during LVBO. 31

14 Comparison of how LVBO converges with different distance types. . . 34

15 Latent space for Bayesian GPLVM, with uncertainty shown. 37

List of Tables

1 Classification Error % for the various clothing items. 28

2 Parameters that were varied. 29

3 LVBO Success % for different surrogate models and distances. 32

4 Comparison of distance types for Gaussian process as surrogate model. 33

5 Comparison of distance types for Bayesian GPLVM as surrogate model. 33

6 Time taken by LVBO for different surrogate models and distances. . . 35

7 Hyperparameters for VAE training. 41

8 Hyperparameters for Bayesian GPLVM training. 41

1 Introduction

Bayesian optimization is a popular technique employed to find the minimum or maximum of an objective function, which it treats as a black box. It is mainly used for objective functions that are computationally expensive to evaluate [1]. However, Bayesian optimization does not work well when the dimensionality of the space in which the objective function lies is large (i.e., generally when the number of dimensions is more than 10) [2].

Figure 1: Example image of size 28×28 pixels.

An example would clarify the problem. Consider that the objective is to find an "optimal" image that closely matches the image of size 28×28 pixels in Figure 1. Assuming that we have to find a binary optimal image, in which each pixel can either be black or white (i.e., each pixel can have a value of either 0 or 1), the number of possible images that exist is $2^{(28\times28)}$. This number is much larger than the number of atoms in the universe and it would be very difficult to find the optimal image by searching in a space of this size.

Although some work has been done on this problem of Bayesian optimization not working well in high dimensional spaces, the problem remains open. One approach that has been applied to solve the problem is Random EMbedding Bayesian Optimization (REMBO) [2], which randomly selects a low dimensional subspace (embedding) from the original space and carries out Bayesian optimization in that subspace. Another approach decomposes the high dimensional objective function into several lower dimensional "additive" functions and optimizes each of them separately [3].

Recently, in [4], a probabilistic dimensionality reduction method called the Variational Autoencoder was used to learn a low dimensional space "latent" space from the high dimensional space. Bayesian optimization was then applied in the latent

space, and the result obtained was mapped back to the high dimensional space. This approach was applied to the problem of finding the optimal combination of kernels in a Gaussian process and competitive results were obtained.

In this thesis, we follow a similar approach to [4] and call it "Latent Variable Bayesian Optimization (LVBO)". We map the original optimization problem to a lower dimensional latent space using probabilistic dimensionality reduction methods. The main novelty introduced here is the way in which the uncertainty in the latent space is propagated to the optimization.

To test the ideas proposed in this thesis, we address the following problem: given an image of an item of clothing, find the image that matches it closest. The dataset used was Fashion MNIST, which contains 28x28 grayscale images of items of clothing. [5]. Each image in the dataset is associated with one of 10 labels, from t-shirts to sneakers. An example Fashion MNIST image is given in Figure 1.

This thesis is organized as follows. Section 2 goes over the background theory necessary to understand LVBO. Section 3 details the LVBO method and discusses the ways in which we propagate uncertainty. Section 4 presents the experimental setup used to evaluate LVBO and analyses the results obtained. Finally, Section 5 sums up the thesis and presents ideas for further work.

2 Background

In this section, we discuss the theoretical concepts necessary to understand Latent Variable Bayesian Optimization. An overview of Gaussian processes, dimensionality reduction models and Bayesian optimization is presented.

2.1 Gaussian Processes

A Gaussian process (GP) is a distribution of functions, formally defined as "a collection of random variables, any finite number of which have a joint Gaussian distribution [6]." To fully specify a GP, we need to define its mean function $m(\mathbf{x})$ and its covariance function $k(\mathbf{x}, \mathbf{x}')$:

$$f(\mathbf{x}) \sim GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')). \tag{1}$$

The mean function is usually considered to be zero as this simplifies notation. A common covariance function is the Radial Basis Function (RBF), also called the Squared Exponential Function. It gives the covariance between each pair of random variables and is defined as:

$$cov(f(\mathbf{x}_p), f(\mathbf{x}_q)) = k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{x}_p - \mathbf{x}_q)^T M(\mathbf{x}_p - \mathbf{x}_q)\right) \tag{2}$$

where σ_f^2 is a constant that controls the overall variance of the function and M is a matrix that can take a variety of forms. When $M = diag(\boldsymbol{l})^{-2}$, where $\boldsymbol{l}$ is a vector containing a scalar value corresponding to each dimension of $\mathbf{x}$, the kernel is called ARD-RBF [6].

ARD stands for Automatic Relevance Determination and refers to the fact that the scalar values in $\boldsymbol{l}$, called characteristic length-scales, allow the assignment of different levels of importance (relevance) to different dimensions of $\mathbf{x}$. If the characteristic length-scale corresponding to a certain dimension has a very large value, that dimension will be assigned very low relevance. This means that if a feature of $\mathbf{x}$ contributes very little to pattern changes in $f(\mathbf{x})$, its characteristic length-scale will have a very large value and it will effectively be ignored [6].

An important thing to note about Equation 2 is that, when M is a diagonal matrix, $(\mathbf{x}_p - \mathbf{x}_q)^T M(\mathbf{x}_p - \mathbf{x}_q)$ is the square of the Euclidean distance between $\mathbf{x}_p$ and $\mathbf{x}_q$, calculated after each dimension has been scaled by its characteristic length-scale.

2.1.1 GP Regression Model

Now, consider that we have some training data points $\mathbf{x}_i$, noisy values $y_i = f(\mathbf{x}_i) + \varepsilon$ where $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$, and some prior knowledge of the relationship between the variables in $\mathbf{x}$ (i.e., how the variables depend on each other). The objective is to predict the value $f(\mathbf{x}*)$ for a test data point $\mathbf{x}*$. We use the GP regression model, detailed in the following paragraphs, to do so.

We can say that the covariance between the noisy values y_i is:

$$cov(y_p, y_q) = k(\mathbf{x}_p, \mathbf{x}_q) + \sigma_n^2 \delta_{pq} \tag{3}$$

where δ_{pq} is a Kronecker delta (which has a value of 1 if and only if $p = q$ and a value of 0 otherwise) [6].

This can also be written as:

$$cov(\mathbf{y}) = K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I} \tag{4}$$

where $\mathbf{y}$ is a vector containing the y_i values, $\mathbf{X}$ is a matrix with training data points $\mathbf{x}_i$ as its rows, $\mathbf{I}$ is the identity matrix and $K(\mathbf{X}, \mathbf{X})$ is a matrix of the covariances calculated for all pairs of training data points [6].

Denoting the matrix of the covariances calculated for all pairs of training and test data points as $K(\mathbf{X}, \mathbf{X}*)$, and similarly defining $K(\mathbf{X}*, \mathbf{X})$ and $K(\mathbf{X}*, \mathbf{X}*)$, we can say that the joint distribution of the training and test function values is given by:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(\mathbf{X}, \mathbf{X}) + \sigma_n^2 \mathbf{I} & K(\mathbf{X}, \mathbf{X}*) \\ K(\mathbf{X}*, \mathbf{X}) & K(\mathbf{X}*, \mathbf{X}*) \end{bmatrix}\right). \tag{5}$$

To get the predictive distribution for a single test data point $\mathbf{x}*$, we use the property that Gaussians are closed under marginalization and conditioning. Therefore, the predictive distribution is also Gaussian. Conditioning the joint distribution in Equation 5 on $\mathbf{X}$ and $\mathbf{y}$, the predictive distribution $\mathcal{N}(\overline{f}_*, V[f_*])$ for $\mathbf{x}*$ is given by:

$$\overline{f}_* = \mathbf{k}_*^{\mathsf{T}} (K + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y} \tag{6}$$

$$V[f_*] = k(\mathbf{x}*, \mathbf{x}*) - \mathbf{k}_*^T (K + \sigma_n^2 \mathbf{I})^{-1} \mathbf{k}_* \tag{7}$$

where K denotes $K(\mathbf{X}, \mathbf{X})$ and $\mathbf{k}_*$ denotes the vector containing the covariances between $\mathbf{x}*$ and the training data points [6].

2.1.2 Learning Hyperparameters of GP Regression Model

In the previous section, the structure of the GP regression model was discussed. This section details how the model is trained. Training the GP regression model means finding optimal values for the kernel hyperparameters. This is done by maximizing the marginal likelihood of the GP. The marginal likelihood is:

$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{f}, \mathbf{X}) p(\mathbf{f}|\mathbf{X}) d\mathbf{f} \tag{8}$$

where $p(\mathbf{y}|\mathbf{f}, \mathbf{X})$ is the likelihood, $p(\mathbf{f}|\mathbf{X})$ is the prior and the marginalization takes place over the function values $\mathbf{f}$ [6].

Solving the integral in Equation 8 and taking the log yields the following log marginal likelihood:

$$\log\left(p(\mathbf{y}|\mathbf{X})\right) = -\frac{1}{2}\mathbf{y}^T(K + \sigma_n^2\mathbf{I})^{-1}\mathbf{y} - \frac{1}{2}\log|K + \sigma_n^2\mathbf{I}| - \frac{n}{2}\log 2\pi. \tag{9}$$

The likelihood in Equation 9 is implicitly conditioned on $\boldsymbol{\theta}$, the hyperparameters of the covariance function, as well. (For the ARD-RBF kernel in Equation 2, $\boldsymbol{\theta}$ refers to σ_f^2 and M.) We make this condition explicit:

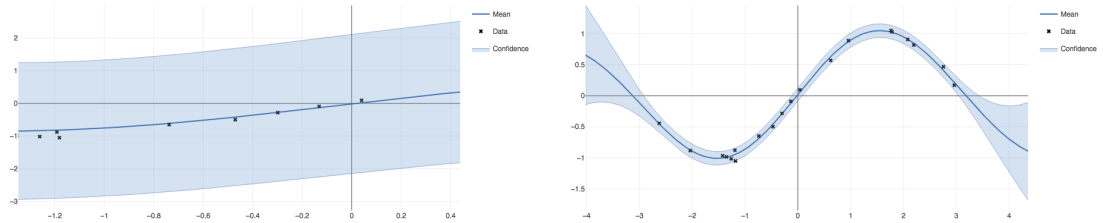
$$\log\left(p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta})\right) = -\frac{1}{2}\mathbf{y}^T(K + \sigma_n^2\mathbf{I})^{-1}\mathbf{y} - \frac{1}{2}\log|K + \sigma_n^2\mathbf{I}| - \frac{n}{2}\log 2\pi. \tag{10}$$

optimization techniques, such as gradient descent, are used to find the values of $\boldsymbol{\theta}$ that maximize the marginal likelihood in Equation 10 [6]. The equation used for the gradient is:

$$\frac{\partial}{\partial\theta_j}\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}) = \frac{1}{2}\mathbf{y}^\mathsf{T}K^{-1}\frac{\partial K}{\partial\theta_j}K^{-1}\mathbf{y} - \frac{1}{2}\,tr\left(K^{-1}\frac{\partial K}{\partial\theta_j}\right) \tag{11}$$

$$= \frac{1}{2}\,tr\left((K^{-1}\mathbf{y})(K^{-1}\mathbf{y})^\mathsf{T} - K^{-1})\frac{\partial K}{\partial\theta_j}\right) \tag{12}$$

Figure 2 shows an example of GP regression in 1D: the prior with the initial kernel hyperparameters and the posterior with the optimized hyperparameters.



