

DVMS 1.5 : A User Manual

(The Data Visualisation & Modeling System)

DHARMESH M. MANIYAR AND IAN T. NABNEY



ASTON UNIVERSITY & *Pfizer* Central Research

February 2005

This copy of the user manual has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with its author(s) and that no quotation from the manual and no information derived from it may be published without proper acknowledgement.

DVMS 1.5 : A User Manual

(The Data Visualisation & Modeling System)

DHARMESH M. MANIYAR AND IAN T. NABNEY

Summary

The data available during the drug discovery process is vast in amount and diverse in nature. To gain useful information from such data, an effective visualisation tool is required. To provide better visualisation facilities to the domain experts (screening scientist, biologist, chemist, etc.), we developed a software which is based on recently developed principled visualisation algorithms such as Generative Topographic Mapping (GTM) and Hierarchical Generative Topographic Mapping (HGTM). The software also supports conventional visualisation techniques such as Principal Component Analysis, NeuroScale, PhiVis, and Locally Linear Embedding (LLE).

The software also provides global and local regression facilities. It supports regression algorithms such as Multilayer Perceptron (MLP), Radial Basis Functions network (RBF), Generalised Linear Models (GLM), Mixture of Experts (MoE), and newly developed Guided Mixture of Experts (GME).

This user manual gives an overview of the purpose of the software tool, highlights some of the issues to be taken care while creating a new model, and provides information about how to install & use the tool. The user manual does not require the readers to have familiarity with the algorithms it implements. Basic computing skills are enough to operate the software.

Keywords: Drug Discovery, Machine Learning, Visualisation, Regression, Graphical User Interface

Contents

1	Introduction	4
1.1	Motivation	4
1.2	The approach	5
1.3	Data Visualisation & Modelling System (DVMS)	5
1.3.1	Installing DVMS	6
2	Using DVMS	7
2.1	The <i>configuration</i> file	7
2.2	The <i>data</i> file	9
2.2.1	Data Selection	9
2.2.2	Data Pre-processing	10
3	Creating and using visualisation models	12
3.1	Training a Model	12
3.1.1	Deciding Parameters	12
3.1.2	Interactive training for HGTM model	14
3.1.3	Model evaluation	17
3.2	Visualising trained models	17
3.2.1	Visualisation of a PCA, Neuroscale, or GTM model	18
3.2.2	Visualisation of a HGTM model	19
4	Creating and using regression models	21
4.1	Introduction	21
4.2	Training and using a global model	21
4.3	Training and using a guided local regression model	21

Chapter 1

Introduction

1.1 Motivation

Biological activity data of chemical compounds on different targets collected using technologies such as high-throughput screening (HTS) and the availability of detailed physicochemical properties of chemical compounds and their fingerprint data, could be used to mine useful information [1] to improve the drug discovery process. Screening scientists are required to use these data to effectively screen out compounds from the vast compound library. One of the challenges the screening scientists face is to visualise this vast data to effectively interpret it and take quick decisions from.

Data visualisation is an important means of extracting useful information from large quantities of raw data. It is difficult for a normal human being to visualise data in more than three dimensions. That is why projection of high-dimensional data in lower-dimensional space is useful in understanding data. The term visualisation is used here for a method of projecting high-dimensional vast data in lower dimensions in such a way that the projected data keeps most of the topographic properties and makes it easier for the users to interpret the data to gain useful information from it.

Regression is a category of problems where the objective is to estimate the value of a continuous output variable from some input variables. For example, One of the most useful computational model for hit identification would be to be able to relate physicochemical properties and fingerprint properties of compounds with their biological activity without actually carrying out screening on the HTS. If such robust reliable model is created, then the screening scientist can first predict the biological activity for compounds and then on the basis of that decide which compounds are worth actual testing on HTS.

Use of powerful software such as SpotFire ¹ and PIPELINE PILOT ² is now common in all major pharmaceutical company. Such software provides basic machine learning techniques, such as projection using PCA, regression using GLM, but they still lack the implementation of new principled and powerful machine learning algorithms to provide effective visualisation and regression.

The aim behind developing a new software tool is to facilitate the domain experts (screening

¹Spotfire, <http://www.spotfire.com/>

²SciTegic, http://www.scitegic.com/products_services/pipeline_pilot.htm

scientists, biologist, chemist, etc) with the new visualisation and regression algorithms.

1.2 The approach

For vast high-dimensional datasets such as in drug discovery, traditional visualisation techniques such as principal component analysis (PCA) [2] and NeuroScale [3] are not likely to be sufficient to capture all the interesting aspects. A recently developed principled visualisation technique such as generative topographic mapping (GTM) [4] can be effective. Moreover, a hierarchical visualisation system, such as hierarchical GTM (HGTM) [5], which allows us to explore interesting regions more in detail at deeper levels is desirable for huge datasets as ours. These principled techniques can not only help domain experts to understand the data more effectively but can also help us in development of *guided* local regression models.

Because of the volume and the diversity of the data, trying to develop a single regression model to predict the activity of all the compounds in the library is unlikely to succeed. What could be effective is a group of local models, each of which working on a set of similar compounds, in other words, in different regions of the input space. In addition, we develop *guided* local regression models in such a way that domain experts give us the segmentation of the input space. We develop, implement and compare different regression models to predict the biological activities of compounds using their physicochemical properties.

To obtain an informed segmentation of the input space which then can be used to develop effective local regression models, it is important to understand the structure of the data and present it to the domain experts. Applying principled visualisation techniques first serves that purpose.

