Visualization of High Dimensional Models within the SG++ Data Mining Pipeline

Vincent Bennet Bautista Anguiano

October 10, 2019

1 Abstact

Visualization is an essential aspect during a knowledge discovery process, since it provides the user with information which cannot be easily identified just by looking at raw numbers. The SG++ Datamining Pipeline is a component of the SG++ Toolbox used to generate such data mining models by utilizing sparse grids methods. Until recently, there was no way the end user could visualize these models and therefore a new visualization module was implemented. In this report, a description of these new implemented module will be given, explaining the approach taken when handling data on higher dimensions. Additionally, some visual results will be shown for density estimation and classification models for high dimensional data sets.

2 Introduction

The Spatially Adaptive Sparse Grid Toolbox SG++ is an open source library project for sparse grids methods and its combination technique originally created by Plfüger [1]. This library is comprised of several components, each one developed specifically to solve a specific kind of problem like Partial Differential Equation Solvers, Function Interpolation, Uncertainty Quantification etc.

One of these component is the SG++ Datamining Pipeline, a piece of software which enables the end user to generate data mining models using sparse grid methods in a simplified manner, in order words, without the user getting into the mathematical aspects of the sparse grids themselves. So far, the pipeline generates models for density estimation, classification and regression models and it is comprised of the following modules:

- Datasource: Module in charge of obtaining the data samples from a specified source.
- Fitter: Module in charge of training the specified model using sparse grids methods.
- Scorer: Module which delivers an accuracy score of the trained model after each iteration.
- Hyperparameter Optimizer: Module developed by Koepe [2] in charge of obtaining the optimal hyperparameters for a model through algorithms like Bayesian Optimization.

Until now, a way to visualize the output of the pipeline was not available to the end user. High dimensional data visualization is a challenging task, not only because of our inability of visualizing more than 3 dimensions, but also because conventional linear dimensionality reduction techniques (like PCA) fail to capture the non linear structure of the data, resulting in poor and misleading visualization results. In recent years, a handful of algorithms have been developed in order to circumvent this issue like t-SNE [3], Uniform Manifold Approximation and Projection [4], Isomaps [5] and LargeVis [6] among others.

In this report, the implementation of the new visualization module for the SG++ Datamining Pipeline will be presented along with two approaches taken to visualize high dimensional classification and density estimation models, which are generated by the pipeline. These approaches are the t Stochastic Neighbor Embedding (t-SNE) algorithm and a typical projection approach in which cuts made by planes at fixed values are plotted.

3 The t Stochastic Neighbor Embedding (t-SNE) for Visualization

This algorithm formulated by L. Van der Maaten and G. Hinton is a non linear dimensionality reduction algorithm designed with the purpose of visualizing data in high dimensions. A general overview of the algorithm will now be given. A more detailed description can be found at [3].

3.1 Basic Definitions

Let it be the sets $X \in \mathbb{R}^m$ and $Y \in \mathbb{R}^n$ with $n < m$, each with n datapoints. For each pair of data points $x_i, x_j \in X$ the following conditional probability distribution is defined:

$$p_{i|j} = \frac{\exp(-\frac{||x_i - x_j||^2}{2\sigma_i^2})}{\sum_{i \neq k} \exp(-\frac{||x_i - x_k||^2}{2\sigma_i^2})} \tag{1}$$

This distribution models the probability that point x_i chooses point x_j as its neighbor in $\mathbb{R}^m$, if x_i were to choose x_j randomly under a Gaussian centered in itself. To determine the variance σ_i, a binary search for the the best value of σ_i is done by the algorithm. This search is done such that the σ_i produces a distribution P_i with a predefined perplexity. This perplexity value is given by the user as an hyperparameter. The perplexity is formally defined as:

$$\text{Perplexity}(P_i) = 2^{H(P_i)} \tag{2}$$

with $H(P_i)$ being the Shannon entropy of P_i given by:

$$H(P_i) = -\sum p_{j|i} \log_2 p_{j|i} \tag{3}$$

This perplexity value can be interpreted as a smooth measure of the effective number of neighbors.

