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

1 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.

2 Introduction

Suppose we have a dataset $D = \{x_1, x_2, .., x_M\} \in \mathbf{R}^d$ 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 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. Kernel density estimation is the most widely used parametric density estimation method. 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 [7].

2.1 Grid-based Density Estimation

Density estimation methods like Kernel Density Estimation are sensitive with respect to the parameters and have long runtime for large datasets. In practice, kernel density estimation is often restricted to dimension less than 4. Grid based methods however overcome these limitations. However, similar to Kernel Density Estimation, 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 [6].

2.2 Sparse Grids

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 $\mathcal{O}(h_n^{-1}.log(h_n^{-1})^{d-1})$ 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 [3].

For clear understanding of the concept of basis functions, we refer to the comprehensive description provided in [6].

2.2.1 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

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

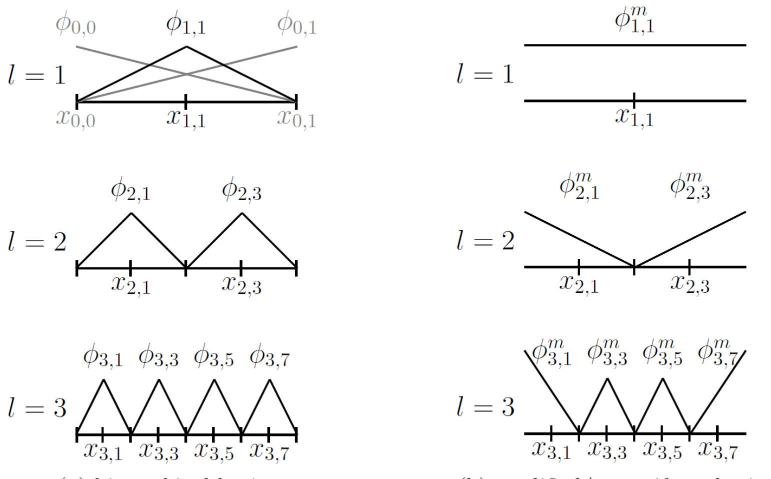
The one-dimensional hierarchical hat function $\varphi_{l,i}$ centered at the grid point $x_{l,i} = i.2^{-l}$ is the result dilation and translation of φ,

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

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). [6]

2.2.2 Modified Linear Basis

We employ Sparse Grid with a modified linear basis for our density estimation method, since it has a correspondence to the 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

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

2.3 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 [6].

2.4 Coarsening the Grid

We can further improve the efficieny of density estimation by deleting 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 [4]. 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 efficiency of computation.

3 Density Estimation using Sparse Grid

We estimate $\widetilde{P}$ defined as the Probability density function of an exponential family. In which, the required function is approximated using sparse grid discretization technique. In other terms MAP (Maximum A Posteriori) approach with sparse grid discretization.

$$\widetilde{P} = argmin_{f \in V} \int_{\Omega} (f(x) - p_{\epsilon}(x))^2 dx + \lambda || \wedge f||^2_{L^2} \tag{5}$$

Minimization is performed using variational methods: The solution of the underlying

$$\textstyle\sum_{j=1}^{N} \alpha_j^{i+1} a(\varphi_k, \varphi_j) = \frac{1}{n} \sum_{i=1}^{N} \varphi_k(x_i) - \int \varphi_k(x) \frac{exp(\sum_{j=1}^{N} \alpha_j^i \varphi_j(x))}{\int exp(\sum_{j=1}^{N} \alpha_j^i \varphi_j(z)) dz} dx \tag{6}$$

In the matrix form we have

$$A\alpha^{i+1} = q - \Phi(\alpha^i) \tag{7}$$

Where A represents the Regularization operator

$$A = a(\varphi_i, \varphi_j) = \lambda \int \bigtriangledown \varphi_i(x) \bigtriangledown \varphi_i(x) dx \tag{8}$$

And q can be computed using the definition of the basis functions

$$q = \frac{1}{n} \textstyle\sum_{i=1}^{n} \varphi_k(x_i) \tag{9}$$

The challenge lies in solving the following Non-Linear part

$$\Phi(\alpha^i) = \int \varphi_k(x) \frac{exp(\sum_{j=1}^{N} \alpha_j^i \varphi_j(x))}{\int exp(\sum_{j=1}^{N} \alpha_j^i \varphi_j(z)) dz} dx \tag{10}$$