We use the informed segmentation obtained to develop powerful localised linear and/or non-linear regression models. Analytically it is possible to use the *soft* segmentation obtained from the principled visualisation technique, such as HGTM, to develop *guided* variants of popular local regression models such as Mixture of Experts (ME) [6].

An interactive software tool which supports these algorithms is provided to the domain experts.

1.3 Data Visualisation & Modelling System (DVMS)

We developed a Data Visualisation and Modelling System (DVMS) to facilitate domain experts with visualisation and regression algorithms. The DVMS v1.5 currently supports visualisation using PCA, NeuroScale, PhiVis [7], LLE [8], GTM, and HGTM algorithms. DVMS is designed as an easy to use, interactive graphical user interface (GUI) tool to help the users to visualise and understand data. The software can be used to visualise any data, as long as it is in the required format (which is discussed in 2.2). The software provides regression facilities as well. It also helps domain experts to provide us *informed* segmentation of the dataset which can be used to develop *guided* local regression [9] models.

Figure 1.1 demonstrates a top-level information flow diagram for DVMS.

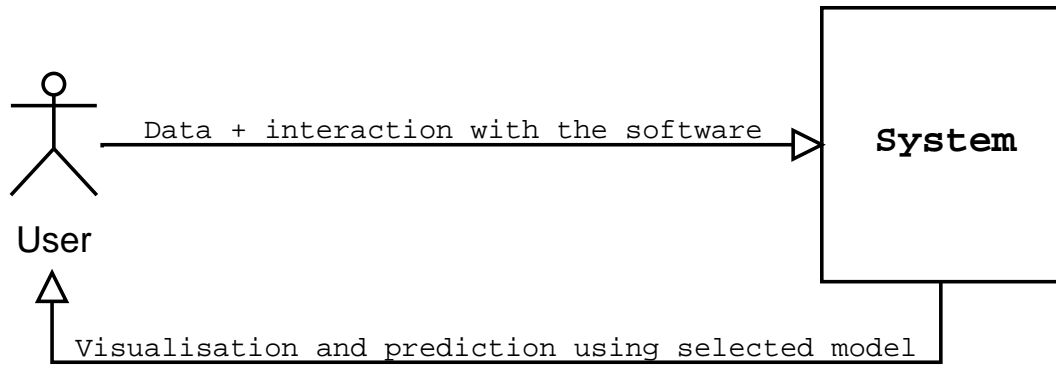


Figure 1.1: Top-level information flow diagram for DVMS

The software is developed in MATLAB³ using the NETLAB toolbox [10]. It can work as a stand-alone application on Microsoft Windows and GNU/LINUX platforms.

1.3.1 Installing DVMS

The DVMS software can be used without the user needing a MATLAB installation. DVMS is provided on a CD. Total stand alone version of DVMS v1.5 with all the required libraries is around 250MB. User should carry out following steps to run DVMS on a machine:

- Copy the entire DVMS v1.5 directory from the DVMS v1.5 CD on to the hard disk.
- Double click the batch file (`dvms.bat`) available in `.pdf` format in the DVMS v1.5 directory.

Alternatively, DVMS v1.5 can directly be run from the CD, which might be comparatively slower. This document is available in the `doc` directory in the DVMS v1.5 directory.

³The MathWorks Inc., <http://www.mathworks.com/>

Chapter 2

Using DVMS

This chapter provides information about how to use the DVMS software. The entire process of developing new models using DVMS can be broken into 4 steps as below:

1. Creating the *configuration* file
2. Creating the *data* file
3. Creating *models*
4. Using the *models*

The section 2.1 and the section 2.2 describes the 1st and the 2nd steps respectively. The chapter 3 highlights important issues to be taken care of while creating a new visualisation model and using it. The chapter 4 describes creation of a regression model and its use.

2.1 The *configuration* file

The *configuration* file contains information about preprocessing required on the data (normalisation) and the properties of the data. It also hold information about options according to which the output is generated. The *configuration* file can be created using a text editor. An example of a sample configuration file is given below:

```
Begin Header
$LABELING
1
$NORMALISATION
1
$NO_VARIABLES
15
$PROPERTYHEADER
0
$PREDICTION
0
$GMM_KL
```

0

End Header

The *configuration* file must start with the `Begin Header` row and it must have the `End Header` row as the last row. For the current version of DVMS, there should be six different type of options stored in the *configuration* file. All options start with the `$` sign on separate row. Details about each of the options are discussed below.

- **\$LABELING** : This option is used to specify if the records in the data file have labeling information. As described in the section 2.2, last field of the *data* file can be a label field. If the *data* file has the label information, then this option should be set to 1. Otherwise it should be set to 0. Example of the *data* file shown in the section 2.2 has the label information, so the *configuration* file should have **\$LABELING** option set to 1. As demonstrated in the example.
- **\$NORMALISATION** : This option is used to specify whether to normalise the data or not. If it is set to 0, the data will not be normalised. If it is set to 1, the data is normalised by treating each variable as independent and normalising them to have a mean of zero and standard deviation of one, as

$$\tilde{x}_i^n = \frac{x_i^n - \bar{x}_i}{\sigma_i}. \quad (2.1)$$

where $n = 1, \dots, N$ labels the patterns, \bar{x}_i and σ_i^2 represent mean and variance of variable i respectively.

If this flag is set to 2, the data is normalised using the whitening technique [2].

Data having diverse scales should be normalised for useful results. Since, drug discovery data generally have variables with diverse scales, it is recommended to always normally normalise the data. Hence most of the time this option should be set to 1.

