INTER-DISCIPLINARY PROJECT

Estimation of Probability Density Functions and Graphical Models using regularized Sparse Grids.

Karthikeya Sampa Subbarao

Technical University of Munich, Department of Informatics
Boltzmannstr. 3, D-85748 Garching bei Munich, Germany
karthikeya108@gmail.com

Advisor: **Valeriy Khakhutskyy**

Examiner: **Univ.-Prof. Dr. Hans-Joachim Bungartz**

May 2015

Contents

Abstract

In this paper, we study the application of sparse grid based methods for estimation of probability density and also device a method to coarsen the grid by identifying and removing, less important ANOVA components from the model. This process also reveals the underlying structure of the dataset. We employ Markov Chain Monte Carlo sampling for estimating the expected values of sufficient statistics of the model.

1 Introduction

Suppose we have a dataset $D = \{x_1, x_2, .., x_M\}, where x_i \in \mathbf{R}^d$ drawn from an unknown distribution $p(X)$ of a random variable X. Density estimation is the construction of an estimate $\hat{p}$ of the probability density function p based on the dataset D. Density estimation has always been a very important problem in statistics and data mining [14]. Density estimation can be used in exploration and presentation of data and can be used to solve various types of problems. Classification and Sampling are two such applications.

Density estimation methods can be parametric or nonparametric. Parametric density estimation methods assume that the form of the underlying distribution is known whereas nonparametric density estimation methods only use given data samples. There are various nonparametric density estimation methods [8]. Kernel density estimation (KDE) is the most widely used nonparametric density estimation method. The (one-dimensional) estimator $\hat{p}(x) = 1/M \sum_{i=1}^{M} K((x - x_i)/h)$ is a linear combination of kernel functions K centered at the datapoints $x_i \in D$. The performance of the estimator depends on the choice of the kernel function K and the bandwidth $h > 0$ and the selection of the bandwidth is a non-trivial task.

Despite the fact that multivariate kernel density estimation is an important technique in multivariate data analysis and has a wide range of applications, its performance worsens exponentially with high dimensional data sets, this phenomenon is called *Curse of Dimensionality*, where there is exponential growth in combinatorial optimization as the dimension of the dataset increases [1]. For thorough understanding of density estimation and its applications we refer to [12].

1.1 Grid-based Density Estimation

In Kernel Density Estimation, the evaluation of $\hat{p}(x)$ depends on the number M of data points D. Thus, in order to evaluate the estimated density function $\hat{p}(x)$, all M kernel functions centered at all data points have to be evaluated. One remedy is to divide the data into a small number of bins and place a kernel function at each bin (which is also called 'gridding the data'). However, the number of bins increases exponentially with the dimension of the data points. In practice, kernel density estimation is often limited to, say, four dimensions. Note that approaches based on Fast Fourier Transforms (FFT) also rely on grids and thus are hardly considered in more than four dimensions [3].

Density estimation methods like KDE are sensitive with respect to the parameters and have long runtime for large datasets. Grid based methods overcome these limitations to a certain extent. However, similar to KDE, full grid approach also suffers from *Curse of Dimensionality*. The number of grid points in the full grid, grows exponentially with the dimension of the data points. For more details on the grid-based methods, we refer to [11].

1.2 Sparse-Grid-based Density Estimation

As mentioned earlier, estimation of density using Grid based methods also suffer from the curse of dimensionality for higher dimensional problems. Hence we switch

to a hierarchical grid instead of an equidistant grid. It is shown that we can construct a grid which retains the high accuracy of the full mesh grid with much less grid points. L2 regularised density estimation model has been developed using the Sparse Grid techniques which has proven to be effective in dealing with the curse of dimensionality [11].

Sparse grid method is a special discretization technique. It is based on a hierarchical basis, a representation of a discrete function space which is equivalent to the conventional nodal basis, and a sparse tensor product construction. For the representation of a function f defined over a d-dimensional domain the sparse grid approach employs $\mathcal{O}(h_n^{-1} \cdot log(h_n^{-1})^{d-1})$ grid points in the discretization process, where n is the discretization level and $h_n = 2^{-n}$ denotes the mesh size [2]. In depth details of the topic can be found in [6].

For clear understanding of the concept of basis functions, we refer to the comprehensive description provided in [11].

1.2.1 Hierarchical Basis

Let $\mathcal{V}_l$ be the sparse grid space of level l corresponding to the domain $\Omega = [0,1]^d \in \mathbf{R}^d$, which consists of the so called hierarchical basis. In the standard case, piecewise linear hat functions are used as basis functions. One-dimensional standard hat function $\varphi : [-1,1] \to \mathbf{R}$ is defined as

$$\varphi(x) = max(1 - |x|, 0). \tag{1}$$

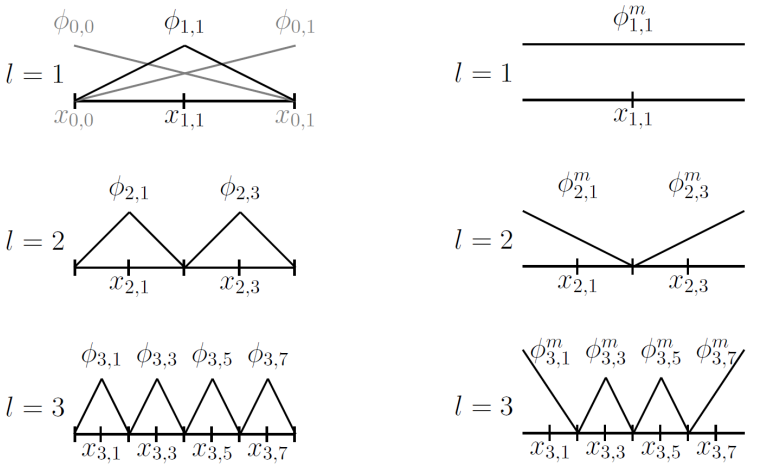
The one-dimensional hierarchical hat function $\varphi_{l,i}$ centered at the grid point $x_{l,i} = i.2^{-l}$ is the result dilation and translation of φ,

$$\varphi_{l,i}(x) = \varphi(2^l x - i). \tag{2}$$

We extend the hat function to d-dimensional case by using a tensor product approach

$$\varphi_{\mathbf{l},\mathbf{i}}(x) = \prod_{j=1}^{d} \varphi_{l_j,i_j}(x_j), \tag{3}$$

where $l = (l_1, ..., l_d)$ and $i = (i_1, ..., i_d)$ denote the level and index respectively. The corresponding grid point $x_{l,i} = [x_{l_1,i_1}, ..., x_{l_d,i_d}]^T$ is the center of the support of $\varphi_{\mathbf{l},\mathbf{i}}$.