(a) Prior with the initial kernel hyperparameters.

(b) Posterior with the optimized kernel hyperparameters.

Figure 2: GP regression example. The shaded region represents the 95% confidence region for every possible x value. The dark blue line represents the mean of the GP and the black dots are the training data points. Figures have been taken from [7].

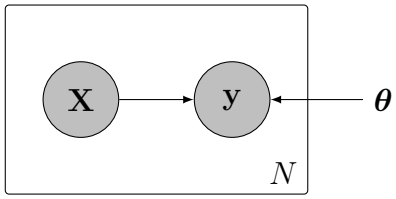


Figure 3: Graphical model of GP regression. Arrows indicate dependency. Shaded circles represent observed variables.

2.2 Dimensionality Reduction Methods

Dimensionality reduction methods are employed on data that has a large number of features containing different amounts of useful information. The aim of dimensionality reduction is to represent the data points with only a few features that encode most of the useful information. Most often, this is equivalent to finding a low dimensional manifold, on which the data points actually lie, within the original higher dimensional space [8].

For instance, consider the Fashion MNIST dataset which has images of size 28x28 pixels; some examples are shown in Figure 4. Considering each pixel to be a single feature, each image has 784 features (or dimensions). If the task at hand is to classify the image and select a category for it from a few different options, this can be done by considering only a few features (numbering far less than 784). Therefore, a dimensionality reduction method can be deployed to come up with a low dimen-

sional representation of the image.

(a) Sneaker. (b) Pullover. (c) Trouser.

Figure 4: Examples of images in the Fashion MNIST dataset.

Employing dimensionality reduction methods before doing further statistical analysis/machine learning reduces the amount of memory required to store the data and the amount of time required to train the models. Additionally, if the data is reduced to 2 or 3 dimensions, it can be easily visualized. Dimensionality reduction can even improve the accuracy of the model by removing multi-collinearity[1] from the data [9].

Dimensionality reduction methods can be classified in many different ways; for example, linear or non-linear. Linear methods, as opposed to non-linear methods, employ a linear transformation to come up with a low dimensional representation of the data. As non-linear methods are generally more suited to real-world data, we will only consider these in this thesis.

Another way of classification is non-probabilistic methods versus probabilistic methods [10]. For a test data point, the former only return a corresponding point in the lower dimensional space while the latter also return the uncertainty associated with it. Since this additional information can be meaningfully used in achieving the objective of this thesis[2], only probabilistic methods are considered.

Dimensionality reduction methods can also be divided into categories based on the approach used to find the low dimensional manifold. A popular approach used to find the low dimensional manifold is to optimize a function using a Gaussian process. This approach, called the 'Bayesian Gaussian Process Latent Variable Model (Bayesian GPLVM)' [11], is the first method explored in this thesis. Another approach is to train a neural network called an autoencoder. A probabilistic version

[1]Multi-collinearity refers to redundant features; for example, one feature might be length in centimeters and another might be length in inches.

[2]As described in Section 1.

of the autoencoder is called the Variational Autoencoder (VAE) [12]. VAE is the second method explored in this thesis. Both Bayesian GPLVM and VAE are non-linear and probabilistic.

2.2.1 Gaussian Process Latent Variable Model

A latent variable model is a statistical model that describes the relationship between observed variables and latent (hidden) variables. In the context of dimensionality reduction, the observed variables are the variables in the high dimensional space and the latent variables are the variables in the low dimensional space. Considering the example of the Fashion MNIST dataset, the observed variables are the pixels in the image. For the same example, although the latent variables do not carry any inherent meaning themselves, they encode information about the item of clothing in the image.

The Gaussian Process Latent Variable Model (GPLVM) uses a Gaussian process to define a mapping between the latent and data spaces [6]. While the GPLVM is a non-linear model, it is helpful to briefly discuss a linear latent variable model before delving into the details of the GPLVM.

Consider the following case: there are N data points in the high dimensional space (called the data space). The dimension of the data space is D and the dimension of the latent space that you want to create is Q. The data points are stacked into an $N \times D$ matrix $\mathbf{Y}$. $\mathbf{Y}$ is centered by subtracting the mean of the N rows from each row. The aim is to find a matrix $\mathbf{X}$, of size $N \times Q$, which contains the latent space points corresponding to the data space points in $\mathbf{Y}$.

The linear latent variable model assumes that a data space point and its corresponding latent space point are related by the following equation:

$$\mathbf{y} = \mathbf{W}\mathbf{x} + \boldsymbol{e} \tag{13}$$

where $\mathbf{e}$ indicates noise and $\mathbf{W}$ is the transformation matrix that relates the data space to the latent space [13].

A common linear latent variable model is called the Probabilistic Principal Component Analysis (PPCA). In PPCA, $\mathbf{e}$ is considered to be Gaussian with zero mean and covariance $\beta^{-1}\mathbf{I}$ (β being a constant and $\mathbf{I}$ the identity matrix). The likelihood for a particular data point $\mathbf{y}_n$ can thus be written as:

$$p(\mathbf{y}_n|\mathbf{W},\beta) = \int p(\mathbf{y}_n|\mathbf{x}_n,\mathbf{W},\beta)p(\mathbf{x}_n)d\mathbf{x}_n \tag{14}$$

where $p(\mathbf{x}_n) = \mathcal{N}(\mathbf{x}_n|\mathbf{0},\mathbf{I})$, and $p(\mathbf{y}_n|\mathbf{x}_n,\mathbf{W},\beta) = \mathcal{N}(\mathbf{y}_n|\mathbf{W}\mathbf{x}_n,\beta^{-1}\mathbf{I})$ [14].

β and $\mathbf{W}$ can be found by making the assumption that data points are independent and identically distributed and maximizing the whole dataset's likelihood, which is given by:

$$p(\mathbf{Y}|\mathbf{W},\beta) = \prod_{n=1}^{N} p(\mathbf{y}_n|\mathbf{W},\beta). \tag{15}$$

An alternative to this approach of marginalizing (or integrating out) $\mathbf{x}_n$ is marginalizing $\mathbf{W}$ instead. In this alternative approach, called dual PPCA, the prior over $\mathbf{W}$ is given by:

$$p(\mathbf{W}) = \prod_{i=1}^{D} \mathcal{N}(\mathbf{w}_i|\mathbf{0},\alpha^{-1}\mathbf{I}) \tag{16}$$

where $\mathbf{w}_i$ is the i^{th} row of the matrix $\mathbf{W}$ and α is a constant [14].

The corresponding likelihood for the whole dataset is given by:

$$p(\mathbf{Y}|\mathbf{X},\beta) = \frac{1}{(2\pi)^{\frac{DN}{2}}|\mathbf{K}|^{\frac{D}{2}}} \exp\left(-\frac{1}{2}tr(\mathbf{K}^{-1}\mathbf{Y}\mathbf{Y}^T)\right) \tag{17}$$

where $\mathbf{K} = \alpha\mathbf{X}\mathbf{X}^T + \beta^{-1}\mathbf{I}$.

After taking the log of this likelihood, and equating the gradient of the log likelihood (with respect to $\mathbf{X}$) to 0, we see that the likelihood is maximized when:

$$\mathbf{X} = \mathbf{U}_Q\mathbf{L}\mathbf{V}^T \tag{18}$$

where $\mathbf{U}_Q$ is an $N \times Q$ matrix that has the eigenvectors of $\mathbf{Y}\mathbf{Y}^T$ as its columns, and $\mathbf{L}$ is a $Q \times Q$ diagonal matrix whose j^{th} element is $l_j = (\frac{\lambda_j}{\alpha D} - \frac{1}{\beta\alpha})^{\frac{1}{2}}$, where λ_i is the j^{th} eigenvalue of $\mathbf{Y}\mathbf{Y}^T$. $\mathbf{V}$ is an arbitrary $Q \times Q$ orthogonal matrix [14].

Similarly, β is given by:

$$\frac{1}{D-Q}\sum_{j=Q+1}^{D} \lambda_j \quad [15]. \tag{19}$$

We can see that maximizing the likelihood directly yields the latent space points corresponding to the data space points. The key realization here, which was used in [14] to create the GPLVM, is that the likelihood in Equation 17 is the product of D independent Gaussian processes, each of which has $\mathbf{K}$ as the covariance function. While $\mathbf{K}$ is linear, it can easily be replaced with a non-linear covariance function to yield a non-linear dimensionality reduction method (which is consequently called the GPLVM).

The covariance function used in [14] was the Radial Basis Function (RBF) kernel, discussed in Section 2.1. The equation of this kernel, after accounting for noise, is:

$$k_{n,m} = \sigma_f^2 \exp\left(-\frac{\gamma}{2}(\mathbf{x}_n - \mathbf{x}_m)^T(\mathbf{x}_n - \mathbf{x}_m)\right) + \delta_{nm}\,\beta^{-1} \tag{20}$$

where $k_{n,m}$ is the element in the n^{th} row and m^{th} column of $\mathbf{K}$, σ_f^2 controls the overall variance of the function, γ is the length-scale parameter and δ_{nm} is the Kronecker delta [14].

Having a non-linear covariance function means that a closed form solution for $\mathbf{X}$ no longer exists. Therefore, gradient descent, with the likelihood as the objective function, is used to find it. Since the objective function has multiple local optima, the solution is not unique. The parameters for the covariance function, $\boldsymbol{\theta}$, can be found using the same method [13] . (When the RBF kernel is used, $\boldsymbol{\theta}$ refers to σ_f^2 and γ.)

In practice, the abovementioned Maximum Likelihood Estimation (MLE) is not used. Rather, a prior is specified over each of $\mathbf{X}$, σ_f^2, and γ, and Maximum A Posteriori (MAP) estimates are found for each of them. The only difference between MLE and MAP is the inclusion of the prior in MAP; the likelihood is, in effect, weighted by the prior [14].

$\boldsymbol{\theta}$ is used to find the predictive distribution mentioned in Section 2.1.1, whose parameters are given in Equations 6 and 7. The predictive distribution gives the data space representation for a test latent space point.

A simple inference model can be used to get the latent space representation of a test data space point. This model is made from the covariance function, likelihood and posterior distribution of the GPLVM and applies gradient descent based optimization to return a point estimate of the latent space point corresponding to the given data space point [16].

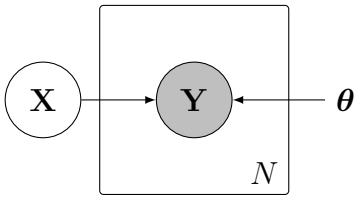


Figure 5: Graphical model of the GPLVM. Arrows indicate dependency. Shaded and unshaded circles represent observed and latent variables, respectively.

2.2.2 Bayesian GPLVM

The GPLVM has the limitation that only point estimates of $\mathbf{X}$ are returned. It is preferred to get a distribution over $\mathbf{X}$ instead. The Bayesian GPLVM [11] allows us to do this by finding the distribution $p(\mathbf{X}|\mathbf{Y})$, called the posterior. Since finding the parameters of the true posterior is intractable, the parameters of an approximation are found using gradient descent based optimization.