Based on the distribution $p_{i|j}$, the joint probability distribution is approximated by:

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2n} \tag{4}$$

This assures that $\forall p_{ij} p_{ij} > \frac{1}{2n}$ in case that outliers are present within the data.

For each set of data points $y_i, y_j \in Y$ the following joint distribution is defined

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{l \neq k} (1 + ||y_l - y_k||^2)^{-1}} \tag{5}$$

which models the probability that the point y_i chooses point y_j in $\mathbb{R}^n$.

The objective of this algorithm is to find the sets of data points in Y such that the Kullback-Leibler divergence of p_{ij} and q_{ij} is minimized. In other words:

$$\arg\min_Y \sum_i \sum_j p_{ij} \frac{p_{ij}}{q_{ij}} \tag{6}$$

which can be solved through gradient descent using the following gradient:

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + ||y_i - y_j||^2)^{-1} \tag{7}$$

3.2 Algorithm Speedup

Since calculating all p and q distributions requires $n(n-1)$ combinations, solving this problem has a running time bounded by $O(n^2)$. In order to speedup this algorithm, the following adjustments were designed and presented by Van der Maaten at [8]. Here, the most important aspects are presented.

3.2.1 Speeding up the p distributions calculations

Since the probabilities of points far away from point p_i will tend to go to zero only the the probabilities of the u nearest neighbors are calculated, with the rest being set to zero. Vaan der Maaten sets the number of nearest neighbors to be determined by the perplexity as $u = 3$ Perplexity

The way of obtaining the nearest neighbors is through a Vantage-Point tree. This data structure created by Yianilos[7] is a tree structure optimized for nearest neighbor search in metric spaces. The space is partitioned by choosing a position in the space (the vantage point) and splitting the points into ones that are close to the vantage point than a certian threshold τ and those who are not. Recursively applying this method, neighbors in the tree are highly likely to be neighbors in the space. A Vantage Point Tree has an approximate time cost complexity of $O(n \log n)$ for building it and one of $O(\log n)$ for executing a nearest-neighbor search, so this step has a total time complexity of $O(n \log n + u \log n) = O(n \log n)$. This is a significant improvement over the original $O(n^2)$

3.2.2 Speeding up the gradient calculation

Multiplying and dividing the gradient by $Z = \sum_{l \neq k}(1 + ||y_l - y_k||^2)^{-1}$ and simplifying we obtain:

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij}) q_{ij} Z (y_i - y_j) \tag{8}$$

which then can be separated into two independent parts

$$\frac{\delta C}{\delta y_i} = 4(F_{attr} + F_{rep}) = 4(\sum_j (p_{ij} q_{ij} Z (y_i - y_j)) + \sum_j (q_{ij}^2 Z (y_i - y_j))) \tag{9}$$

The first term F_{attr} can be easily be computed in $O(uN)$ using the same approach as with the p distributions. Computing the term F_{rep} naively is however still in $O(n^2)$. Vaan der Mateen developed a Barne-Huts approximation [9] in order to speedup the calculation to $O(n \log n)$.

This algorithm constructs a quadtree or octree of the current points' embedding. A quadtree is a tree data structure created by Finkel and Bentley [10], whose internal nodes have four children, each representing the partition of a 2D space into four rectangular cells. The octree is the quadtree's equivalent in 3D.

All of the points which belong to a certain cell are stored as children of the node that represents the cell. The cell is recursively partitioned until each cell contains only one data point. The nodes contain a summarized information of the cell, in this case, the center of mass of all of its children defined as y_{cell}.

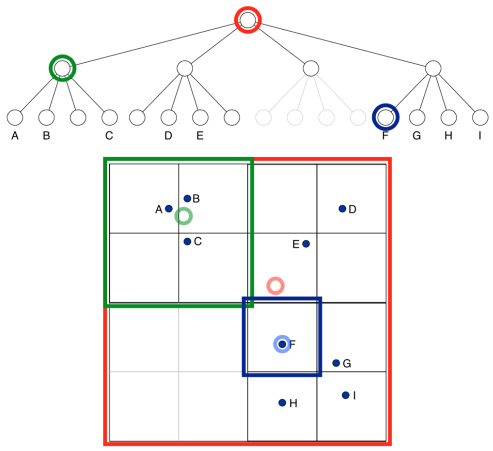


Figure 1: t-SNE Quadtree approximation. Taken from [8]. Each node contains the summarized information of its children. If the condition is fulfilled, the information at the parent node is taken to calculate the q probabilities for all of its children

