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**

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

August 2015

Contents

Abstract

In this paper, we study the application of sparse grid based methods for estimation of probability density function 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 conditional dependence structure of the dataset's random variables. 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 $x_i \in \mathbb{R}^d$ 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 [20]. 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 problems.

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 [10]. 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 $h > 0$; 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 relates to *Curse of Dimensionality*, a term coined by Bellman to describe the problem caused by the exponential increase in volume associated with adding extra dimensions to Euclidean space [2]. It refers to the fact that for an estimator of a function, the number of observations required per variable increase exponentially with the number of variables, to maintain a given level of accuracy. For thorough understanding of density estimation and its applications we refer to [18].

In the remaining part of this section we describe grid-based density estimation method. In Section 2 we describe the sparse-grid based density estimation using maximum a posteriori (MAP) approach. Section 3 will provide the details on the performance improvement strategies for the density estimation method along with the algorithms. In section 4 we provide the details of experiments and interpret the results.

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, which results in the curse of dimensionality. 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.

Density estimation methods like KDE are sensitive to the choice of 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, one can look into [16].

1.2 Sparse-grid-based density estimation

To remedy the curse of dimensionality of the grid-based methods, 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 [3]. L_2 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 [16].

Sparse grids method is a spacial 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 [4]. In depth details of the topic can be found in [3].

For clear understanding of the concept of basis functions, one can read through the comprehensive description provided in [16].

1.2.1 Hierarchical Basis

For ease of presentation we will consider the domain $\Omega = [0,1]^d$ here and in the following. Let $l = (l_1, ..., l_d) \in \mathbb{N}^d$ denote a multi-index. We define the anisotropic grid Ω_l on $\bar{\Omega}$ with a mesh size $h_l = (h_{l_1}, ..., h_{l_d}) = (2^{-l_1}, ..., 2^{-l_d})$, it has different, but equidistant mesh sizes in each coordinate direction t. This way the grid Ω_l consists of the points

$$x_{\mathbf{l},\mathbf{i}} = (x_{l_1,i_1}, ..., x_{l_d,i_d}), \tag{1}$$

with $x_{l_t,i_t} = j_t \cdot h_{l_t} = j_t \cdot 2^{-l_t}$ and $j_t = 0, ..., 2^{l_t}$.

Let $\mathcal{V}_l$ be the function space corresponding to the domain $\Omega = [0,1]^d \subset \mathbb{R}^d$, which is spanned by the so called hierarchical basis up to level l. In the standard case, piecewise linear hat functions are used as basis functions. One-dimensional standard hat function $\phi : [-1,1] \to \mathbb{R}$ is defined as

$$\phi(x) = max(1 - |x|, 0). \tag{2}$$

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

$$\phi_{l,i}(x) = \phi(2^l x - i). \tag{3}$$

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

$$\phi_{\mathbf{l},\mathbf{i}}(\mathbf{x}) = \prod_{j=1}^{d} \phi_{l_j,i_j}(x_j), \tag{4}$$

where $l = (l_1, ..., l_d)$ and $i = (i_1, ..., i_d)$ denote the level and index respectively. The corresponding grid point $x_{\mathbf{l},\mathbf{i}} = [x_{l_1,i_1}, ..., x_{l_d,i_d}]^T$ is the center of the support of $\phi_{\mathbf{l},\mathbf{i}}$. The hierarchical basis functions for level $l = 1$ to $l = 3$ are shown in Figure 1.

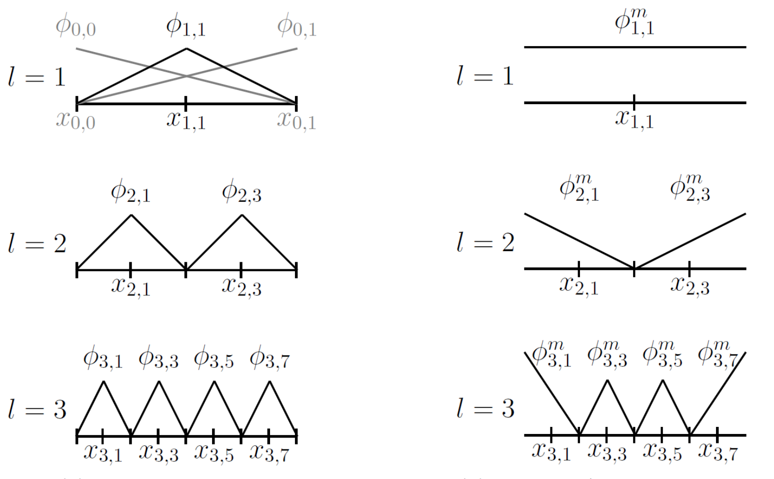


Figure 1: Hierarchical basis functions for level $l = 1$ to $l = 3$ are shown in (a). Modified basis functions are shown in (b). Source: [16].

The space $\tilde{\mathcal{H}}_k$ of piecewise d-linear functions is spanned by the nodel point basis

$$\tilde{\varphi}_k = \left\{ \phi_{\mathbf{k},\mathbf{i}} | i \in \tilde{\mathfrak{I}}_k \right\}, \tag{5}$$

with the index set

$$\tilde{\mathfrak{I}} = \left\{ i \in \mathbb{N}^d | 1 \leq i_j < 2^{k_j}, 1 \leq j \leq d \right\}. \tag{6}$$

We also introduce the hierarchical increments $\mathcal{W}_k$ spanned by

$$\hat{\varphi}_k = \left\{ \phi_{\mathbf{k},\mathbf{i}} | i \in \hat{\mathfrak{I}}_k \right\}, \tag{7}$$

where

$$\hat{\mathfrak{I}} = \left\{ i \in \mathbb{N}^d | 1 \leq i_j < 2^{l_j}, i_j \notin 2\mathbb{N}, 1 \leq j \leq d \right\}. \tag{8}$$

The function space

$$\mathcal{V}_l = \bigoplus_{|l|_1 \leq l+d-1} \mathcal{W}_k \tag{9}$$

with $|l|_1 = \sum_i |l_i|$ is called the sparse grid space of level l and dimension d. Figure 2 shows the grid points of the two-dimensional sparse grid of level three, the grid points of the corresponding hierarchical increments $\mathcal{W}_l$, and the grid points of the hierarchical subspaces $\tilde{\mathcal{H}}_k$ with $|k|_1 \leq l + d - 1$.

