Dimensionality Reduction in Complex Models

A. Boukouvalas, D. M. Maniyar and D. Cornford

Neural Computing Research Group
Aston University
Aston Street, Birmingham B4 7EG, UK
Tel. +44-121-204-3652, Fax. +44-121-204-3685
Email: boukouva, maniyard, d.cornford @aston.ac.uk

Abstract

As a part of the Managing Uncertainty in Complex Models (MUCM) project, research at Aston University will develop methods for dimensionality reduction of the input and/or output spaces of models, as seen within the emulator framework. Towards this end this report describes a framework for generating toy datasets, whose underlying structure is understood, to facilitate early investigations of dimensionality reduction methods and to gain a deeper understanding of the algorithms employed, both in terms of how effective they are for given types of models / situations, and also their speed in applications and how this scales with various factors. The framework, which allows the evaluation of both screening and projection approaches to dimensionality reduction, is described. We also describe the screening and projection methods currently under consideration and present some preliminary results. The aim of this draft of the report is to solicit feedback from the project team on the dataset generation framework, the methods we propose to use, and suggestions for extensions that should be considered.

Technical Report NCRG/2007/001

May 2007

1 Introduction

Within MUCM, Aston has the role of tackling high dimensional data sets. In order to allow the Aston MUCM team to gain increased familiarity with various aspects of the dimension reduction problem, we have created a series of 'toy data sets'. These have been designed to simulate some of the key features of the sorts of problems we aim to address within MUCM. The generative formulation requires us to address issues such as, how do we think the high dimensional data we might see from 'real models ' might have been generated? The actual data sets also allow us to experiment with the methods we intend to apply in a problem setting where the correct solution is known, which will not generally be the case in real model situations.

To help us understand both the screening and projection approaches to dimension reduction we have generated two series of data sets. For the screening examples we take an initial set of inputs, pass these through a function (our simulator), then augment the input space with correlated and/or uncorrelated covariates, as described in Section 2. We then seek to determine, using a range of screening methods, the relevant inputs and apply these in an emulator of the simulator. We attempt this over a range of functions (simulators) with a variety of non-linear behaviours, as described in Section 3. When dealing with dimension reduction using projection, we generate a random non-linear projection into a high dimensional feature space, and attempt to recover a mapping to an appropriate low dimensional space, in which we again construct an emulator.

Within all these experiments we have to take some care to explore the generic aspects of the problem and not simply the given noise realisation, thus wherever a random element enters into the equations are careful to sample repeatedly over this distribution to find the average ability of the methods to recover the correct sub-space of the inputs/outputs and the variability of this over the noise realisations. Since the effect of noise is likely to be quite simple (we consider additive, independent noise models), we do not anticipate having to create huge numbers of realisations.

2 The data generation framework

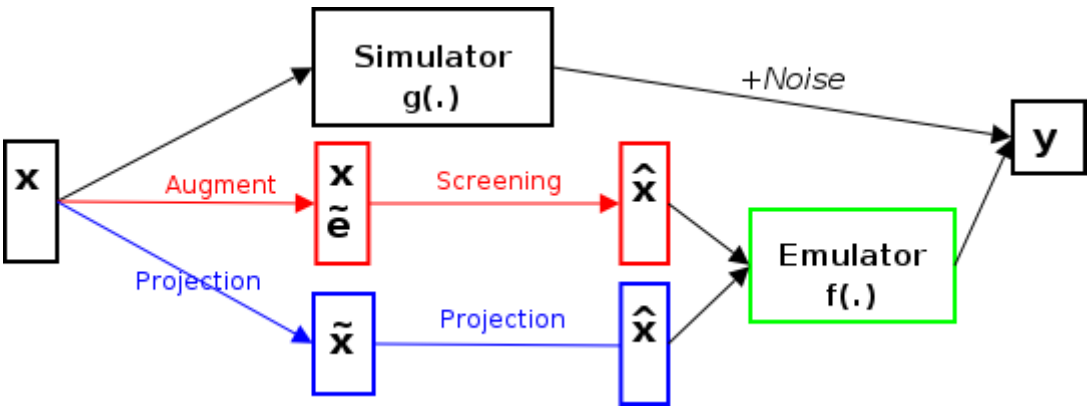


Figure 1: The data generation framework.

For both, screening and projection methods, the first part of the data generation framework is shared to generate base data (relevant inputs and their simulator outputs with desired levels of noise). In the first part we,

- Sample N base vectors, $\mathbf{x}$, of dimensionality d using a Latin hypercube sampling from uniform distribution [11]. All dimensions have the same domain $[l, u]$. We are aware that experimental design is relevant to dimension reduction methods in general and screening in particular and are seeking guidance in this area.
- Normalise all vectors $\mathbf{x}$ such that they have zero mean and unit variance by applying the transform $\mathbf{x} = \frac{\mathbf{x}-\mu_{\mathbf{x}}}{\sigma_{\mathbf{x}}}$. This step simplifies subsequent analysis.

- Obtain noiseless model outputs using a desired simulator (see section 3).
- Corrupt the model output with independent identically distributed Gaussian noise

$$y = g(\mathbf{x}) + \eta, \tag{1}$$

where $\mathbf{x}$ is a d dimensional input vector, $g(\mathbf{x})$ is the generative function selected previously and η is distributed as zero mean additive Gaussian noise. The variance of the noise term η is set to be $\frac{1}{10}_{th}$ of the signal variance, i.e. $0.1 \times Var(g(\mathbf{x}))$. [1]

The last step to obtain toy datasets to evaluate the screening and projection methods in the emulator setting is described in Section 2.1 and Section 2.2 respectively.