(a) hierarchical basis (b) modified/nonuniform basis

Figure 1: Hierarchical Basis Functions for level $l = 1$ to $l = 3$ are shown in (a). Modified or nonuniform basis functions are shown in (b). Source: [11].

1.2.2 Modified Linear Basis

We employ Sparse Grid with a modified linear basis for our density estimation method, since it has a correspondence to the Analysis of Variance (ANOVA) like decomposition. Above figure shows both hierarchical and modified or nonuniform basis functions. These basis functions extrapolate towards the boundary and so do not require boundary points to approximate functions with non-zero values at the boundary. The hierarchical basis function $\varphi_{l,i}$ is modified in the following way

$$\varphi_{l,i}^{m}(x) = \begin{cases} 1 & \text{if } l = 1, i = 1, \\ \left\{ \begin{array}{cc} 2 - 2^l \cdot x & \text{if } x \in [0, 2^{-(l-1)}] \\ 0 & \text{else} \end{array} \right\} & \text{if } l > 1, i = 1, \\ \left\{ \begin{array}{cc} 2^l \cdot x + 1 - i & \text{if } x \in [1 - 2^{-(l-1)}, 1] \\ 0 & \text{else} \end{array} \right\} & \text{if } l > 1, i = 2^l - 1, \\ \varphi(x \cdot 2^l - i) & \text{else.} \end{cases} \tag{4}$$

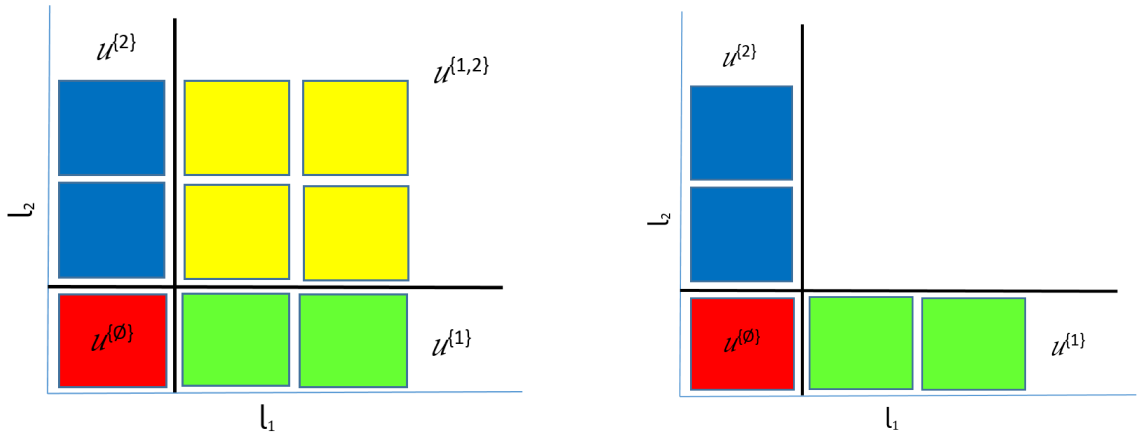
1.3 Analysis of Variance

We consider a decomposition of the d-dimensional function h as

$$\begin{aligned} h(x_1, ..., x_d) = h_0 + \sum_{j_1}^{d} h_{j_1}(x_{j_1}) + \sum_{j_1 < j_2}^{d} h_{j_1,j_2}(x_{j_1}, x_{j_2}) \\ + \sum_{j_1 < j_2 < j_3}^{d} h_{j_1,j_2,j_3}(x_{j_1}, x_{j_2}, x_{j_3}) + \cdots + h_{j_1,...,j_d}(x_{j_1}, ..., x_{j_d})). \end{aligned} \tag{5}$$

Where, h_0 is a constant function, h_{j_1} are one-dimensional functions, h_{j_1,j_2} are two-dimensional functions, and so on. This type of decomposition is known in statistics under the name analysis of variance (ANOVA) which is used to identify important variables and important interactions between variables in high-dimensional models. Note that (5) is a finite expansion of h into 2^d different terms. Such a decomposition can be gained by a tensor product construction of a splitting of the one-dimensional function space into its constant subspace and its remainder. For the detailed understanding of the topic we refer to [4].

Here we are interested in the decomposition of a sparse grid function $u \in \mathcal{V}_l$ (sparse grid space) with respect to a subspace $\tilde{\mathcal{H}}_{\mathcal{K}}$, the hierarchical subspaces of the sparse grid structure. This means, the constant function h_0 in (5) is replaced by a function $\tilde{u} \in \tilde{\mathcal{H}}_{\mathcal{K}}$ as well as all further functions $u_{j_1}, u_{j_1,j_2}, ...$ are constructed with respect to $\tilde{\mathcal{H}}_{\mathcal{K}}$. The figure 2($a$) shows the two-dimensional ANOVA-like decomposition with respect to $\tilde{\mathcal{H}}_{(2,2)}$.



(a) (b)

Figure 2: The two-dimensional ANOVA-like decomposition with respect to $\tilde{\mathcal{H}}_{(2,2)}$: The bold cross indicates which hierarchical increments belong to which component. Figure (b) shows the decomposition after deletion of the components corresponding to a factor.

1.4 Coarsening the Grid

Now we will look into a method which can further improve the efficieny of density estimation. As stated earlier, the sparse grirds use much less grid points than the full grid. As an extension of the hypothesis, we might be able to achieve the same level of accuracy by further coarsening the grid, i.e., to delete the grid points which are less important. This can be achieved by formulating a relationship between the grid points and ANOVA components of the model. The existence of a relation between sparse grids and dimension decomposition has been studied and sparse grid approach to dimension decomposition which works similar to ANOVA decompositions has been shown in [7]. In this project we employ sparse grid with modified basis which corresponds to ANOVA like decomposition and further associate the components with the factors of a factor graph. Thus we formulate an approach to compute, analyze and visualize ANOVA components which will help us to identify unwanted or less important components and remove them in order to reduce the complexity of the problem and hence improving the computational efficiency. The figure $2(b)$ shows the structure after removing grid points corresponding to one of the components from the grid.