- **\$NO_VARIABLES** : It represents dimension of the data space. It should be a number specifying number of variables the *data* file contains. As explained in the section 2.2, the number of variables in the *data* file is total number of fields in the *data* file except the ID field (the first field) and the labeling field (if any, it should be the last field).
- **\$PROPERTYHEADER** : If the data file has the first row as property names, this option is set to 1, else it should be 0.
- **\$PREDICTION** : If the experiment you want to carry out using this configuration file involves regression, this flag should be set to 1. Else it should be 0. The first variable (after the ID field) is treated as the output variable for the regression model.
- **\$GMM_KL** : This field is set 1 to quantitatively check separation given by different visualisation algorithms. If it is set to 1, DVMS fits Gaussian Mixture Model (GMM) on different labels and

calculates Kullback-Leibler (KL) divergence [11] between them. This should be set to 1 only if the \$LABELING flag is set to 1 and user wants a quantitative idea about the quality of the visualisation.

2.2 The *data* file

The *data* file should contain the raw data inputed to the system in the Comma Separated Value (CSV) file format. If the \$PROPERTYHEADER option in the *configuration* file is set to 1, the data file should have first row as the list of property names. Otherwise, the data file should not contain any header row. As the format suggests, the columns in the *data* file should be separated by “comma” and each record is on separate row. The first column is the ID (in the case of drug discovery data, generally it will be the compound ID). Subsequent columns, until the last column, should be the values(DVMS v1.5 supports only numeric values) of different variables (for example, screening results, physicochemical data, etc.). The last column is for the labeled data (data with known classes). It is useful to have a good labeling, particularly while visualising the data with the HGTM algorithm. If the flag \$LABELING is set to 1 in the *configuration* file, then the user has to provide last column of data file as label information. Otherwise the last column should be the last variable of the actual data.

An example of a *data* file . . .

```
1,2.6,-2.2, ... 0.45,6.68,1
5,37.6,0.7, ... 2.82,8.47,1
8,-49,0.6, ... 4.44,7.36,2
10,-9.2,4.4, ... 3.81,0.67,2
11,15.8,4.4, ... 3.46,55.38,2
13,50.6,0.4, ... 4.831,42.56,3
14,13,-3.2, ... 2.319,8.17,3
15,-2,16.4, ... 2.958,1.58,3
...
...
```

Data file in the above format can be directly generated using the widely used PIPELINE PILOT¹ tool at *Pfizer* and tools such as Microsoft Excel.

Following issues should be taken care of while creating a data file.

2.2.1 Data Selection

Data selection is a vital part of the process because if the variables that are chosen do not contain useful information, it is impossible to get any insight from a visualisation tool. However, this does not mean that every possible variable should be included. The reason for this is that if too many variables are used, then the interesting underlying relationships in the data can be obscured by unimportant variations (or ‘noise’) on other variables. Luckily, visualisation by its very nature helps the user explore a dataset, and so the results can be used to guide variable selection.

¹SciTegic, http://www.scitegic.com/products_services/pipeline_pilot.htm

2.2.2 Data Pre-processing

DVMS v1.5 requires all variables to be expressed as numbers (i.e. it does not explicitly cater for discrete variables). It is also helpful if all the variables are measured on a similar scale. For example, if the range of one variable is -1000 to 1000 , and the range of the others is -1 to 1 , then the first variable will dominate the results. A common technique is to *normalise* each variable to have a mean of zero and standard deviation of one. DVMS v1.5 provides this facility as described in the section 2.1.

Normalisation works well in most circumstances, but problems can still arise if there are significant *outliers*: data values which are *very* different from the norm. This may prevent the model from being trained successfully, but more usually, the visualisation plot shows the bulk of the data in one large indistinguishable cluster and just a few data points well separated from it. One of the advantages of using visualisation is that it enables the user to see the presence of these unusual points and then exclude them from the main analysis. Alternatively, if HGTM is used, then sub-models can be placed to split the outliers from the rest of the data.

The DVMS v1.5 requires every entry in the data matrix to have a value. It is possible to train GTM (and HGTM) on datasets where some values are missing, this is a future extension for the tool. If some values are missing, then either the data point should be excluded or they should be replaced by the mean value for the given variable.

It is often useful to include a ‘label’ information that classifies data points. This classification is used to colour the data points, which helps to understand the relationships in the dataset. It also helps the user to see if the different classes are clearly separated in the visualisation plots, which is a useful criterion for determining whether the visualisation process is complete during the training of hierarchical visualisation models.

For example, in screening data, labeling compounds by the number of screens they were active on were used in previous work [12]. In logP visualisations, the logP variable was discretised into bands, and was used as a way of colouring the data points.

It is required to load the *configuration* and *data* files before training a model. It can be done using the main interface of the DVMS as shown in the Figure 2.1. The interface has ‘Conf’ and ‘Data’ menu to do the task or the user can also use intuitive ‘Load Configuration’ and ‘Load Data’ section of the main interface.

