Mixed IDEAs

Peter A.N. Bosman
peterb@cs.uu.nl

Dirk Thierens
Dirk.Thierens@cs.uu.nl

Institute of Information and Computing Sciences, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands

December 2000

Abstract

Building and using probabilistic models to perform stochastic optimization in the case of continuous random variables, has so far been limited to the use of factorizations as the structure of probabilistic models. Furthermore, the only probability density function (pdf) that has been successfully tested on a multiple of problems, is the normal pdf. The normal pdf however strongly generalizes the data and cannot cope with non–linear interactions among the samples. In this paper, we show how clustering algorithms can be used to overcome this problem. We also show how the normal mixture pdf can be used in the case of a general factorization instead of the normal pdf. We formalize the notion of a probabilistic model and propose to use two practical instances for the model structure, which are the factorization and the mixture of factorizations. We propose to use metrics to find good factorizations and thereby eliminate a complexity parameter κ that was required in previous continuous approaches in the case of a general factorization. We also show the background of the metrics through general model selection on the basis of likelihood maximization, which demonstrates their connection with previously used factorization selection algorithms. We use the IDEA framework for iterated density estimation evolutionary algorithms to construct new continuous evolutionary optimization algorithms based on the described techniques. Their performance is evaluated on a set of well known epistatic continuous optimization problems.

1 Introduction

The Iterated Density Estimation Evolutionary Algorithm (IDEA) framework has been used to apply probabilistic models to continuous stochastic optimization [8, 10, 11, 12, 13]. Algorithms within the IDEA framework, build a probabilistic model from a selection of samples and subsequently draw more samples from the probability distribution imposed by the probabilistic model. These samples are then incorporated among the currently available samples, after which selection takes place again. As this process is iterated, the algorithm is intented to converge to a good solution. Estimating and sampling from a probability distribution each iteration can be seen as a combination of the recombination and mutation steps in evolutionary algorithms. Since a selection of the available samples is used to create more solutions, the IDEA framework can be seen as an evolutionary algorithm with an explicit use of statistics.

In previous work [8, 12], the normal pdf and and the normal kernels pdf have been used as elementary probability density functions (pdfs). Using these pdfs, the estimated probability distribution over the complete domain of problem variables is constructed. The normal pdf has proven to be computationally effective and to result in acceptable optimization performance as well [10, 11]. The normal kernels pdf on the other hand has proven to be promising in optimization performance, but also to be very hard to handle [10]. Furthermore, its computational requirements are quite large. It seems that the use of a normal mixture pdf is a good trade–off between the normal pdf and the normal kernels pdf. The computational requirements of the normal mixture pdf are smaller than those of the normal kernels pdf. Furthermore, its parameters can be estimated using the EM algorithm, which eliminates the need for finetuning certain parameters by hand. In this paper, we show how the normal mixture pdf can be used within the IDEA framework.

The normal pdf itself is unable to describe non–linear interactions in a sample set. The normal mixture pdf can be used to overcome this problem. Another approach is to cluster the samples in a preprocessing phase and to use the normal pdf in each cluster. Assuming that the clustering phase effectively breaks up the non–linear interactions between the variables, this should result in an effective estimation of the selected samples. In a pilot study by Pelikan and Goldberg, clustering has been applied to optimization algorithms that use probabilistic models [25]. Here, we formalize the notion of clustering as part of the model selection process and investigate what clustering algorithms are effective in both computational running time as well as clustering performance.

In previous optimization algorithms that use continuous probabilistic models, only little attention has been paid to search metrics that guide the search for a good probabilistic model. In this paper, we propose to use metrics that effectively prefer simpler models if they can describe the samples in a similar manner as can more complex models. We are also show its correspondence with significance testing using statistical hypotheses and the notion of likelihood as a description of the goodness of an estimation.

We show how to use a higher level of modeling in the case of continuous probabilistic models using mixtures of distributions. Furthermore, we also provide useful search metrics to find continuous probabilistic models within iterated density estimation evolutionary algorithms. The combination of the proposed techniques leads to new algorithms. In this paper, we evaluate their performance using a set of test functions.

The remainder of this paper is organized as follows. In section 2 we formalize the notion of a probabilistic model. We also show how previous approaches have used a special instance for the probabilistic model structure, which is called the factorization. In section 3, we go into model selection. First, we discuss two approaches to model selection in general. Subsequently, we show how these approaches can be used to find a good factorization. Finally, we investigate the use of clustering algorithms to use a new higher order instance of probabilistic models in $\mathbb{IDEA}$s, which is a mixture of factorizations. In section 4, we show how the normal pdf and the normal mixture pdf can be fit to any factorization and how we can draw samples from them. Subsequently, in section 5, we go over the $\mathbb{IDEA}$ framework and show the difference between previous approaches and the new approaches described in this paper. We also elaborate on the difference of the $\mathbb{IDEA}$ approach and Evolution Strategies. In section 6, we test the new algorithms on a set of well known test functions. In section 7, we discuss some topics of importance and possible future research. Finally, this paper is concluded in section 8.

2 Probabilistic models

Given a vector of samples from a space defined by a cost function that without loss of generality we seek to minimize, our aim is to build and use a probabilistic model that describes these samples. The construction of this model should be effective in computation time requirements. Furthermore, the model itself should be effective in descriptive power. Once a model has been built, more samples are acquired using the probabilistic model. Subsequently, a new vector of samples is selected from all of the available samples, after which again a probabilistic model is built. As this process is iterated, the requirement that the computation time for model building should be reasonable, is clear as this is the main ingredient of our non–deterministic inductive search. On the other hand, the probabilistic model should efficiently describe the vector of selected samples in order for the algorithm to sample from the promising regions of the search space. If the model is not effective in its description, the regions may not be separated properly and the search does not propagate effectively over most of the regions. Instead, only a single region could for instance be explored and the algorithm is more likely to be mislead.