2 Density Estimation using Sparse Grid

Suppose we have an initial guess p_ϵ of the density function underlying the data, $D = \{x_1, ..., x_M\}$ and let f resemble a member of exponential family, we look for $\tilde{p}$ in the function space $\mathcal{V}$ such that

$$\tilde{p} = argmin_{f \in \mathcal{V}} \int_\Omega (f(x) - p_\epsilon(x))^2 dx + \lambda || \wedge f||^2_{L^2}. \tag{6}$$

The left term of (6) makes sure that $\tilde{p}$ stays close to p_ϵ, the right term $|| \wedge f||^2_{L^2}$ is a regularization term which imposes smoothness constraint. For instance, $\wedge$ can be chosen to be ∇. The regularization parameter $\lambda > 0$ controls the trade-off between fidelity and smoothness.

Let Ψ_l be the set of hierarchical basis functions of the sparse grid space $\mathcal{V}_l$ of level l. If we set $p_\epsilon = \frac{1}{M} \sum_{i=1}^{M} \varphi(x_i)$, we obtain after some transformations (refer [10]), the variational equation of the form

$$\int_{\Omega} \varphi(x)\hat{f}(x)dx + \lambda \int_{\Omega} \wedge \hat{f}(x) \cdot \wedge \varphi(x)dx = \frac{1}{M}\sum_{i=1}^{M} \varphi(x_i), \tag{7}$$

which holds for all $\varphi \in \Psi_l$ and $\hat{f}(x) = \frac{exp(f(x))}{\int exp(f(z))dz}$.

We discretize the function $f(x)$ using sparse grid discretization technique. The function $f(x)$ as a linear combination of basis functions is written as follows:

$$f(x) = \sum_{j=1}^{N} \alpha_j \varphi_j(x), \tag{8}$$

where N is the number of grid points. α represents the coefficients and $\varphi(x)$ represents the basis functions which is defined in section 1.2.

We use Maximum a Posteriori approach to estimate the density and we adapt the modified version of the algorithm described in [5]. One (k^{th}) iteration of the algorithm in the mathematical form is given as

$$\sum_{j=1}^{N} \alpha_j^{i+1} a(\varphi_k, \varphi_j) = \frac{1}{M}\sum_{i=1}^{M} \varphi_k(x_i) - \int \varphi_k(x) \frac{exp(\sum_{j=1}^{N} \alpha_j^i \varphi_j(x))}{\int exp(\sum_{j=1}^{N} \alpha_j^i \varphi_j(z))dz} dx. \tag{9}$$

We can write (9) in the Matrix form as follows,

$$A_{k,j}\alpha^{i+1} = E_{empir} - \Phi(\alpha^i). \tag{10}$$

Where A represents the Regularization operator

$$A_{k,j} = a(\varphi_k, \varphi_j) = \lambda \int \bigtriangledown \varphi_k(x) \bigtriangledown \varphi_j(x)dx. \tag{11}$$

And E_{empir} can be computed using the definition of the basis functions

$$E_{empir} = \frac{1}{M}\sum_{i=1}^{M} \varphi_k(x_i). \tag{12}$$

The challenge lies in solving the following nonlinear part, which represents the expected value of the function $\varphi_k(x)$,

$$\Phi(\alpha^i) = \int \varphi_k(x) \frac{exp(\sum_{j=1}^{N} \alpha_j^i \varphi_j(x))}{\int exp(\sum_{j=1}^{N} \alpha_j^i \varphi_j(z))dz} dx. \tag{13}$$

We employ Markov Chain Monte Carlo (MCMC) sampling for computing this nonlinear term. Different approaches to solve this term, like Monte Carlo Integration has been evaluated in [13].

2.1 Computating the Expected Value using MCMC

The nonlinear term (13) can be written as follows:

$$\Phi(\alpha^i) = \int \varphi_k(x) l(x) dx. \tag{14}$$

For the sake of simplicity let us rewrite the equation as

$$\Phi(\alpha^i) = \int g(x) l(x) dx, \tag{15}$$

where $l(x) = \frac{exp(\sum_{j=1}^{N} \alpha_j^i \varphi_j(x))}{\int exp(\sum_{j=1}^{N} \alpha_j^i \varphi_j(z))dz}$ which gives the probability with which we need to sample the input values x for the function $g(x)$. Hence, (15) actually represents the expected value of $g(x)$ and can be written as

$$E[g(x)] = \int g(x)l(x)dx. \tag{16}$$

Following are the methods we employed to compute this expected value:

- We can compute the denominator of $l(x)$, i.e., $\int exp(\sum_{j=1}^{N} \alpha_j^i \varphi_j(z))dz$ using Monte Carlo Integration and use the value to compute $E[g(x)]$, which is also done with Monte Carlo Integration. This method has high runtime and hence becomes infeasible for high dimensional datasets [13].

- We can sample values from $l(x)$ using Markov Chain Monte Carlo (MCMC) by creating a model based on the Sparse Grid structure and assuming particular distribution (preferable Uniform) for the random variables. Once we sample the values from $l(x)$ using MCMC, we can evaluate the function $g(x)$ on these values and compute the mean which will be the expected value of the function $g(x)$. For computing the expected value of the function iteratively based on the updated model, we can use the sample values obtained in the previous sampling run. We can either store the sampled values and use them to initiate the sampling process in the current iteration, or we can sample few new points and compute the expected value by reusing the output of the function values from the previous iteration. Either way it should result in faster convergence of the sampling run.

2.2 Estimating the coefficients of the basis functions using MAP approach

Having discussed all the fundamental concepts, let us have a look at the basic algorithm for estimating the density underlying the given dataset. Given a d-dimensional dataset of length n and a modlinear grid, we are required to estimate the coefficients corresponding to each grid point.

At first we initialize the parameters ω, the learning rate, ϵ, a measure to compare the residual and decide when to stop the algorithm and $imax$ which represents the maximum number of iterations allowed, which in our case we set to the number of grid points.