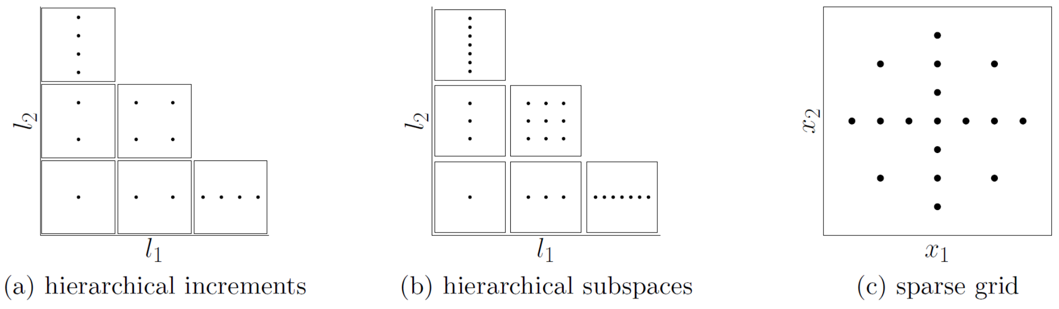


Figure 2: In (a) the grid points corresponding to the hierarchical increments $\mathcal{W}_l$ of a sparse grid of level three arranged in the hierarchical scheme are shown. In (b) we plotted the grid points of the hierarchical subspaces $\tilde{\mathcal{H}}_l$ and in (c) the corresponding sparse grid. Source: [16].

1.2.2 Modified Linear Basis and ANOVA-like Decomposition

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

$$\begin{aligned} h(x_1, ..., x_d) = h_0 + \sum_{j_1}^{d} h_{j_1}(x_{j_1}) + \sum_{j_1<j_2}^{d} h_{j_1,j_2}(x_{j_1}, x_{j_2}) \\ + \sum_{j_1<j_2<j_3}^{d} h_{j_1,j_2,j_3}(x_{j_1}, x_{j_2}, x_{j_3}) + \cdots + h_{j_1,...,j_d}(x_{j_1}, ..., x_{j_d})). \end{aligned} \tag{10}$$

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 (10) 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 [5].

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 $\tilde{\mathcal{H}}_k$, the hierarchical subspaces of the sparse grid structure, where k is a list of the grid levels in every dimension. Equation (11) shows the decomposition of the function space $\mathcal{V}_l$ in to ANOVA like components.

$$\begin{aligned} \mathcal{V}_l &= \bigotimes_{j=1}^{d}(1_j \oplus \mathcal{W}_j) \\ &= 1_1 \otimes \cdots \otimes 1_d \\ &\bigoplus_{i=1}^{d} 1_1 \otimes \cdots \mathcal{W}_i \cdots \otimes 1_d \\ &\bigoplus_{i=1}^{d}\bigoplus_{i<j} 1_1 \otimes \cdots \mathcal{W}_i \cdots \mathcal{W}_j \cdots \otimes 1_d \\ &\bigoplus_{i=1}^{d}\bigoplus_{i<j}\bigoplus_{i<j<k} 1_1 \otimes \cdots \mathcal{W}_i \cdots \mathcal{W}_j \cdots \mathcal{W}_k \cdots \otimes 1_d \\ &\cdots \\ &\oplus \mathcal{W}_1 \otimes \cdots \otimes \mathcal{W}_d \end{aligned} \tag{11}$$

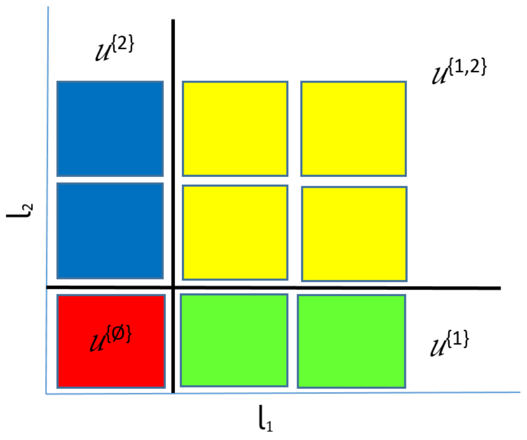
Here $1_j = 1$ and $\mathcal{W}_j = \mathcal{W}$; we use the index j merely to indicate the respective coordinate direction. A compact representation in terms of $u \in \tilde{\mathcal{H}}_k$ would be

$$\mathcal{V}_l = u^{\{\emptyset\}} \quad \bigoplus_{i=1}^{d} u^{\{i\}} \quad \bigoplus_{i=1}^{d}\bigoplus_{i<j} u^{\{i,j\}} \cdots \tag{12}$$

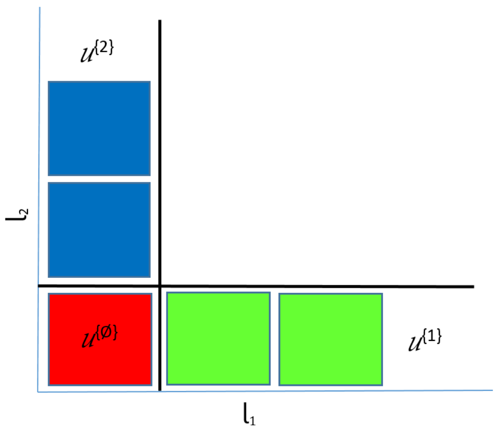
This means, the constant function h_0 in (10) is replaced by a function $u^{\{\emptyset\}}$ as well as all further functions $u^{\{i\}}, u^{\{i,j\}}, ...$ are constructed with respect to $\tilde{\mathcal{H}}_k$. The figure $2(a)$ shows the two-dimensional ANOVA-like decomposition with respect to $\tilde{\mathcal{H}}_{(2,2)}$.

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. Figure 1 shows both hierarchical and modified 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 $\phi_{l,i}$ is modified in the following way

$$\phi^m_{l,i}(x) = \begin{cases} 1 & \text{if } l = 1, i = 1, \\ \left\{ \begin{array}{cc} 2 - 2^l \cdot x & \text{if } \; x \in [0, 2^{-(l-1)}] \\ 0 & \text{else} \end{array} \right\} & \text{if } l > 1, i = 1, \\ \left\{ \begin{array}{cc} 2^l \cdot x + 1 - i & \text{if } \; x \in [1 - 2^{-(l-1)}, 1] \\ 0 & \text{else} \end{array} \right\} & \text{if } l > 1, i = 2^l - 1, \\ \phi(x \cdot 2^l - i) & \text{else.} \end{cases} \tag{13}$$



(a)



(b)

Figure 3: The two-dimensional ANOVA-like decomposition of the function $u \in \mathcal{V}_l$ 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.

1.3 Factor Graph

A factor graph represents the factorization of a function of several variables. It is often used to represent the dependencies or the interaction between the variables. The interaction may involve two or more variables. Figure 4 shows a factor graph of 3 variables and factors up to 3-variable interactions, which is the max number of interactions possible for a 3 variable function. This can be interpreted as all the variables interacting with one another or all the variables being dependent on one another.