Once the tree is constructed, the algorithm traverses the tree for each point y_i trough depth search and in each node it checks whether the current node can be use as a summary of the underlying cells to calculate the probability q_{ij}. The check is done by using the following expression:

$$\frac{r_{cell}}{||y_i - y_{cell}||} < \theta \tag{10}$$

Where θ is the hyperparameter between 0 and 1 that defines the level of approximation of the gradient and r_{cell} the radius of the cell. Note that if

$\theta = 0$ then the computation is reduced to the naive one. Figure 1 shows a graphical example of how this is achieved.

4 Visualization Module Implementation

A new module was implemented in order for the end user to obtain a graphical output of the model generated by the pipeline during the fitting process. This visualization module was implemented for density estimation and classification models but can be extended for other future ones.

The module does not generate a graphic file per se, but rather a file which can be used to generate one. So far the module delivers its output in two formats: csv and json.

Specifically, the json format serves as an input for the plotly graphic library. This opens source graphic library is available in Python, R, Matlab and Javscript and allows the user to generate interactive plots based on its json specification.

4.1 Output Types

The visualization module currently produces an output of the following types:

- t-SNE Compression: This output contains the information of the compressed training data using the t-SNE algorithm while being evaluated by the model.
- Density Estimation Heatmap: This output contains the information to generate a heatmap based on a density estimation model. If the model has three or more dimensions, a series of 2D cuts are generated per graph. The modules generates all possible combinations while taking maximum four dimensions at a time. The rest of the dimensions are set to a value of 0.5. If the output is json, the grid is also included in the output.
- Density Estimation Linear Cuts: This output contains the information to generate linear graphs based on a density estimation model. If the

model has two or more dimensions, a series of 1D cuts are generated per graph. The module generates all possible combinations while taking maximum 3 dimensions at a time. The rests of the dimensions are set to a value of 0.5.

- Classification Heatmap: This output contains the information to generate a heatmap based on a classification model. If the model has three or more dimensions, a series of 2D cuts are generated per graph. The modules generates all possible combinations while taking maximum four dimensions at a time. The rest of the dimensions are set to a value of 0.5. If the output is json, the grid per each class is also included in the output as also the data points used to train the model. In the case of the cuts, the data points are projected directly over the cut itself.

  As a side note, since the Fitter Module generates a density estimation model per class to obtain the classification model, so does the visualization module while generating outputs. Not only the Classification Heatmap and t-SNE graph is generated for the classification model but also the Density Estimation Heatmaps and Linear Cuts for each defined class.

4.2 Configuration Parameters

Similarly to the rest of the modules of the pipeline, the new visualization module is also configured from within a json config file. It is comprised of two main dictionaries:

- generalConfig: The configuration for general aspects of the module, like the output directory and the type of output. Table 1 shows a complete description of each parameter.

- parameters: The parameters to run the selected dimensionality reduction algorithm defined in the generalConfig. Table 2 shows a complete description of each attribute

Attribute Name	Valid value range	Default Value	Comments
algorithm	“tsne” plus other algorithms for visualization which could be implemented in the future	tsne	Algorithm to be executed to reduce the dimensionality of the model. (Note that at the current version of the pipeline the t-SNE algorithm is still in progress of integration)
targetDirectory	Absolute or relative path	./output	Path to the file in which the data will be stored after the algorithm is applied.
targetFileType	csv, json	csv	Format of the file in which to present the output of the visualization algorithm.
numBatches	$[1, \inf)$	1	Number which determine after how many batches the visualization module will be executed. Note that the first batch is always executed.

Table 1: Attributes for generalConfig

Attribute Name	Valid value range	Default Value	Comments
perplexity	$5, 50$	30	Perplexity value used to run the t-SNE algorithm.
theta	$(0, 1]$	0.5	Summarization threshold of a set of points used by the algorithm to approximate the probability distributions.
seed	$[0, \inf)$	csv	Format of the file in which to present the output of the visualization algorithm.
maxNumberIterations	$[1, \inf)$	1000	Maximum number of iterations in which the gradient descent will be run.
targetDimension	1,2,3	2	The number of dimensions to which the data will be reduced.