$q(\mathbf{X})$ is given by:

$$q(\mathbf{X}) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_n, \mathbf{S}_n) \tag{21}$$

where N is the number of training data points, and $\boldsymbol{\mu}_n$ and $\mathbf{S}_n$ are the parameters that need to be found. In the following description, $\mathbf{S}_n$ is considered to be a diagonal covariance matrix for simplicity [11].

To find the parameters of $\boldsymbol{\mu}_n$ and $\mathbf{S}_n$, we need to optimize the marginal likelihood. However, the equation used for marginal likelihood is different from the one used for the GPLVM; the difference being that $\mathbf{X}$ is integrated out as well. The equation used is:

$$p(\mathbf{Y}) = \int p(\mathbf{Y}|\mathbf{X})p(\mathbf{X})d\mathbf{X} \tag{22}$$

where $p(\mathbf{X})$, the prior for $\mathbf{X}$, is given by:

$$p(\mathbf{X}) = \prod_{n=1}^{N} \mathcal{N}(\mathbf{x}_n|\mathbf{0}, \mathbf{I}). \tag{23}$$

$\log p(\mathbf{Y})$ is intractable, so an approximation is maximized instead [11]. This approx-

imation is the Jensen's lower bound, given by:

$$F(q) = \widetilde{F}(q) - KL(q||p) \tag{24}$$

where:

$$KL(q||p) = \int q(\mathbf{X}) \log \frac{q(\mathbf{X})}{p(\mathbf{X})} \tag{25}$$

$$\widetilde{F}(q) = \sum_{d=1}^{D} \int q(\mathbf{X}) \log p(\mathbf{y}_d|\mathbf{X}) d\mathbf{X} = \sum_{d=1}^{D} \widetilde{F}_d(q) \tag{26}$$

The KL divergence can be found analytically as both $q(\mathbf{X})$ and $p(\mathbf{X})$ are Gaussian distributions. However, $\widetilde{F}(q)$ cannot be found like this as the integral $\widetilde{F}_d(q)$ is intractable. A closed-form lower bound is found for $\widetilde{F}_d(q)$ instead. (D is the total number of dimensions of the data space and d refers to each dimension.)

Details of this lower bound are not mentioned here as its derivation is quite involved and would require introducing concepts like inducing inputs and Ψ statistics, which are not directly relevant to this thesis. [11] includes these details.

An important thing to note is that the parameters of $q(\mathbf{X})$, $\boldsymbol{\mu}_n$ and $\mathbf{S}_n$, appear in the lower bound for $\widetilde{F}_d(q)$. This bound is substituted into Equation 26 and $F(q)$ in Equation 24 is found. The values of $\boldsymbol{\mu}_n$ and $\mathbf{S}_n$ are then found by using gradient descent based optimization techniques to maximize $F(q)$ with respect to $\boldsymbol{\mu}_n$ and $\mathbf{S}_n$. The parameters of the covariance function, $\boldsymbol{\theta}$, can also be found in a similar manner. $\boldsymbol{\theta}$ can then be used to get the data space representation of a test latent space point, in exactly the same manner as the GPLVM.

The latent space representation of a test data space point is also obtained in the same manner as for the GPLVM: by using an inference model. However, a difference is that the inference model returns the variational posterior of the latent space point instead of a point estimate. In the following sections of this thesis, we refer to this inference model as Bayesian GPLVM inference model.

2.2.3 Variational Autoencoder

A Variational Autoencoder (VAE) [12] is another dimensionality reduction method. It has two main components: an encoder and a decoder. The encoder is a neural network that finds the latent space representation given a data point. This representation is a multivariate Gaussian probability distribution with a particular mean

vector and covariance matrix. The mean of this distribution is considered to be the latent space point. The decoder, on the other hand, is a neural network that generates the data space representation given a particular latent space representation. In the math shown below, we denote the encoder as $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ and the decoder as $p_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z})$, where $\mathbf{x}$ is a point in the data space, $\mathbf{z}$ is the corresponding point in the latent space, $\boldsymbol{\theta}$ represents the parameters of the encoder and $\boldsymbol{\phi}$ represents the parameters of the decoder [17].

The key purpose of the VAE is not only to find a good latent space representation of a given training data point but to do so while allowing new data points that are similar to the training data points to be generated. The former aim is achieved by minimizing reconstruction loss, which is an indication of the closeness between a particular training data point and the data point generated from its latent space representation. The latter aim means that the multivariate Gaussian distribution found for each training data point should be sufficiently wide, allowing a range of plausible samples to be generated from it. The wider the distribution, the closer it is to the unit Gaussian, which has a zero vector as the mean and an identity matrix as the covariance.

Therefore, the objective function optimized when training the VAE consists of two terms: the reconstruction loss term and the regularization term. The reconstruction loss term is the negative expected value of $\log p_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z})$, found with respect to $\mathbf{z}$, where $\mathbf{z}$ comes from the encoder. The regularization term yields a value representing the diversity of plausible samples that can be generated from the Gaussian distribution corresponding to a particular training data point. More specifically, it is the Kullback-Leibler divergence, which measures the closeness between $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ and the unit Gaussian [17].

For a single training data point x_i, the objective function is therefore given by:

$$l_i(\boldsymbol{\theta}, \boldsymbol{\phi}) = -E_{\mathbf{z}\sim q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}_i)}[\, \log p_{\boldsymbol{\phi}}(\mathbf{x}_i|\mathbf{z})\,] + KL(\, q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}_i)\,||\,p(\mathbf{z})\,). \tag{27}$$

The objective function actually used in training, covering all N training data points, is given by:

$$\sum_{i=1}^{N} l_i(\boldsymbol{\theta}, \boldsymbol{\phi}) \tag{28}$$

Gradient descent is used to optimize this objective function and obtain the trained VAE. The trained model is used to get the latent representation of a test data point. This is done by feeding the data point to the encoder, and getting the parameters of

the Gaussian distribution corresponding to that point. The data space representation corresponding to a latent space point is obtained by simply feeding that point to the decoder [17].

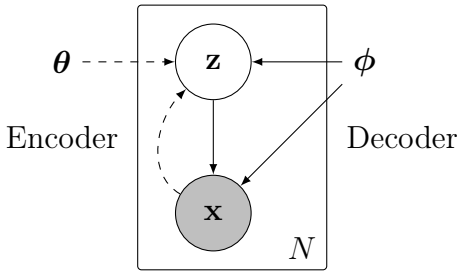


Figure 6: Graphical model of the VAE. Arrows indicate dependency. Shaded and unshaded circles represent observed and latent variables, respectively.

2.3 Bayesian Optimization

Optimization involves finding the minimum or maximum of a function within a specified domain. Considering the minimization case, this can be stated as:

$$x_M = \arg\min_{x \in \mathcal{X}} f(x) \tag{29}$$

where the domain is $\mathcal{X}$ [1].

Bayesian optimization is an optimization technique that treats the function being optimized (i.e., the objective function) as a black box. It works well even if the objective function is expensive to evaluate, stochastic (has an additive noise term), non-convex[3], or discontinuous.

The key idea of Bayesian optimization is to replace the original objective function with a probabilistic model. This model is used to build an acquisition function that is used to query new points in the domain when searching for the global optimum of the objective function. The typical Bayesian optimization algorithm has the following steps [1]:

1. Create a set D that consists of a few $(x, f(x))$ pairs, where x is randomly selected from the domain.[4]

[3]In convex functions, any local minimum is a global minimum.
[4]There are other alternatives to random selection as well.

2. Initialize a probabilistic regression model using D.

3. For some number of iterations, repeat:

    - Select a new location x in the domain by optimizing an acquisition function (which uses the regression model in place of the actual, computationally expensive objective function).
    - Evaluate the objective function at the new location and append the resulting $(x, f(x))$ pair to D.
    - Use D to update the regression model.

The probabilistic regression model mentioned above is a statistical model that takes as input a set of independent variable values (in our case x) and makes a prediction about the value of a dependent variable (in our case $f(x)$). The model also indicates how certain it is about the prediction. There are several types of probabilistic regression models; examples include random forests, tree Parzen estimators and Gaussian processes [1]. In this thesis, we focus on Gaussian processes.

Since a Gaussian process is used as a surrogate for the objective function f, we can say that:

$$f \sim GP(\mu, K) \tag{30}$$

Then, for a given x, the prediction for $f(x)$ is given by the predictive distribution described in Section 2.1.1.

The acquisition function mentioned in the Bayesian optimization algorithm above defines the balance between exploring new areas in the domain, and exploiting areas that have already been explored and are known to have small objective function values (or, in the case of maximization, large objective function values). Selecting a location in an already explored area means that you are guaranteed to get a good result, but at the same time, means that you are ignoring an area which could yield an even better result. Conversely, selecting a location in a new area comes with the caveat that it could yield a poor result, effectively wasting the computational resources used up in evaluating the objective function at that location.

Optimizing the acquisition function yields the location that gives the ideal trade-off between exploration and exploitation. There are several acquisition functions. Maximum Probability of Improvement (MPI), Upper Confidence Bound (UCB) and

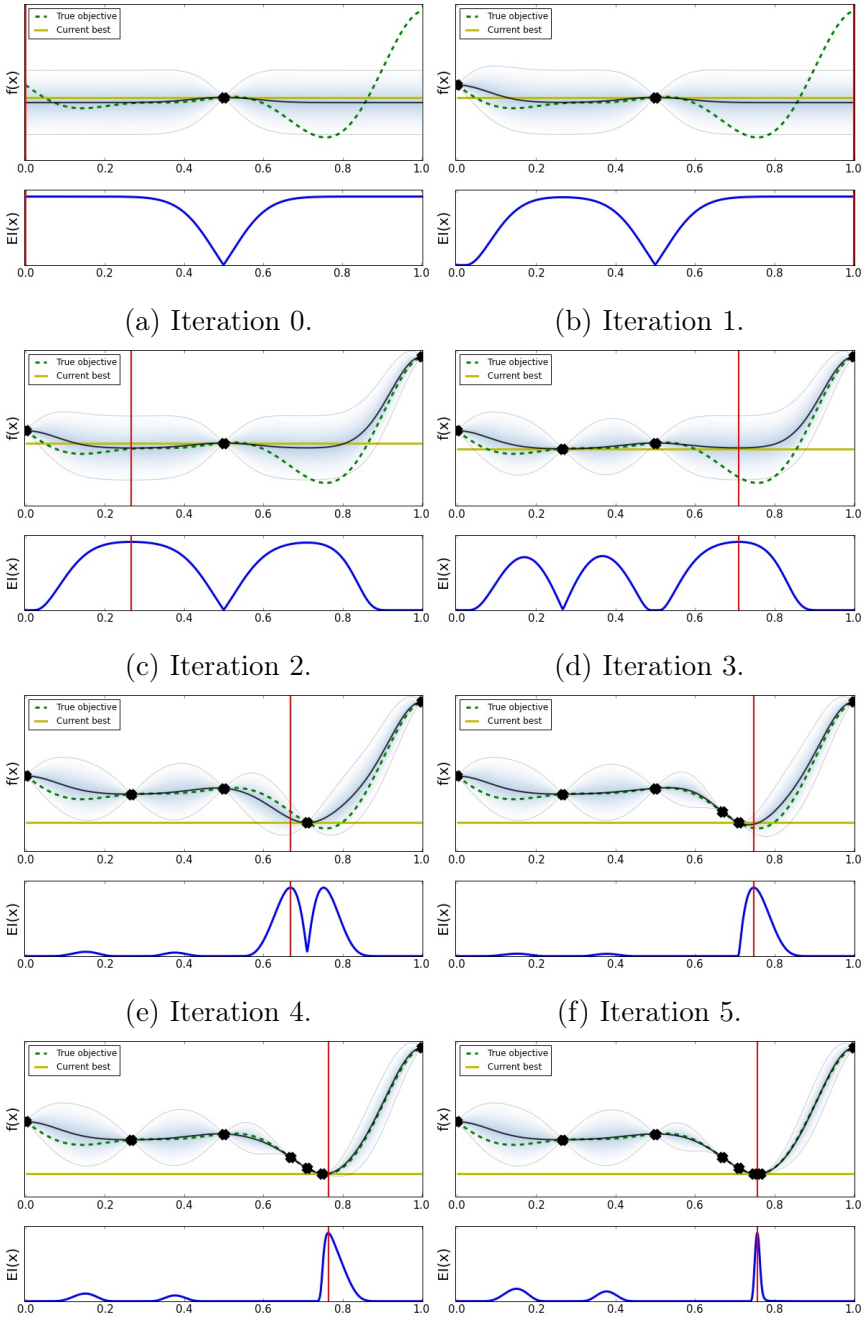
Expected Improvement (EI) are some of the most common ones. We use EI in this thesis.