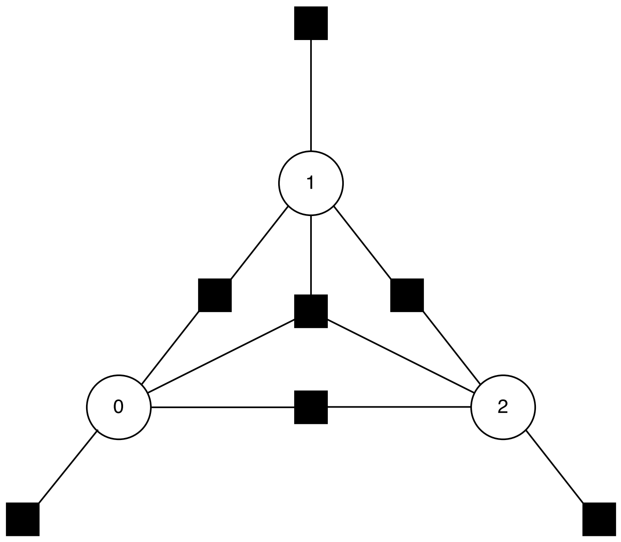


Figure 4: A fully connected factor graph with 3 variables. The circular nodes denote variables and the square nodes denote the factors. Square nodes connected to only one variable node represent the individual factors. Square nodes connecting two variables represent the interacting factor of order 2. Square node connecting all three variables represent the interacting factor of order 3.

1.4 Coarsening the Grid

As stated earlier, the sparse grids use much less grid points than the full grid hence improving the efficiency of density estimation. Now we will look into a method which can further improve the efficiency. 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 dimensional decomposition has been studied and sparse grids approach to dimensional decomposition which works similar to ANOVA decompositions has been shown in [7].

In this project we employ sparse grids with modified basis which corresponds to ANOVA like decomposition. Also, the factors of a factor graph corresponds to the components of ANOVA decomposition, hence we can associate the components of the model 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 in our model and remove them in order to reduce the complexity of the problem and hence improving the computational efficiency. Figure 3(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 [1], we look for $\tilde{p}$ in the function space $\mathcal{V}$ such that

$$\tilde{p} = \underset{f \in \mathcal{V}}{argmin} \int_\Omega (f(x) - p_\epsilon(x))^2 dx + \lambda || \wedge f ||^2_{L^2}. \tag{14}$$

The left term of (14) 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, the regularization operator $\wedge$ 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 $p_\epsilon = \frac{1}{M}\sum_{i=1}^{M}\phi(x_i)$, we obtain after some transformations (refer [8]), the variational equation of the form

$$\int_\Omega \phi(x)\tilde{p}(x)dx + \lambda\int_\Omega \wedge\tilde{p}(x)\cdot\wedge\phi(x)dx = \frac{1}{M}\sum_{i=1}^{M}\phi(x_i), \tag{15}$$

which holds for all $\phi \in \Psi_l$ and since f belongs to an exponential family, we can write $\tilde{p}(x) = \frac{exp(u(x))}{\int exp(u(z))dz}$.

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

$$u(x) = \sum_{j=1}^{N}\alpha_j\phi_j(x), \tag{16}$$

where N is the number of grid points. α represents the coefficients and $\phi(x)$ 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 [6]. One (i^{th}) iteration of the algorithm in the mathematical form is given as

$$\sum_{j=1}^{N}\alpha_j^{i+1}a(\phi_k,\phi_j) = \frac{1}{M}\sum_{i=1}^{M}\phi_k(x_i) - \int\phi_k(x)\frac{exp(\sum_{j=1}^{N}\alpha_j^i\phi_j(x))}{\int exp(\sum_{j=1}^{N}\alpha_j^i\phi_j(z))dz}dx. \tag{17}$$

We can write (17) in the Matrix form as follows,

$$A\overrightarrow{\alpha}^{i+1} = E_{empir} - \Phi(\overrightarrow{\alpha}^i). \tag{18}$$

Where A represents the regularization operator

$$(A)_{k,j} = a(\phi_k,\phi_j) = \lambda\int \nabla\phi_k(x)\nabla\phi_j(x)dx. \tag{19}$$

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

$$E_{empir} = \frac{1}{M}\sum_{i=1}^{M}\phi_k(x_i). \tag{20}$$

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

$$\Phi(\alpha^i) = \int\phi_k(x)\frac{exp(\sum_{j=1}^{N}\alpha_j^i\phi_j(x))}{\int exp(\sum_{j=1}^{N}\alpha_j^i\phi_j(z))dz}dx. \tag{21}$$

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 [19].

2.1 Computating the Expected Value using MCMC

The nonlinear term (21) can be written as follows:

$$\Phi(\alpha^i) = \int \phi_k(x)\tilde{p}(x)dx. \tag{22}$$

where $\tilde{p}(x)$ gives the probability with which we need to sample the input values x for the function $\phi_k(x)$. Hence, (22) actually represents the expected value of $\phi_k(x)$ and can be written as

$$E[\phi_k(x)] = \int \phi_k(x)\tilde{p}(x)dx. \tag{23}$$

Following are the alternative methods we can employ to compute this expected value:

- We can compute the denominator of $\tilde{p}(x)$, i.e., $\int exp(\sum_{j=1}^{N} \alpha_j^i \phi_j(z))dz$ using Monte Carlo Integration and use the value to compute $E[\phi_k(x)]$, which is also done with Monte Carlo Integration. This method has high runtime and hence becomes infeasible for high dimensional datasets [19].
- We can sample values from the posterior distribution $\tilde{p}(x)$ using Markov Chain Monte Carlo (MCMC) by creating a model based on the Sparse Grid structure and assuming particular prior distribution (preferably Uniform) for the random variables. Once we sample the values from $\tilde{p}(x)$ using MCMC, we can evaluate the function $\phi_k(x)$ on these samples and compute the mean which will be the expected value of the function $\phi_k(x)$. 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 [14].

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 M and a modlinear grid, we are required to estimate the coefficients corresponding to each grid point.

At first we initialize the learning rate (ω), a measure (ϵ) to compare the residual and decide when to stop the algorithm and the maximum number of iterations allowed ($imax$), which in our case we set to the number of grid points. The Newton-Raphson procedure to solve (18) and determine the coefficients is given in Algorithm 1.

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

$$\tilde{p}(x) = \frac{exp(\sum_{j=1}^{N} \alpha_j^i \phi_j(x))}{\int exp(\sum_{j=1}^{N} \alpha_j^i \phi_j(z))dz} \tag{24}$$