Table 2: Attributes for parameters

5 Data Set Description

Four datasets where used in order to test the new visualization module. Tables 3 and 4 show a general description of each of these.

Dataset Name	Model Type	# Points	# Classes
5D Gaussians Same variance	Density Estimation	1000 per Gaussian	N/A
5D Gaussians Different variance	Density Estimation	1000 per Gaussian	N/A
Two Moons Data Set	Classification	180	2
5D Gaussians 3 Classes	Classification	300 points per Gaussian	3

Table 3: Datasets description used in this run, part 1

Dataset Name	Characteristics
5D Gaussians Same variance	Gaussian means: (0,0,0,0,0) and (20,20,20,20,20) Covariance: Identity for both
5D Gaussians Different variance	Gaussian means: (0,0,0,0,0) and (20,20,20,20,20) Covariance: Identity and 10 times the identity respectively
Two Moons Data Set	
5D Gaussians 3 Classes	Gaussian means: Class 1: (0,0,0,0,0) Class 2: (-5,-5,-5,-5) Class 3: (5,5,5,5,5) Covariance: Identity

Table 4: Datasets description used in this run, part 2

Additionally, these were the parameters used in the config file while running the visualization module:

- Perplexity: 30
- Theta: 0.5
- Number of Iterations: 1000
- Random seed: 50
- Grid Level: 5 and 7 for the Two moon dataset
- Grid Level: 4 for the remaining datasets

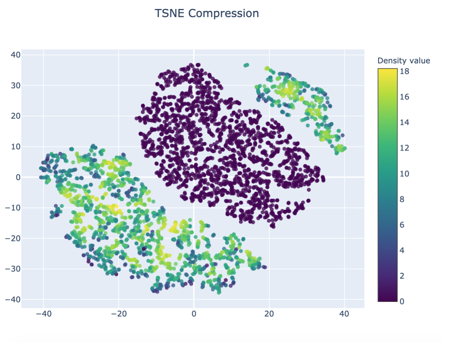
6 Test Results

The results obtained by running the module on the previously mentioned datasets are now presented in this section. Firstly, the results of the visualization of density estimation models will be presented, followed by the ones of the classification models.

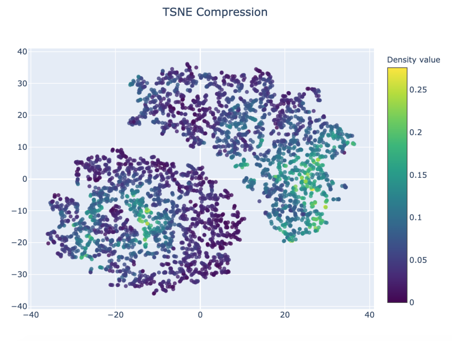
6.1 Density Estimation

Figure 2a shows the t-SNE compression of the dataset with the gaussians with different variance. It is curious to observe that even though the points of the second gaussian are more separately spaced in the high dimensional space due to the high variance, the t-SNE algorithm manages to identify them as another cluster. Only with the help of the model can the user identify that these points have little to no existent density in comparison the ones in the other gaussian and therefore play a smaller role in the fitting of the model.

On the other hand, Figure 2b shows the case where both gaussians have the same variance and indeed the t-SNE algorithm identifies the 2 separate Gaussians and with the help of the model the user can identify that the points have similar distributions and densities.