Before we can discuss how to effectively find a good probabilistic model, we have to define exactly what such a model consists of. In section 2.1 we therefore formally define our concept of probabilistic models that we shall use in our algorithms. One of the most fundamental concepts is the *factorization*. We elaborate on the use of this basic concept in section 2.2. Finally, in section 2.3 we give an overview of the use of probabilistic models in previous work. Also, we indicate how we shall expand this use to more complex probabilistic models and present a general overview that classifies previous work and the current work on the basis of the probabilistic model.

2.1 Inside a probabilistic model

We write $\boldsymbol{a} = (\boldsymbol{a}_0, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_{|\boldsymbol{a}|-1})$ for a vector $\boldsymbol{a}$ of length $|\boldsymbol{a}|$. Furthermore, we introduce the notation $a\langle \boldsymbol{c}\rangle = (a_{\boldsymbol{c}_0}, a_{\boldsymbol{c}_1}, \ldots, a_{\boldsymbol{c}_{|\boldsymbol{c}|-1}})$, meaning that $(a\langle \boldsymbol{c}\rangle)_i = a_{\boldsymbol{c}_i}, i \in (0, 1, \ldots, |\boldsymbol{c}| - 1)$. We assume that the dimensionality of our source is l and write $\boldsymbol{\mathcal{L}} = (0, 1, \ldots, l-1)$. Given a vector of l–dimensional data points $\boldsymbol{\mathcal{S}} = (\boldsymbol{y^0}, \boldsymbol{y^1}, \ldots, \boldsymbol{y^{|\mathcal{S}|-1}})$, $\boldsymbol{y^i} = y^i\langle \boldsymbol{\mathcal{L}}\rangle = (y^i_0, y^i_1, \ldots, y^i_{l-1})$, we identify l continuous random variables Y_0, Y_1, …, Y_{l-1}. The probabilistic model that we define in this section, describes a probability distribution over $\boldsymbol{\mathcal{Y}} = (Y_0,\ Y_1,\ \ldots,\ Y_{l-1})$. The contents of the probabilistic model determines the complexity of the probability distribution.

As the probabilistic model describes a probability distribution, the iterated density estimation approach is to find a probabilistic model and to draw new samples from the probability distribution. To this end, we note that any probability distribution over continuous random variables consists of *probability density functions*. The probability density function (pdf) is the most fundamental part of a probability distribution and therefore of a probabilistic model. Let $\boldsymbol{a} \sqsubseteq \boldsymbol{\mathcal{L}}$, meaning that $\boldsymbol{a}$ contains only elements of $\boldsymbol{\mathcal{L}}$. As a basis, we first state the definition of the multivariate joint pdf. Let $dy\langle \boldsymbol{a}\rangle = \prod_{i=0}^{|\boldsymbol{a}|-1} dy_{\boldsymbol{a}_i}$. The multivariate joint pdf $f_{\boldsymbol{a}}(y\langle \boldsymbol{a}\rangle)$ over random variables $Y\langle \boldsymbol{a}\rangle$ can be written as:

$$\int_{b_0}^{c_0}\int_{b_1}^{c_1}\ldots\int_{b_{|\boldsymbol{a}|-1}}^{c_{|\boldsymbol{a}|-1}} f_{\boldsymbol{a}}(y\langle \boldsymbol{a}\rangle)dy\langle \boldsymbol{a}\rangle = P(Y\langle \boldsymbol{a}\rangle \in A) \tag{1}$$

$$\text{such that } \int\int\ldots\int f_{\boldsymbol{a}}(y\langle \boldsymbol{a}\rangle)dy\langle \boldsymbol{a}\rangle = 1,\ \ f_{\boldsymbol{a}}(\cdot) \geq 0, \text{ and } A = \left(\prod_{i=0}^{|\boldsymbol{a}|-1}[b_i, c_i]\right) \subseteq \mathbb{R}^{|\boldsymbol{a}|}$$

We write $P(Y\langle \boldsymbol{a}\rangle)(y\langle \boldsymbol{a}\rangle)$ for $f_{\boldsymbol{a}}(y\langle \boldsymbol{a}\rangle)$, making $P(Y\langle \boldsymbol{a}\rangle)$ a pdf. In this way, we can use the common probability notation $P(Y\langle \boldsymbol{a}\rangle)$ over random variables $Y\langle \boldsymbol{a}\rangle$ instead of writing $f_{\boldsymbol{a}}(y\langle \boldsymbol{a}\rangle)$ for a density function. Furthermore, the use of this notation allows us to catch the properties of probability distributions in both the elementary case of a single pdf as well as higher order cases in which the distribution is factorized. One such a definition is that of *conditional probability*, which will prove to be one of the most important. We let $\boldsymbol{b} \sqsubseteq \boldsymbol{\mathcal{L}}$ and define $\boldsymbol{a} \sqcup \boldsymbol{b}$ to be the splicing of the two vectors so that the elements of $\boldsymbol{b}$ are placed behind the elements of $\boldsymbol{a}$, giving $|\boldsymbol{a} \sqcup \boldsymbol{b}| = |\boldsymbol{a}| + |\boldsymbol{b}|$. Using the definition of multivariate probability, we can write conditional probability as:

$$P(Y\langle \boldsymbol{a}\rangle | Y\langle \boldsymbol{b}\rangle) = \frac{P(Y\langle \boldsymbol{a} \sqcup \boldsymbol{b}\rangle)}{P(Y\langle \boldsymbol{b}\rangle)} \tag{2}$$