2.1 Screening

For screening the last step in the generation of the dataset involves augmentation with extra 'noise dimensions'. Assuming a specified number of extra dimensions d_{extra} and a base input data matrix $\mathbf{X}$, with N examples in dimension d, a noise design matrix can be constructed as

$$\tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{X} \\ \mathbf{B}\mathbf{X}^{\tau} + \epsilon \end{bmatrix}, \tag{2}$$

where $\tilde{\mathbf{X}}$ is the noise design matrix of dimension $(d_{extra} + d) \times N$,$\mathbf{B}$ is a correlation determination matrix with dimensions $d_{extra} \times d$ and ϵ is distributed as zero mean Gaussian noise $N(0, I)$. The form of the matrix $\mathbf{B}$ determines the correlation, if any, between noise and model variables:

- $\mathbf{B} = 0$ constructs noise variables that are uncorrelated to the model variables.
- $\mathbf{B} = col - rank(k)$ where $col - rank$ refers to linearly independent columns and $k = 0.5 \times d_{extra}$. Thus k of the noise variables are linearly correlated to single model variables. The k noise variables are assigned randomly to model variables. The exact correlation coefficient is determined by the $\mathbf{B}$ matrix element b_j where $1 \leq j \leq k$ and for framework simplicity is fixed $\{b_j\} = \{+0.5\}$
- $\mathbf{B}$ is a full matrix. Rather than generating a random matrix with large number of model variables interacting, we limit the number of interacting model variables to two. Thus at most two elements in each row of $\mathbf{B}$ are non zero. The coefficients are selected randomly from a set $\{b_j\} = \{-0.2, -0.5, +0.5, +0.7\}$. The number of rows selected is $k = 0.8 \times d_{extra}$.

2.2 Projection

The final step to generate datasets to evaluate the use of projection methods (described in Section 4.2) to dimensionality reduction is to artificially project the base input data ($\mathbf{X}$), living in lower dimension d, into a higher dimensional space q. The artificial projection can be linear or non-linear. This can be described as:

$$\tilde{\mathbf{X}} = \mathbf{W}\mathbf{\Phi}(\mathbf{X}), \tag{3}$$

where $\tilde{\mathbf{X}}$, the resulting data projected into higher dimensions q, is a $q \times N$ matrix. $\mathbf{W}$ is a $q \times d$ weight matrix and $\mathbf{\Phi}(\cdot)$ are basis functions which are responsible for introducing non-linearity into the projection. A typical choice that we will employ for such projective mappings is to use Radial Basis Functions (RBF). Specific examples of such projections will be discussed in Section 5 as and when we use them.

3 Simulator functions

Various functions (simulators) are considered. For screening, the initial goal is to demonstrate the weakness of simple methods such as forward selection or backward elimination by introducing interaction

[1] Thus this system is ergodic since the $\mu_\eta = \mu_{\mathbf{x}} = 0$ and $\sigma_\eta = c\sigma_{\mathbf{x}}$ where c is a constant.

terms between inputs. We expect that a number of functions might be added to the list of functions we use in time, as different issues arise in the real model settings we will explore.

The proposed generative functions are grouped below by the number of input dimensions they operate on.

1-D

- $$f(x) = x(sin(\frac{1}{2}x)) + 1/(x^2 + x + .001), \tag{4}$$

  where the function output landscape is not homogeneous (see figure 2(a)).

2-D

- $$f(\mathbf{x}) = sin(x_1 x_2), \tag{5}$$

  which has the interaction term within a non-linear function.

- Rosenbrock's function
  $$f(\mathbf{x}) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2, \tag{6}$$

  which is often used to test optimisation routines due to the long narrow valley near the global minimum at $[1, 1]$(see figure 2(b)).

3-D

- The function
  $$f(\mathbf{x}) = 1 + x_1 + x_1^2 + x_1 x_2 + 2x_1 x_2 x_3, \tag{7}$$

  where x_2 and x_3 only appear in interaction terms and are unlikely to be relevant unless used in conjunction with x_1 and x_2. This function is used in [2] to generate simulated data.

18-D

- The function
  $$\begin{aligned} f(\mathbf{x}) = &\frac{5x_{12}}{1 + x_1} + 5(x_4 - x_{20})^2 + x_5 + 40x_{19}^3 - 5x_{19} + .05x_2 + .08x_3 - .03x_6 + .03x_7 - .09x_9 \\ &- .01x_{10} - .07x_{11} + .25x_{13}^2 - .04x_{14} + .06x_{15} - .01x_{17} - .03x_{18} \end{aligned} \tag{8}$$

  which contains small effects factors,non-linear effects and interactions. This function is used in [8] to generate simulated data along with two augmented noise variables.

N-D

- $$f(\mathbf{x}) = \sin(\mathbf{x}^\tau)\mathbf{b}, \tag{9}$$

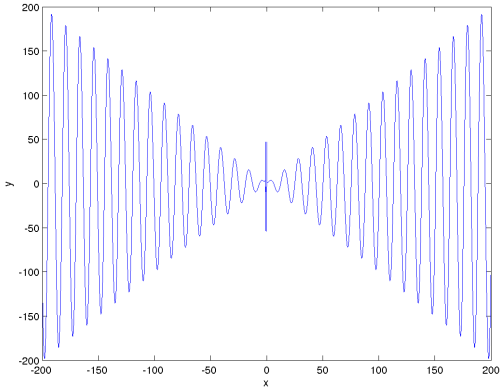
  where $\mathbf{b}$ is vector of coefficients for predictor variables. See figure2(d) for an example.