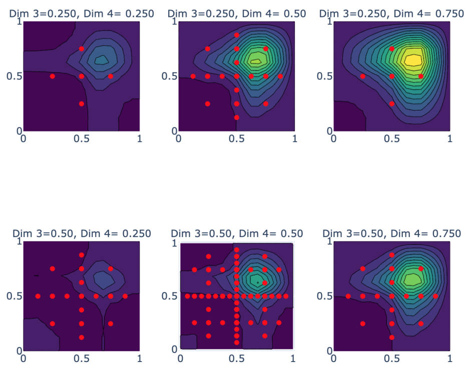


(a) Gaussians with different covariances

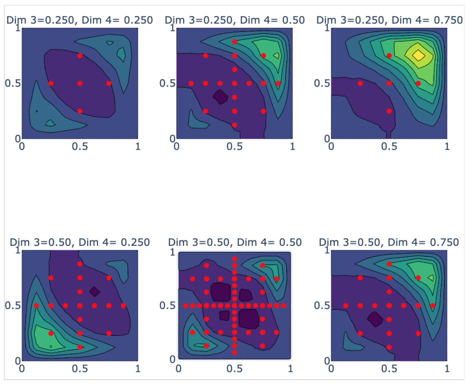


(b) 5D Gaussians with same covariances

Figure 2: t-SNE Compression for Density Estimation 5D Gaussians. Notice that with the help of the model evaluation, the end user can give himself a general idea how distributed the data points are in the high dimensional space. In Figure a, we can deduce that the points with low density value are more spaced in the high dimensional space than the points with high density value. On the other hand in figure b, we can deduce that the points of both clusters are similarly distributed.



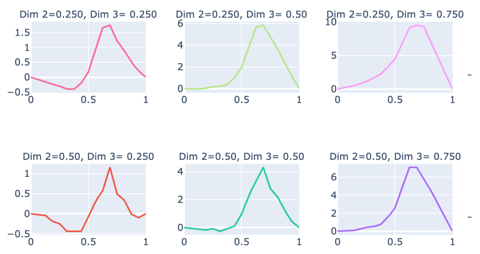
(a) 5D Gaussians with different variance



(b) 5D Gaussians with same covariances

Figure 3: Density Estimation Heatmaps. Figure a shows the presence of only one density region in the high dimensional space, while figure b shows the presence of 2 density regions of almost same value in the high dimensional space.

Figures 3a and 3b confirms these findings by taking a look into the density estimation heatmaps. Again, in the set with different variances there is only one area of density defined and in the set with same variances both areas of similar density appear.


(a) 5D Gaussians with different variance

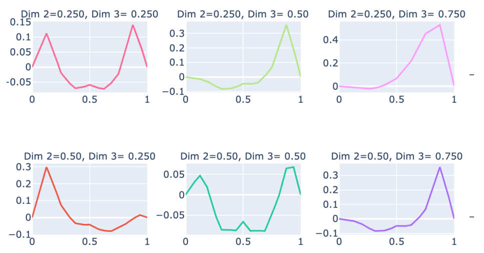

(b) 5D Gaussians with same covariances

Figure 4: Density Estimation Linear Cuts. Here, the results of the heatmaps are repeated albeit in finer detail. The end user can see the exact value of the densities and how much these vary across the space.

Figures 4a and 4b can help the user see and compare how similar the densities are for both gaussians. On the same variance dataset, the cuts shows for both gaussians a similar density value, while in the difference variance dataset only one area of great density appears.

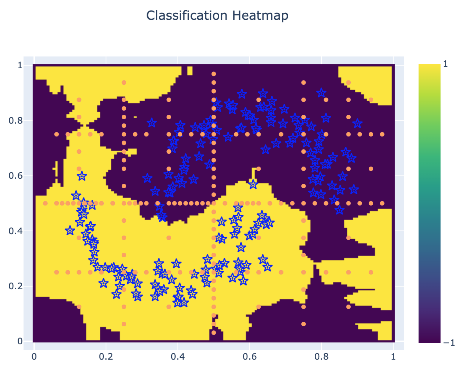
6.2 Classification

The results of the visualization of classification models will now be presented. This is split into 2 subsections due to the higher amount of graphs in comparison to the density estimation models.

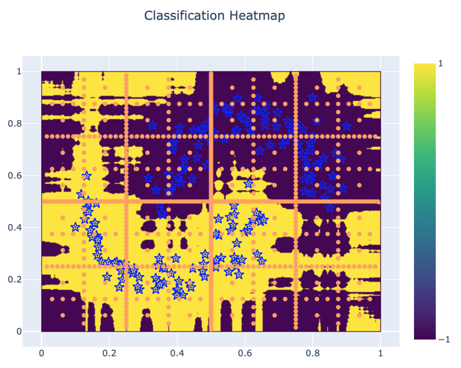
6.2.1 Two Moons Dataset

Figures 5a and 5b show the result of the classification heatmaps using different levels of the grid. It is of great importance to note that the higher the level of the grid the finer the resolution will be and the more precisely the class assigning will be in a certain region.