Combining the multivariate joint pdf from equation 1 with the multivariate joint probability from equation 2, we can construct a probability distribution over $\boldsymbol{\mathcal{Y}}$. This can be done by specifying a single multivariate joint pdf, but this can also be done by constructing a *factorization*. Since the product of two pdfs is again a pdf, the probability distribution over $\boldsymbol{\mathcal{Y}}$ can be factored such that it becomes a product of multivariate pdfs (either joint or conditional). Actually, we can restrict any factorization to use only multivariate conditional pdfs from equation 2, since we have that:

$$P(Y\langle \boldsymbol{a}\rangle) = \prod_{i=0}^{|\boldsymbol{a}|-1} P(Y_{\boldsymbol{a}_i} | Y_{\boldsymbol{a}_{i+1}}, Y_{\boldsymbol{a}_{i+2}}, \ldots, Y_{\boldsymbol{a}_{|\boldsymbol{a}|-1}}) \tag{3}$$

When it is more beneficial to model the samples with a joint pdf $P(Y\langle \boldsymbol{a} \sqcup \boldsymbol{b}\rangle)$ than with a product of two joint pdfs $P(Y\langle \boldsymbol{a}\rangle)P(Y\langle \boldsymbol{b}\rangle)$, there is a dependency between variables $Y\langle \boldsymbol{a}\rangle$ and $Y\langle \boldsymbol{b}\rangle$. In this case, we speak of an *unconditional dependency*. For any pair of variables Y_i and Y_j, we have that it is either beneficial to use $P(Y_i|Y_j)$ instead of $P(Y_i)$ or it is not. If it is more beneficial to use the conditional probability, we have a *conditional dependency* between variables Y_i and Y_j.

The most notable difference between the two dependencies is that conditional dependencies are directed, whereas unconditional dependencies are not.

Even though we can express any multivariate joint pdf with multivariate conditional pdfs, it can still be beneficial to use only multivariate joint pdfs in a factorization. The reason for this is that conditional pdfs can be analytically and computationally more difficult to handle.

To find a good probabilistic model, we can thus for instance attempt to find a good factorization of the probability distribution and then fit a pdf over each multivariate joint or multivariate conditional pdf in the factorization. We have however refrained so far from giving the actual definition of a factorization. In section 2.2 we go into the details of factorizations and formalize them. Here, we shall restrict ourselves to an example of a factorization and subsequently use it in the definition of a probabilistic model.

We denote a factorization by $\mathfrak{f}$. An example of a factorization is $P_{\mathfrak{f}}(Y_0, Y_1, Y_2) = P(Y_0)P(Y_1, Y_2)$. Again, this is a pdf, but as we use a product of elementary pdfs, we refer to it as a probability distribution. On the other hand, the probability distribution that is factorized into a single factor $P_{\mathfrak{f}}(Y_0, Y_1, Y_2) = P(Y_0, Y_1, Y_2)$ is equal to the unfactorized probability distribution and is equal to an elementary pdf. Still, we assume for each element in the factorization that we fit a pdf over it. We call the resulting probability distribution a *factorized* probability distribution.

With the exception of the work by Pelikan [25], the approaches in the field of building and using probabilistic models have so far used a factorization as the structure of the probabilistic model. The usefulness of a factorization becomes apparent when we increase the dimensionality of the problem. For instance in the case of discrete variables, we estimate the full joint probabilities by counting the amount of times some combination occurs in the sample vector and divide it by the total amount of samples. One of these combinations does not have to be computed as it is one minus the sum over the probabilities of the other combinations. For l dimensions and binary random variables, this means that we have to estimate $2^l - 1$ parameters. However, if we bound the amount of variables that are allowed to interact in a joint probability distribution by κ, we get $2^\kappa - 1$ parameters. If κ is fixed, this amount becomes a constant. This exponential scaling behavior is the most important reason why a factorization can be useful. If the interactions between the problem variables can be effectively caught in a bounded complexity order probabilistic model, the optimization process is efficient. However, if the amount of parameters that has to be computed to fit the pdf is computationally expensive, such as exponential, optimization by using probabilistic models is inherently a non–scalable approach. Note that whether or not this is possible, depends both on the algorithm that searches for a factorization as well as the optimization problem.

A factorization of a probability distribution along with the parameters for the pdfs to fit over the factors, could thus serve as the definition of a probabilistic model. Even though this definition has been used in most approaches so far, it does not allow for modeling at a higher level, being a mixture of factorizations. Such a mixture allows to apply a clustering strategy first, after which a factorization can be found in each cluster separately. This way, non–linearity and symmetry in the sample vector can effectively be tackled and processed.

Even though a mixture of factorizations is a very general structure that allows for many probabilistic models, we use a more general notation that allows for any type of structure. In this paper, the instances of the structure in the probabilistic model so defined, will be the factorization and the mixture of factorizations. In general, we define a probabilistic model to have some sort of structure ς and a vector of parameters $\boldsymbol{\theta}$ for the pdfs as implied by ς. We call ς the *probabilistic model structure* and $\boldsymbol{\theta}$ the *probabilistic model parameters*. As noted before, ς is likely to be constrained as using a maximum complexity structure does not scale up. Therefore, if ς is a factorization, it is usually constrained to bounded order interactions between variables. We denote the space of all ς that are allowed within such constraints by $\mathfrak{C}_\varsigma$. Usually, such constraints are not placed on the parameters $\boldsymbol{\theta}$, but we denote the constrained space for the parameters by $\mathfrak{C}_{\boldsymbol{\theta}}$. Formally, a probabilistic model $\mathcal{M}$ can thus be defined as a tuple just as can be done for its complete constrained space $\mathfrak{C}$ of allowed models:

$$\mathcal{M} = (\varsigma, \boldsymbol{\theta}) \ \in \ \mathfrak{C} = (\mathfrak{C}_\varsigma, \mathfrak{C}_{\boldsymbol{\theta}}) \tag{4}$$

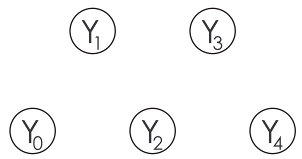

Figure 1: An empty factorization containing only the variables.