Algorithm 1: Estimating the coefficients of the basis functions using MAP approach. Source: [6].

```
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

In this section we will discuss different ways to improve the performance of our density estimation method along with the implementation details in the form of algorithms. Section 3.1 provides a brief description about dynamic learning rate using Armijo line search technique. In Section 3.2 we discuss different methods for coarsening the grid.

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 [6], 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.

In this project we deal with an optimization problem, in which we minimize the objective function, which is the negative log likelihood of the given dataset [12]. We employ Newton-Raphson procedure to minimize 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:

$$t(x_k + \omega p_k) \leq t(x_k) + c_1 \omega \nabla t_k^T p_k \tag{25}$$

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 $c_1 \in (0, 1)$.

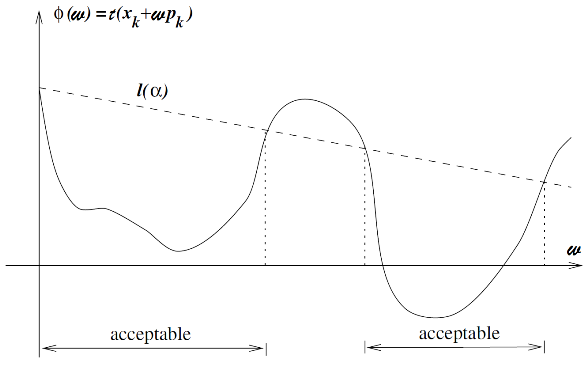


Figure 5: Armijo Condition (Sufficient decrease condition) for choosing the right step size in the descent direction. Source: [15].

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 (25) is sometimes called Armijo condition. The sufficient decrease condition is illustrated in Figure 5. The right-hand-side of (25), which is a linear function, can be denoted by $l(\alpha)$. The function $l()$ has negative slope $c_1 \omega \nabla t_k^T p_k$, but because $c_1 \in (0, 1)$, it lies above the graph of for small positive values of ω. The sufficient decrease condition states that ω is acceptable only if $\Phi(\omega) \le l(\omega)$. The intervals on which this condition is satisfied are shown in Figure 5. In practice, c_1 is chosen to be quite small, say $c_1 = 10^{-4}$. For detailed understanding of the concept we refer to [15].

Algorithm 2 is the improved form of Algorithm 1 with dynamic learning rate and grid coarsening techniques which we will discuss in the following Section.

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

```
Data: $grid$, $\alpha$, $factor\_graph$, $data$          // Modified Linear Basis Grid
Result: $\alpha$ 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 $\omega$ (Section 3.1) such that $t(x_k + \omega p_k) \le t(x_k) + c_1 \omega \nabla t_k^T p_k$
    $\alpha^{i+1} = \alpha^i + \omega \cdot p_k$
    Calculate model expected values: $\Phi(\alpha^{i+1})$ (Section 2.1)
    Compute $\nabla\mathcal{L} = E_{empir} - \Phi(\alpha^{i+1}) - A \cdot \alpha$
    $grid$, $\alpha$ = grid_coarsening($grid$, $\alpha$, $factor\_graph$)
    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
```

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 strategies for identifying the unwanted or less important factors. Algorithm 3 provides implementation details.

Mean Coefficient Thresholding

In order to determine the importance of the grid points, we first estimate the coefficients of all the grid points and then fetch the grid point indices and corresponding factors in the factor graph model. Each factor can have association with multiple coefficients. For each 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.

Mean Volume Thresholding

Volume of a grid point represents the L_2 norm of basis function, weighted by corresponding basis function coefficients. Volume is a better measure for gauging the importance of factors in the model since it also considers the basis function values along with the coefficient values. Mean volume thresholding strategy is same as mean coefficient thresholding, only difference is that we use volumes in place of coefficient values.

Choosing the Threshold

Choosing the threshold 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. (We do not provide the details of these experiments in this paper). Mean of the absolute values of all the coefficients, estimated in the very first execution of our algorithm turned to be a good choice. However with more such experiments, it was evident that dynamically updating the threshold at every iteration with the mean absolute values of the coefficients (or volumes in case of Mean Volume Thresholding) of interacting factors, is a better choice. We chose the later because the estimated coefficients are closer to the true coefficients at every iteration and hence we obtain a better threshold at every iteration.

Algorithm 3: Coefficient Thresholding

```
Procedure coefficient_thresholding(grid, $\overrightarrow{\alpha}$, factor_graph)
    Set $\alpha_t$ = mean of $|\overrightarrow{\alpha}|$
    Set dimension = factor_graph → dimension
    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_alphas[factor] = $\overrightarrow{\alpha}$[grid_point]
    end
    for factor, alphas_of_factor in factor_alphas do
        if mean of |alphas_of_factor| < $\alpha_t$ then
            add factor to delete_list
        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
```

Updating the Grid

When we throw away nodes 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 nodes in factor graph. Hence it is possible to make sure that crucial grid points are not deleted when finer level grid points exist in the grid; which otherwise might lead to inconsistency in the grid. Implementation details are provided in Algorithm 4.

Algorithm 5: Grid Coarsening