Figures 6a and 6b show the density estimation heatmaps for class -1 using the different grid levels while figures 7a and 7b show them for class 1. Note that both areas match exactly the ones in the classification and just as with the classification heatmaps, the resolution and detail increase with the use of a higher level grid.

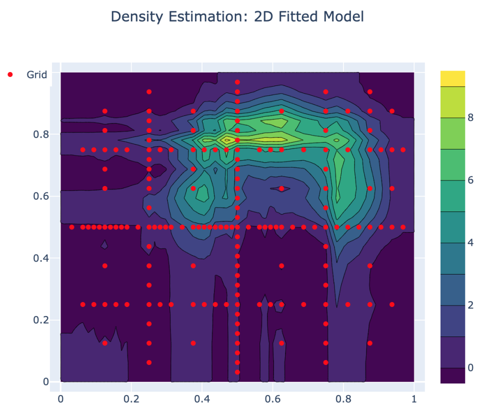


(a) Level 5 grid

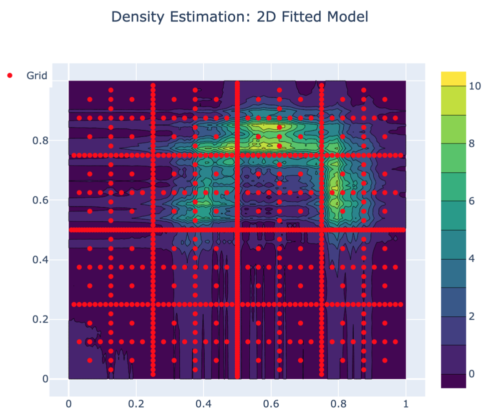


(b) Level 7 grid

Figure 5: Classification Heatmaps for the Two Moon Datasets. The higher the level, the finer the resolution of the heatmap and the more precisely the class distribution is

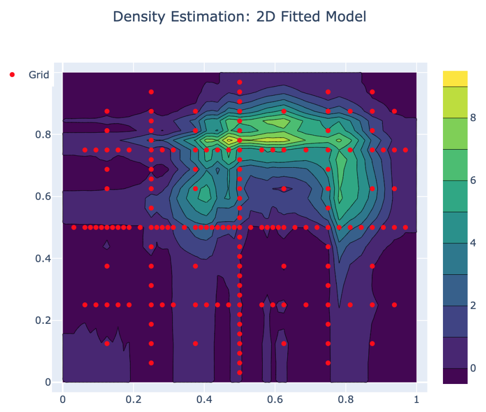


(a) Level 5 grid

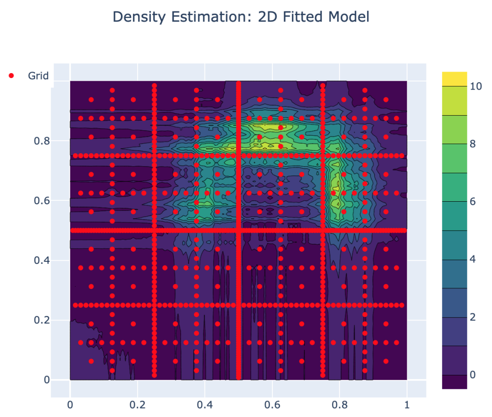


(b) Level 7 grid

Figure 6: Class -1 Density Estimation Heatmap. Notice that the areas of high density match the areas of the corresponding class in the classification heatmap.



(a) Level 5 grid



(b) Level 7 grid

Figure 7: Class 1 Density Estimation Heatmap. Notice that the areas of high density match the areas of the corresponding class in the classification heatmap.