The EI acquisition function defines, at a given location x, the expected improvement over the previous best objective function value (denoted f_{best}). optimizing the EI function means finding the x that maximizes the integral:

$$\int_{f_{best}}^{\infty} (y - f_{best}) p(y|\mathbf{x}, D) dy \tag{31}$$

where $y = f(\mathbf{x})$ [1].

Figure 7 shows an example of Bayesian optimization on the 1D Forrester function.



(a) Iteration 0. (b) Iteration 1.

(c) Iteration 2. (d) Iteration 3.

(e) Iteration 4. (f) Iteration 5.

(g) Iteration 6. (h) Iteration 7.

Figure 7: Bayesian optimization to find the minimum of the 1D Forrester function. In each subfigure, the top plot represents the objective function. The dotted green line represents the true objective function, the light green line represents the best (minimum) value of the objective function found so far, the black dots represent the locations at which the objective function has been evaluated, and the shaded region represents the 95% confidence region corresponding to each x value. The bottom plot in each subfigure represents the acquisition function - the objective function is evaluated at locations where the acquisition function is maximum. It can be seen from subfigure (h) that the minimum value of the objective function has been found.

3 Latent Variable Bayesian Optimization

As discussed in Section 2.3, a global optimization problem has the form:

$$\mathbf{x}_M = \arg\min_{\mathbf{x}\in\mathcal{X}} f(\mathbf{x}) \tag{32}$$

where $f(\mathbf{x})$ is the objective function and $\mathcal{X}$ is the domain of $f(\mathbf{x})$. $\mathcal{X}$ can be considered to be the space in which $\mathbf{x}$ lies, or equivalently, the set of all possible $\mathbf{x}$ values.

In this thesis, we apply the Bayesian optimization method on this problem for the case when $\mathcal{X}$ has a large number of dimensions. Bayesian optimization generally performs poorly when $\mathcal{X}$ has more than 10 dimensions and the aim of this thesis is to present a solution to this problem. The reason for the poor performance is that finding the global minimum requires a good coverage of $\mathcal{X}$ but as the dimensions of $\mathcal{X}$ increase, the number of evaluations required for good coverage grows exponentially [2].

The key idea in our approach, which we call "Latent Variable Bayesian Optimization (LVBO)", is to apply dimensionality reduction methods on some training data to learn a latent space and then use this latent space as the domain in the optimization. For LVBO to work well, the latent space must capture the geometry of the high dimensional "data" space; the dimensionality reduction method must ensure that the ordering of the points in the high dimensional space is represented in the latent space.

Denoting the dimensionality reduction method as the function g and the latent space representation of a particular data space point as $\mathbf{w}$, we can say that $\mathbf{w} = g(\mathbf{x})$. Then, the optimization problem becomes:

$$\mathbf{w}_M = \arg\min_{\mathbf{w}\in\mathcal{W}} f(g^{-1}(\mathbf{w})) \tag{33}$$

where $\mathcal{W}$ is the latent space and $g^{-1}(\mathbf{w})$ is the data space representation generated from the latent space representation $\mathbf{w}$. Once $\mathbf{w}_M$ has been found by solving the optimization problem in Equation 33, $\mathbf{x}_M$ is simply given by $g^{-1}(\mathbf{w}_M)$.

3.1 Specific Use Case

To investigate LVBO, we focus on one specific use case throughout this thesis: given an image of an item of clothing, find the image that matches it closest. Intuitively, this might seem odd; why find the closest matching image by using optimization when we can simply compare each option and see which one is the closest match for the given image. The confusion can be cleared by considering the following instance: we have two given images and need to find a hybrid that has some features from one image and some from the other. This hybrid needs to match both images closely, but none of the options available does so. LVBO, being a generative method as described below, can easily be employed to find the best hybrid.

Finding the closest matching image for a given image can be framed as the following optimization problem in the "image" space[5]:

$$\mathbf{x}_M = \arg\min_{\mathbf{x}\in\mathcal{X}} f_{\mathbf{i}}(\mathbf{x}) \tag{34}$$

where $\mathbf{x}$ is an image within the space $\mathcal{X}$, $\mathbf{i}$ is the given image, and $f_{\mathbf{i}}(\mathbf{x})$, the objective function, is a measure of how closely $\mathbf{x}$ matches $\mathbf{i}$.

$\mathcal{X}$, the space in which each image lives, has 784 dimensions. This is because each image in Fashion MNIST, the dataset we use in this thesis, is of size 28x28 pixels. Also, since each image is grayscale and has integer pixel values ranging from 0 to 255, $\mathcal{X}$ is a discontinuous space.[6]

$f_{\mathbf{i}}(\mathbf{x})$ can be chosen to denote various types of distance between images $\mathbf{x}$ and $\mathbf{i}$. These include Hausdorff, Tangent and Euclidean distances. We considered their relative advantages and disadvantages and chose Euclidean distance as it is the simplest and fastest. The Euclidean distance between $\mathbf{x}$ and $\mathbf{i}$ is given by:

$$e(\mathbf{x}, \mathbf{i}) = \sqrt{\sum_{j=1}^{K} (\mathbf{x}_j - \mathbf{i}_j)^2} \tag{35}$$

where $\mathbf{x}_j$ and $\mathbf{i}_j$ denote pixel values at corresponding locations in $\mathbf{x}$ and $\mathbf{i}$ and K is the number of pixels (784 in our case).

[5]The "image" space is also referred to as the "data" space.

[6]Fashion MNIST actually has pixel values between 0 and 1 as each integer value has been divided by 255. This does not change the fact that $\mathcal{X}$ is discontinuous.

Applying LVBO to this use case, the optimization problem becomes:

$$\mathbf{w}_M = \arg\min_{\mathbf{w}\in\mathcal{W}} f_{\mathbf{i}}(g^{-1}(\mathbf{w})) \tag{36}$$

where $\mathbf{x}_M$ can be reconstructed by $g^{-1}(\mathbf{w}_M)$.

$g^{-1}(\mathbf{w})$ allows us to generate new images that look like the given image $\mathbf{i}$. This is why LVBO is a generative method. Both $g(\mathbf{x})$ and $g^{-1}(\mathbf{w})$ are unknown and have to be learned from data.

3.2 LVBO Metamodels

LVBO relies on two metamodels:

1. A dimensionality reduction model g.
2. A surrogate model[7] to carry out Bayesian optimization in the latent space learnt by 1.

Figure 8 visually indicates how the metamodels fit in the overall context of LVBO. Sections 3.3 and 3.4 discuss each of the metamodels in detail.

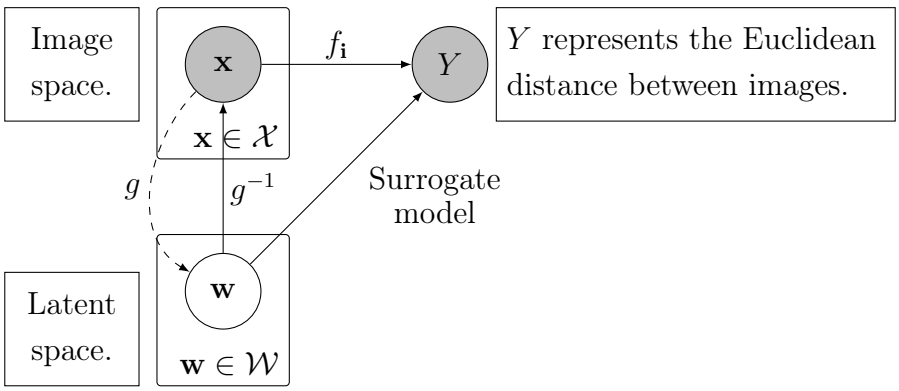


Figure 8: Graphical model for LVBO. Arrows indicate dependency. Shaded and unshaded circles represent observed and latent variables, respectively.

[7]Surrogate model is referred to as "regression model" in Section 2.3.

3.3 Dimensionality Reduction Model

In this thesis, two dimensionality reduction models were explored for use with LVBO: the Bayesian GPLVM and the Variational Autoencoder.

When the Bayesian GPLVM is used as the dimensionality reduction model, g refers to the Bayesian GPLVM inference model used to get the latent space representation of a test data space point (as mentioned towards the end of Section 2.2.2) while g^{-1} refers to the trained Bayesian GPLVM itself. When the Variational Autoencoder is used, g refers to the encoder and g^{-1} to the decoder.

The performance of Bayesian GPLVM and the Variational Autoencoder was compared on the Fashion MNIST dataset. The Bayesian GPLVM was then selected for use with LVBO. The method employed to compare the two models and the reason Bayesian GPLVM was selected are detailed in Section 4.1.

Every point in the latent space learnt by the Bayesian GPLVM has an uncertainty associated with it. In fact, as discussed in Section 2.2.2, the latent space does not consist of point estimates but consists of variational posteriors. When we refer to "points" in the latent space, we are actually referring to the means of these latent space variational posteriors.

3.4 Surrogate Model

For the surrogate model, we choose between the standard Gaussian process (with fixed inputs) and a Bayesian GPLVM in which $p(\mathbf{X})$ is fixed to the distribution of the latent space learnt by the dimensionality reduction model in Section 3.3.

Both the Gaussian process and the Bayesian GPLVM use a kernel that in this thesis we fix to be the ARD-RBF kernel. This is given by:

$$k(\mathbf{w}_p, \mathbf{w}_q) = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{w}_p - \mathbf{w}_q)^T M(\mathbf{w}_p - \mathbf{w}_q)\right). \tag{37}$$

When M is a diagonal matrix, $(\mathbf{w}_p - \mathbf{w}_q)^T M(\mathbf{w}_p - \mathbf{w}_q)$ is the square of the Euclidean distance between $\mathbf{w}_p$ and $\mathbf{w}_q$, calculated after each dimension has been scaled by its characteristic length-scale.