We employ Markov Chain Monte Carlo sampling for computing the Non-linear term mentioned above. Different approaches to solve the Non-linear term, like Monte Carlo Integration has been evaluated in [8]

4 Computation of Non-Linear term using MCMC

We need to compute this Non-Linear term.

$$\Phi(\alpha^i) = \int \varphi_k(x) \frac{exp(\sum_{j=1}^{N} \alpha_j^i \varphi_j(x))}{\int exp(\sum_{j=1}^{N} \alpha_j^i \varphi_j(z)) dz} dx \tag{11}$$

We can write the above in as follows:

$$\Phi(\alpha^i) = \int \varphi_k(x) p(x) dx \tag{12}$$

Further

$$\Phi(\alpha^i) = \int f(x)p(x)dx \tag{13}$$

Where $p(x) = \frac{exp(\sum_{j=1}^{N} \alpha_j^i \varphi_j(x))}{\int exp(\sum_{j=1}^{N} \alpha_j^i \varphi_j(z))dz}$

$p(x)$ gives the probability with which we need sample the input values 'x' for the function $f(x)$.

We can compute $\Phi(\alpha^i)$ in two ways:

- We can compute the denominator of $p(x)$ which is $\int exp(\sum_{j=1}^{N} \alpha_j^i \varphi_j(z))dz$ using Monte Carlo Integration and use the value in computing $\Phi(\alpha^i)$ which is also computed with Monte Carlo Integration. This method has high runtime and hence becomes infeasible for high dimensional datasets [8].
- We can sample values from $p(x)$ 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 $p(x)$ using MCMC, we can evaluate the function $f(x)$ on these values and compute the mean which will be the expected value of the function $f(x)$.

  $E[f(x)] = \int f(x)p(x)dx$

  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 of $p(x)$ and use them to initiate the sampling process which should result in faster convergence of the sampling run, or we can sample few new points and calculate the expected value by reusing the output of the function values.

5 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 [5]:
$f(x_k + \alpha p_k) \leq f(x_k) + c_1 \alpha \nabla f_k^T p_k$
where p_k is the descent direction at x_k and $c_1 \in (0, 1)$

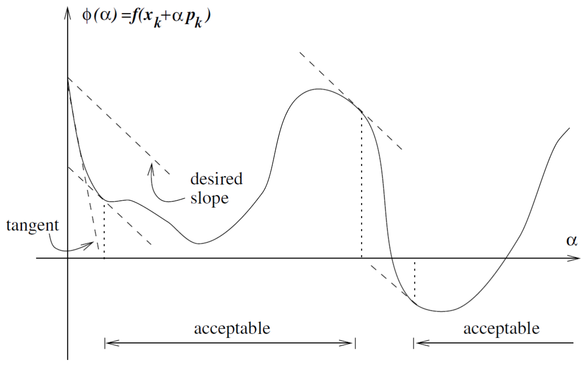


Figure 2: Armijo Condition for choosing the right step size in the descent direction. [5]

6 Dependency Modeling and Structure Learning

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 estimation of the Coefficients, unwanted factors are removed from the factor graph and the grid is coarsened accordingly. This will reduce the computation complexity of estimating the Coefficients. Since we derive the underlying structure of the dataset, we perform few experiments to validate the performance of our algorithm in structure learning. Different strategies for identifying the unwanted or less important factors are employed and evaluated as follows:

6.1 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 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 the factors that have been marked to be deleted, if there are no higher order factors that have this factor as a subset.

6.2 Choosing 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 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. Following are the results of these experiments which clearly demonstrate that the model is capable of determining the dependencies and therefore retain or throw away the factors accordingly.

6.3 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.