```
Procedure grid_coarsening(grid, $\overrightarrow{\alpha}$, factor_graph)
    factor_graph = coefficient_thresholding(grid, $\overrightarrow{\alpha}$, factor_graph)
    Set dimension = factor_graph → dimension
    for grid_point in grid do
        for d in dimension do
            if level(d) of the grid_point != 1 then
                factor = factor + (d)        // concatenate (d) to the tuple
            end
        end
        if factor not in factor_graph then
            $\overrightarrow{\alpha}$[grid_point] = 0
            delete grid_point
        end
    end
    return grid, $\overrightarrow{\alpha}$                    // returns coarsenend grid
```

4 Experiments and Results

We evaluate the sparse-grid based density estimation (SGDE) algorithms introduced in Section 3, in different ways using artificial datasets and other publicly available datasets. We create two types of artificial data sets. One type of data sets are sampled in each dimension independently from normal distribution with mean 0.5 and standard deviation 0.1. Another type of data sets are sampled based on a particular structure, defined in terms of dependencies of the variables. For all our experiments, we set the regularization parameter $\lambda = 10^{-1}$ (unless otherwise mentioned) since it provided the best results in our experiments. This is in contrast to the findings observed in [16], in which, setting the regularization parameter to much smaller value $\lambda = 10^{-5}$ gave best results.

Validation of density estimation results

Figure 6 shows the comparision of the density distribution of the data obtained from posterior sampling of our model and the true distribution. As one can see, our method was able to estimate the underlying distribution accurately. In order to quantify the measure of accuracy we used Kolmogorov-Smirnov (KS) two-sample test [9], a nonparametric method for comparing two samples and determine whether they are from the same probability distribution or not. In this case, the test returned a p-value greater than 70%, which means that we cannot reject the hypothesis that the two samples are from the same distribution.

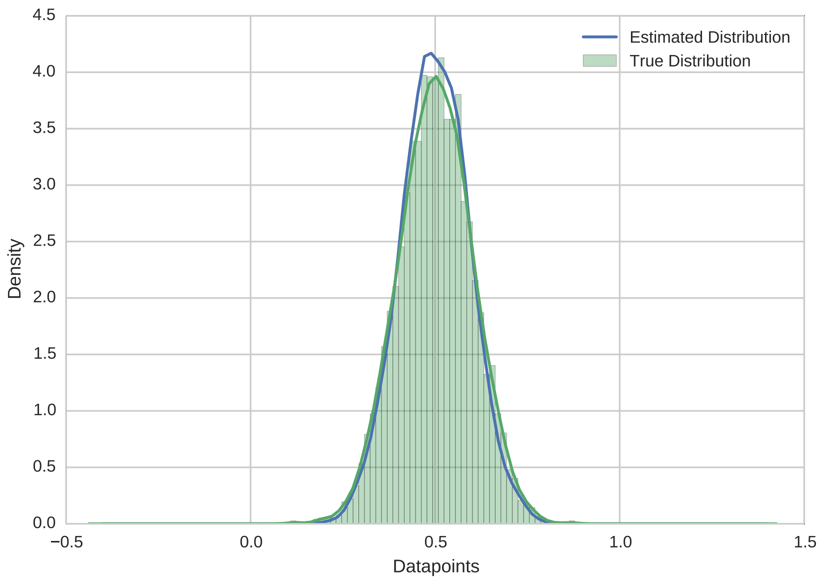


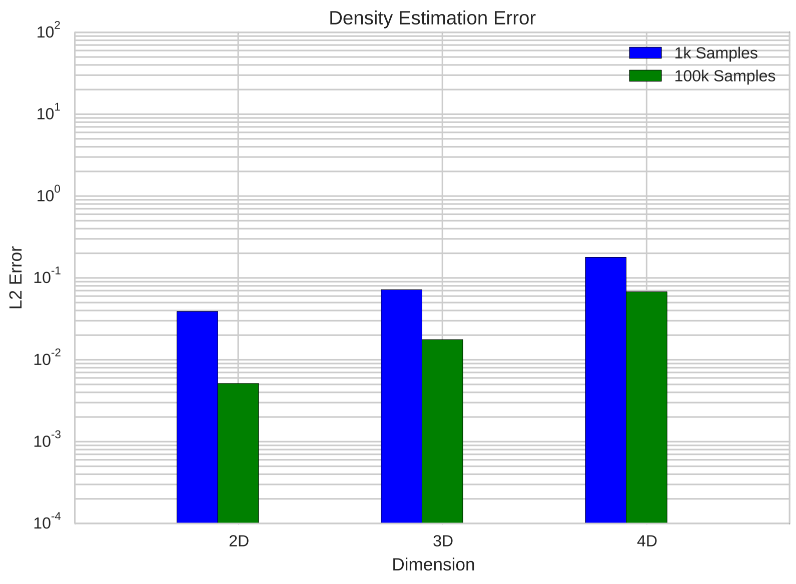
Figure 6: True and estimated distribution of 1D normal dataset.

In case of artificial datasets, we know the exact density function p and hence we can compute the average L_2 error

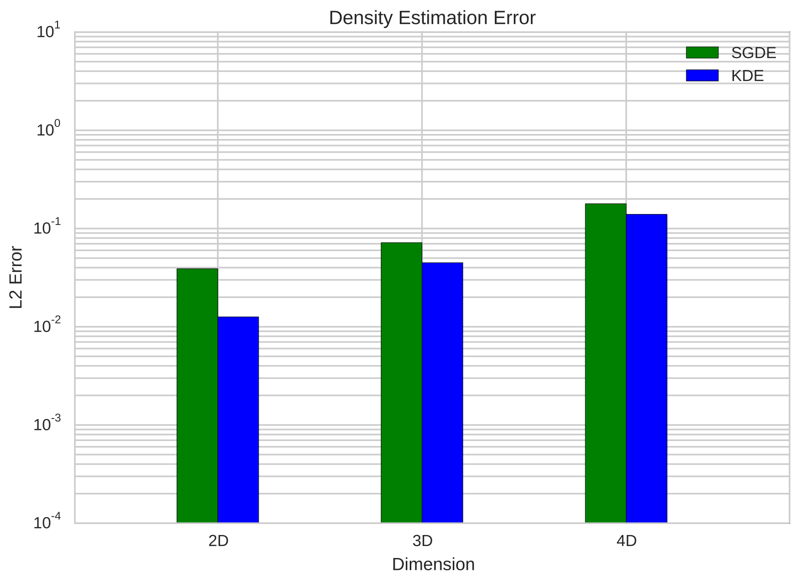
$$\frac{1}{M}\sqrt{\sum_{x \in D}(p(x) - \hat{p}(x))^2}, \tag{26}$$

where M is the number of datapoints in the dataset D.

Figure 7(a) shows the comparision of L_2 error for two multivariate normal datasets of different magnitude. The blue and green bars in the graph indicate the error measure for datasets with $M = 1000$ and $M = 100,000$ datapoints respectively. Note that the error corresponding to the dataset with more number of datapoints, is lower. This infers that our method estimates a better model when supplied with more data.



(a)



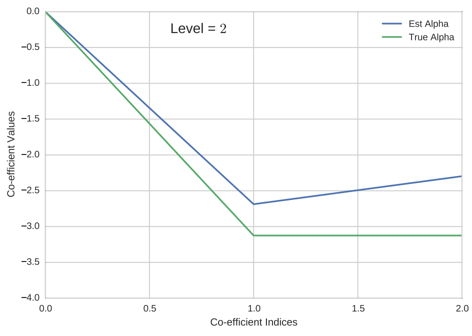
(b)

Figure 7: Comparison of our density estimation method based on sparse grids (SGDE) for datasets of different magnitude is shown in (a) and the comparision of SGDE with kernel density estimation (KDE) is show in (b). Dimensions of the datasets are marked on the x axes and y axes shows the L_2 error.