Note that we are using $\mathbf{w}$ in Equation 37, instead of $\mathbf{x}$. This is because we are referring to the latent space points. As the latent space learnt by the Bayesian GPLVM consists of variational posteriors, $\mathbf{w}_p$ and $\mathbf{w}_q$ are distributions, not point estimates. When we find the Euclidean distance between $\mathbf{w}_p$ and $\mathbf{w}_q$, we are actually finding the distance between their means and ignoring their covariance matrices.

Disregarding the length-scale parameters for notational simplicity, Equation 37 can be written as:

$$k(\mathbf{w}_p, \mathbf{w}_q) = \sigma_f^2 \exp\left(-\frac{1}{2} d(\mathbf{w}_p, \mathbf{w}_q)^2\right) \tag{38}$$

where $d(\mathbf{w}_p, \mathbf{w}_q)$ represents the Euclidean distance measure.

To incorporate the uncertainty of the latent space into the search for the closest match, $d(\mathbf{w}_p, \mathbf{w}_q)$ can be made to denote distance measures that do not ignore the covariance matrices of $\mathbf{w}_p$ and $\mathbf{w}_q$. We study two such measures: the Hellinger distance and the Geodesic distance.

3.4.1 Hellinger Distance

The Hellinger distance is a particular case of the so-called f-divergence. The f-divergence measures the similarity between two probability distributions and is given by:

$$D_f(P||Q) := \mathbb{E}_Q\left[\frac{dP(x)}{dQ(x)}\right] \tag{39}$$

where P and Q are probability distributions and $x \in \mathcal{X}$ [18].

As the variational posteriors $\mathbf{w}_p$ and $\mathbf{w}_q$ are multivariate Gaussians, we are only concerned with the Hellinger distance between two multivariate Gaussians [19]. Denoting these distributions by $\mathbf{w}_p \sim \mathcal{N}(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_q)$ and $\mathbf{w}_q \sim \mathcal{N}(\boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)$, the Hellinger distance between them is given by:

$$d(\mathbf{w}_p, \mathbf{w}_q) = \sqrt{1 - \frac{\det(\boldsymbol{\Sigma}_p)^{\frac{1}{4}} \det(\boldsymbol{\Sigma}_q)^{\frac{1}{4}}}{\det(\frac{\boldsymbol{\Sigma}_p + \boldsymbol{\Sigma}_q}{2})^{\frac{1}{2}}} \exp\left[-\frac{1}{8}(\boldsymbol{\mu}_p - \boldsymbol{\mu}_q)^T \left(\frac{\boldsymbol{\Sigma}_p + \boldsymbol{\Sigma}_q}{2}\right)^{-1} (\boldsymbol{\mu}_p - \boldsymbol{\mu}_q)\right]} \tag{40}$$

3.4.2 Geodesic Distance

The latent space learnt by the Bayesian GPLVM can be viewed as an uncertain manifold embedded into the data space. As an example, consider that the data space has three dimensions and the manifold is a sphere lying within the data space. Since every point on the sphere can be described using two values (coordinates), the manifold is two dimensional. To find the distance between two points on the sphere, we could just find the Euclidean distance between those points' data space representations. However, this would not be correct as the curvature of the sphere would not be accounted for [20].

Drawing a parallel between the example above and the general case, the curvature of the sphere can be considered to be a distortion in the latent space. To obtain the correct distance between two latent space points, the distortion needs to be accounted for. The distance measure that accounts for it is called Geodesic distance, "geodesic" referring to the shortest curve that connects the points [20].

The Geodesic distance is calculated by employing a concept of Riemannian geometry, called the metric tensor. For the Bayesian GPLVM, the metric tensor comes from a probability distribution called the Wishart distribution. Taking the expectation over this distribution, the expected metric tensor is given by:

$$\mathbf{G} = \boldsymbol{\mu}^T \boldsymbol{\mu} + D\boldsymbol{\Sigma} \tag{41}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the mean and covariance of $\frac{\delta p(\mathbf{Y}|\mathbf{X})}{\delta \mathbf{X}}$, and D is the number of dimensions of the data space [21].

$\mathbf{G}$ is used to solve the second order differential equation:

$$\gamma'' = -\frac{1}{2}\mathbf{G}^{-1}\left[\frac{\delta vec\mathbf{G}}{\delta \gamma}\right](\gamma' \otimes \gamma') \tag{42}$$

where γ refers to the equation of Geodesic curve, $vec\mathbf{G}$ stacks the columns of $\mathbf{G}$ and $\otimes$ refers to the Kronecker product. γ is then integrated to find the Geodesic distance. An important thing to note is that the geodesic curve avoids regions of high uncertainty in the latent space [21].

The above process is quite expensive computationally, so a heuristic approach was applied in this thesis to get an approximate Geodesic distance. This approach was taken from [20] and yields the following formula for the Geodesic distance between

$\mathbf{w}_p$ and $\mathbf{w}_q$:

$$d(\mathbf{w}_p, \mathbf{w}_q) = \sqrt{(\boldsymbol{\mu}_p - \boldsymbol{\mu}_q)^T \frac{\mathbf{G}_p + \mathbf{G}_q}{2} (\boldsymbol{\mu}_p - \boldsymbol{\mu}_q)} \tag{43}$$

where $\mathbf{G}_p$ and $\mathbf{G}_q$ are the expected metric tensors calculated at $\mathbf{w}_p$ and $\mathbf{w}_q$ respectively. $\mathbf{w}_p \sim \mathcal{N}(\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_q)$ and $\mathbf{w}_q \sim \mathcal{N}(\boldsymbol{\mu}_q, \boldsymbol{\Sigma}_q)$, as before.[8]

3.5 LVBO Algorithm

The full LVBO algorithm is presented below:

1. Initialize s points in latent space and store in matrix called $\mathbf{W}$.[9]
2. Store the diagonals of the covariance matrices corresponding to the points in $\mathbf{W}$ in a matrix called $\mathbf{W_{var}}$.[10]
3. Store the objective function values corresponding to $\mathbf{W}$ in a matrix called $\mathbf{Y}$.
4. Initialize kernel to be used in surrogate model and specify which distance type (Euclidean, Hellinger or Geodesic) is to be used.
5. Initialize surrogate model to either a Gaussian process or a Bayesian GPLVM.
6. For k iterations, repeat:
   (a) Update kernel with $\mathbf{W}$, $\mathbf{W_{var}}$ and $\mathbf{Y}$.
   (b) Update surrogate model with $\mathbf{W}$ and $\mathbf{Y}$ if it is a Gaussian process or with $\mathbf{W}$, $\mathbf{W_{var}}$ and $\mathbf{Y}$ if it is a Bayesian GPLVM.
   (c) Optimize the surrogate model.
   (d) Give optimized surrogate model as input to acquisition function.
   (e) Optimize acquisition function and store the latent space point that is returned in a vector called $\mathbf{w_{next}}$.

[8] Note that $\boldsymbol{\Sigma}_p$ and $\boldsymbol{\Sigma}_q$ are used in the calculation of $\mathbf{G}_p$ and $\mathbf{G}_q$.

[9] There are two ways of initialization: using a random number generator for each latent space dimension, or randomly selecting latent space representations of the data points used to train the Bayesian GPLVM dimensionality reduction model. Both approaches are compared in Section 4.6.

[10] The covariance matrices are diagonal as the latent space dimensions are independent from each other.

(f) Find the diagonal of the covariance matrix corresponding to $\mathbf{w_{next}}$ and store it in a vector called $\mathbf{w_{nextvar}}$.[11]

(g) Update $\mathbf{W}$ with $\mathbf{w_{next}}$.

(h) Update $\mathbf{W_{var}}$ with the $\mathbf{w_{nextvar}}$.

(i) Evaluate the objective function at $\mathbf{w_{next}}$ and update $\mathbf{Y}$ with the result.

The objective function $f_{\mathbf{i}}(g^{-1}(\mathbf{w}))$ that is used with the above LVBO algorithm is:

1. Take as input a latent space point $\mathbf{w}$.
2. Use the trained Bayesian GPLVM to get the data space representation $g^{-1}(\mathbf{w})$.
3. Return as output the Euclidean distance between $\mathbf{i}$ (the given image) and $g^{-1}(\mathbf{w})$.

[11] It is not possible to get the covariance matrix corresponding to $\mathbf{w_{next}}$ directly. To get around this problem, we map $\mathbf{w_{next}}$ to the data space using the Bayesian GPLVM dimensionality reduction model and then map the data space representation back to the latent space using the Bayesian GPLVM inference model. The latter operation returns a mean as well as a covariance matrix.

This back-and-forth mapping is actually done in the objective function as well, to ensure the correctness of LVBO. However, we do not mention it in the objective function algorithm, as it would needlessly complicate the algorithm. The full LVBO algorithm, with the back-and-forth mapping, and the complicated version of the objective function algorithm are given in Appendix C.

4 Experiments

This section describes the experimental setup used to evaluate LVBO and the results that were obtained for different LVBO parameter settings. The code used to run the various experiments can be found online - details are given in Appendix A.

4.1 Selecting a Dimensionality Reduction Model

As explained in Section 3, the first task when implementing LVBO is to train a dimensionality reduction model. We trained two such models, the Variational Autoencoder (VAE) and the Bayesian GPLVM, and selected one for use in the LVBO. The hyperparameters used for training both are presented in Appendix B.

Instead of the full Fashion MNIST dataset, 5000 training images were used to train the models. This was because larger training dataset sizes led to memory errors in GPy, the library used for training the Bayesian GPLVM. The dimensionality of the latent space was selected to be 2 for both models as a 2D space can easily be visualized. Another reason was that a larger number of latent space dimensions would slow down the LVBO process.

Figures 9 and 10 show the latent space learnt by the Bayesian GPLVM and the VAE. The points plotted are the latent space representations of the 5000 training data points (images).

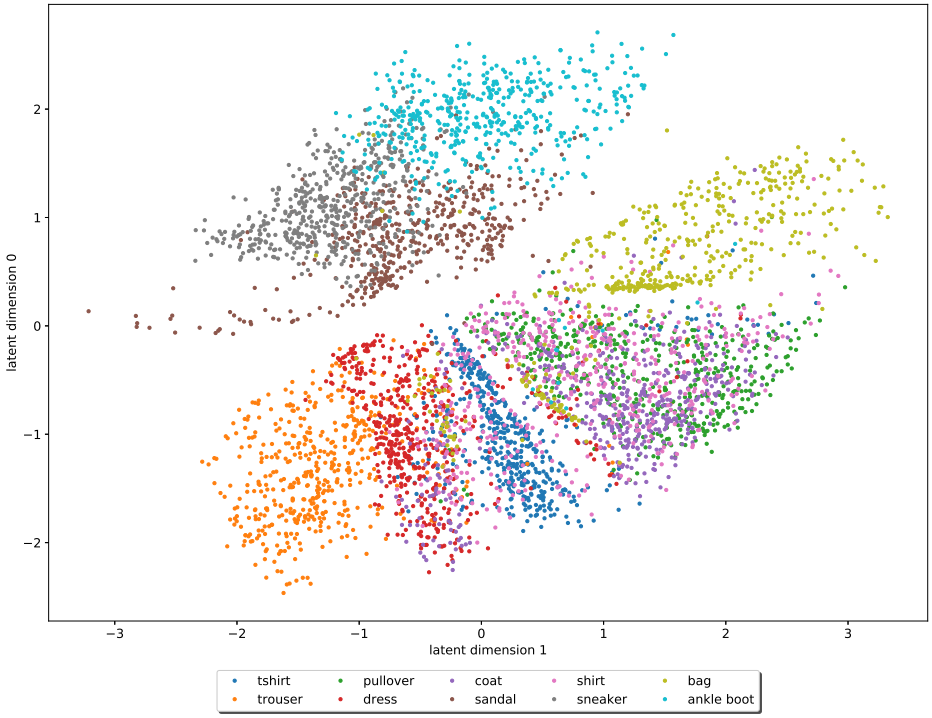


Figure 9: Latent space representations of training data points, as learnt by the VAE.