6.2.2 5D 3 Class Gaussians

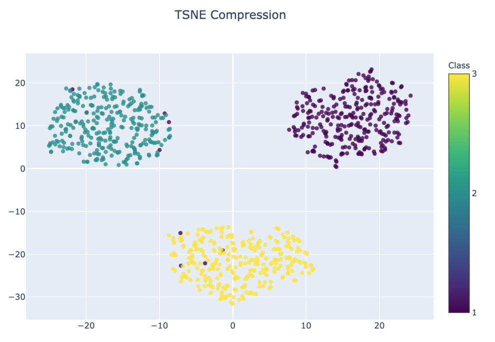
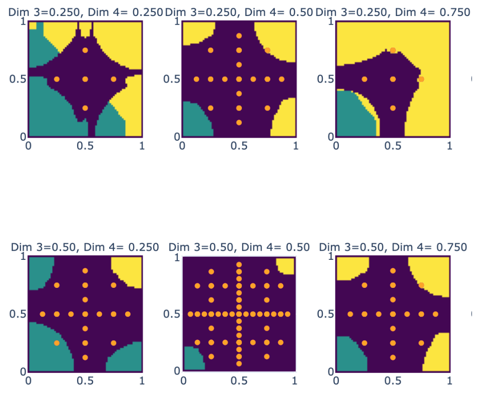
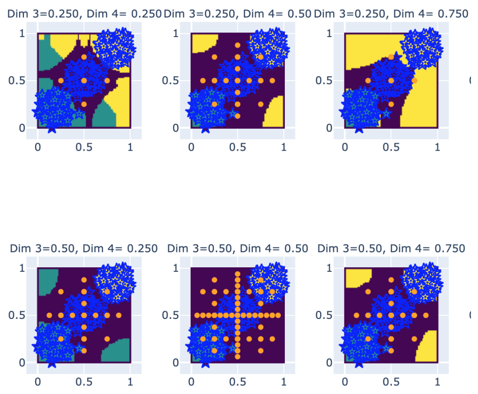


Figure 8: 5D 3 Class Gaussians: t-SNE Compression. Here, the algorithms separates the 3 clusters correctly and their assigned classes. It can be seen that some points are also misclassified, giving the end user a useful insight of the accuracy of the model.

Figure 8 shows the t-SNE compression graph result. The algorithm manages to identify the 3 separate gaussians though one can see that some points appear as misclassified from the model. This can be really useful for the end user to visually appreciate the accuracy of the models and tune the respective parameters to obtain better results.



(a) Heatmaps without data points

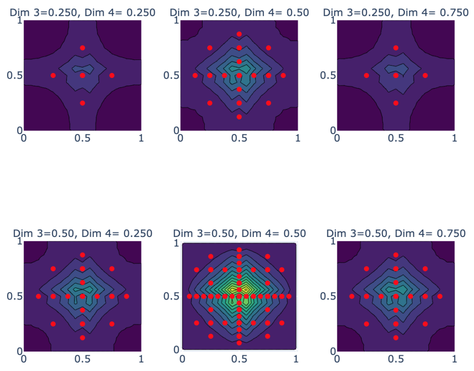


(b) Heatmaps with data points

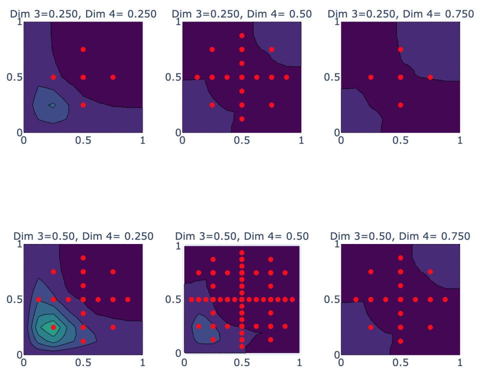
Figure 9: 5D 3 Class Gaussians: Classification Heatmaps. Again we see the classification areas are correctly distributed through the space.

Just like in the previous sections figures 9a and 9b show the Classification Heatmaps for the model. Since this is a 5D model, 30 different graphs where generated for this graph. The one shown in the figures corresponds to the one in which the dimension 5 is kept fixed to a value of 0.5 and the plane cuts are defined by dimension 3 and 4, while each graph maps dimension 1 against dimension 2.