- Inverted Schwefel function:
  $$f(\mathbf{x}) = 418.98288 \times n + \sum_{i=1}^{n} -x_i \sin(\sqrt{|x_i|}), \tag{10}$$

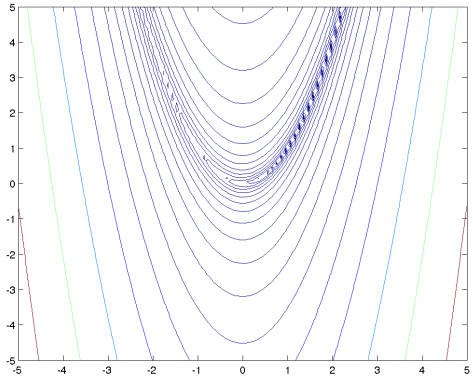
  where n is the dimensionality of $\mathbf{x}$ (see figure 2(c)). This function is non-stationary.

4 Dimensionality reduction methods

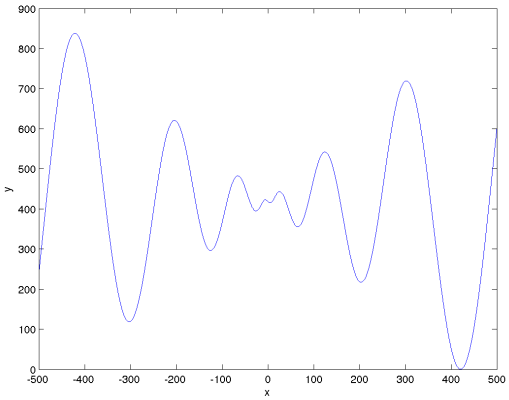
In this section we briefly describe the initial set of screening and projection methods we employ.



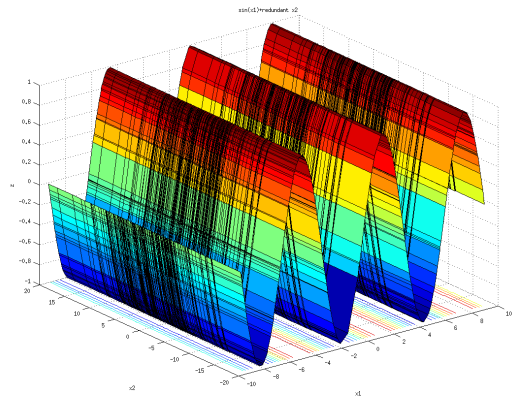
(a) $f(x) = x(sin(\frac{1}{2}x)) + 1/(x^2 + x + .001)$



(b) Contour plot Rosenbrock's function $f(\mathbf{x}) = (1-x_1)^2 + 100(x_2 - x_1^2)^2$



(c) Inverted Schwefel function for one dimension



(d) $sin(x_1)$+redundant x_2

Figure 2: Simulator functions

4.1 Screening methods

Figure 1 provides an overview of variable selection within the wider context of dimensionality reduction in an emulator setting. Variable selection methods have been broadly categorised in three categories([5]):

- Variable Ranking. Input variables are ranked according to the prediction accuracy of each $\hat{\mathbf{x}}_i$ calculated against the emulator output t. A commonly used metric is the Pearson correlation coefficient
$$R(i) = \frac{cov(\hat{\mathbf{x}}_i, t)}{\sqrt{var(\hat{\mathbf{x}}_i)var(t)}}. \tag{11}$$

- Wrapper methods. The emulator is used to assess the predictive power of subsets of variables. As is suggested in [5] the algorithm to generate subsets needs to be defined, as well as the method used to assess the prediction performance of the emulator to guide the search and halt it. Typical search strategies are forward selection where variables are progressively incorporated in larger and larger subsets and exhaustive search where all possible subsets are considered. Performance assessments are typically performed using a validation set, cross-validation or in a Bayesian setting employing model evidence.

- Embedded methods. For both variable ranking and wrapper methods, the emulator is considered a perfect black box. In embedded methods, the variable selection is done as part of the training of the emulator. An embedded method commonly employed in the context of Gaussian Processes

is Automatic Relevance Determination (ARD) where the characteristic length scales λ_i determine the input relevance ([13]):

$$k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 exp(-\frac{1}{2}(\mathbf{x}_p - \mathbf{x}_q)^\tau diag(\lambda)^{-2}(\mathbf{x}_p - \mathbf{x}_q)) + \sigma_n^2 \delta_{pq}, \tag{12}$$

where σ_n is the variance of the assumed i.i.d Gaussian noise and σ_f is the signal variance.

4.2 Projection methods

As it is shown in Figure 1, we propose to use projection methods, as a pre-processing step, to reduce the dimensionality of the input space before developing an emulator. In this section we briefly introduce some of the projection methods under consideration. Future work will consider how we might incorporate dimension reduction into the emulator inference framework directly, in a sequential setting, but at present these ideas remain speculative, and are not developed herein.

4.2.1 Principal Component Analysis (PCA)

Principal component analysis (PCA) is a multivariate procedure which rotates the data such that maximum variance is projected onto the principal axes. Essentially, a set of correlated variables are transformed into a set of uncorrelated variables which can be ordered by reducing variance. The uncorrelated variables, often called principal components, are linear combinations of the original variables, and in many cases the last few of these principal components can be removed with minimum loss of information, since these are often, but not always essentially noise variables. If the data covariance matrix has full rank, there can be as many principal components as there are variables.

Given a set of observations $\tilde{\mathbf{x}}_n \in \mathbb{R}^Q, n = 1, \ldots, N$, which are centred, $\sum_{n=1}^N \tilde{\mathbf{x}}_n = 0$, in PCA we find the principal components by diagonalising the covariance matrix given by:

$$\boldsymbol{C} = \frac{1}{N} \sum_{n=1}^{N} \tilde{\mathbf{x}}_n \tilde{\mathbf{x}}_n^\top, \tag{13}$$

which is achieved in practice using an eigen-decomposition:

$$\mathbf{CU} = \mathbf{U}\boldsymbol{\Lambda}, \tag{14}$$

where $\mathbf{U}$ is a $Q \times Q$ matrix which has the unit length eigenvectors with columns $\mathbf{u}_1, \ldots, \mathbf{u}_Q$ and Λ is a diagonal matrix with the corresponding eigenvalues, $\lambda_1, \ldots, \lambda_Q$. The eigenvectors are the principal components and the eigenvalues are the corresponding variances.