7 Algorithms

Algorithm 1: Estimation of Co-efficients corresponding to the Basis Functions - 'Newton-Raphson' procedure for penalized maximum likelihood

```
Data: ModLineaerGrid, Co-effecients, Factor graph, Set of DataPoints
      $X = x_1, .., x_M$
Result: Coefficients corresponding with each Basis Function
Choose Parameters $\epsilon > 0 \, and \, i_{max} = Grid\ Size$;
Calculate $A$;
Compute empirical expected values - E_empir;
Calculate model expected values - E_model $\Phi(\alpha^{initial})$;
Compute $b = E\_empir - E\_model - A.\alpha$;
while $likelihood\_grad\_norm > \epsilon$ and $i <= i_{max}$ do
    Estimate alpha_direction by solving $A = \tilde{\alpha}b$;
    Calculate learning_rate using Armijo line search;
    $\alpha = \alpha + learning\_rate * alpha\_direction$;
    Calculate model expected values - E_model $\Phi(\alpha^{i+1})$;
    Compute $b = E\_empir - E\_model - A.\alpha$;
    $likelihood\_grad\_norm = \|b\|$;
    Update Factor Graph;
    Coarsen the Grid;
    if Grid updated then
        Update Coefficient vector - retaining only non zero values;
        Calculate $A$;
        Compute empirical expected values - E_empir;
        Calculate model expected values - E_model $\Phi(\alpha^{initial})$;
        Compute $b = E\_empir - E\_model - A.\alpha$;
    end
    $i = i + 1$;
end
```

Algorithm 2: Update Factor Graph through Coefficient Thresholding

```
Data: ModLineaerGrid, Co-effecients, Factor graph
Result: Updated Factor Graph
Choose Parameters: Coefficient Threshold Value;
for grid_point_index in Grid do
    for d in dimension do
        if level(d) of the grid point $! = 1$ then
            $factor = factor + (d)$
        end
    end
    $factor\_alpha\_collection[factor] = alpha[grid\_point\_index]$;
    for factor and alphas_of_factor in factor_alpha_collection do
        if mean of $|alphas\_of\_factor| <$ mean of $|alphas|$ then
            add factor to delete_list
        end
    end
end
for factor in delete_list do
    if factor not $\subset$ higher order factor then
        Delete the factor from the Factor Graph
    end
end
```

Algorithm 3: Coarsening the Grid based on the Factor Graph

```
Data: ModLineaerGrid, Factor graph
Result: Coarsened Grid
for grid_point_index in Grid do
    for d in dimension do
        if level(d) of the grid point $! = 1$ then
            $factor = factor + (d)$
        end
    end
    if factors not $\subset$ factor graph then
        $alpha[grid\_point\_index] = 0$
    end
    if $alpha[grid\_point\_index] = 0$ then
        Delete $grid[grid\_point\_index]$
    end
end
```

8 Experiments and Results

Upon obtaining the α and the *Grid*, I estimate the density of the input data points using:

$$f(x) = \exp\left(\sum_{i=1}^{n} \alpha_i \varphi_i(x)\right)$$

8.1 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

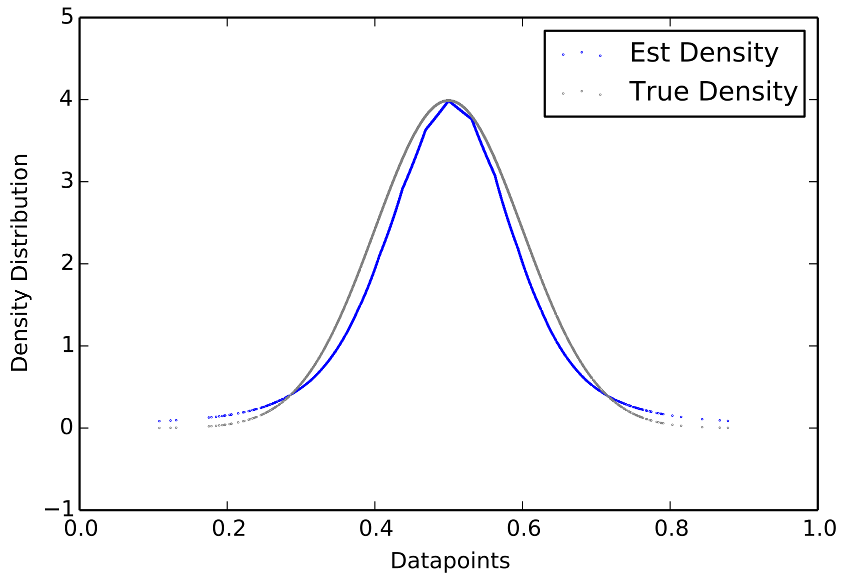


Figure 3: True and Estimated distribution of 1D Normal dataset

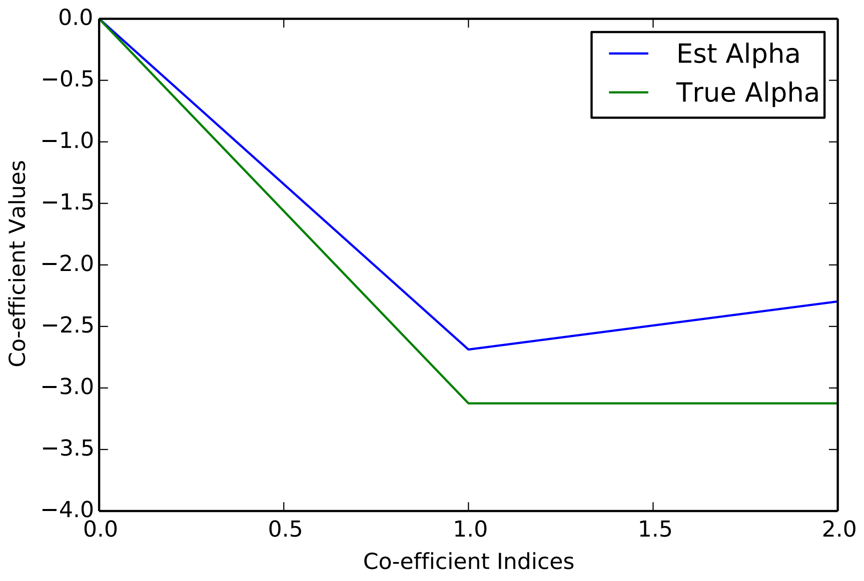


Figure 4: Ture and Estimated values of the Coefficients of 1D Toy DataSet

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 $\begin{pmatrix} 0.01 & 0 \\ 0 & 0.01 \end{pmatrix}$

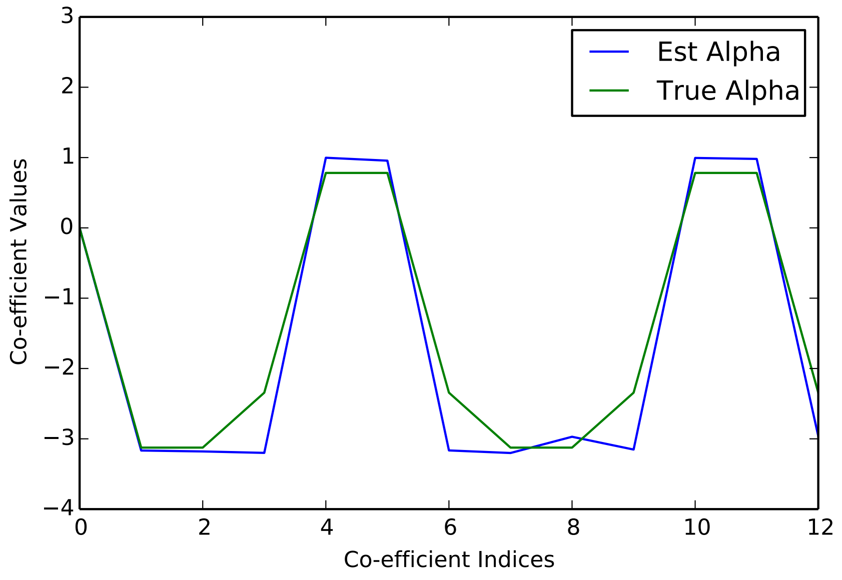


Figure 5: Ture and Estimated values of the Coefficients of 2D Toy DataSet

Toy DataSet (3D):