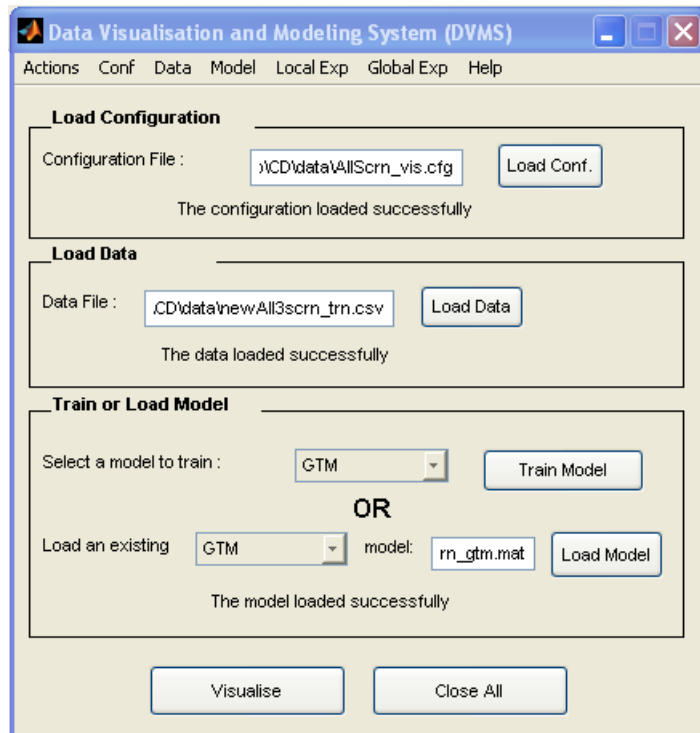


Figure 2.1: Main interface of the DVMS

Chapter 3

Creating and using visualisation models

This chapter discusses different issues one should consider during the development of visualisation models. There is more to developing a good visualisation model than simply running a training algorithm. Model development is a process, and each stage must be carefully considered if the end result is to be useful. Two important issues in creating a good model data selection and data pre-processing, are discussed in Section 2.2.1 and Section 2.2.2 respectively. Two other important steps; model training and model evaluation, are discussed here. The discussion on model development in this chapter is not applicable for PCA and LLE as they do not require separate training.

3.1 Training a Model

The purpose of training a model is to adjust the model parameters (sometimes known as weights) so that the model fits well to the data. The quality of the fit is measured using an *error function*: the smaller the value of the error function (which may be negative) the better the fit. Note that the error function for GTM and HGTM is quite different from that for Neuroscale, and hence the values cannot be compared between these models.

The key question is how well the model fits the underlying generator of the data; we say that a good model *generalises* well to new data. This can be measured by *testing* the model (i.e. evaluating the error function) on a separate dataset. It is this property of generalisation that enables the user to train the model on a smaller sub-sample of the data (usually a relatively slow process) and then visualise the main dataset (usually a fast process).

3.1.1 Deciding Parameters

When training a model, there are certain macro-level parameters that the user needs to determine. Adjustable parameters settings for training a Neuroscale or GTM model can be seen in the Figure

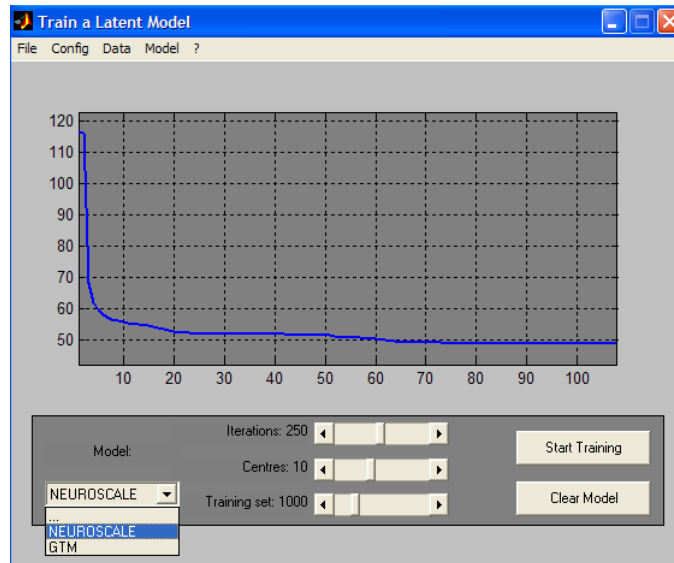


Figure 3.1: Interface for training a Neuroscale or GTM model

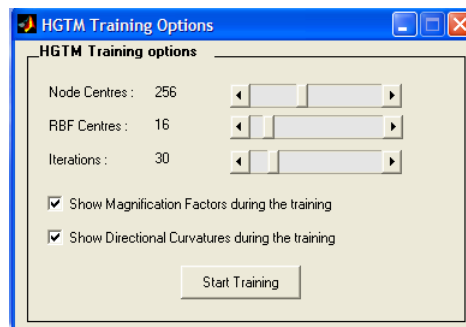


Figure 3.2: HGTM adjustable parameters during the training

3.1 and for the HGTM model can be seen in the Figure 3.2. The main architectural parameter for NeuroScale is the number of RBF centres. For GTM and HGTM they are the number of node centres (Gaussians) and number of RBF centres.

Model complexity

These consist the size and structure of the model. Typically, large numbers (number of RBF centres or node centres) allow the model to be more complicated. If the number is too small, then the model will be too simple and will have a large error on the training data. If the number is too large, then the model will have a low error on the training data, but a larger error on new data because the model is too specific to the details of the training data (i.e. the model is *overtrained* or *overfitted* to the data).

One way to determine a good value for architectural parameters is to train several models with a range of values and compare the generalisation performance. We should look for the simplest model that generalises well.

NeuroScale: number of hidden units (RBF centres). The larger the number, the more complex the

projection function can be.

GTM: The GTM can be interpreted as a two-dimensional rubber sheet in data space: spherical blobs placed on the sheet capture the fact that the data lies near to, but not exactly on, the sheet.

1. Number of node centres. The Gaussians are the spherical blobs: the more that there are, the better the data can be modelled. However, the number of training iterations is proportional to the number of Gaussians, so using too many can make training very slow. It is harder to overfit, although this is possible.
2. Number of RBF centres. This governs the complexity of the map from the computer screen to data space: effectively the amount of stretch and curvature of the rubber sheet. The larger the number, the more complex the map.