PCA is the most commonly used of the projection method. The fact that PCA defines a linear, orthogonal sub-space gives it favourable computational properties, but it is also its main limitation since any non-linear relationship between variables will not be captured.

4.2.2 Neuroscale

Neuroscale [7] is a neural network based implementation of Sammon's mapping. It is a projection method which attempts to preserve relative 'similarity' (in this case characterised by *Euclidean* distance) between points in the higher-dimensional input space ($\tilde{\mathbf{X}}$) and lower-dimensional projection space ($\hat{\mathbf{X}}$), as shown in Figure 3.

The distance-preserving nature of the transformation is imposed by the 'stress' term which accounts for the preservation of inter-point similarity

$$E = \sum_{i}^{N} \sum_{j>i}^{N} (d_{ij}^* - d_{ij})^2 \ , \tag{15}$$

where

$$d_{ij}^* = \| \tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j \| \ , \tag{16}$$

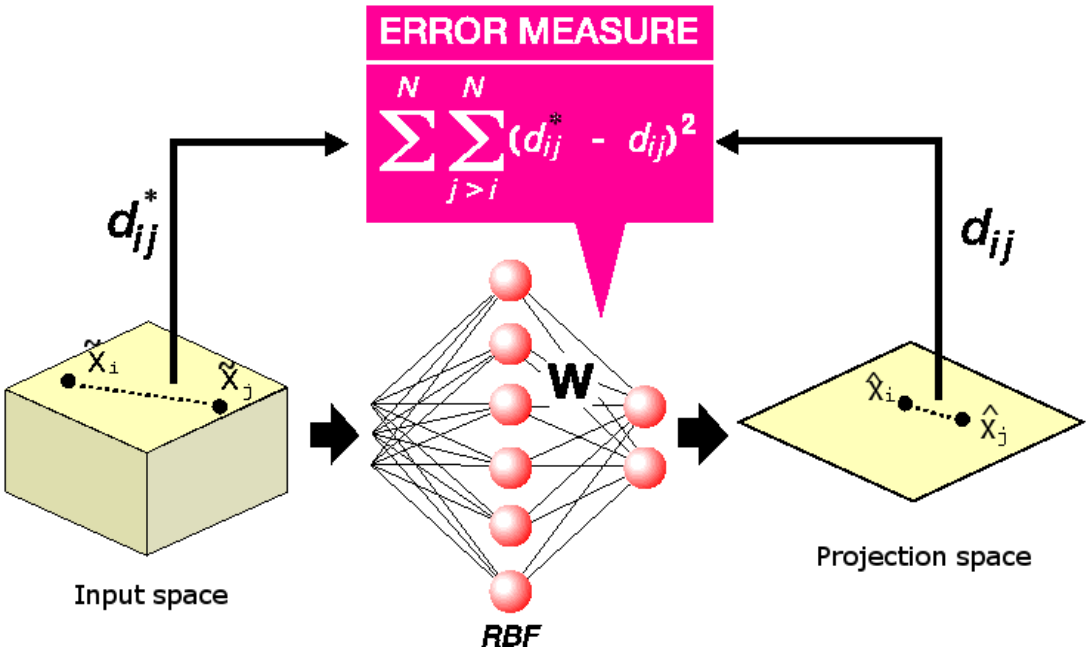


Figure 3: Schematic representation of the Neuroscale model (adapted from [12]).

and

$$d_{ij} = \| \hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j \| \ . \tag{17}$$

Points are projected from the data space onto the latent (projection) space by means of a Radial Basis Function (RBF) neural network. The RBF parameters are defined by minimising the 'stress' term, rather than the traditional residual error minimisation that might be employed in a regression setting. The high dimensional targets are projected onto the low dimensional representation using the RBF:

$$\hat{\mathbf{X}} = \mathbf{W}\boldsymbol{\phi}(\tilde{\mathbf{X}}), \tag{18}$$

where $\mathbf{W}$ is a weight matrix and $\boldsymbol{\phi}(\cdot)$ are the basis functions.

4.2.3 Generative Topographic Mapping

The Generative Topographic Mapping (GTM) models a constrained probability distribution in the high-dimensional data space, $\mathcal{Q} \subset \mathbb{R}^Q$, by means of low-dimensional latent, or hidden, variables [1]. The data is projected / visualised in the latent space, $\mathcal{H} \subset \mathbb{R}^L$.

As demonstrated in Figure 4 (adapted from [1]), we cover the latent space, $\mathcal{H}$, with an array of M latent space centres, $\mathbf{z}_i \in \mathcal{H}, i = 1, 2, ..., M$. The non-linear GTM transformation, $f : \mathcal{H} \Rightarrow \mathcal{Q}$, from the latent space to the data space is defined using a RBF network. We define in the latent space with a set of K fixed non-linear basis functions (we use Gaussian functions of the same width σ), $\boldsymbol{\phi} : \mathcal{H} \Rightarrow \mathbb{R}$, on a regular grid in the latent space. Given a point $\mathbf{z} \in \mathcal{H}$ in the latent space, its image under the map f is

$$f(\mathbf{z}_i, \mathbf{W}) = \boldsymbol{\phi}(\mathbf{z}_i)\mathbf{W}, \tag{19}$$

where $\boldsymbol{\phi}(\mathbf{z}_i) = (\phi_1(\mathbf{z}_i), ..., \phi_K(\mathbf{z}_i))^T$ and $\mathbf{W}$ is a $K \times Q$ matrix of weight parameters.