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

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

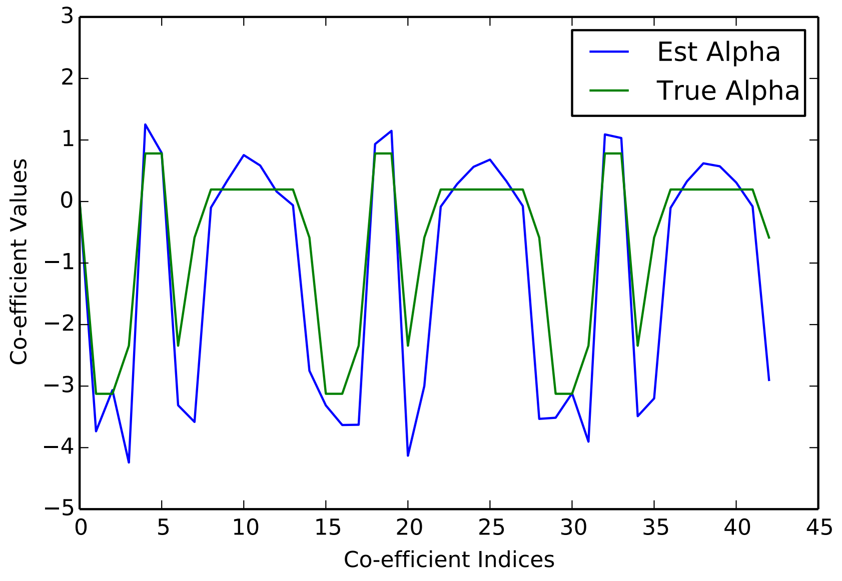


Figure 6: Ture and Estimated values of the Coefficients of 3D Toy DataSet

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

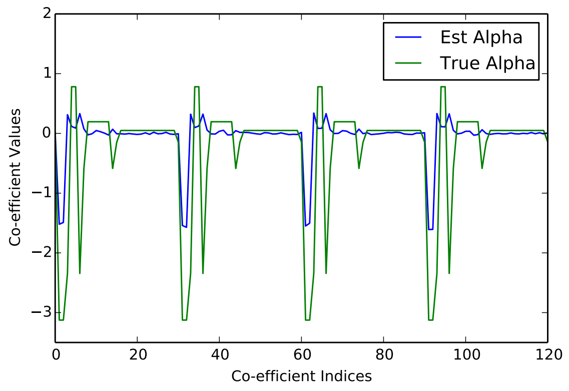


Figure 7: Ture and Estimated values of the Coefficients of 4D Toy DataSet

Toy DataSet (4D):

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

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

Parameter	Reg Strategy	Reg Param	Grid Level
Values	laplace	0.1	5

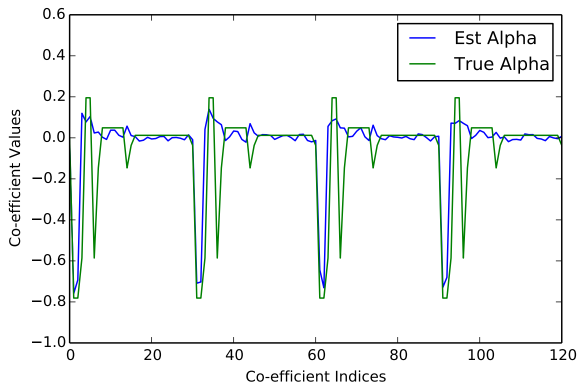


Figure 8: Ture and Estimated values of the Coefficients of 4D Toy DataSet

8.2 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

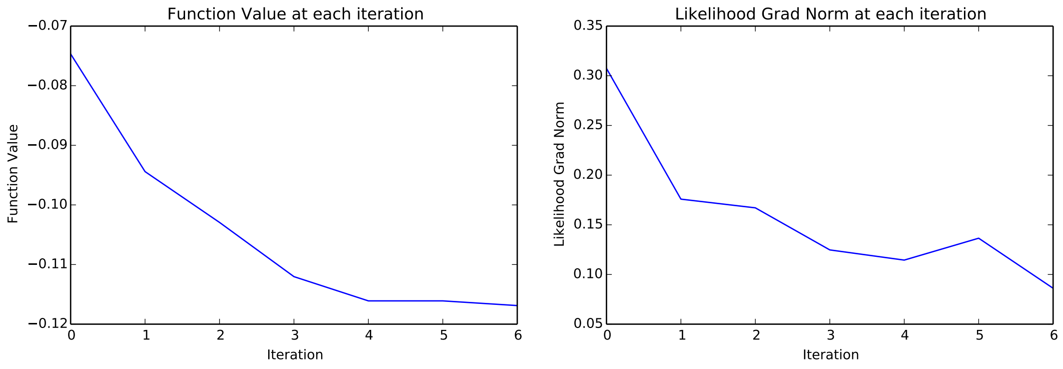


Figure 9: Function Value and Likelihood Gradient Norm per iteration

Bupa Liver Dataset (6D):