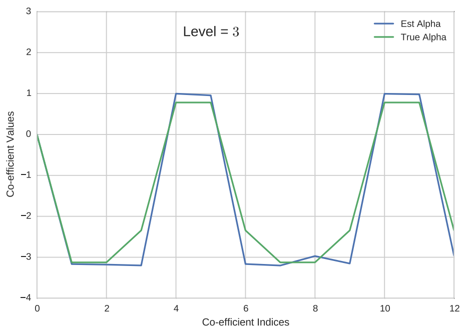
Since statistical tests like KS test are applicable only in one-dimensional cases, we need another method to evaluate the results obtained using SGDE algorithm for multivariate distribution. Hence we compare the results with kernel density estimation (KDE) for more than one-dimensional datasets. Note that we compare our results with KDE implemented in Python Scipy [11] package which uses Silverman's rule to choose the bandwidth parameter. Figure 7(b) shows the comparision of the density estimation method based on sparse grids and kernel density estimation in terms of L_2 error. The figure shows that our method is competitive with KDE even though we keep the regularization parameter λ fixed over all data sets and all dimensions. One more observation from these graphs is that the error increases significantly with the increase in the dimension of the dataset.

Validation of estimated coeffecients

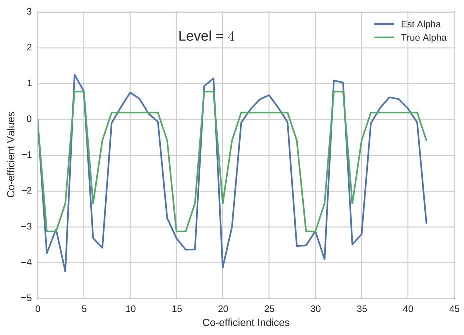
Since our algorithm is estimating the values of the coefficients corresponding to the basis functions of the sparse grid, the best way to validate our algorithm is to compare the estimated coefficient values with the true coefficient values. In case of artificial dataset sampled independently in different dimensions, it is possible for us to compute the true coefficient values. Figure 8 shows the comparision of the coefficient values of the datasets of dimensions one, two, three and four, each having $M = 5000$ datapoints.



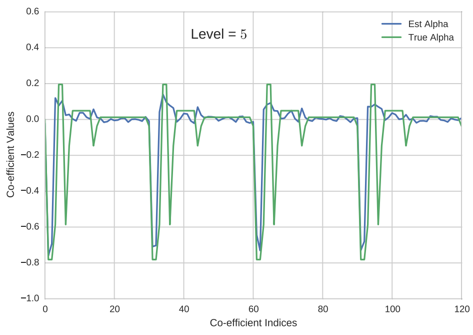
(a) one dimension



(b) two dimensions



(c) three dimensions



(d) four dimensions

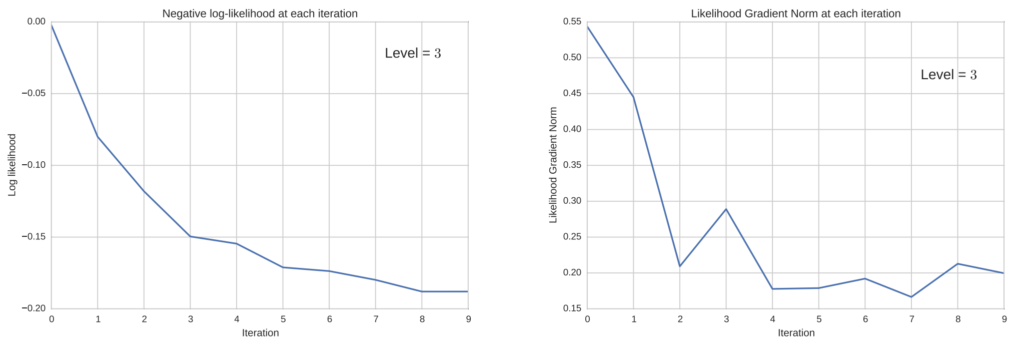
Figure 8: True and Estimated values of the coefficients.

As shown in figure 8, similarity exists between the estimated and true values of the coefficients. Based on these observations, we can conclude that the method employed to compute the expected value (23), is successful in estimating the values approximately. However, the similarity decreases with the increase in the dimension of the data set.

Convergence

Now we evaluate our method on two publicly available datasets. Since we do not know the exact density functions for these datasets, we analyze the behavior of the algorithm by observing the negative log likelihood and likehood gradient norm at each iteration.

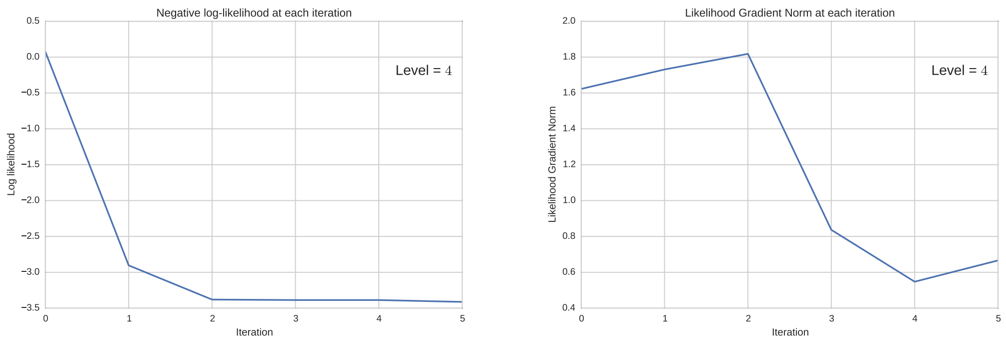
Ripley dataset is a two-dimensional dataset which consists of 250 datapoints [17].



(a) (b)

Figure 9: Negative Log-likelihood and Likelihood Gradient Norm per iteration for Ripley dataset.

The BUPA Liver Disorders data set from Irvine Machine Learning Database Repository is a six-dimensional dataset which consists of 345 datapoints [13].



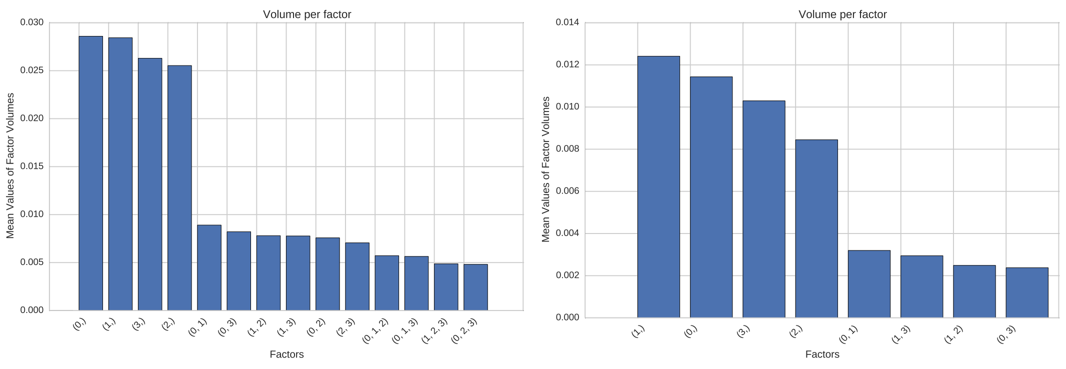
(a) (b)

Figure 10: Negative Log-likelihood and Likelihood Gradient Norm per iteration for BUPA Liver dataset.

For both these datasets, we can see that the negative log likelihood decrease sharply in first few iterations and at further iterations it does not change much. This is an indication that the algorithm has converged. The likelihood gradient norm on the other hand can be interpreted as the step size, which also decreases with the number of iterations however not as smooth as the log likelihood.