HGTM: as this consists of a tree of GTM models, the architectural parameters for the GTM need to be set as each individual model is trained. In addition, the user will need to decide the number of levels and the number of child nodes at each level. To a large degree, this is a matter of how well the current set of visualisation plots explains the data. The issue is discussed further in Section 3.1.2.

Training iterations

The user has to decide how many iterations the algorithms should run for. The principle of determining when to stop training the single models (GTM and NeuroScale) is straightforward: each model should be trained until the error value has converged. During training, graphs are used to see the logarithm of error values. Once the error plot has reached a plateau (as shown in Figure 3.1), no more training is required. If the error curve has not reached a plateau when the training algorithm terminates, then the model should be trained further. Training a hierarchical model is recursive: once the top level GTM has been trained, every leaf node in the tree can be extended with child models. The next section provides more information on issues concerning training an HGTM model.

3.1.2 Interactive training for HGTM model

The additional aspects of training a hierarchical model are: how to add child plots; when and why to add child plots and when to stop.

How to add child GTMs. Child models are added to a leaf node in the current tree. The user selects points, $\mathbf{c}_i \in \mathcal{H}, i = 1, 2, \dots, A$, in the latent space that correspond to centres of the subregions they are interested in. The points \mathbf{c}_i are then transformed via the map f to the data space. Then the subregions are formed using Voronoi compartments [13].

Adding child GTMs using DVMS is easy. The user can left click on the parent GTM plot to select centres for the submodels and right click when the selection of submodels is finished.

```

$LABELING
0
$NORMALISATION
1
$NO_VARIABLES
16
$PROPERTYHEADER
1
Selected data file:C:\ForCDNew Folder\Data\Drug\8screens_mauai.csv

**** Training the Root Model ****
Cycle 1 Error 20224.822061
Cycle 2 Error 11558.371586
Cycle 3 Error 11397.362896
Cycle 4 Error 11240.657255
Cycle 5 Error 11171.522740
Cycle 6 Error 11129.275570
Cycle 7 Error 11096.215089
Cycle 8 Error 11070.082496
Cycle 9 Error 11047.987618
Cycle 10 Error 11028.278160
Warning: Maximum number of iterations has been exceeded

Press C or c (while the ErrorLog plot is highlighted) to continue training.
Press any other key or mouse button to finish the optimisation.

=====
=====

Using the left mouse click on the plot, explore the latent space in data space.
To explore the latent space using the data space, left click a point in the latent space to visualise its properties in the data space.
Right click in the latent space figure when finished.

=====
=====

Place mouse pointer into the visualisation plot area
Use Left Click to select submodels positions!
Use Right Click to finish the selection of submodels

Use Magnification Factor and Directional Curvatures plots to choose effective submodels.

```

Figure 3.3: Typical interaction during the HGTM training

Relevant instructions are provided on the plot and in the DVMS interaction window (as shown in Figure 3.3) during the training process.

When to add child GTMs. GTM models the data as a curved and stretched two-dimensional sheet. However, if the data points that a leaf model in the tree is responsible for do not lie close to such a surface, then the visualisation plot will be misleading. So, the basic principle of adding new child GTMs is to partition the data so that *locally* it lies close to a two-dimensional sheet. We can use the *parallel coordinate* facility provided by DVMS to explore patterns of nearest points (Euclidean distance) from the point selected in the latent space as shown in Figure 3.5. This can be very useful in understanding different regions of the latent space as user can see the corresponding data space value.

Thus the user should add a child GTM to a leaf model if:

1. The plot is cluttered with too many points and we can not see separate clusters.
2. With the help of the curvature plots, the user decides if the model is not flat. It is particularly helpful to put child models on either side of bands of large curvature, as this ‘slices’ the data into two simpler segments. For example, notice in Figure 3.4 there is a strong curvature band in the bottom-right corner of submodel 2. Having two submodels either side of this curvature could be useful. From the labels (color code) of the data points, it can be confirmed that it was a good decision.