Parameter	Reg Strategy	Reg Param	Grid Level
Values	laplace	0.1	3

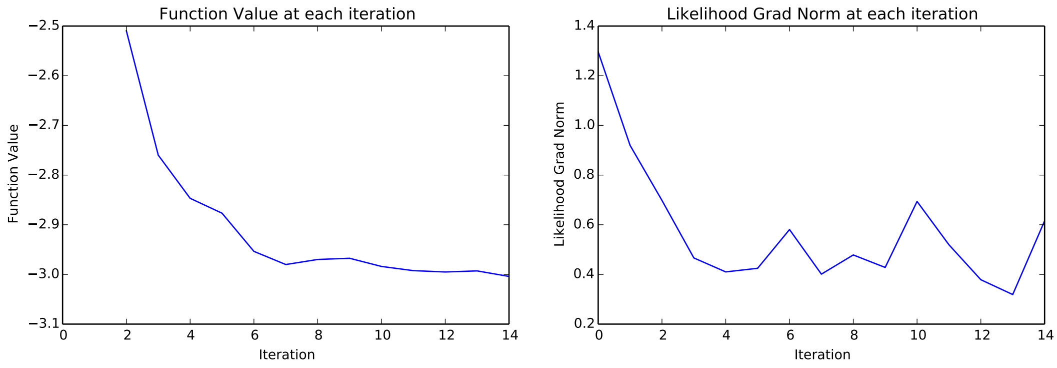


Figure 10: Function Value and Likelihood Gradient Norm per iteration

8.3 Structure Learning

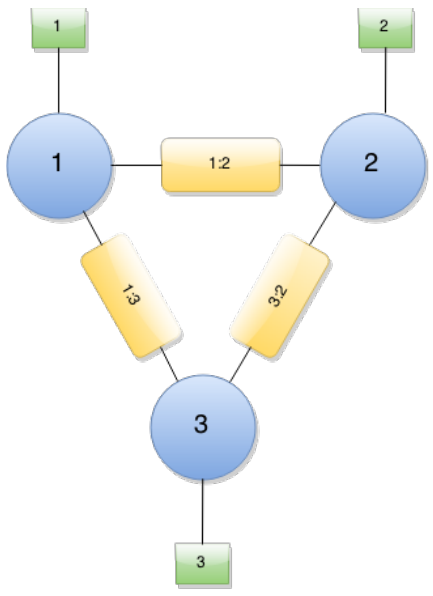
Toy DataSet with Dependencies (3D):

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

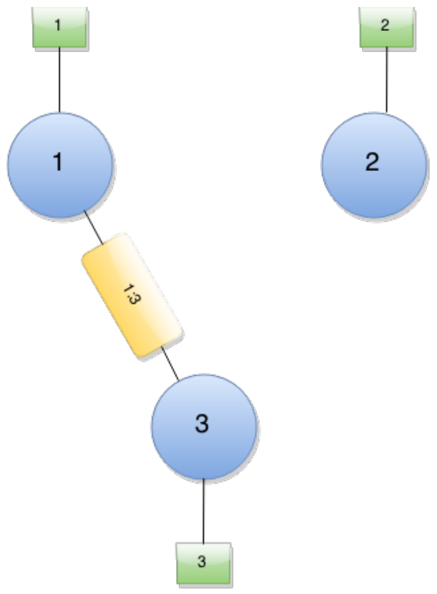
$$\text{Mean} \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \end{pmatrix} \text{ Co-variance Matrix } \begin{pmatrix} 0.01 & 0 & 0.01 \\ 0 & 0.01 & 0 \\ 0.01 & 0 & 0.01 \end{pmatrix}$$

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

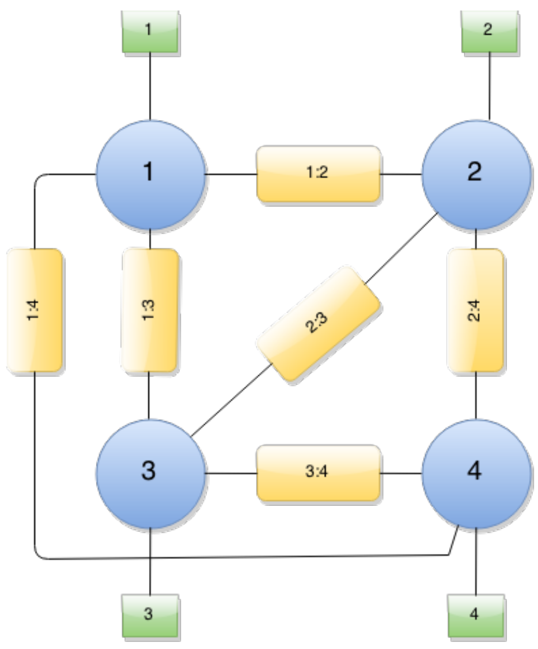
Toy DataSet with Dependencies (4D):