In the case that we use a single factorization for ς, we have $\varsigma = \mathfrak{f}$. We shall also make use of a mixture of factorizations. We denote the amount of components in the mixture by k and let $\boldsymbol{\mathcal{K}} = (0, 1, \ldots, k-1)$ be a vector of k numbers. The probabilistic model structure in the case of a mixture of factorizations is a vector of such factorizations $\varsigma = \boldsymbol{\mathfrak{f}} = (\mathfrak{f}_0, \mathfrak{f}_1, \ldots, \mathfrak{f}_{k-1}) = \mathfrak{f}\langle\boldsymbol{\mathcal{K}}\rangle$.

With any probabilistic model $\mathcal{M}$, a probability distribution is associated. This probability distribution is a composition of pdfs. The composition as well as the parameters themselves can defined by the model. As our goal is to use the probabilistic model in optimization, we have to approximate the probability distribution of the given sample vector $\boldsymbol{\mathcal{S}}$. The resulting probability distribution that is based on the model that we find, is an approximation to the *true* probability distribution. Therefore, we write the probability distribution implied by model $\mathcal{M}$ as $\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})$, which is an approximation to the true underlying probability distribution $P(\boldsymbol{\mathcal{Y}})$ over $\boldsymbol{\mathcal{S}}$.

The amount of parameters $|\boldsymbol{\theta}|$ depends on the choice of the pdfs to use as well as on the structure ς. However, the pdf to fit over every factor implied by ς is chosen on beforehand. The way in which the parameters $\boldsymbol{\theta}$ are fit, is also predefined on beforehand. We denote the parameter vector that is obtained in this manner by $\boldsymbol{\theta} \xleftarrow{fit} \varsigma$. The amount of parameters $|\boldsymbol{\theta}|$ is thus completely determined by the structure ς. Therefore, we will write $\hat{P}_{\varsigma}(\boldsymbol{\mathcal{Y}})$ instead of $\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})$ to indicate that the model is based on the structure ς and that the parameters $\boldsymbol{\theta} \xleftarrow{fit} \varsigma$ that are implied by the structure given preselected pdfs to use, are fit using a predefined method.

Now that we have formally defined the contents of a probabilistic model, we can define algorithms that construct such a probabilistic model and sample from the resulting probability distribution to achieve an iterated search procedure. However, we have already remarked that we shall restrict our attention to the use of factorizations and combinations thereof. To be able to build factorizations, we first elaborate on them in the next section.

2.2 Factorizations and their graphs

A factorization implies a product of pdfs. If such a product is a function of variables $y\langle\boldsymbol{a}\rangle$, it is a valid probabilistic model structure over random variables $Y\langle\boldsymbol{a}\rangle$. The product itself is closely related to the dependencies between the variables. We can start by noting that two variables are either unconditionally dependent or they are not. Furthermore, we may state that variable Y_i is either *conditionally dependent* on variables $Y\langle\boldsymbol{a}\rangle$ with $i \notin \boldsymbol{a}$ or it is not. These relations constitute a *factorization*. This structure can be modeled using graphs. If we identify a vertex for every random variable Y_i and introduce an arc (Y_i, Y_j) if and only if Y_j is conditionally dependent on Y_i, we get the conditional factorization graph. In the case of unconditional dependence, a connected component in the graph implies that all variables in the connected component are jointly dependent. As such, a connected component models a multivariate joint pdf.

A conditional dependence arc imposes a conditional probability in the probability distribution over $\boldsymbol{\mathcal{Y}}$. For example, figure 1 shows an empty factorization over 5 variables. The underlying factorization can be specified as $P(\boldsymbol{\mathcal{Y}}) = P(Y_0)P(Y_1)P(Y_2)P(Y_3)P(Y_4)$. Using only unconditional dependencies, figure 2 is an example of an unconditional factorization graph that models $P(\boldsymbol{\mathcal{Y}}) = P(Y_0, Y_2)P(Y_1, Y_3, Y_4)$. As a final example, figure 3 shows a factorization graph using conditional dependencies that models $P(\boldsymbol{\mathcal{Y}}) = P(Y_0|Y_2)P(Y_1|Y_2Y_3Y_4)P(Y_2|Y_3Y_4)P(Y_3|Y_4)P(Y_4) = P(Y_0|Y_2)P(Y_1, Y_2, Y_3, Y_4)$. The latter inequality holds because of equation 3.

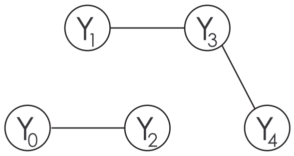

Figure 2: A non–empty unconditional factorization representing only joint dependencies.

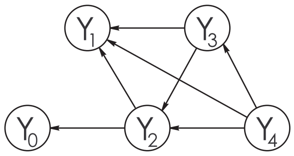

Figure 3: A non–empty conditional factorization representing conditional dependencies.