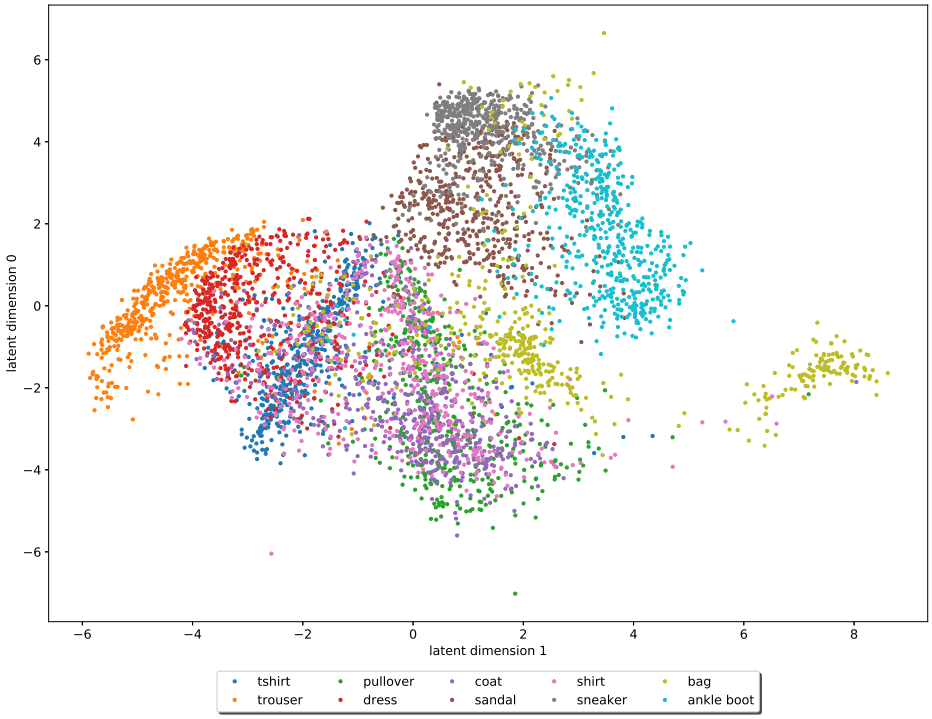


Figure 10: Latent space representations of training data points, as learnt by the Bayesian GPLVM.

500 test images were used to compare the two dimensionality reduction models. First, each test image's latent space representation was found. Then the test image was given the label of the training image that had the nearest latent representation to the test image's own latent representation. For each type of clothing, the proportion of the number of images classified incorrectly to the total number of test images for that type was found. This proportion is given as a Classification Error % in Table 1.

Type of Clothing	**VAE**	**Bayesian GPLVM**
T-shirt	33	44
Pullover	59	63
Coat	60	66
Shirt	75	73
Bag	24	32
Trouser	16	9
Dress	37	38
Sandal	27	41
Sneaker	25	36
Ankle boot	19	14
All items	**38**	**42**

Table 1: Classification Error % for the various clothing items.

It can be seen from Table 1 that the Classification Error %'s for pullovers, coats and shirts are very high compared to the other items of clothing. The reason is that they look quite similar and mostly occupy the same region in the latent space, as can be seen in Figures 9 and 10. Another observation that can be made from Table 1 is that trousers have the least Classification Error %. The reason is that trousers are least similar to the other items of clothing and thus least likely to be confused with them. Figures 9 and 10 corroborate this; the region occupied in the latent space by trousers is largely exclusively occupied by them.

While the overall Classification Error % for the VAE is less than that for the Bayesian GPLVM, we selected the latter as the dimensionality reduction model for LVBO. The Bayesian GPLVM was deemed the logical choice as LVBO relies heavily on Gaussian processes, which are discussed in much more detail in this thesis than the VAE/neural networks.

4.2 Experiment Parameters

Once the Bayesian GPLVM had been selected as the dimensionality reduction model, various LVBO experiments were run. The parameters that were kept constant in all the experiments were:

- Kernel: ARD-RBF
- Acquisition function: Expected Improvement
- Size of the design space[12]: Dimension 1 = (-7, 7) and Dimension 2 = (-7, 7)
- Number of samples to initialize surrogate model: 5

The parameters that were varied, and the different options for each of them, are listed in Table 2.

Parameter	**Options**
Surrogate model to carry out the search	- Gaussian process - Bayesian GPLVM
Number of iterations	- 10 - 25 - 50
Distance type used in surrogate model kernel	- Euclidean - Hellinger - Geodesic
Initialization scheme	- Randomly selected points in latent space - Latent representations of randomly selected training data points

Table 2: Parameters that were varied.

Sections 4.4 - 4.6 discuss each of the parameters mentioned in Table 2 and the results that were obtained by varying them.

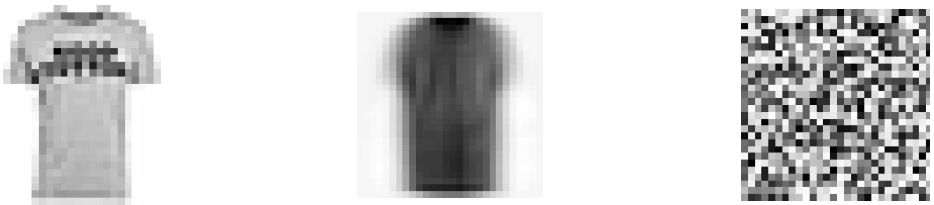
Throughout the rest of this Section, whenever the Bayesian GPLVM is being discussed in the context of the surrogate model, this will be explicitly mentioned. All

[12]The size was selected by visually inspecting the latent space representations learnt by the dimensionality reduction model. A vast majority of these representations, as can be seen in Figure 10, lie within (-7, 7) for both dimensions.

other references to the Bayesian GPLVM or the Bayesian GPLVM inference model must be considered to mean the dimensionality reduction model. This is to prevent one being confused as the other.

4.3 LVBO vs Bayesian Optimization

Before seeing what effect varying the parameters mentioned in Table 2 has on the performance of LVBO, it is necessary to establish that LVBO works well and can give a huge advantage over standard Bayesian Optimization (BO) for our specific use case. This is demonstrated using Figure 11, which shows that for a given image of a t-shirt the closest match generated using LVBO can easily be identified as a t-shirt while the closest match generated using BO is just noise.



(a) Test image (b) LVBO (c) BO

Figure 11: Closest matches generated for test image using LVBO and BO. The parameters used for LVBO are: surrogate model - Gaussian process; number of iterations - 25; distance type used in surrogate model kernel - Euclidean; initialization scheme - latent representations of randomly selected training data points. The parameters used for BO are: number of domain dimensions - 784; number of iterations - 25.

Figure 12 shows the convergence for the images in Figure 11 and reiterates the fact that LVBO massively outperforms BO.

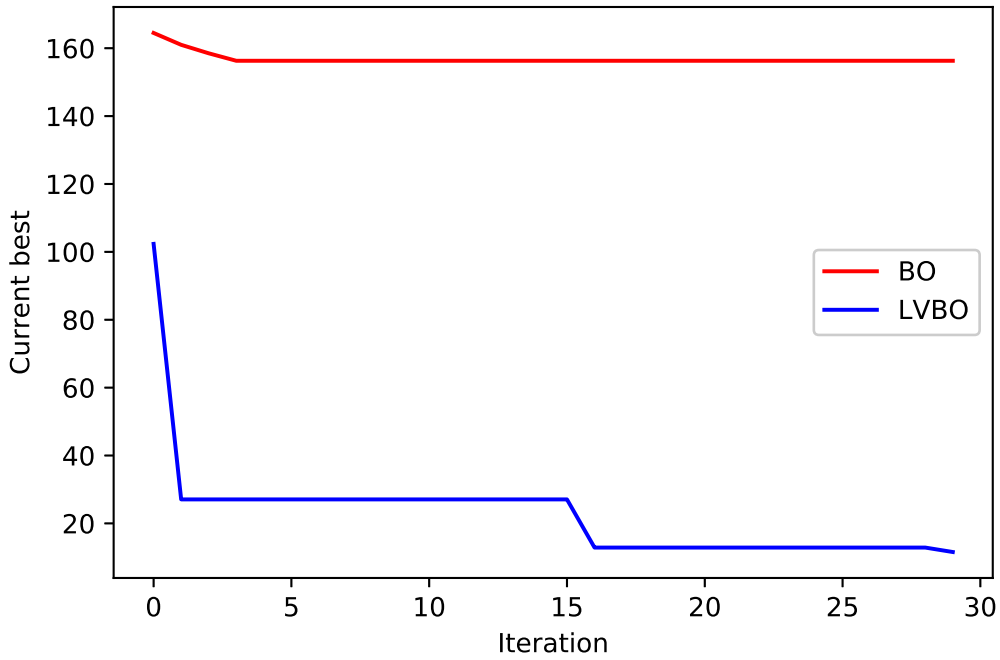
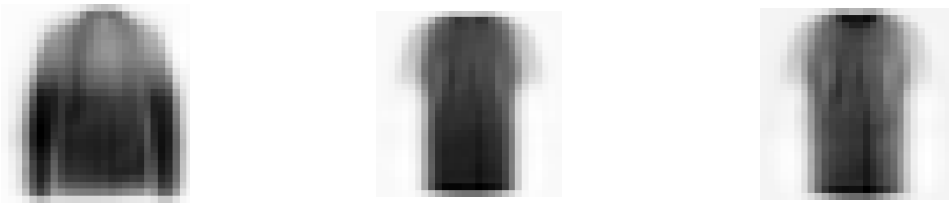


Figure 12: Convergence for the images in Figure 11. The 'Current best' value at each iteration is the Euclidean distance between the test image and the closest match generated for it so far.

Figure 13 shows the images corresponding to LVBO after 0, 10 and 25 iterations. The image at iteration 0 is that of a pullover because LVBO is initialized, as mentioned in the Figure 11 caption, with the latent space representations of randomly selected training data points. In this case, the initial training data point happens to be a pullover. The image at iteration 10, being a t-shirt, is far closer to the test image than the image at iteration 0. The difference between the images at iterations 10 and 25 is less stark, but it can be surmised that the image at iteration 25 is closer to the test image because it has a less dark color.