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

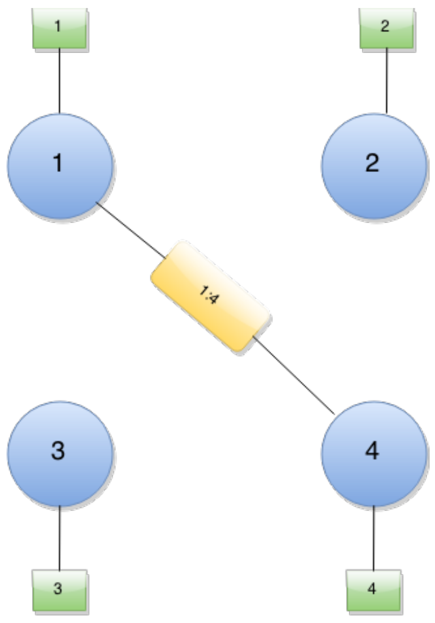
$$\text{Mean} \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{pmatrix} \text{Co-variance Matrix} \begin{pmatrix} 0.02 & 0 & 0 & 0.02 \\ 0 & 0.02 & 0 & 0 \\ 0 & 0 & 0.02 & 0 \\ 0.02 & 0 & 0 & 0.02 \end{pmatrix}$$

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.

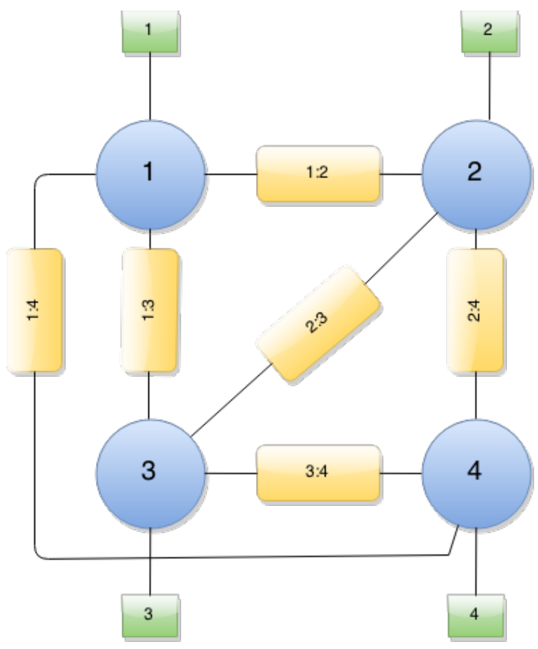
Toy DataSet with Dependencies (4D):

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

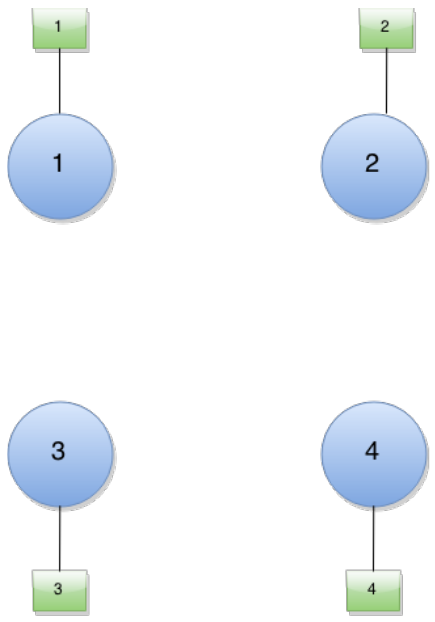
$$\text{Mean} \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{pmatrix} \text{Co-variance Matrix} \begin{pmatrix} 0.02 & 0 & 0 & 0.01 \\ 0 & 0.02 & 0 & 0 \\ 0 & 0 & 0.02 & 0 \\ 0.01 & 0 & 0 & 0.02 \end{pmatrix}$$

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.

9 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] Michael Griebel Hans-Joachim Bungartz. Sparse grids. 2004.

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

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

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

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

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