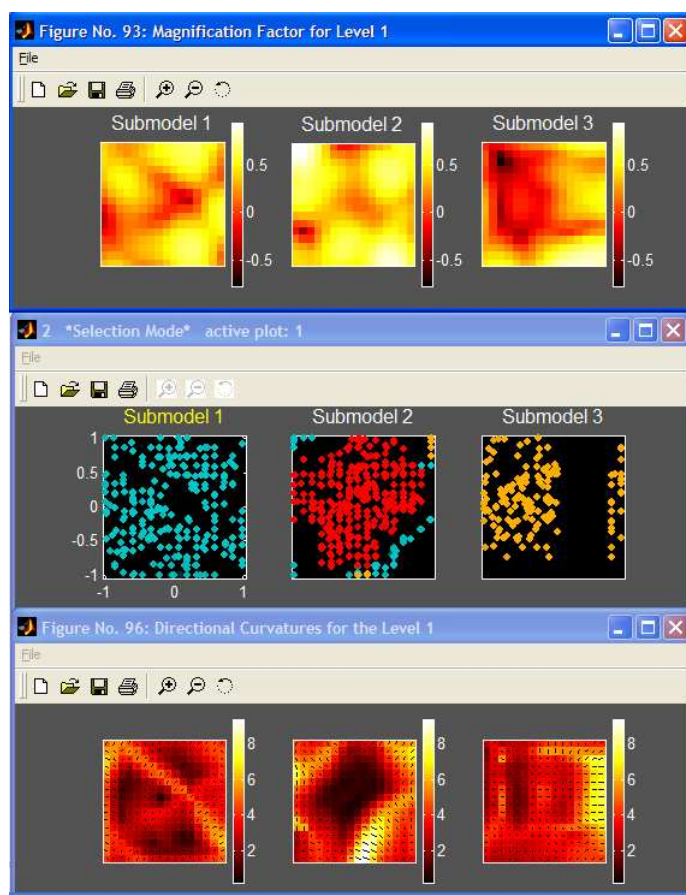


Figure 3.4: An example of plots during the HGTM training

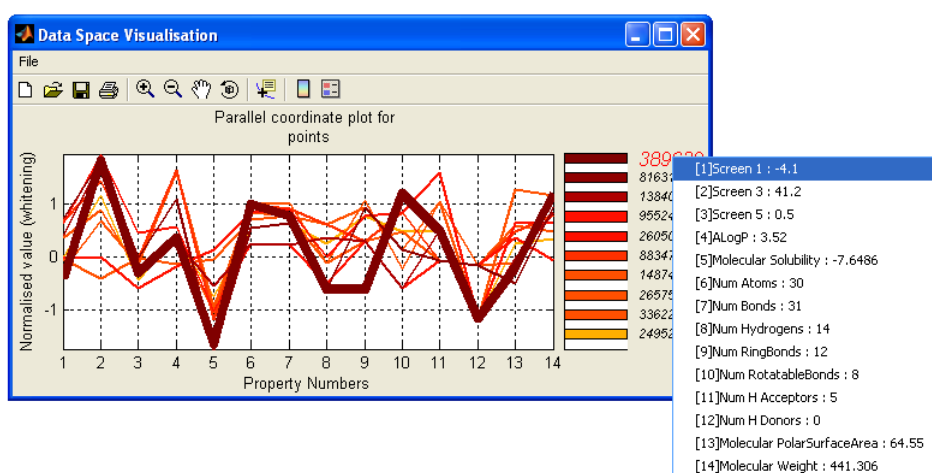


Figure 3.5: Exploring the data space using *parallel coordinate* technique

3. The magnification factor plot shows that some areas of the map are being stretched a long way. Again, putting child models in regions of high data density, create child plots that are flatter.

When to stop. One should stop adding models when the visualisation plots are telling everything that one needs to know. One way of deciding this is when the leaf node plots look similar to their parents.

If we are visualising the data, and not trying to build predictive models, then it is not necessary to create a single GTM plot for each significant data cluster; it is enough if the leaf nodes show well separated clusters of data.

Training effectiveness is shown using a similar error graph as shown in Figure 3.1. We should look for the training error to end with a plateau, which means that the learning algorithm is approximating to a minimum of the learning cost function. As this stage, we can change parameters and start training the model again or can decide to train the same model further. Once a model is trained, the user can save it and test it on the testing set.

3.1.3 Model evaluation

There are two main aspects of model evaluation: how well the model fits the underlying data generator and how informative the visualisation plot is.

The first of these is best measured by generalisation performance: computing the error measure on a testing dataset. A good model should have a similar value of error per data point on the test set and the training set.

Assessing the quality of the visualisation plots themselves is something that is subjective. The magnification factor and curvature plots (for GTM) can help with this, as can a more detailed exploration of local regions with the visualisation of nearest points in data space (using the parallel coordinate technique as shown in Figure 3.5). Some experimentation with the model architecture and the variables that are included is an inevitable part of exploring the data and improving the model. Once a good visualisation model is created the visual results are relatively easy to understand the data for the domain experts. Quantitative measure of the quality of results can be obtained by KL divergence amongst the GMM fit on the latent space. It can be obtained in DVMS by setting the `$GMM_KL` flag as 1.

3.2 Visualising trained models

Model trained as explained in the section 3.1, can be loaded to visualise the data. Loading an existing model is simple. It can be done using the ‘Model’ menu or the ‘Load Model’ button interface on the main screen of the DVMS (Figure 2.1). Status of the model loading is displayed just below the ‘Load

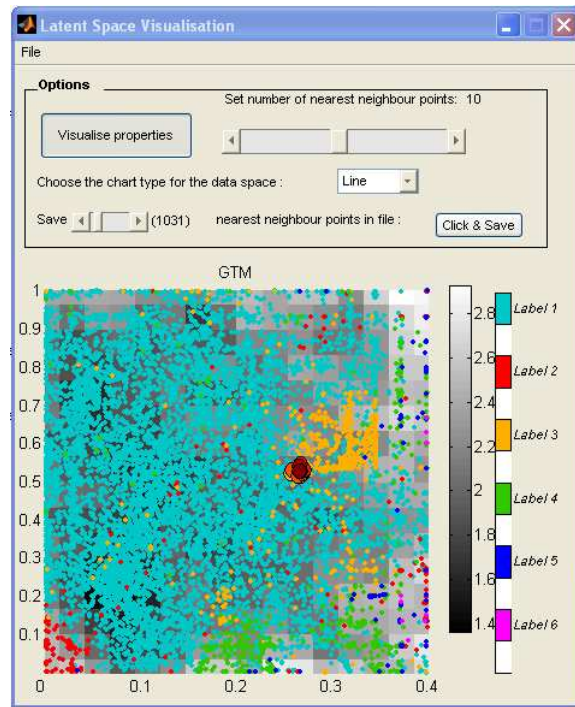


Figure 3.6: Data visualisation in latent space using a GTM model

Model’ textbox. The ‘Visualise’ button on the main interface of the DVMS (Figure 2.1), allows the data to be visualised in the latent space according to the loaded data and model.