Upon obtaining these coefficients, i.e., the α-vector, we estimate the density of the data points using,

$$d(x) = \exp\left(\sum_{i=1}^{n} \alpha_i \varphi_i(x)\right). \tag{17}$$

Algorithm 1: Estimating the coefficients of the basis functions using MAP approach

Data: ModLineaerGrid, Coeffecients, Set of DataPoints $X = x_1, .., x_M$
Result: Coefficients corresponding to each basis function of the grid
Choose Parameters $\omega > 0$, $\epsilon > 0$ *and* $i_{max} = Grid\ Size$;
Calculate q;
Calculate A;
Calculate $\Phi(\alpha^0)$ as describe in section 2.1;
while *residual* $> \epsilon$ *and* $i <= i_{max}$ **do**
  $b = q - \Phi(\alpha^i)$;
  Solve $A = \tilde{\alpha} b$;
  $\alpha^{i+1} = \alpha^i + \omega\tilde{\alpha}$;
  Calculate $\Phi(\alpha^{i+1})$ as describe in section 2.1;
  residual $= \|A\alpha^{i+1} - q + \Phi(\alpha^{i+1})\|$;
  $i = i + 1$;
end

3 Performance Improvement Strategies

3.1 Dynamic Learning Rate using Armijo Line Search

Even though having an assumed and fixed learning rate for the estimation of coefficients gave us good results, it would not suit for all kinds of problems and would take more time to converge. In order make the algorithm more efficient and reliable, we employ Armijo type line search to determine the learning rate at every iteration.

The problem that we are trying to solve is a minimization problem where we are trying to minimize the objective function which is the negative log likelihood of the given dataset. We employ Newton-Raphson procedure for minimizing the objective function. This involves determining the descent direction and the step size of the descent. Once we obtain the descent direction for our objective function, we need to pick the right step size. Taking a step too large might result in the function value being greater than the current value or if the step size is too small it might take forever to converge. Armijo's condition basically suggests that the 'right' step size is such that we have a sufficient decrease in the function value at the new point. This is mathematically represented as follows:

$$t(x_k + \omega p_k) \leq t(x_k) + c_1 \omega \nabla t_k^T p_k \tag{18}$$

where t is the function we are trying to minimize, ω is the learning rate and p_k is the descent direction at x_k and $c_1 \in (0, 1)$.

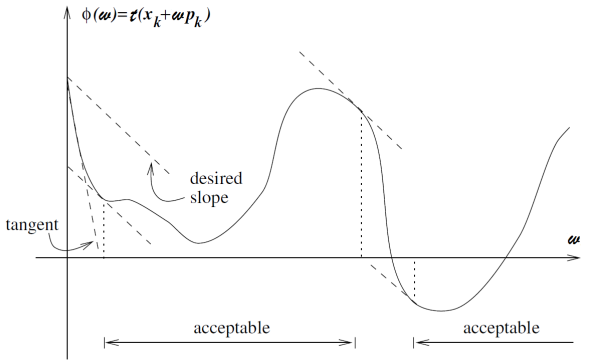


Figure 3: Armijo Condition for choosing the right step size in the descent direction. Source: [9].

The reduction in t should be proportional to both the step length ω and the directional derivative $\nabla t_k^T p_k$. The inequality (18) is sometimes called Armijo condition. For detailed understanding of the concept we refer to [9].

3.2 Grid Coarsening Strategies

As mentioned earlier in section 1.2.4, we can improve the performance of our algorithm by coarsening the grid. Now we discuss the strategy to associate the grid structure with ANOVA-like components and a method to identify the components which are less important to our model. We use *factor graph* to model the dependencies among the variables (dimensions). Initially we model the dependency of all the variables being dependent on one another by creating a fully connected factor graph. Then gradually based on the estimated coefficient values, unwanted factors are removed from the factor graph and the grid is coarsened accordingly. This will reduce the computational complexity of estimating the coefficients. Now we describe a strategy for identifying the unwanted or less important factors.

3.2.1 Mean Coefficient Thresholding

First step is to estimate the coefficients of all the grid points. Then we fetch and store the grid point index and corresponding factors that it is involved with. For each such factors, we list all the corresponding coefficients. If the mean of the coefficients is less than some *chosen threshold* value, then the factor is marked to be deleted. Once we have evaluated all the factors, we delete each factor in the delete list if there are no higher order factors that have this factor as a subset.

3.2.2 Choosing the Coefficient Threshold

Choosing the threshold as a decision measure for deleting the dependencies from the model is non-trivial. However, experiments with few artificial datasets with known co-variance and structure helped in determining a strategy to calculate the threshold. Mean of the absolute values of all the coefficients, estimated in the very first execution of our algorithm turned to be a good choice. Similar choices with slight variations like dynamically updating the threshold at every iteration with the mean of the absolute values of the estimated coefficients, also performed well for some datasets. Algorithm 3 in section 3.3 provides implementation details.

3.2.3 Updating the Grid

As and when we throw away components from the factor graph / ANOVA components and drive the corresponding coefficient values to zero, we also need to update the grid accordingly by deleting the corresponding grid points. This reduction in the grid contributes to the improvement in the performance of the algorithm. Entire logic which controls the deletion of the components in the model is laid out in the form of a factor graph and driven using the coefficient values which corresponds to the components in factor graph. Hence it is possible to make sure that crucial grid points are not deleted when there are higher level grid points present in the grid; which otherwise might lead to inconsistency in the grid. Algorithm 4 in section 3.3 provides implementation details.

3.3 Algorithms

In this section we provide the algorithms for the concepts described earlier. Algorithm 2 is the improved form of Algorithm 1 descibed earlier.

Algorithm 2: Estimating the coefficients of the basis functions - 'Newton-Raphson' procedure for penalized maximum likelihood

Data: $grid$, α, $factor\_graph$, $data$ `// Modified Linear Basis Grid`
Result: α corresponding to each basis function
Initialize Parameters: $\epsilon > 0 \, and \, i_{max} =$ size of $grid$
Calculate A
Compute E_{empir}
Calculate model expected values: $\Phi(\alpha^0)$ (section 2.1)
Compute $b = E\_empir - \Phi(\alpha^0) - A \cdot \alpha$
while $||\nabla\mathcal{L}|| > \epsilon$ *and* $i <= i_{max}$ **do**
  Solve $A = \tilde{\alpha}b$
  choose ω (section 3.1) such that $t(x_k + \omega p_k) \leq t(x_k) + c_1\omega\nabla t_k^T p_k$
  $\alpha^{i+1} = \alpha^i + \omega * p_k$
  Calculate model expected values: $\Phi(\alpha^{i+1})$ (section 2.1)
  Compute $\nabla\mathcal{L} = E_{empir} - \Phi(\alpha^{i+1}) - A \cdot \alpha$
  $grid$, α = `grid_coarsening(`$grid$, α, $factor\_graph$`)`
  if $grid$ updated **then**
    Update Coefficient vector, retaining only non zero values
    Calculate A
    Compute E_{empir}
    Calculate model expected values: $\Phi(\alpha^i)$ (section 2.1)
    Compute $\nabla\mathcal{L} = E\_empir - \Phi(\alpha^i) - A \cdot \alpha$
  end
  $i = i + 1$
end