GTM defines a generative probabilistic model in the data space by placing a radially symmetric multi-variate Gaussian with zero mean and inverse variance β around images, under f, of the latent space centres $\mathbf{z}_i \in \mathcal{H}, i = 1, 2, ..., M$. We refer to the Gaussian density associated with the centre $\mathbf{z}_i$ by $p(\tilde{\mathbf{x}} \mid \mathbf{z}_i, \mathbf{W}, \beta)$. Defining a uniform prior over $\mathbf{z}_i$, the density model in the data space provided by the GTM is

$$p(\tilde{\mathbf{x}} \mid \mathbf{W}, \beta) = \frac{1}{M}\sum_{i=1}^{M} p(\tilde{\mathbf{x}} \mid \mathbf{z}_i, \mathbf{W}, \beta). \tag{20}$$

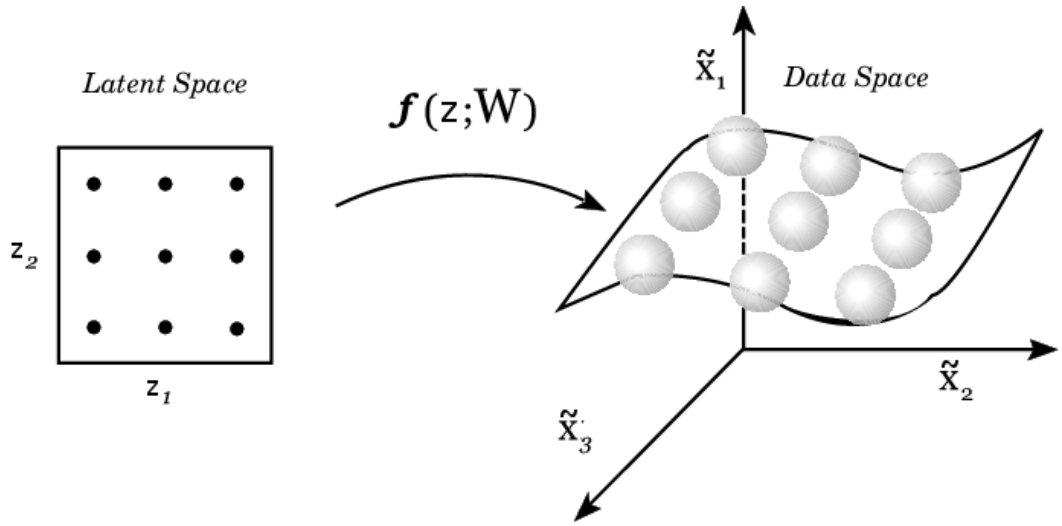


Figure 4: Schematic representation of the GTM model.

For the purpose of data projection, we use Bayes' theorem to invert the transformation f from the latent space $\mathcal{H}$ to the data space $\mathcal{Q}$. The posterior distribution on $\mathcal{H}$, given a data point $\tilde{\mathbf{x}}_n \in \mathcal{Q}$, is a sum of delta functions centred at centres $\mathbf{z}_i$, with coefficients equal to the posterior probability $R_{i,n}$ that the i-th Gaussian (corresponding to the latent space centre $\mathbf{z}_i$) generated $\tilde{\mathbf{x}}_n$:

$$R_{i,n} = \frac{p(\tilde{\mathbf{x}}_n \mid \mathbf{z}_i, \mathbf{W}, \beta)}{\sum_{j=1}^{M} p(\tilde{\mathbf{x}}_n \mid \mathbf{z}_j, \mathbf{W}, \beta)}. \tag{21}$$

The latent space representation of the point $\tilde{\mathbf{x}}_n$, i.e. *the projection of* $\tilde{\mathbf{x}}_n$, is taken to be the mean, $\sum_{i=1}^{M} R_{in}\mathbf{z}_i$ of the posterior distribution on $\mathcal{H}$. The parameters of the GTM (weights $\mathbf{W}$ and inverse variance β) are inferred from data using a variant of the Expectation Maximisation (EM) algorithm. The f–image of the latent space $\mathcal{H}, \Omega = f(\mathcal{H}) = \{f(\mathbf{z}) \in \mathbb{R}^Q \mid \mathbf{z} \in \mathcal{H}\}$, forms a smooth L-dimensional manifold in the data space. We refer to the manifold Ω as the *projection manifold* of the GTM.

5 Experimental results

5.1 Screening methods

The generative function(7) was used to generate the data. The dataset configuration is shown in table 1.

Parameter	Value
Input domain	[-1,1]
Number of observations	200
Relevant dimensions	3
Extra dimensions	3
Output noise ratio	0.1

Table 1: Screening dataset configuration

All the Gaussian Processes (GP) used in the experimental set-ups were implemented using using the NETLAB toolkit [9] with a squared exponential covariance function (although Dan is not totally happy about this!). The hyperparameters of the covariance function are estimated to determine their Maximum A Posteriori value (MAP). The hyperprior for all parameters is a single spherical Gaussian with unit mean and variance. The optimization was performed using the scaled conjugate descent optimisation algorithm with maximum number of iterations set to 50.

Two measures are presented: the root mean square error (RMSE) to show the accuracy of the selected subset when used in an emulator setting, and the wall-clock elapsed time to measure the computational cost, derived from Matlab. The RMSE was computed as:

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(t - y(\mathbf{x}))^2} \ . \tag{22}$$

There was no stopping rule used in the algorithms, rather the number of relevant dimensions was assumed to be known and equal to 3; hence each algorithm was targeted to find the best subset of three variables. In future work we will generalise this approach.