It follows from equation 3 that using unconditional factorizations that imply joint probabilities, is a special case of using conditional factorizations. Using these conditional factorizations, we can specify any factorization. Note that a factorization is valid if and only if its factorization graph is *acyclic*. This can be formalized [8] by introducing a function $\pi(\cdot)$ that returns a vector of parent variable indices $\pi(i) = (\pi(i)_0, \pi(i)_1, \dots, \pi(i)_{|\pi(i)|-1})$. The vector $\pi(i)$ denotes the indices of the variables that Y_i is conditionally dependent on. Furthermore, we introduce a vector of ordering variable indices $\boldsymbol{\omega} = \omega\langle\boldsymbol{\mathcal{L}}\rangle$. Using $(\pi, \boldsymbol{\omega})$, constraints can be specified to enforce that the factorization graph is acyclic. Furthermore, scanning the variables in the order $Y_{\omega_{l-1}}, Y_{\omega_{l-2}}, \dots, Y_{\omega_0}$ ensures that the parent variables that some variable is conditioned on, will already have been regarded. A conditional factorization can be uniquely specified by a pair $(\pi, \boldsymbol{\omega})$ as follows:

$$P_{(\pi,\boldsymbol{\omega})}(\boldsymbol{\mathcal{Y}}) = \prod_{i=0}^{l-1} P(Y_{\omega_i} | Y\langle\pi(\omega_i)\rangle) \tag{5}$$

$$\text{such that} \quad \forall_{i\in\boldsymbol{\mathcal{L}}}\langle \omega_i \in \boldsymbol{\mathcal{L}} \wedge \forall_{k\in\boldsymbol{\mathcal{L}}-\{i\}}\langle \omega_i \neq \omega_k\rangle\rangle$$
$$\forall_{i\in\boldsymbol{\mathcal{L}}}\langle \forall_{k\in\pi(\omega_i)}\langle k \in \{\omega_{i+1}, \omega_{i+2}, \dots, \omega_{l-1}\}\rangle\rangle$$

Using this definition, the factorization in figure 3 can be specified as $\omega_0 = 0, \omega_1 = 1, \omega_2 = 2, \omega_3 = 3, \omega_4 = 4$ and $\pi(0) = (2)$, $\pi(1) = (2,3,4)$, $\pi(2) = (3,4)$, $\pi(3) = (4)$ and $\pi(4) = ()$.

If we use a product of multivariate joint pdfs, the factorization becomes a *marginal product model*. We define the *node vector* $\boldsymbol{\nu}$ to be a vector of vectors of connected component indices $\boldsymbol{\nu}_i = ((\boldsymbol{\nu}_i)_0, (\boldsymbol{\nu}_i)_1, \dots, (\boldsymbol{\nu}_i)_{|\boldsymbol{\nu}_i|-1})$. We write $\boldsymbol{\nu}_i = (\nu_0^i, \nu_1^i, \dots, \nu_{|\boldsymbol{\nu}_i|}^i)$, so $(\boldsymbol{\nu}_i)_j = \nu_j^i$. The vector $\boldsymbol{\nu}_i$ denotes the indices of variables that are jointly dependent. Any unconditional factorization can be uniquely specified using $\boldsymbol{\nu}$:

$$P_{\boldsymbol{\nu}}(\boldsymbol{\mathcal{Y}}) = \prod_{i=0}^{|\boldsymbol{\nu}|-1} P(Y\langle\boldsymbol{\nu}_i\rangle) \tag{6}$$

$$\text{such that} \quad \forall_{0\le i\le|\boldsymbol{\nu}|-1}\langle \boldsymbol{\nu}_i \sqsubseteq \boldsymbol{\mathcal{L}} \wedge \forall_{0\le j\le|\boldsymbol{\nu}|-1, j\neq i}\langle \boldsymbol{\nu}_i \sqcap \boldsymbol{\nu}_j = ()\rangle\rangle$$
$$\bigsqcup_{i=0}^{|\boldsymbol{\nu}|-1} \boldsymbol{\nu}_i = \boldsymbol{\mathcal{L}}$$

The above constraints enforce that the vector of available random variables $\mathcal{Y}$ is partitioned into $|\boldsymbol{\nu}|$ disjoint subvectors. Over each of these subvectors, a single multivariate joint pdf is fit to obtain the factorized probability distribution. Using this definition, the factorization in figure 2 can be specified as $\boldsymbol{\nu}_0 = (0, 2), \boldsymbol{\nu}_1 = (1, 3, 4), |\boldsymbol{\nu}| = 2$.

We conclude this section by noting that we have formalized two general types of factorizations. One is defined using multivariate conditional probabilities whereas the other is defined using multivariate joint probabilities. These are the two factorizations that we use:

$$\mathfrak{f} \in \{(\pi, \boldsymbol{\omega}), \boldsymbol{\nu}\} \tag{7}$$

2.3 An overview of things that were and of things to come

In previous work on continuous models [8, 10, 11, 12, 13, 22, 18, 30, 31], a factorization $\mathfrak{f}$ was used as the structure ς of the probabilistic model $\mathcal{M}$. In some cases, constraints in addition to the general acyclic graph constraints have been applied. Furthermore, with the exception of the work by Bosman and Thierens [12] and Gallagher, Fream and Downs [18], a single multivariate normal pdf was used. In the work by Bosman and Thierens [12], the normal kernels pdf is proposed. However, the normal kernels pdf is very hard to handle and has therefore not been put to practice as much as has been done for the normal pdf. Gallagher, Fream and Downs [18] propose the use of the flexible multivariate normal mixture pdf, but only with the univariate factorization in which every variable is regarded independent of the others. In this paper, we show how the multivariate normal mixture pdf can be used in the case of a general factorization.

Roughly, we can say that the research on the use of probabilistic models in continuous domains has been limited to models where the structure is a factorization and the parameters are those of a normal pdf (at least in the general factorization case). If we denote the normal pdf by $f_{\mathcal{N}}$, we can indicate this by writing $\varsigma = \mathfrak{f}$ and $\boldsymbol{\theta} \Leftarrow f_{\mathcal{N}}$. In this paper, we show how the normal mixture pdf can be used in the general factorization case, giving $\boldsymbol{\theta} \Leftarrow f_{\mathcal{N}_M}$.

By using the normal mixture pdf, we can apply a powerful density estimator in our optimization procedure. However, this does not necessarily allow for the breaking of symmetry or the fitting of non–linearity in the optimization problem. To achieve this, we propose to use a mixture of factorizations. To this end, we cluster the sample vector into subvectors and apply density estimation to each subvector. We thereby change the structure of the model to $\varsigma = \mathfrak{f}\langle\boldsymbol{\mathcal{K}}\rangle$.