Algorithm 3: Coefficient Thresholding

Procedure `coefficient_thresholding(`$grid$, α, $factor\_graph$`)`
  Set $alpha\_threshold$ = mean of $|\alpha|$
  for $grid\_point$ *in* $grid$ **do**
    for d in $dimension$ **do**
      if $level(d)$ of the $grid\_point$ $!= 1$ **then**
        $factor = factor + (d)$
      end
    end
    $factor\_alpha\_collection[factor] = \alpha[grid\_point]$
    for $factor$, $alphas\_of\_factor$ in $factor\_alpha\_collection$ **do**
      if mean of $|alphas\_of\_factor| < alpha\_threshold$ **then**
        add $factor$ to $delete\_list$
      end
    end
  end
  for $factor$ in $delete\_list$ **do**
    if $factor$ not a subset of higher order factors **then**
      delete the $factor$ from the $factor\_graph$
    end
  end
  return $factor\_graph$

Algorithm 5: Grid Coarsening

```
Procedure grid_coarsening(grid, α, factor_graph)
    factor_graph = coefficient_thresholding(grid, α, factor_graph)
    for grid_point in grid do
        for d in dimension do
            if level(d) of the grid_point != 1 then
                factor = factor + (d)
            end
        end
        if factor not in factor_graph then
            α[grid_point] = 0
            delete grid[grid_point]
        end
    end
    return grid, α                                  // returns coarsenend grid
```

4 Experiments and Results

4.1 Validation of results

In order to validate the estimated values of the Coefficients, we compute the true values of the Coefficients and compare them with the estimated values.

Toy DataSet (1D):

A Toy Dataset generated using Numpy. Normalized One Dimensional Dataset with:

Mean : 0.5
Variance : 0.1

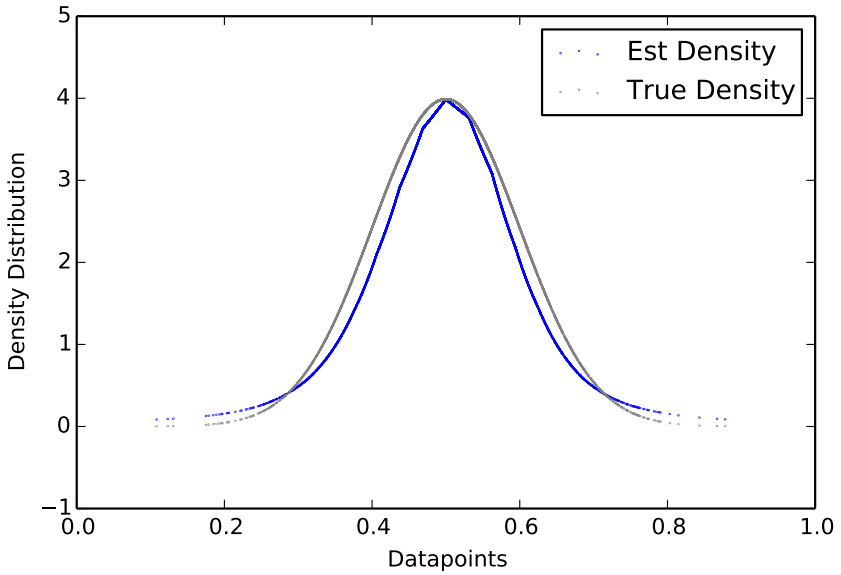


Figure 4: True and Estimated distribution of 1D Normal dataset

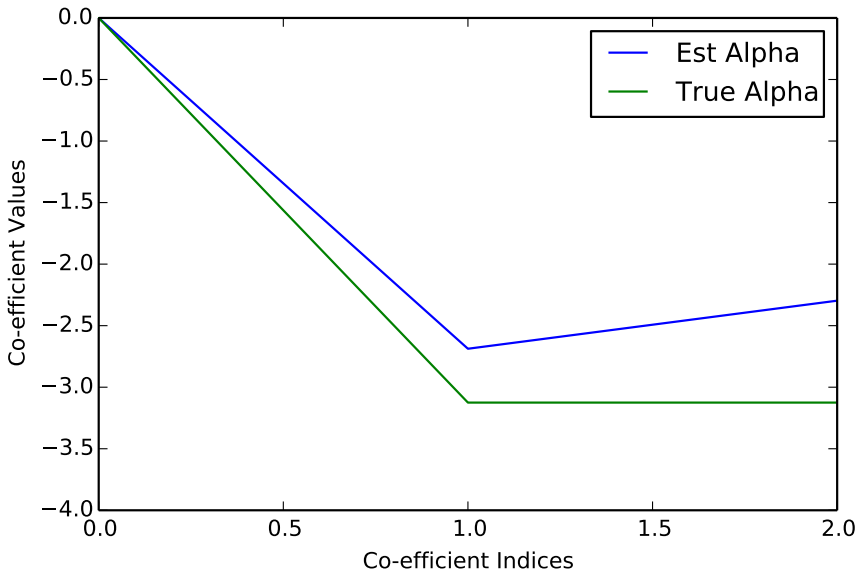


Figure 5: Ture and Estimated values of the Coefficients of 1D Toy DataSet

Toy DataSet (2D):

A Toy Dataset generated using Numpy. Normalized Two Dimensional Dataset with:

Mean $\begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$ Co-variance Matrix $\begin{pmatrix} 0.01 & 0 \\ 0 & 0.01 \end{pmatrix}$

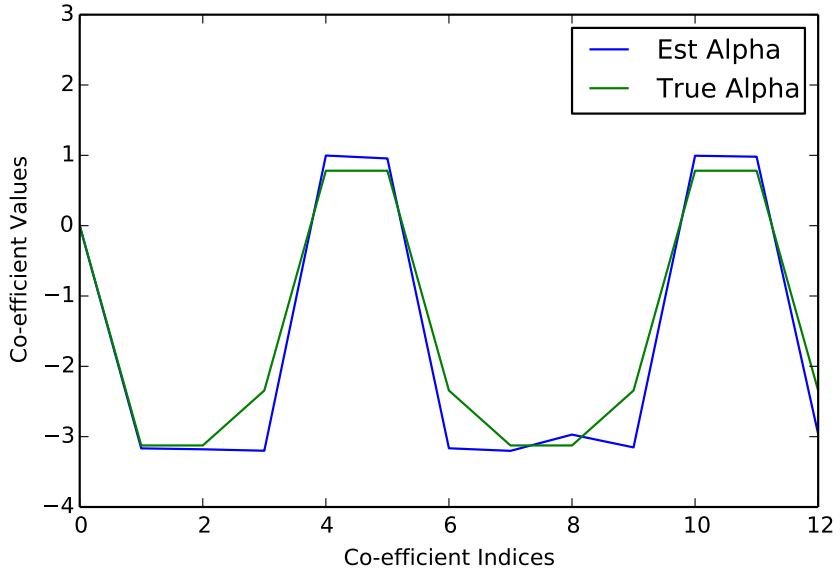


Figure 6: Ture and Estimated values of the Coefficients of 2D Toy DataSet

Toy DataSet (3D):