Structure of the dataset

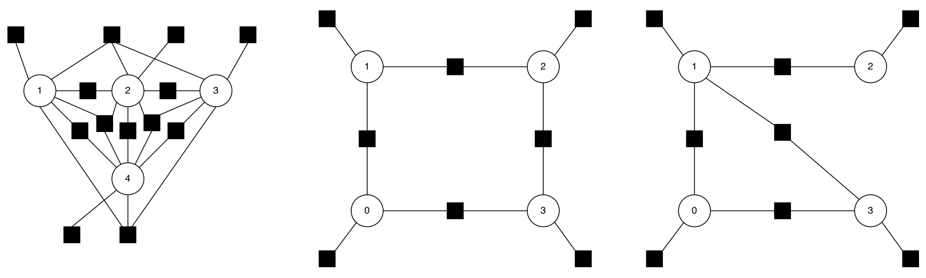
In Section 3.2 we explained strategies for coarsening the grid. Now we will see how exactly the algorithm measures the importance of the factors using the mean volume thresholding. In order to illustrate this, we generate four-dimensional dataset with a known structure. We initiate the Algorithm 2 with a 3-factor interaction factor graph model, shown in figure $12(a)$ and observe how the algorithm measures the importance of individual factors gradually deletes the less important ones.



(a) (b)

Figure 11: Volume per factor.

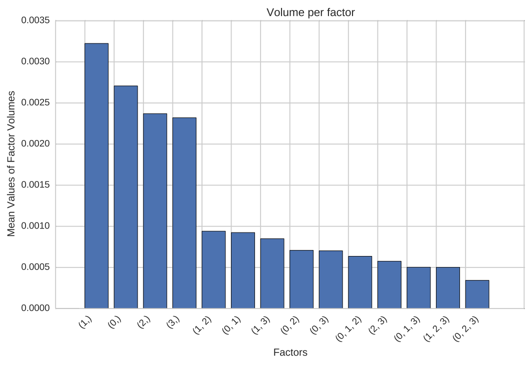
Figure 11(a) shows the volumes corresponding to each factor at the end of second iteration of our algorithm. At this stage we have not coarsened the grid. As expected, the volumes corresponding to the factors decreases as the number of interactions increases. Hence the higher order interactions are less important and are deleted first. Figure 11(b) shows the factors remaining at the end of all the iterations. Figure 12(b) shows the actual structure of the dataset and figure 12(c) shows the predicted structure. The algorithm was able to identify all the important factors except one. The volumes of the factors at every iteration is shown in figure 15 in Appendices.



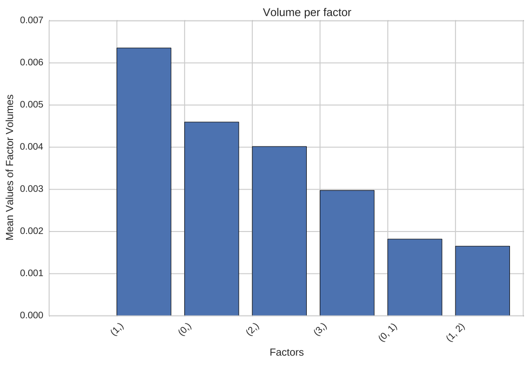
(a) Initial Structure (b) Actual Structure (c) Predicted Structure

Figure 12: Underlying structure of the dataset in the form of factor graph.

We perform one more experiment with a different structure as shown in figures 14(b) and the predicted structure is shown in figure 14(c). Figure 13(a) shows the volumes corresponding to each factor at the end of second iteration and figure 13(b) shows the factors remaining at the end of all the iterations. Even in this experiment, the algorithm was able to identify the important factors to a certain extent. The volumes corresponding to the factors at every iteration is shown in figure 16 in Appendices.

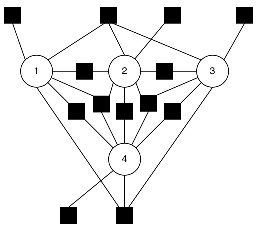


(a)

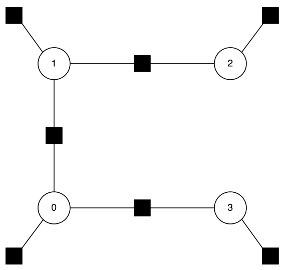


(b)

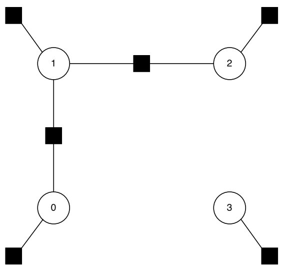
Figure 13: Volume per factor.



(a) Initial Structure



(b) Actual Structure



(c) Predicted Structure

Figure 14: Underlying structure of the dataset in the form of factor graph.

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, we devised and evaluated different methods to coarsen the grid, based on the importance, which improved the performance of sparse-grids based density estimation method. Second, we employed and evaluated a method based on Markov Chain Monte Carlo sampling, to estimate the expected values of sufficient statistics of the model. Third, we devised and evaluated a technique to predict the underlying structure of the dataset.

Next step would be to focus on improving the grid coarsening strategy by devicing a more accurate method to measure the importance of factors in the model. One more important improvement would be with respect to the regularization, which had a significant effect on the performance of the algorithm and hence needs to be experimented with.

References