(a) Iteration 0 (b) Iteration 10 (c) Iteration 25

Figure 13: How the image produced by LVBO evolves as number of iterations increases.

4.4 Effect of Surrogate Model on LVBO

The key metric used to judge how well LVBO performed on a test image was based on the values of:

- A: The Euclidean distance between the test image and the closest match generated for it using LVBO.
- B: The Euclidean distance between the test image and its reconstruction. The reconstructed image is obtained by using the Bayesian GPLVM inference model to find the latent space variational posterior corresponding to the test image, and then using the Bayesian GPLVM to find the data space representation of the mean of that posterior. Details about how this is done are in Section 2.2.2.

It was found experimentally that for some test images A even equalled B (up to two decimal places), suggesting that LVBO worked quite well. For other images however, A came close to B but was still slightly larger. Therefore, the performance of LVBO under different parameter settings was measured by running it on several test images and measuring the percentage of test images for which A came within 10% of B. We call this proportion the LVBO Success %; the higher the LVBO Success %, the better the performance. The number of test images was arbitrarily selected to be 30. A larger number was not feasible because of the large amount of computational time that would be taken up (the slowest parameter configuration, for 30 images, takes 80 hours to run on a 3.3 GHz Intel Core i7 CPU with 16 GB DDR3 RAM).

	Surrogate model	
Distance type	Gaussian process	Bayesian GPLVM
Euclidean	80	10
Hellinger	80	40
Geodesic	30	16.7

Table 3: LVBO Success % for different surrogate models and distances.

Table 3 shows the LVBO Success % for different parameter settings. It can clearly be seen that using the Gaussian process as the surrogate model yields much better results than the Bayesian GPLVM, for each distance type. This suggests that accounting for uncertainty in the latent space through the surrogate model is not

beneficial, contrary to what we might intuitively expect. However, this result might be specific to our example problem of finding closest matches to fashion product images, and not generally the case for LVBO. Therefore, it needs to be investigated further.

Another observation that can be made from Table 3 is that the Hellinger and Geodesic distances yield better results than the Euclidean distance when the Bayesian GPLVM is used as the surrogate model, but do no better than it when the Gaussian process is used. Section 4.5 elaborates on this.

4.5 Effect of Surrogate Model Kernel Distance on LVBO

	Quality of LVBO results		
Distance type	Best	Second-best	Worst
Euclidean	0.8	0.1	0.1
Hellinger	0.1	0.8	0.1
Geodesic	0.1	0.1	0.8

Table 4: Comparison of distance types when surrogate model is Gaussian process.

	Quality of LVBO results		
Distance type	Best	Second-best	Worst
Euclidean	0.2	0.2	0.6
Hellinger	0.5	0.4	0.1
Geodesic	0.3	0.4	0.3

Table 5: Comparison of distance types when surrogate model is Bayesian GPLVM.

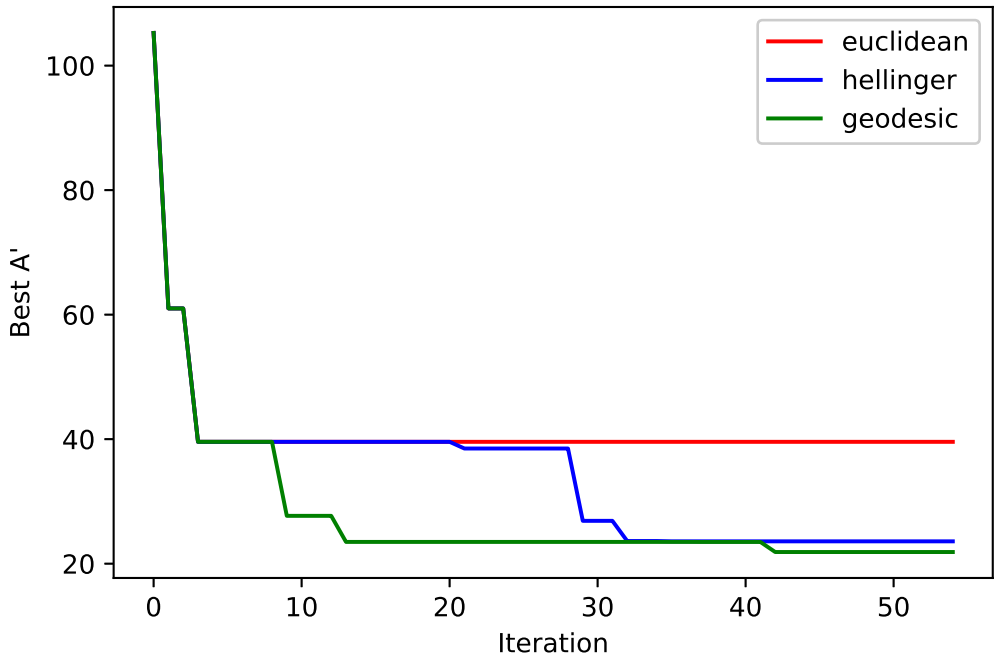


Figure 14: A comparison, on a single test image, of how LVBO converges with different distance types when Bayesian GPLVM is used as the surrogate model. The Best A′ value at each iteration is the Euclidean distance between the test image and the closest match generated for it so far. To isolate the effect of distance type on convergence, the results were averaged over 5 different initializations and the same 5 initializations were used for each distance type.

Table 4 shows how the three distance types compare with each other when the Gaussian process is used as the surrogate model. Table 5 shows the comparison when the Bayesian GPLVM is used as the surrogate model. The comparison was made by carrying out LVBO and finding A, thrice for each of the 30 test images: once with each of the three distance types. The following paragraph explains how the tables were filled by using an example.

Consider the top left index of Table 4. The value 0.8 was obtained in the following manner. First, we found the number of test images for which using the Euclidean distance in the surrogate model kernel yielded an A value better than the A values obtained using the Hellinger and Geodesic distances.[13] Then we divided this number by the total number of test images, 30, to get 0.8. We consider 0.8 to represent the probability that using the Euclidean distance in the latent space would lead to better

[13]As A is the distance between a test image and the closest match generated for it using LVBO, the lower it is, the better.

LVBO results than using the other two distance types when a Gaussian process is used as the surrogate model. The other values in Table 4 and Table 5 were obtained in the same manner.

Table 4 shows that Euclidean is clearly the best performing distance type when the Gaussian process is used as the surrogate model. The second-best distance type is Hellinger while the worst performing is Geodesic.

However, Table 5 shows that Hellinger performs the best when the Bayesian GPLVM is used as the surrogate model, followed by Geodesic and then Euclidean. Figure 14 compares, for a single test image, how LVBO converges with different distances and reiterates that Hellinger and Geodesic distances outperform the Euclidean distance.

These observations suggest that accounting for uncertainty in the latent space through a distance measure is only beneficial when the uncertainty is also being accounted for through the surrogate model. As opposed to the Bayesian GPLVM surrogate model, the Gaussian process does not account for uncertainty in the latent space; therefore, the Hellinger or Geodesic distances give no benefit over Euclidean distance when the Gaussian process is used as the surrogate model.

	Surrogate model	
Distance type	Gaussian process	Bayesian GPLVM
Euclidean	0.35	16.60
Hellinger	0.37	1.94
Geodesic	127	159

Table 6: Time taken by LVBO in minutes for different surrogate models and distances.

Table 6 shows the amount of time taken for different parameter settings when LVBO is run on a single test image. The times shown were obtained by measuring the total time it took for LVBO to run on 30 images and then dividing by 30. The machine used was a 3.3 GHz Intel Core i7 CPU with 16 GB DDR3 RAM.

It can be seen from Table 6 that LVBO is much faster when the Gaussian process is used as the surrogate model. This is because it is much simpler to optimize the Gaussian process than it is to optimize the Bayesian GPLVM; unlike the former, the latter takes into account the covariance matrices of the latent space variational posteriors and involves the computation of Ψ statistics.

For both surrogate models, the slowest results are obtained when Geodesic distance is used. The time taken is more than 2 orders of magnitude more than the times taken for other distance types. This is because unlike the other two, the formula for Geodesic distance involves a significant time sink: the computation of the expected metric tensor.

When the Gaussian process is used as the surrogate model, faster results are obtained with Euclidean distance than with Hellinger distance. Computing Euclidean distance does not involve covariance matrices and is thus faster.

However, when the Bayesian GPLVM is used as the surrogate model, the opposite result is seen: the time taken with Euclidean is about an order of magnitude slower than the time taken with Hellinger. This is because the number of times the distance function is called during LVBO is much larger when Euclidean is used.

Considering all the results mentioned in this subsection together, it can be concluded that:

- When the surrogate model is the Gaussian process, the Euclidean distance should be used.
- When the surrogate model is the Bayesian GPLVM, the Hellinger distance should be used. Not only does it take the least amount of time by far, it also yields the best results.
- Geodesic distance is not worth implementing for either surrogate model because of the comparatively enormous amount of time it takes.

4.6 Effect of No. of Iterations and Initialization on LVBO

In the experiments discussed above, the default number of iterations for which LVBO was run was 25. This number was selected after running LVBO on the 30 test images for 10, 25 and 50 iterations. For all parameter settings discussed in the previous subsections, it was seen that the LVBO Success % increased significantly when the number of iterations was increased from 10 to 25 but remained around the same when the number of iterations was increased from 25 to 50.

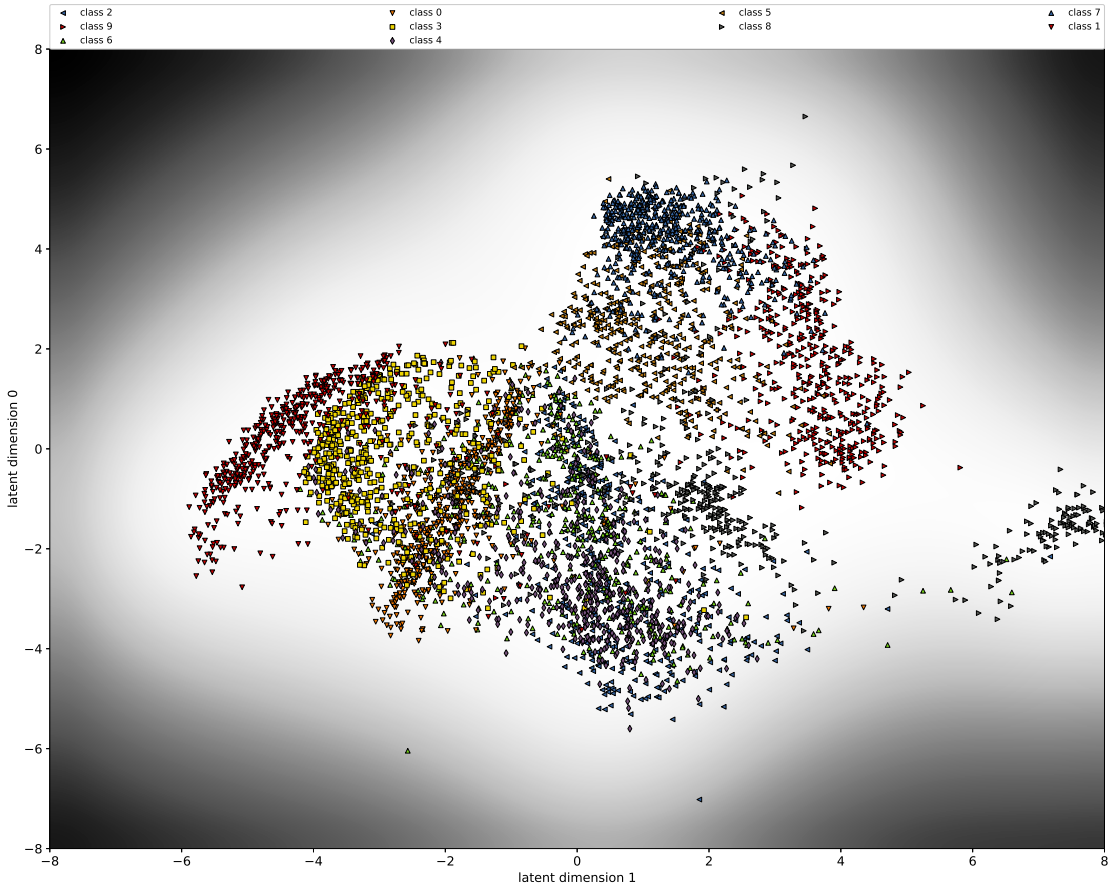


Figure 15: Latent space representations of training data points, as learnt by the Bayesian GPLVM. The representations are the same as those in Figure 10. The grayscale background represents uncertainty; the darker a latent space point, the more uncertainty associated with it.