A Toy Dataset generated using Numpy. Normalized Three Dimensional Dataset with:

Mean $\begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}$ Co-variance Matrix $\begin{pmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.01 \end{pmatrix}$

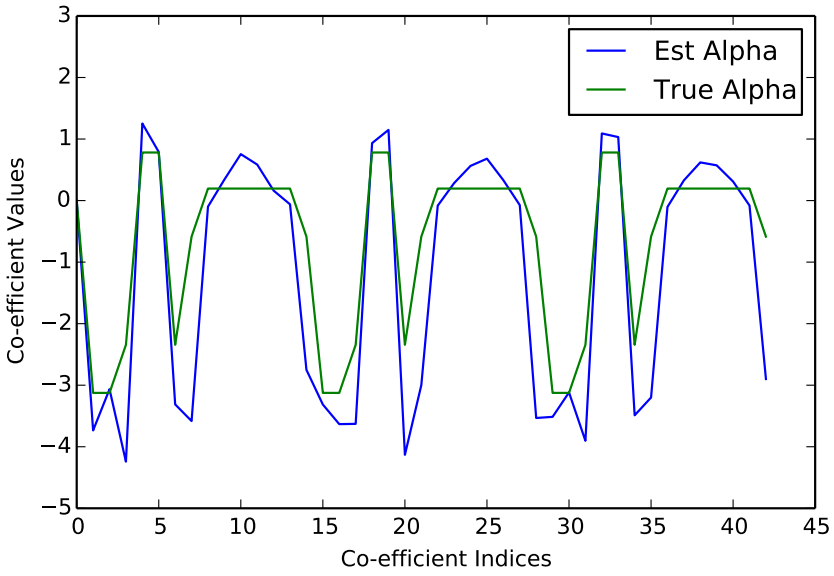


Figure 7: Ture and Estimated values of the Coefficients of 3D Toy DataSet

Note: Following experiments were run using factor graph (ANOVA components) with interacting factors of order 2. This results in faster convergence, however the results are not as good as with full factor graphs.

Toy DataSet (4D):

A Toy Dataset generated using Numpy. Normalized four Dimensional Dataset with:

$$\text{Mean} \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{pmatrix} \text{Co-variance Matrix} \begin{pmatrix} 0.01 & 0 & 0 & 0 \\ 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0.01 \end{pmatrix}$$

Parameter	Reg Strategy	Reg Param	Grid Level
Values	laplace	0.1	5

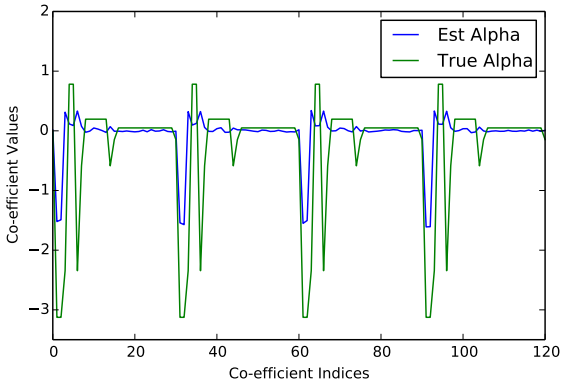


Figure 8: Ture and Estimated values of the Coefficients of 4D Toy DataSet

Toy DataSet (4D):

A Toy Dataset generated using Numpy. Normalized four Dimensional Dataset with:

$$\text{Mean} \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{pmatrix} \quad \text{Co-variance Matrix} \begin{pmatrix} 0.02 & 0 & 0 & 0 \\ 0 & 0.02 & 0 & 0 \\ 0 & 0 & 0.02 & 0 \\ 0 & 0 & 0 & 0.02 \end{pmatrix}$$

Parameter	Reg Strategy	Reg Param	Grid Level
Values	laplace	0.1	5

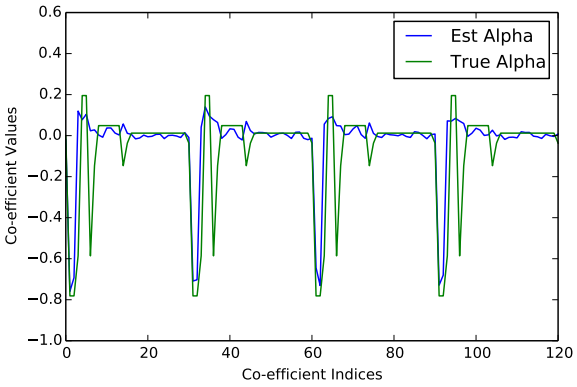


Figure 9: Ture and Estimated values of the Coefficients of 4D Toy DataSet

4.2 Minimization the objective function

One more way to validate our approach is to verify the reduction in the objective function value. Since our goal is to minimize the objective function, with estimation of the Coefficients at every iteration the likelihood gradient norm should reduce and ultimately reach the minimum. Following experiments confirm the same.