Figure 4 graphically depicts the previous work and the work that we describe in this paper. The approaches in the top row make use of a single factorization $\mathfrak{f}$. This is what has been done so far for continuous optimization problems. In the case of a general factorization, the approaches have been limited to the use of a single normal pdf, which is the top left entry in the table in figure 4. The approaches in the right column make use of the normal mixture pdf. The approaches in the bottom row use a mixture of factorizations. In this paper, we use clustering algorithms to partition the sample vector into subvectors. A factorization is found for each so constructed subvector, thereby achieving the mixture of factorizations. Concluding, in the general case, the approaches so far for continuous optimization problems have used the top left entry in the table in figure 4. In this paper, we show how the remaining three entries in the table of figure 2.3 can be established in search of an improvement over the approaches for continuous domains so far.

3 Model selection

Given a vector $\boldsymbol{\mathcal{S}}$ of data points, it very valuable to be able to infer from what probability distribution these data points were sampled. Given such information, we can draw conclusions on the source of these data points and make predictions for future data coming from similar sources. In some applications we are not concerned with finding the full probability distribution, but certain aspects of it under certain conditions. The use of linear regression to find out whether variables or events are correlated, is an example of such an approach. Analysis of this sort is applied whenever

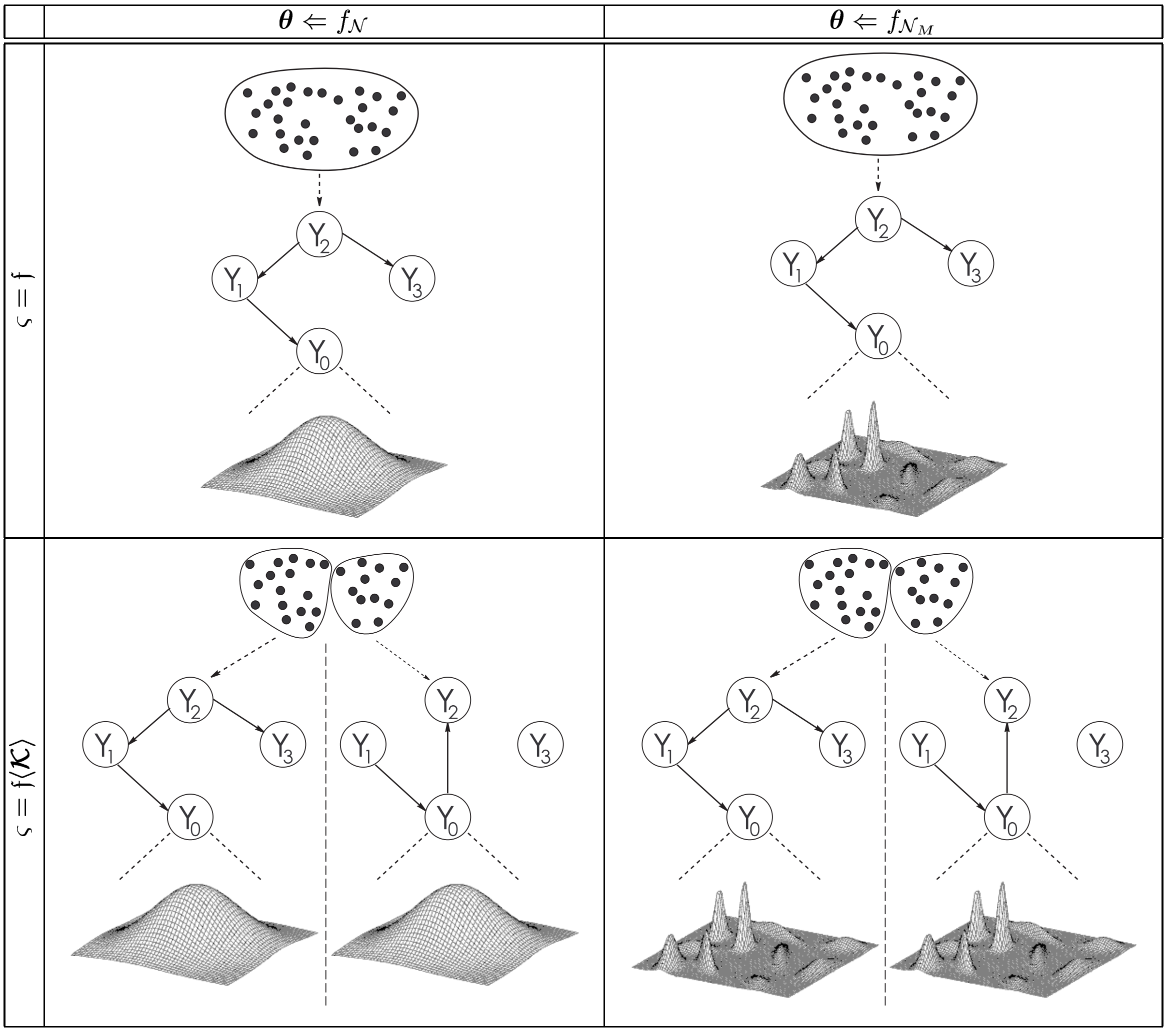

Figure 4: An overview of the algorithms that build and use probabilistic models. The top left entry represents the current algorithms in the case of a general factorization. The other entries represent algorithms that are proposed in this paper.

induction is to be performed, given certain observations. Real life examples are for instance observations of hospitalized patients and the question of whether one symptom is caused by another or whether a certain disease is always accompanied by high fever, so that this can be taken into account whenever a new patient with this disease arrives.

The least restrictive of these inductional questions, which is the desire to know the underlying probability distribution, is called *model selection*. The data points in vector $\boldsymbol{\mathcal{S}}$ are the result of sampling from some *model*. It is this model that we set out to find or approximate. To be more precise, we utilize this inferred model information in order to perform global optimization.