[1] K. Balakrishnan. *Exponential Distribution: Theory, Methods and Applications*. Taylor & Francis, 1996.

[2] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1 edition, 1957.

[3] Hans-Joachim Bungartz and Michael Griebel. Sparse grids. *Acta Numerica*, 13:1–123, 2004.

[4] J. Garcke. Sparse grids in a nutshell. In J. Garcke and M. Griebel, editors, *Sparse grids and applications*, volume 88 of *Lecture Notes in Computational Science and Engineering*, pages 57–80. Springer, 2013.

[5] M. Griebel. *Sparse Grids and Related Approximation Schemes for Higher Dimensional Problems.* Sonderforschungsbereich 611, Singuläre Phänomene und Skalierung in Mathematischen Modellen. SFB 611, 2005.

[6] P. Hahnen. Nichtlineare numerische Verfahren zur multivariaten Dichteschätzung. Diplomarbeit, Institut für Numerische Simulation, Universität Bonn, November 2006.

[7] M. Hegland. Adaptive sparse grids. In K. Burrage and Roger B. Sidje, editors, *Proc. of 10th Computational Techniques and Applications Conference CTAC-2001*, volume 44, pages C335–C353, April 2003.

[8] Markus Hegland, Giles Hooker, and Stephen Roberts. Finite element thin plate splines in density estimation, 2000.

[9] M. Hollander and D.A. Wolfe. *Nonparametric Statistical Methods.* Wiley Series in Probability and Statistics. Wiley, 1999.

[10] Alan Julian Izenman. Recent developments in nonparametric density estimation. *Journal of the American Statistical Association*, 86(413):pp. 205–224.

[11] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed 2015-07-04].

[12] M. G. Kendall, A. Stuart, and J. K. Ord, editors. *Kendall's Advanced Theory of Statistics.* Oxford University Press, Inc., New York, NY, USA, 1987.

[13] M. Lichman. UCI machine learning repository, 2013.

[14] Kevin P Murphy. *Machine learning: a probabilistic perspective.* Cambridge, MA, 2012.

[15] J. Nocedal and S. Wright. *Numerical Optimization.* Springer Series in Operations Research and Financial Engineering. Springer, 2006.

[16] Benjamin Peherstorfer. *Model Order Reduction of Parametrized Systems with Sparse Grid Learning Techniques.* Dissertation, Department of Informatics, Technische Universität München, October 2013.

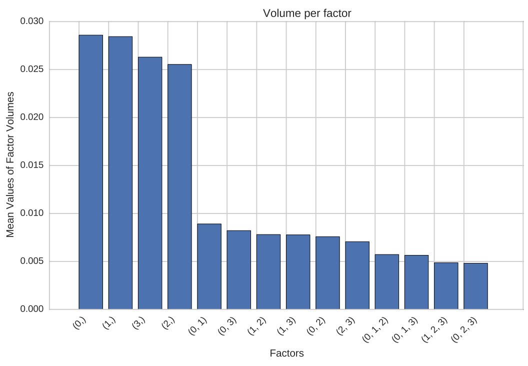
[17] Brian D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, 1996.

[18] B.W. Silverman. *Density Estimation for Statistics and Data Analysis.* Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1986.

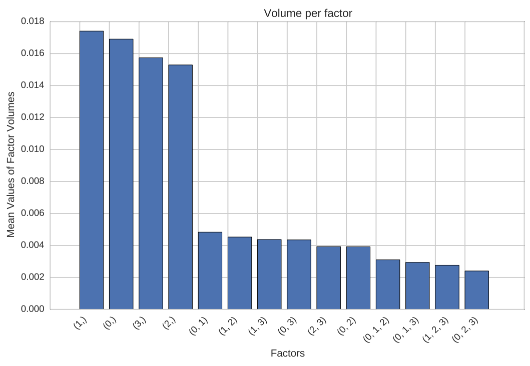
[19] Sebastian Soyer. Nonlinear density estimation with applications in astronomy. 2014.

[20] V.N. Vapnik. *Statistical learning theory.* Adaptive and learning systems for signal processing, communications, and control. Wiley, 1998.

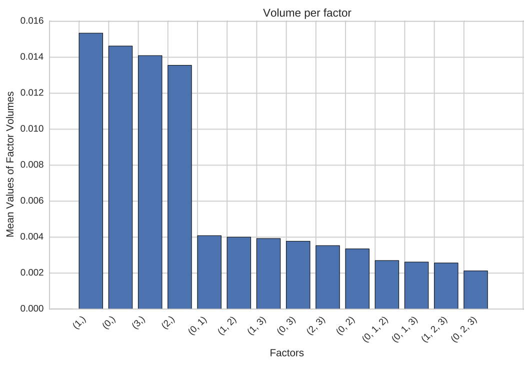
Appendices



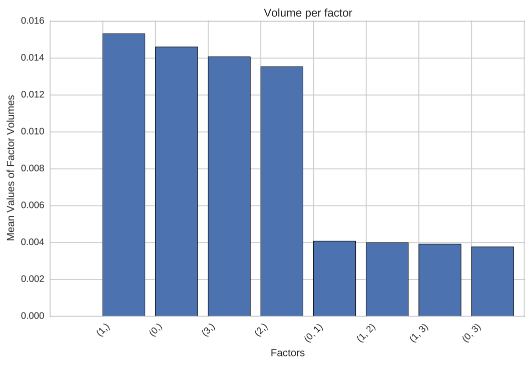
(a)



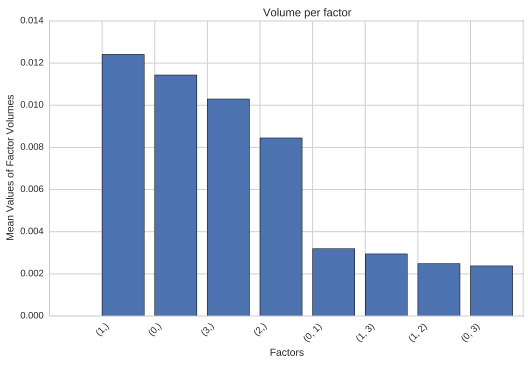
(b)



(c)

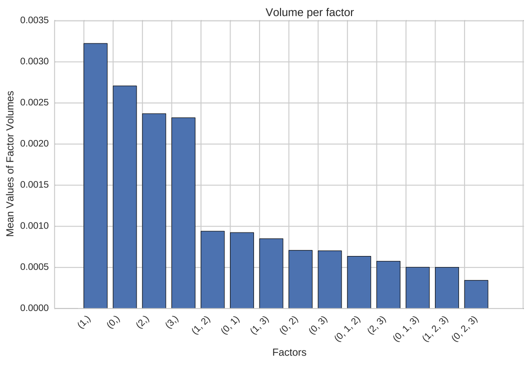


(d)

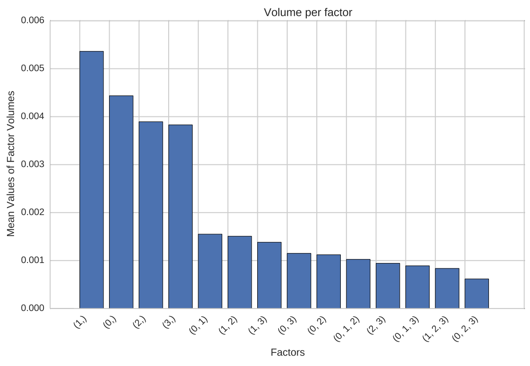


(e)

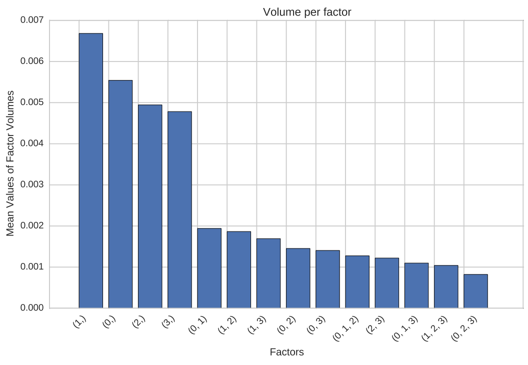
Figure 15: Volume per factor at different iteration steps.



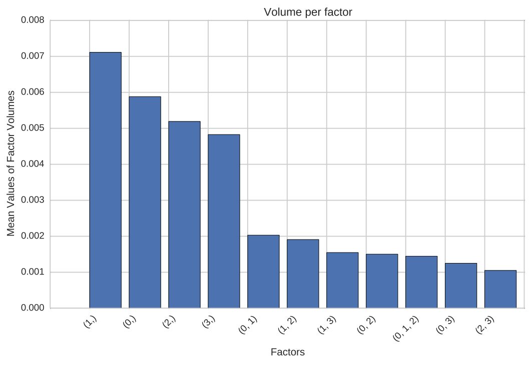
(a)



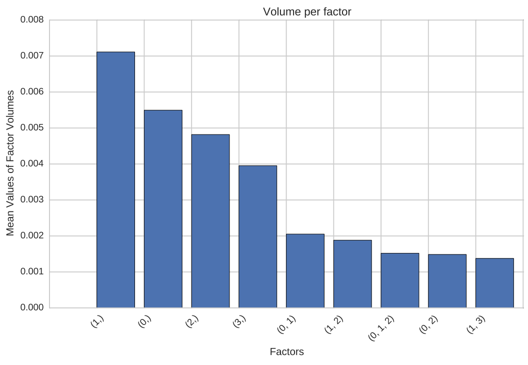
(b)



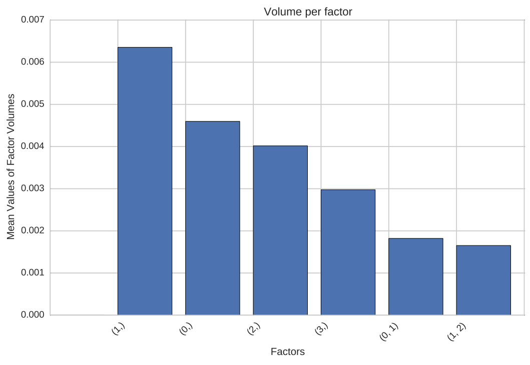
(c)



(d)



(e)



(f)

Figure 16: Volume per factor at different iteration steps.