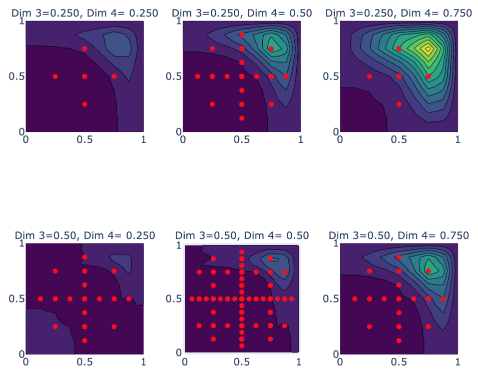
Interesting to note here is the presence of the data points in the cuts. This can help the user to determine in which region of the high dimensional space the points are if their class matches with the class's region in a certain cut.



(a) Class 1



(b) Class 2



(c) Class 3

Figure 10: 5D 3 Class Gaussians: Density Estimation Heatmaps. Each density estimation heatmap matches its corresponding class in the classification heatmap. Additionally, comparisons can be made between densities to understand why a class was selected within a certain region.

Figures 10c, 10b and 10c show the density estimation heatmaps for each class using the same dimensions configuration. Again, the areas match to the ones in the classification heatmap. Comparisons can also be made between densities to understand why a class was selected within a certain region.

7 Conclusions and Future Work

From the previous sections it can be seen that the results provide useful insights to end user about the models generated by the library. There are however the following issues in which further work needs to be done:

- The linear cuts and heatmaps are currently being generated for all possible combinations of dimensions. This is extremely problematic when handling high dimensions since the complexity is in $O(d!)$ with d being the number of dimensions.
- Even though t-SNE generates really accurate results, the influence of the sparse grid points cannot be shown since the algorithm's objective is to find a mapping that best resembles the original internal structure of the high dimensional data and the grid points are not part of this.
- The t-SNE algorithm, even with its speedup, does not scale exactly well when handling extremely large data sets.
- Using higher grid levels, the heatmap generation, specially the classification heatmap, slows down to a significant degree. The main reason is that the resolution increase exponentially with the level and so the number of density estimation models to evaluate. This becomes worse if there is a significant number of classes within the data.
- The approaches taken here are not applicable to regression models, due to the fact that in these models other attributes are expected to be visualized, like the statistical features of them. New approaches need to be considered

Additionally to this issues, an exploration of other dimensionality reduction algorithms could also be done so that the end user has flexibility in selecting the one that better suits its needs. Some suggestions are:

- Self Organizing Maps
- Uniform Manifold Approximation and Projection
- Isomaps
- LageVis

Nevertheless, the results of this first implementation is a good base from where future research can be conducted and hopefully with better results.

References

[1] D. Pflüger(2010), *Spatially Adaptive Sparse Grids for High-Dimensional Problems*, Institut für Informatik, Technische Universität München

[2] E. J. Koepe(2018) *Optimizing Hyperparameters in the SG++ Datamining Pipeline*, Department of Informatics, Technische Universität München

[3] L. Van der Maaten, G. Hinton (2008) *Visualizing Data using t-SNE*, Journal of Machine Learning Research, 9, 2579-2605

[4] L. McInnes, J. Healy, J. Melville *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*, Tutte Institute for Mathematics and Computing

[5] J. B. Tenenbaum, V. de Silva, J. C. Langford (2000) *A Global Geometric Framework for Nonlinear Dimensionality Reduction*, Science, 290, 2319-2322

[6] J. Tang, J. Liu, M. Zhang, Q. Mei (2016) *Visualizing Large-scale and High dimensional Data*, Microsoft Research Asia, Pekin University

[7] P. N. Yianilos(1993), *Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces*

[8] L. Van der Maaten (2014), *Accelerating t-SNE using Tree-Based Algorithms* , Journal of Machine Learning Research, 15, 3221-3245

[9] J. Barnes and P. Hut (1986), *A hierarchical O(N log N) force-calculation algorithm* Nature, 324 (4), 446–449

[10] R. A. Finkel, J. L. Bentley (1974) *Quad Trees A Data Structure for Retrieval on Composite Keys*, Acta Informatica, 4, 1-9