Ripley-Garcke Dataset (2D):

Parameter	Reg Strategy	Reg Param	Grid Level
Values	laplace	0.1	3

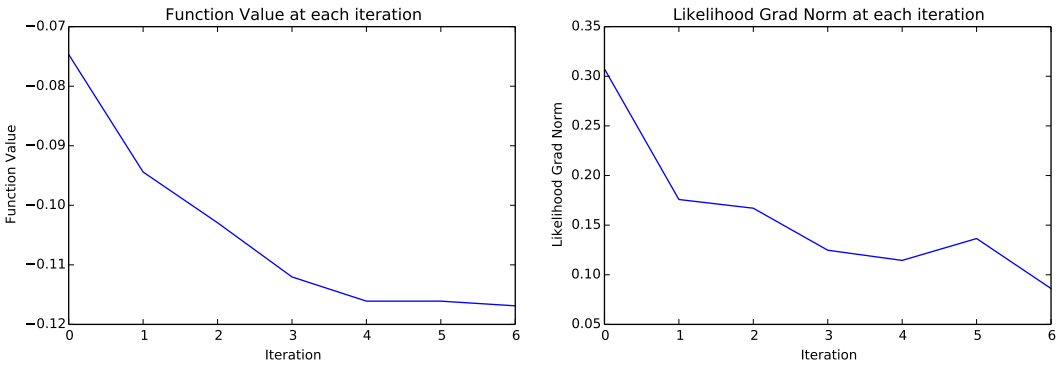


Figure 10: Function Value and Likelihood Gradient Norm per iteration

Bupa Liver Dataset (6D):

Parameter	Reg Strategy	Reg Param	Grid Level
Values	laplace	0.1	3

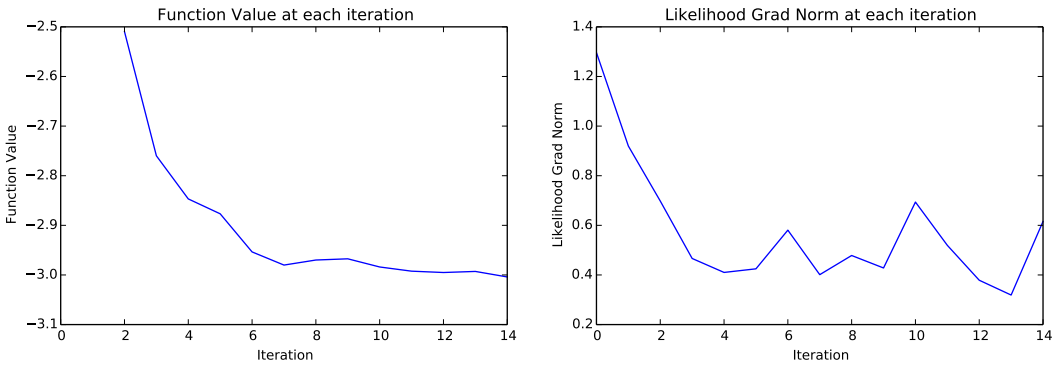


Figure 11: Function Value and Likelihood Gradient Norm per iteration

4.3 Structure Learning

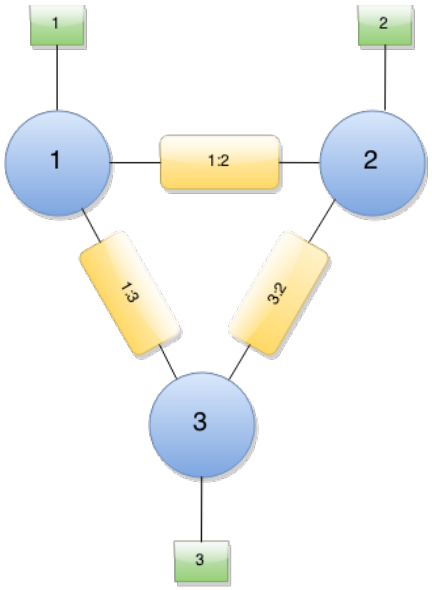
Toy DataSet with Dependencies (3D):

A Toy Dataset generated using Numpy. Normalized Three Dimensional Dataset with:

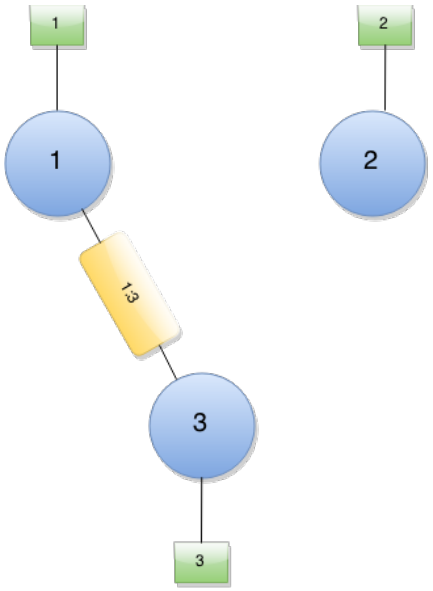
$$\text{Mean} \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \end{pmatrix} \text{Co-variance Matrix} \begin{pmatrix} 0.01 & 0 & 0.01 \\ 0 & 0.01 & 0 \\ 0.01 & 0 & 0.01 \end{pmatrix}$$

We run our estimation algorithm with factor of order 2. The following figures show the factor graphs before and after the run. The Co-variance matrix clearly shows that there is a dependency between dimension one and three and the result of the experiment is consistent with this fact.

Parameter	Reg Strategy	Reg Param	Grid Level
Values	laplace	0.1	4



(a) Factor graph of 3D dataset



(b) Reduced factor graph of 3D dataset

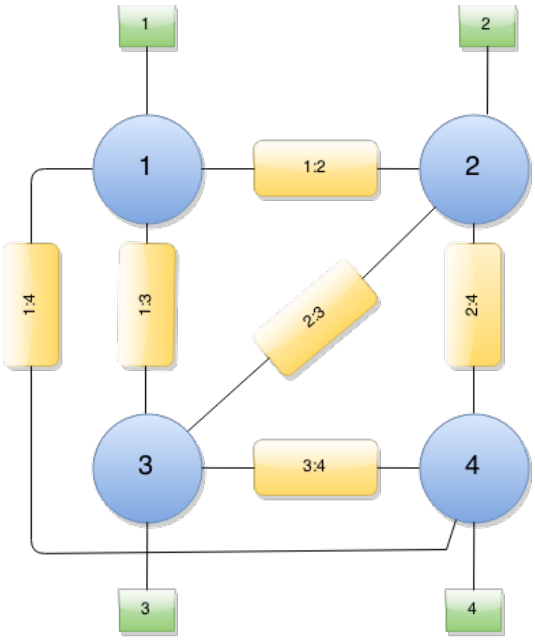
Toy DataSet with Dependencies (4D):

A Toy Dataset generated using Numpy. Normalized four Dimensional Dataset with:

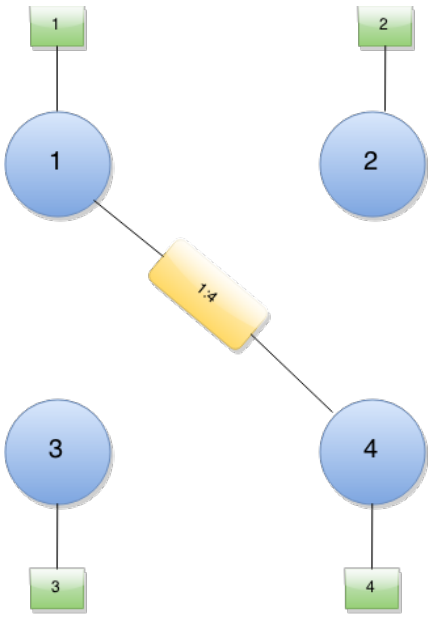
Mean $\begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}$ Co-variance Matrix $\begin{pmatrix} 0.02 & 0 & 0 & 0.02 \\ 0 & 0.02 & 0 & 0 \\ 0 & 0 & 0.02 & 0 \\ 0.02 & 0 & 0 & 0.02 \end{pmatrix}$

We run our estimation algorithm with factor or order 2. The following figures show the factor graphs before and after the run. The Co-variance matrix clearly shows that there is a dependency between dimension one and four and the result of the experiment is consistent with this fact.