The default initialization scheme used in the above experiments was random selection of latent space representations of training data points. The other option of using a random number generator to generate points in the latent space was not selected as it was presumed that it would cause slower convergence. Figure 15 shows that uncertainty in the latent space is low at latent space representations of training data points and high at other regions of the latent space, and since it is likely that the latent space representation of the closest match to a test image is also in a low uncertainty region of the latent space, this presumption made sense.

However, it was found that initializing from training data points did not yield much improvement over initializing randomly; convergence was achieved after 25 iterations even when random initialization was used.

5 Conclusion and Further Work

In this thesis, we applied a technique called Latent Variable Bayesian Optimization (LVBO) to the use case of finding the closest match for an image of a clothing item. The key conclusions that we can make from the results obtained are given below:

1. LVBO works much better than standard Bayesian optimization when the latent space learnt by the dimensionality reduction model captures the geometry of the original high dimensional space. It yields a close match for the given image while Bayesian optimization just yields noise.

2. Propagating uncertainty of the latent space into the optimization by specifying the surrogate model to be a Bayesian GPLVM instead of a standard Gaussian process does not improve the results obtained. The reason for this needs to be investigated further - a probable reason might be the variational approximation $q(\mathbf{X})$ in the Bayesian GPLVM.

3. Another way in which the uncertainty can be propagated is by using Hellinger or Geodesic distances, instead of Euclidean distance, to measure how far apart points in the latent space points are. When the surrogate model is specified to be the Bayesian GPLVM, Hellinger distance yields a significant improvement over Euclidean distance while Geodesic distance yields a minor improvement over Euclidean distance.

4. The time taken for Geodesic distance to be computed is about 2 orders of magnitude more than the time taken for Euclidean distance. This means that the minor improvement in performance over Euclidean distance, as mentioned in the previous point, is not worth the enormous additional time.

Besides the suggestion for further investigation made in point 2 above, other avenues for further work include:

1. Implementing other f-divergences and seeing whether they yield better results than Hellinger distance.

2. Applying LVBO to other use cases to find out which results obtained in this thesis are use case specific and which results are generally true for LVBO.

References

[1] I. Dewancker, M. McCourt, and S. Clark, "Bayesian optimization primer." `https://sigopt.com/static/pdf/SigOpt_Bayesian_Optimization_Primer.pdf`. Accessed: 2018-08-15.

[2] Z. Wang, F. Hutter, M. Zoghi, D. Matheson, and N. de Freitas, "Bayesian Optimization in a Billion Dimensions via Random Embeddings," *ArXiv e-prints*, Jan. 2013.

[3] K. Kandasamy, J. Schneider, and B. Poczos, "High Dimensional Bayesian Optimisation and Bandits via Additive Models," *ArXiv e-prints*, Mar. 2015.

[4] X. Lu, J. Gonzalez, Z. Dai, and N. Lawrence, "Structured variationally auto-encoded optimization," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, (Stockholmsmässan, Stockholm Sweden), pp. 3267–3275, PMLR, 10–15 Jul 2018.

[5] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms," *ArXiv e-prints*, Aug. 2017.

[6] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[7] "Gaussian process regression tutorial." `http://nbviewer.jupyter.org/github/SheffieldML/notebook/blob/master/GPy/basic_gp.ipynb`. Accessed: 2018-08-15.

[8] C. O. S. Sorzano, J. Vargas, and A. P. Montano, "A survey of dimensionality reduction techniques," *ArXiv e-prints*, Mar. 2014.

[9] R. D. De Veaux and L. H. Ungar, "Multicollinearity: A tale of two nonparametric regressions," in *Selecting Models from Data* (P. Cheeseman and R. W. Oldford, eds.), (New York, NY), pp. 393–402, Springer New York, 1994.

[10] A. Damianou, "Non-linear probabilistic dimensionality reduction for dynamical and multi-modal vision datasets." `http://adamian.github.io/talks/KTHtalk.pdf`. Accessed: 2018-08-15.

[11] M. Titsias and N. D. Lawrence, "Bayesian gaussian process latent variable model," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (Y. W. Teh and M. Titterington, eds.), vol. 9 of *Proceedings of Machine Learning Research*, (Chia Laguna Resort, Sardinia, Italy), pp. 844–851, PMLR, 13–15 May 2010.

[12] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *ArXiv e-prints*, Dec. 2013.

[13] "Probabilistic pca with gplvm." `https://wiki.math.uwaterloo.ca/statwiki/index.php?title=probabilistic_PCA_with_GPLVM`. Accessed: 2018-08-15.

[14] N. D. Lawrence, "Gaussian process latent variable models for visualisation of high dimensional data," in *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, (Cambridge, MA, USA), pp. 329–336, MIT Press, 2003.

[15] M. E. Tipping and C. M. Bishop, "Probabilistic principal component analysis," *Journal of the Royal Statistical Society, Series B*, vol. 21/3, p. 611–622, 01 1999.

[16] "Source code for inferencex." `http://gpy.readthedocs.io/en/deploy/_modules/GPy/inference/latent_function_inference/inferenceX.html`. Accessed: 2018-08-15.

[17] J. Altosaar, "Tutorial - what is a variational autoencoder?." `https://jaan.io/what-is-variational-autoencoder-vae-tutorial/`. Accessed: 2018-08-15.

[18] C. Chou, "f-divergences." `http://www.regor.url.tw/cnchou/wp-content/uploads/NTUActiveLearner/statistics/f-divergence/f-divergences-original.pdf`. Accessed: 2018-08-15.

[19] "Hellinger distance." `https://en.wikipedia.org/wiki/Hellinger_distance`. Accessed: 2018-08-15.

[20] M. Zwiessele, "Bringing models to the domain: Deploying gaussian processes in the biological sciences." `http://etheses.whiterose.ac.uk/18492/1/MaxZwiesseleThesis.pdf`. Accessed: 2018-07-21.

[21] A. Tosi, S. Hauberg, A. Vellido, and N. D. Lawrence, "Metrics for Probabilistic Geometries," *ArXiv e-prints*, Nov. 2014.

A Code

The code for LVBO is given in the form of a Jupyter Notebook at:
`https://github.com/aursid/LVBO`.

The code uses the GPy and GPyOpt libraries extensively. Modifications were needed to GPy in some places - these are indicated in the Jupyter Notebook.

B Dimensionality Reduction Model Parameters

The hyperparameters used to train the Variational Autoencoder and the Bayesian GPLVM are shown Table 7 and 8 .

Hyperparameter	**Value**
Batch size	100
No. of batches	50
No. of latent layer units	400
No. of output units	784
No. of latent layers	3
No. of dense layers	2
No. of epochs	50

Table 7: Hyperparameters for VAE training.

Hyperparameter	**Value**
Max. iterations	200
Kernel	ARD-RBF

Table 8: Hyperparameters for Bayesian GPLVM training.

C Detailed LVBO Algorithm

The detailed version of the LVBO algorithm, with the back-and-forth mapping, is:

1. Initialize s points in latent space and store in matrix called $\mathbf{W}$.
2. Store the diagonals of the covariance matrices corresponding to the points in $\mathbf{W}$ in a matrix called $\mathbf{W_{var}}$.
3. Store the objective function values corresponding to $\mathbf{W}$ in a matrix called $\mathbf{Y}$.
4. Initialize kernel to be used in surrogate model and specify which distance type (Euclidean, Hellinger or Geodesic) is to be used.
5. Initialize surrogate model to either a Gaussian process or a Bayesian GPLVM.
6. For k iterations, repeat:
   (a) Update kernel with $\mathbf{W}$, $\mathbf{W_{var}}$ and $\mathbf{Y}$.
   (b) Update surrogate model with $\mathbf{W}$ and $\mathbf{Y}$ if it is a Gaussian process or with $\mathbf{W}$, $\mathbf{W_{var}}$ and $\mathbf{Y}$ if it is a Bayesian GPLVM.
   (c) Optimize the surrogate model.
   (d) Give optimized surrogate model as input to acquisition function.
   (e) Optimize acquisition function and store the latent space point returned in a vector called $\mathbf{w_{next}}$.
   (f) Use the Bayesian GPLVM to get the data space representation $g^{-1}(\mathbf{w_{next}})$.
   (g) Use the Bayesian GPLVM inference model to get the variational posterior corresponding to $g^{-1}(\mathbf{w_{next}})$. Store the posterior's mean in a vector called $\mathbf{w_{nextmean}}$ and the diagonal of the posterior's covariance matrix in a vector called $\mathbf{w_{nextvar}}$.
   (h) Update $\mathbf{W}$ with $\mathbf{w_{nextmean}}$.
   (i) Update $\mathbf{W_{var}}$ with the $\mathbf{w_{nextvar}}$.
   (j) Evaluate the objective function at $\mathbf{w_{next}}$ and update $\mathbf{Y}$ with the result.

The objective function used with the above algorithm is:

1. Take as input a latent space point $\mathbf{w}$.

2. Use the trained Bayesian GPLVM to get the data space representation $g^{-1}(\mathbf{w})$.

3. Use the Bayesian GPLVM inference model to get the latent space point corresponding to $g^{-1}(\mathbf{w})$ [i.e., Find $g(g^{-1}(\mathbf{w}))$]. As discussed in Section 2.2.2, the Bayesian GPLVM inference model returns a variational posterior and not a point estimate. The mean of this posterior is considered to be $g(g^{-1}(\mathbf{w}))$.

4. Use the Bayesian GPLVM to get the data space representation corresponding to $g(g^{-1}(\mathbf{w}))$ [i.e., Find $g^{-1}(g(g^{-1}(\mathbf{w})))$].

5. Return as output the Euclidean distance between $\mathbf{i}$ (the given image) and $g^{-1}(g(g^{-1}(\mathbf{w})))$.