The following algorithms were used in the experiments

- BaseRelevant: Baseline run using the relevant dimensions only. The RMSE was obtained by training a GP on the relevant dimensions. This value can be interpreted as the optimal RMSE value, given the problem design.
- BaseAll: Baseline run using all the dimensions, i.e. relevant + extra. Again the RMSE was obtained by training a GP on this set. The difference BaseAll-BaseRelevant is a measure of the effect of the extra variables on the predictive accuracy of the GP.
- CorrCoef: Pearson Correlation Coefficient. A variable ranking is performed using the formulae 11 and the top 3 variables are selected and used to train a GP.
- LinFS: Employ a forward subset selection strategy using a multivariate linear regression model. The RMSE is obtained from evaluating the GP using the subset selected by the multiple linear regression model.
- GPFS: Again employ forward selection to generate subsets but use a GP rather than a linear model.
- ARD: Employ the ARD method to rank the input variables and select the top 3 to train a GP model.

In the first experiment no correlation was introduced between the noise and the true variables, i.e. the $\mathbf{B}$ matrix in 2 was zero. Note that the first three variable (1,2,3) are the true relevant variables. The results are shown in Table 2.

In the second experiment a simple correlation was introduced between a subset of the noise variables and a single relevant variable, i.e. the second case for matrix $\mathbf{B}$ in equation 2. The matrix $\mathbf{B}$ was

$$\mathbf{B} = \begin{bmatrix} 0 & 0.5 & 0 \\ 0.5 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{23}$$

. The results are shown in table 3.

The final experiment was performed with some noise variables being linearly correlated to two relevant variables, i.e. the full matrix case for matrix $\mathbf{B}$ in equation 2. The matrix $\mathbf{B}$ was

$$\mathbf{B} = \begin{bmatrix} -0.2 & -0.2 & 0 \\ 0.5 & 0.5 & 0 \\ 0 & 0 & -0.5 \end{bmatrix} \tag{24}$$

. The results are shown in table 4.

In all experiments ten realisations of the toy data were used. Also note that the elapsed time includes the training of the final GP used to validate the variable selection. In the ARD case all realisations provided identical rankings demonstrating the robustness of the algorithm. For the other algorithms however we see a substantial difference in the variables selected across different realisations. The CorrCoef and LinFS methods are computationally inexpensive but provide unreliable results. The GPFS method is nearly as robust as the ARD method but at least three times as computationally expensive.

We conclude that from the methods examined, the ARD method provides the best trade-off in terms of prediction accuracy and computational cost for the datasets examined.

Algorithm	Variables Selected	Realisations	RMSE	Elapsed time
BaseRelevant	1,2,3	10	0.2789	19.1049
BaseAll	1,2,3,4,5,6	10	0.2787	26.1944
CorrCoef	1,4,6,5,3,2	2	0.6575	27.9474
CorrCoef	1,3,4,6,5,2	1	0.5833	25.6385
CorrCoef	1,3,2,6,5,4	1	0.3228	13.603
CorrCoef	1,4,2,3,6,5	1	0.5126	18.8505
CorrCoef	1,5,6,4,2,3	2	0.6595	24.4191
CorrCoef	1,6,3,5,4,2	2	0.6130	28.0526
CorrCoef	1,2,6,5,4,3	1	0.4462	40.7532
LinFS	1,4,5	4	0.6722	22.9228
LinFS	1,3,2	1	0.3228	17.9054
LinFS	1,2,4	1	0.5126	16.7976
LinFS	1,3,5	1	0.7405	15.2841
LinFS	1,4,6	2	0.4461	38.8922
LinFS	1,2,5	1	0.4462	45.9387
GPFS	1,2,3	9	0.2790	220.9582
GPFS	1,6,3	1	0.5151	168.947
ARD	1,2,3	10	0.2790	51.7983

Table 2: Results with uncorrelated extra 3 dimensions averaged over 10 realisations

Algorithm	Variables Selected	Realisations	RMSE	Elapsed time
BaseRelevant	1,2,3	10	0.2789	22.0401
BaseAll	1,2,3,4,5,6	10	0.2785	26.013
CorrCoef	1,5,4,6,3,2	2	0.6684	30.0411
CorrCoef	1,5,3,4,6,2	2	0.6011	28.9543
CorrCoef	1,6,2,3,4,5	1	0.4902	67.4304
CorrCoef	1,6,5,3,4,2	3	0.6169	31.2259
CorrCoef	1,5,2,6,4,3	1	0.4462	48.4542
CorrCoef	1,5,6,2,4,3	1	0.5765	26.033
LinFS	1,4,5	4	0.6686	23.2702
LinFS	1,3,2	1	0.3228	14.4951
LinFS	1,2,4	2	0.4921	31.8163
LinFS	1,3,5	1	0.7384	18.138
LinFS	1,2,5	1	0.4468	36.8765
LinFS	1,2,6	1	0.5003	24.6934
GPFS	1,2,3	9	0.2790	279.9373
GPFS	1,6,2	1	0.5151	160.051
ARD	1,2,3	10	0.2790	50.0986

Table 3: Results with extra dimensions correlated to a single relevant variable averaged over 10 realisations

Algorithm	Variables Selected	Realisations	RMSE	Elapsed time
BaseRelevant	1,2,3	10	0.2619	14.2409
BaseAll	1,2,3,4,5,6	10	0.2615	25.743
CorrCoef	1,5,6,3,4,2	3	0.6607	30.8615
CorrCoef	1,5,3,4,6,2	2	0.6060	27.3796
CorrCoef	1,5,2,3,4,6	2	0.4924	36.7053
CorrCoef	1,2,3,6,4,5	1	0.2750	21.1654
CorrCoef	1,5,4,2,6,3	2	0.5715	19.9939
LinFS	1,4,5	3	0.6609	34.6243
LinFS	1,2,6	1	0.5269	25.8873
LinFS	1,2,4	3	0.5126	21.4676
LinFS	1,3,5	1	0.7405	25.2757
LinFS	1,4,6	1	0.7305	24.2757
LinFS	1,2,5	1	0.5226	23.4676
GPFS	1,2,3	10	0.2926	237.8997
ARD	1,2,3	10	0.2750	44.3431

Table 4: Results with noise dimension correlated with two relevant variables averaged over 10 realisations

5.2 Projective methods

In this section we present the results of using the projective methods, discussed in Section 4.2, to reduce the dimensionality of input space prior to constructing an emulator.