Parameter	Reg Strategy	Reg Param	Grid Level
Values	laplace	0.1	5



(a) Factor graph of 4D dataset



(b) Reduced factor graph of 4D dataset

Drawbacks

This strategy has a drawback which is made evident by the following experiment. The model is able to determine only the strongest co-relation but not the weaker ones.

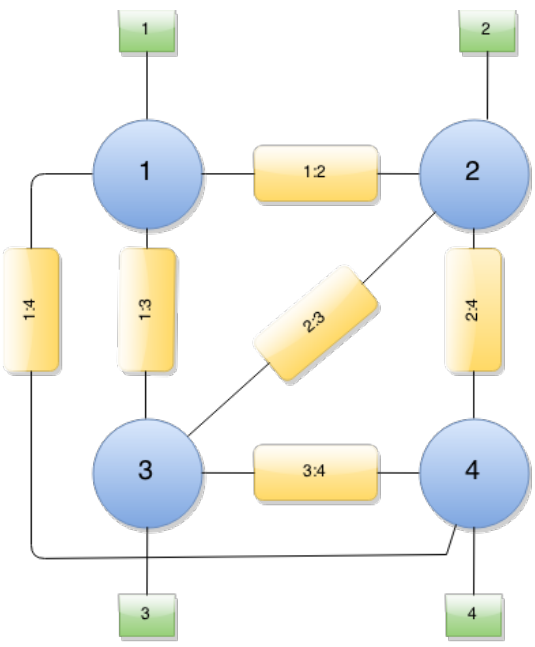
Toy DataSet with Dependencies (4D):

A Toy Dataset generated using Numpy. Normalized four Dimensional Dataset with:

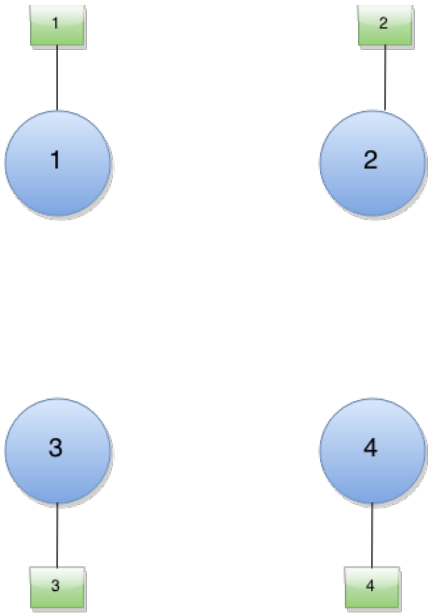
Mean $\begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{pmatrix}$ Co-variance Matrix $\begin{pmatrix} 0.02 & 0 & 0 & 0.01 \\ 0 & 0.02 & 0 & 0 \\ 0 & 0 & 0.02 & 0 \\ 0.01 & 0 & 0 & 0.02 \end{pmatrix}$

We run our estimation algorithm using factor graph with interacting factors of order 2. The following figures show the factor graphs before and after the run. The Co-variance matrix clearly shows that there is a dependency between dimension one and four and the result of the experiment is not consistent with this fact since the co-relation is weaker.

Parameter	Reg Strategy	Reg Param	Grid Level
Values	laplace	0.1	5



(a) Factor graph of 4D dataset



(b) Reduced factor graph of 4D dataset

Note: One more drawback of this approach is that the algorithm is not able to remove the higher order ANOVA components (greater than 2). Hence the experiments have been run using the factor graph with interacting factors of order 2.

5 Conclusion and Future Work

Estimation of probability density using Sparse Grids and modeling the dependencies of dimensions using a factor graph which corresponds to the ANOVA decomposition, proved to be successful. There are three main outcomes of this project. First, improvement in the efficiency of the density estimation algorithm by deleting the unwanted ANOVA components from the model without much effect on the accuracy of results. Second, successful use of Markov Chain Monte Carlo sampling in estimating the expected values of sufficient statistics of the model. Third, learning the underlying structure of the dataset.

Next step would be to focus on improving the algorithm to identify the weaker components in the model even when there are higher order ANOVA components (more than two interacting factors). Also, to device a better method to calculate the Coefficient threshold used in Coefficient thresholding process. One more important factor is the regularization factor which had a significant effect on the performance of the algorithm and hence needs to be experimented with.

References

[1] Jordan Jimmy Crabbe. Handling the curse of dimensionality in multivariate kernel density estimation. 2013.

[2] Jochen Garcke. Sparse grids in a nutshell. 2013.

[3] A G Gray and A W Moore. Nonparametric density estimation: Toward computational tractability. 2003.

[4] Michael Griebel. Sparse grids and related approximation schemes for higher dimensional problems. 2005.

[5] P Hahnen. Nichtlineare numerische verfahren zur multivariaten dichteschaetzung. 2006.

[6] Michael Griebel Hans-Joachim Bungartz. Sparse grids. 2004.

[7] M. Hegland. Adaptive sparse grids. 2003.

[8] A Izenman. Recent developments in nonparametric density estimation. 1991.

[9] Stephen J. Wright Jorge Nocedal. Numerical optimization. 1999.

[10] G Hooker M Hegland and S Roberts. Finite element thin plate splines in density estimation. 2000.

[11] Benjamin Peherstorfer. Model order reduction of parameterized systems with sparse grid learning techniques. 2013.

[12] B W Silverman. Density estimation for statistics and data analysis. 1986.

[13] Sebastian Soyer. Nonlinear density estimation with applications in astronomy. 2014.

[14] V Vapnik. Statistical learning theory. 1998.