3.2.1 Visualisation of a PCA, Neuroscale, or GTM model

Figure 3.6 demonstrates interface for PCA, Neuroscale, and GTM model visualisation. Using the interface, the user can explore nearest points in data space and can even save the latent space points as a comma delimited text file.

It is very useful to relate the visualisation of latent space to the data space. This facility is provided by the ‘Visualise Properties’ button available on the ‘Latent Space Visualisation’ interface (Figure 3.6). The user can adjust the number of nearest neighbors to be displayed and choose the type of chart to be displayed for the data space property visualisation. The user can left click any latent space point to visualise the nearest points to that point in data space. The user should stop the visualise properties action by right clicking on the latent plot. The line chart generated, displaying properties in the data space (as shown in the Figure 3.5), has some interactive facilities too. The user can select particular IDs (the width of the line associated with the ID will increase while it is selected) by clicking on the particular line or the particular ID.

If the properties header was given in the *data* file, right clicking the ID displayed on the right hand side of data space window will give a list of property name with actual data space value for that particular ID. If the properties header was not specified, only the actual data space value is displayed in a list. The example can be seen in 3.5.

Currently up to 6 labels are supported on the plots. In future this will be replaced with the color map to support unlimited number of labels.

3.2.2 Visualisation of a HGTM model

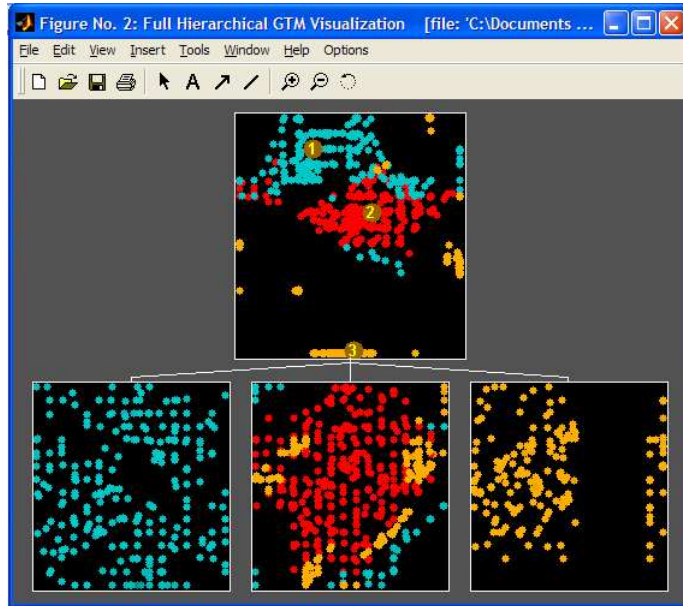


Figure 3.7: An example of hierarchy generated using the HGTM algorithm

HGTM model on data is visualised as a hierarchy of GTMs as shown in Figure 3.7. If the data is labeled, different colours will be used for different labeled points. The interface provides an 'Options' menu which can be used to display magnification factors (as demonstrated in Figure 3.8 and compute & show directional curvatures (as shown in Figure 3.9).

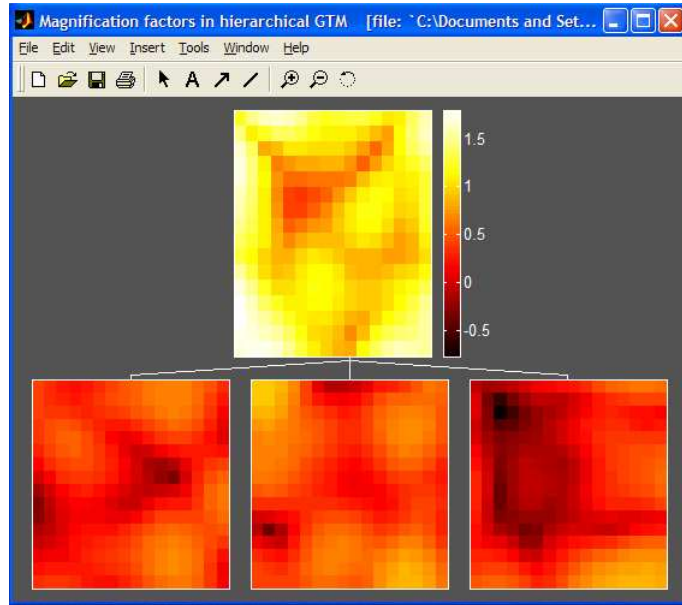


Figure 3.8: Corresponding Magnification Factor for the HGTM hierarchy

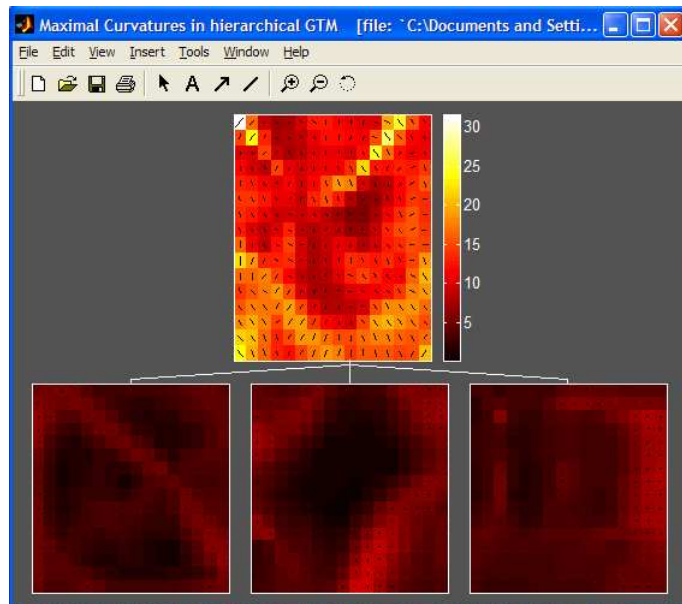


Figure 3.9: Corresponding Directional Curvature for the HGTM hierarchy

Chapter 4

Creating and using regression models

4.1 Introduction

Regression models supported by the DVMS v1.5 can be broadly divided into 2 categories, global and local regression models. Global models use a single model for the problem which covers the entire input space, while the local regression models use a combination of models, each of which works on a smaller part of the input space. Please note that the \$PREDICTION flag in the configuration file should be set to 1 for the creation and evaluation of a regression model.