5.2.1 Toy dataset 1: RBF projection

To generate this dataset we use the generative function (9). Table 5 summarises the dataset configuration.

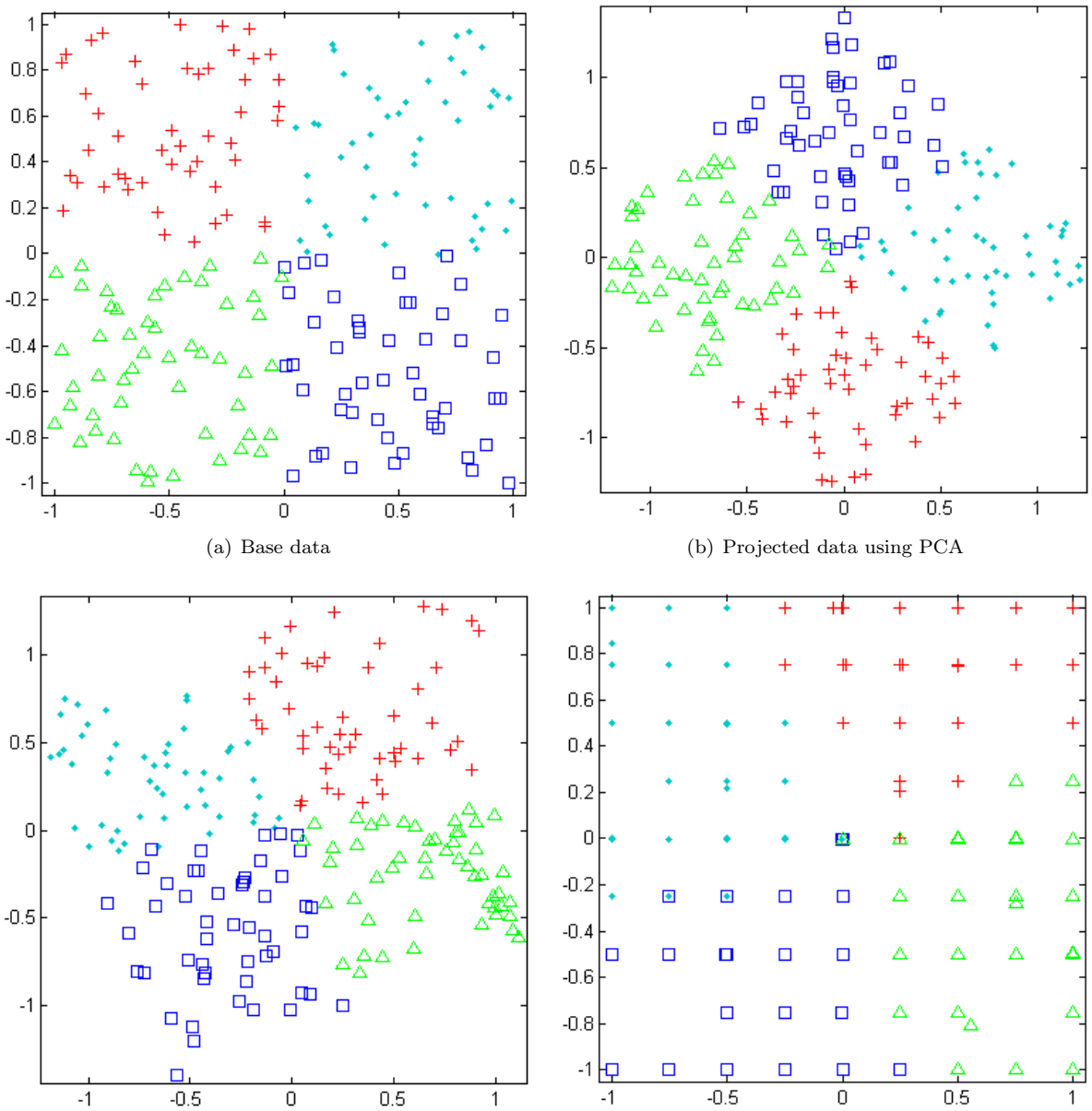
Parameter	Value
Input domain	[-1,1]
Number of observations	200
Dimensions of the base data ($\mathbf{X}$; the relevant inputs)	2
Projection function	RBF
Dimensions of the projected data in higher-dimensions ($\tilde{\mathbf{X}}$)	15
Simulator function	$g(\mathbf{x}) = \sin(\mathbf{x}^\tau)\mathbf{b}$
Output noise ratio	0.1

Table 5: Summary of the projection dataset 1

As summarised in Table 5, the two-dimensional base dataset was transformed into a 15-dimensional dataset ($\tilde{\mathbf{X}}$) using an RBF mapping function with the basis functions, $\boldsymbol{\phi}$, chosen to be 'Gaussian' and the output error function were kept as 'neuroscale' to obtain a non-linear projection in the higher-dimensional space.

We labelled the observations into 4 different classes according to their locations in the base dataset so that the output from the projection methods can be compared intuitively. Figure 5(a) shows a scatter plot of the base data with the observations coloured and marked according to the class information. Note that the class information is just used for visualisation purpose, it is not used in the construction of the emulators or training the projection methods.