We start out in section 3.1 by going over two general model selection techniques. In section 3.2, we formulate an algorithm that systematically and incrementally finds a factorization given a vector of samples. Subsequently, in section 3.3, we go into the selection of a mixture of factorizations using clustering techniques.

3.1 General techniques

In order to decide what model we wish to use to represent the data with, many automated approaches can be used. Depending on the choice of the pdf, specialized statistical tests can be derived to find a factorization for example. In this section however, we wish to refrain from selecting any instances or additional constraints for the model. Instead, we give two general approaches to selecting an appropriate model based upon the data.

In this paper, we see model selection as an iterative process. In any iteration, we have some candidate models that may possibly replace the current model. In section 3.2 we show how such an iterative algorithm can be formalized in order to select a model with $\varsigma = \mathfrak{f}$. Here, we concern ourselves with a single iteration of such an algorithm.

From a set of candidate models, we wish to select the most promising one to replace the current model. If no candidate model seems to be promising, the iterative algorithm halts and the current model is taken to be the result of model selection. We assume that a set of candidate models for some given model is available. In section 3.2 we shall show how such a set can be obtained in the case where we want to find a suitable factorization. In the iterative process, we denote the current model by $\mathcal{M}^0$. In order to select the best model from the set of candidates, we go over each of the candidate models and compare them to $\mathcal{M}^0$. In the remainder of this section, we focus on one such comparison and denote the candidate model by $\mathcal{M}^1$.

In order to compare $\mathcal{M}^1$ to $\mathcal{M}^0$, we distinguish two approaches. One approach is to test whether the use of $\mathcal{M}^1$ is a *significant* improvement over $\mathcal{M}^0$ in representing $\boldsymbol{\mathcal{S}}$. To this end, statistical hypothesis tests can be used. A gentle introduction to statistical hypothesis testing is given in appendix B. We define the general statistical hypothesis test for the comparison of the models as follows:

> **Model Selection Test Hypothesis (MSTH)**
> *Probabilistic model $\mathcal{M}^1$ cannot better describe the given samples than can probabilistic model $\mathcal{M}^0$.*

We can use statistical techniques to test the MSTH. Such a test returns whether or not the hypothesis should be accepted. A certain statistic is shown to follow some distribution. A hypothesis is tested by using so called *critical values* for the statistic, given a certain *significance threshold*. The MSTH is accepted if the statistic is for instance below a critical value associated with the threshold or rejected if it is above the critical value. In such a case we speak of a right–sided test.

Statistical hypothesis tests can be used to select a replacement model from the candidate set. As the test returns whether $\mathcal{M}^1$ is a better descriptive model than is $\mathcal{M}^0$ for the given samples, we can go over the candidates and take the first model from the candidate set to replace $\mathcal{M}^0$ that results in rejecting the MSTH. However, the set of candidates will be generated in an automated fashion. To prevent going over the candidates in the same manner every time so that possibly a certain type of candidate model is always tested before some other type of candidate model, the candidates should be accessed in a random order. In section 3.1.1 we show how a general instance for the MSTH can be formalized based on the likelihood of a probabilistic model.

The second approach to comparing $\mathcal{M}^1$ with $\mathcal{M}^0$ is to use some metric. If we have a metric that informs us of how well some model represents $\boldsymbol{\mathcal{S}}$, we can search amongst the candidate models to find the model that results in the largest increase or decrease of the metric, depending on whether we have to respectively maximize or minimize it. If the metric can no longer be improved upon since every candidate leads to a degradation of the metric, the iterative model selection procedure can be halted. This approach has become a very popular one, especially within the field of Bayesian statistics. In section 3.1.2 we discuss a general, transparent and useful metric that has a clear correspondence with the MSTH instance in section 3.1.1.

Before moving to describe both a general MSTH as well as a metric, we note that an MSTH can actually also serve as a metric. Assuming such a statistic ζ, a critical value ζ_α and a right–sided test, a statistical hypothesis test returns $\boldsymbol{true}$ if and only if $\zeta \geq \zeta_\alpha$. However, the amount that ζ is larger (or smaller, depending on the type of test) than ζ_α can be seen as a measure of how strongly the MSTH is rejected. This can be used in some sort of a metric for which we may find the maximum over all candidate models. The metric to maximize is then equal to $\zeta - \zeta_\alpha$.

3.1.1 Negative log–likelihood statistical hypothesis testing

In this section, we focus on finding a general statistical hypothesis test so as to perform model selection by testing the MSTH. In appendix B more details are presented on statistical hypothesis testing in general. Also an example is given of how unconditional dependencies can be tested using well known statistical measures. Here however, we focus on a hypothesis test that is directly based on the goodness of two fits.

Assuming that the samples were drawn independently from the underlying distribution, the *likelihood* of the samples, given the estimated probability distribution $\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})$, is defined as:

$$\mathfrak{L}(\boldsymbol{\mathcal{S}}|\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})) = \prod_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})(\boldsymbol{y}^i) \tag{8}$$

It is common practise to use the negative logarithm of the likelihood measure as it is often computationally more convenient. The resulting expression is called the *negative log–likehood*:

$$-\ln(\mathfrak{L}(\boldsymbol{\mathcal{S}}|\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}}))) = -\sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \ln(\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})(\boldsymbol{y}^i)) \tag{9}$$