4.2 Training and using a global model

DVMS v1.5 supports Generalised Linear Model (GLM) [10], Multilayer Perceptron (MLP) [2], and Radial Basis Functions network (RBF) [2] as the global models. Training a global model and evaluating it is very simple using DVMS v1.5. Using the ‘Global Exp’ menu on the DVMS main interface (figure 2.1) user can train a global model after loading the configuration and the data files. 30% of training set data is used for validation (to select the best possible model). At the end of the training, training set mean squared error (MSE) [2] & normalised mean squared error (NMSE) [2] and validation set MSE & NMSE are displayed. Once the model is trained, user can save it and evaluate its performance on the test data set.

4.3 Training and using a guided local regression model

The segmentation of input space in the guided local regression models, which we call Guided Mixture of Experts (GME) as it is based on the Mixture of Experts (ME) models, is obtained from the trained HGTM visualisation model. Once the configuration file, the data file, and the trained HGTM

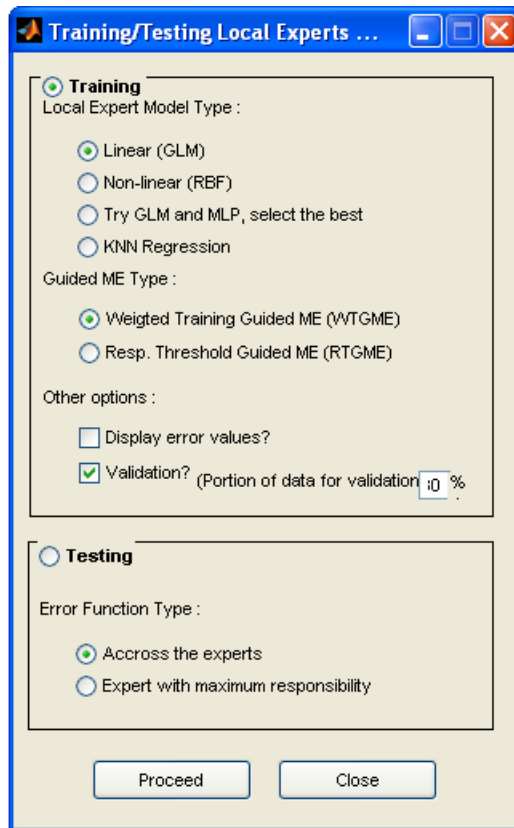


Figure 4.1: Interface to train and evaluate a guided local regression model

visualisation model is loaded successfully, the 'Local Exp' menu in the main DVMS v1.5 interface is used to train a GME. Figure 4.1 shows the interface for training and testing a GME.

Bibliography

- [1] A. C. Good, S. R. Krystek, J. S. Mason, High-throughput and virtual screening: core lead discovery technologies move towards integration, *Drug Discovery Today* 5 (2000) S61–S69.
- [2] C. M. Bishop, *Neural Networks for Pattern Recognition*, 1st Edition, Oxford University Press, 1995.
- [3] D. Lowe, M. E. Tipping, Neuroscale: Novel topographic feature extraction with radial basis function networks, *Advances in Neural Information Processing Systems* 9 (1997) 543–549.
- [4] C. M. Bishop, M. Svensén, C. K. I. Williams, GTM: The generative topographic mapping, *Neural Computation* 10 (1998) 215–234.
- [5] P. Tiño, I. T. Nabney, Constructing localized non-linear projection manifolds in a principled way: hierarchical generative topographic mapping., *IEEE T. Pattern Analysis and Machine Intelligence* 24 (2002) 639–656.
- [6] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, G. E. Hinton, Adaptive mixture of local experts, *Neural Computation* 3 (1991) 79–87.
- [7] M. E. Tipping, C. M. Bishop, Mixtures of probabilistic principal component analysers, *Neural Computation* 11 (2) (1999) 443–482.
URL citeseer.ist.psu.edu/tipping98mixtures.html
- [8] S. T. Roweis, L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding, *Science* 290 (2000) 2323–2326.
- [9] D. M. Maniyar, I. T. Nabney, Guided local regression using visualisation, *Lecture Notes in Computer Science*, Springer, Submitted.
- [10] I. T. Nabney, *Netlab: Algorithms for Pattern Recognition*, 1st Edition, Springer, 2001.
- [11] D. J. MacKay, *Information Theory, Inference, and Learning Algorithms*, Cambridge University Press, 2003.
- [12] P. Tiño, I. T. Nabney, Y. Sun, B. S. Williams, A principled approach to interactive hierarchical non-linear visualization of high-dimensional data, *Computing Science and Statistics* 33.

BIBLIOGRAPHY

- [13] F. Aurenhammer, "Voronoi diagrams - survey of a fundamental geometric data structure", *ACM Computing Surveys* 3 (1991) 345–405.