Then, as a pre-processing step, the projective methods, described in Section 4.2, were applied to project the 15-dimensional input data, $\tilde{\mathbf{X}}$, onto a 2-dimensional manifold. Note that here we chose the latent space to be a 2-dimensional space so that we can visualise the projection obtained using the projection methods applied. One of the very important questions in applying projection methods for dimensionality reduction purpose is to decide the dimensionality of the latent space. We aim to address



(a) Base data

(b) Projected data using PCA

(c) Projected data using Neuroscale

(d) Projected data using GTM

Figure 5: Base data scattered plot and the projections obtained using PCA, Neuroscale and GTM.

this issue in near future. Figure 5(b), Figure 5(c) and Figure 5(d) show the 2D projections obtained using PCA, Neuroscale and GTM respectively.

After the pre-processing step we have following five input datasets:

Base data : the base data ($\mathbf{X}$).

High-dimensional data : the data obtained in 15-dimensions using an RBF projection ($\tilde{\mathbf{X}}$).

PCA projection data : Projection of the high-dimensional data, $\tilde{\mathbf{X}}$, onto a 2-dimensional space using PCA ($\hat{\mathbf{X}}_{PCA}$).

Neuroscale projection data : Projection of the high-dimensional data, $\tilde{\mathbf{X}}$, onto a 2-dimensional space using Neuroscale ($\hat{\mathbf{X}}_{NS}$).

GTM projection data : Projection of the high-dimensional data, $\tilde{\mathbf{X}}$, onto a 2-dimensional space using GTM ($\hat{\mathbf{X}}_{GTM}$).

Each of these datasets were divided into a training and a test set (50% each). A separate Gaussian process model with a squared exponential covariance function structure was constructed for each of the training sets. The results on the corresponding test sets are presented in Table 6.

Dataset	RMSE	Emulator training time (in seconds)
Base data	0.2388	1.7486
High-dimensional data	0.2420	2.3081
PCA projection data	0.2393	1.7545
Neuroscale projection data	0.2474	1.7960
GTM projection data	0.2526	1.6894

Table 6: Results on the test sets

The similar RMSE values in Table 6 suggests that using the projection methods as a pre-processing step to reduce dimensionality in emulator setting gives comparable results. As expected, emulator training time for the base and projected datasets took less time then constructing an emulator on a high-dimensional data. Though this time difference is not significant, it becomes important when we apply these methods in a sequential setting.

6 Discussion and future work

In this report we have presented the dataset generation framework along with a brief overview of screening and projection methods thus far considered. Our preliminary experimental work has obtained promising results of the methods described applied to different forms of generated data. The screening results highlight the effectiveness of the ARD method. In the future we plan to investigate a wider range of methods such as principal variables [3] and Least Angle Regression [4].

The emulators developed using the projection methods as a pre-processing step gave as good RMSE as other an emulator developed without any pre-processing step, see Table 6 although the ratio of dimension to observation set size was very favourable. To further realise the potential of the use of projection methods to reduce dimensionality prior to constructing an emulator, we require to evaluate these techniques for more complex simulator functions and in higher dimensions. In future, we aim to employee other projection methods from Kernel-family such as Kernel-PCA [10] and Gaussian Process Latent Variable Model (GP-LVM) [6]. We also aim to address the issue of determining the intrinsic dimensionality of a higher dimensional data.

We are also considering extensions to the data generation framework such as using different noise models, introducing non-linear interactions of noise variables with true variables and multiple output functions.

References

[1] C. M. Bishop, M. Svensén, and C. K. I. Williams. GTM: The generative topographic mapping. *Neural Computation*, 10:215–234, 1998.

[2] Enrique Castillo, Noelia Sanchez-Marono, Amparo Alonso-Betanzos, and Carmen Castillo. Functional Network Topology Learning and Sensitivity Analysis Based on ANOVA Decomposition. *Neural Comp.*, 19(1):231–257, 2006.

[3] J. A. Cumming and D. A. Wooff. Dimension reduction via principal variables. *Computationa Statistics & Data Analysis*, 2007.

[4] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. Technical report, Department of Statistics, Stanford University, 2002.

[5] Andr Elisseeff Isabelle Guyon. An Introduction to Variable and Feature Selection. *Journal of Maching Learning Research*, 3:1157–1182, 2003.

[6] N. D. Lawrence. Probabilistic non-linear principal component analysis with gaussian process latent variable models. *Journal of Machine Learning Research*, 6:1783–1816, 2005.

[7] D. Lowe and M. E. Tipping. Neuroscale: Novel topographic feature extraction with radial basis function networks. *Advances in Neural Information Processing Systems*, 9:543–549, 1997.

[8] William J. Welch; Robert J. Buck; Jerome Sacks; Henry P. Wynn; Toby J. Mitchell; Max D. Morris. Screening, predicting, and computer experiments. *Technometrics,*, 34(1):15–25, 1992.

[9] I. T. Nabney. *Netlab: Algorithms for Pattern Recognition*. Springer, London, 2001.

[10] B. Schölkopf, S. Mika, A. Smola, G. Rätsch, and K.-R. Müller. Kernel PCA pattern reconstruction *via* approximate pre-images. In L. Niklasson, M. Bodén, and T. Ziemke, editors, *Proceedings of the 8th International Conference on Artificial Neural Networks*, pages 147–152, Berlin, 1998. Springer Verlag.

[11] M. L. Stein. Large sample properties of simulations using latin hypercube sampling. *Technometrics*, 29:143–151, 1987.

[12] M. E. Tipping and D. Lowe. Shadow targets: A novel algorithm for topographic projections by radial basis functions. *Neurocomputing*, 19:211–222, 1998.

[13] Christopher K. I. Williams and Carl Edward Rasmussen. Gaussian processes for regression. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Proc. Conf. Advances in Neural Information Processing Systems, NIPS*, volume 8. MIT Press, 1995.