Momentarily disregarding generalization, in order to find the maximum likelihood probabilistic model $\mathcal{M}$, we have to *maximize* $\mathfrak{L}(\boldsymbol{\mathcal{S}}|\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}}))$ or, equivalently, to *minimize* $-\ln(\mathfrak{L}(\boldsymbol{\mathcal{S}}|\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})))$. The simplest probabilistic model that best fits the equations, can be defined using a single multivariate joint pdf over a full joint factorization for $\mathfrak{f} = \varsigma$:

$$\int_{a_0}^{b_0}\int_{a_1}^{b_1}\cdots\int_{a_{l-1}}^{b_{l-1}} \hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})(y\langle\boldsymbol{\mathcal{L}}\rangle)d_{y\langle\boldsymbol{\mathcal{L}}\rangle} = \frac{\boldsymbol{\mathcal{S}} \sqcap \prod_{i=0}^{l-1}[a_i, b_i]}{|\boldsymbol{\mathcal{S}}|} \tag{10}$$

Clearly, the pdf so constructed is too specific with respect to the sample vector and is therefore said to *overfit* the data. This is a very important aspect of density estimation and is termed *generalization*. Overfitting tends to occur not only in such specific cases as in equation 10 but also often in cases where a combination of many pdfs is used, such as a kernel method. If overfitting takes place, the optimization algorithm that samples more points from the pdf will tend to converge rapidly into a fixed form and give poor results [10]. Therefore, a parametric model with little parameters such as the normal pdf, is usually preferred as it cannot easily overfit a set of samples.

Given a parametric model and thus a set of parameters for the involved pdfs, a general iterative method can be derived to estimate the set of pdf parameters so as to minimize the negative log–likelihood and get a maximum likelihood fit of the estimated probability distribution. Such a method is given by the EM–algorithm, which we discuss in more detail in section 4.3.2. In the EM–algorithm however, the model structure is fixed on beforehand. In order to find the model structure, we can test whether some candidate model $\mathcal{M}^1$ has a lower negative log–likelihood value than the current model $\mathcal{M}^0$. If this is the case, $\mathcal{M}^1$ is a better description than is $\mathcal{M}^0$. Therefore, $\mathcal{M}^1$ should replace $\mathcal{M}^0$. In order to perform the negative log–likelihood test, the involved pdfs first have to be computed. To test whether the likelihood of the probability distribution implied by model $\mathcal{M}^1$ is larger than that of the probability distribution implied by model $\mathcal{M}^0$, we observe:

$$\sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \ln(\hat{P}_{\mathcal{M}^1}(\boldsymbol{\mathcal{Y}})(\boldsymbol{y^i})) - \sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \ln(\hat{P}_{\mathcal{M}^0}(\boldsymbol{\mathcal{Y}})(\boldsymbol{y^i})) \tag{11}$$

If we are given maximum likelihood estimates and if the parameters of model $\mathcal{M}^1$ are a superset of those of model $\mathcal{M}^0$, we have that the negative log–likelihood difference as stated in equation 11 is *always* larger than or equal to 0. This implies that if we incrementally build a model by increasing its complexity, we will always prefer the more complex model if we are able to fit the data with a maximum likelihood. This is however not desirable. If the negative log–likelihood difference is only very small, the more complex model does indeed fit the data slightly better, but this comes at the expense of a larger running time since the model is more complex. By selecting the more complex model, we require a lot more time to perhaps gain a hardly noticeable difference in goodness of fit. This poses a well known trade off between running time and expressional power.

What we require, is to be able to test whether some negative log–likelihood difference is *significant*. If it is, this implies that we should select the more complex model as it would significantly improve the resulting fit. We have then *justified* the selection of the more complex model. To this end, we use a statistical hypothesis test. If we have some random variable R along with N values $r_0, r_1, \ldots, r_{N-1}$ for it, its sample mean $\overline{R}$ and its unbiased sample standard deviation $\tilde{s}_R$ are:

$$\overline{R} = \frac{1}{N}\sum_{i=0}^{N-1} r_i, \qquad \tilde{s}_R = \sqrt{\frac{1}{N-1}\sum_{i=0}^{N-1}(r_i - \overline{R})^2} \tag{12}$$

Because of the central limit theorem, the sample mean is approximately normally distributed. If we now assume to have two of such random variables R^0 and R^1 along with an hypothesized value $\mathfrak{h}$ for the difference of the means, we observe:

$$\mathbb{T} = \frac{\sqrt{N}(\overline{R^0} - \overline{R^1} - \mathfrak{h})}{\sqrt{\tilde{s}^2_{R^0} + \tilde{s}^2_{R^1}}} \tag{13}$$

It can be shown that the $\mathbb{T}$ statistic (see for instance [14]) is distributed according to Student's T distribution (see appendix B) with $\delta = 2(N-1) - |\boldsymbol{\theta^0}| - |\boldsymbol{\theta^1}|$ degrees of freedom, where $\boldsymbol{\theta^i}$, $i \in \{0, 1\}$ is the vector of parameters in model $\mathcal{M}^i$ that have to be estimated.

Now let R^j stand for the negative log–likelihood of the samples under probability distribution $\hat{P}_{\mathcal{M}^j}$, $j \in \{0, 1\}$. We then thus have that:

$$r^j_i = -\ln(\hat{P}_{\mathcal{M}^j}(\boldsymbol{\mathcal{Y}})(\boldsymbol{y^i})) \tag{14}$$

We can now use the $\mathbb{T}$ statistic to test whether the average value of the negative log–likelihood difference is significantly larger than 0. If this is so, candidate model $\mathcal{M}^1$ is preferred over model $\mathcal{M}^0$. To do this, we set $\mathfrak{h} = 0$ and perform a right–sided test. This means that we are testing whether the expected value of $R^0 - R^1$ is greater than 0. Since $\frac{1}{|\boldsymbol{\mathcal{S}}|}x \geq 0 \Leftrightarrow x \geq 0$, we are thus testing whether the difference in equation 11 is significantly larger than 0.

Placing this this test in an iterative algorithm gives us a model selection approach in which each step towards a more complex model is justifyable. However, in all previous continuous approaches that use factorizations of a higher order than the univariate factorization as the model structure, such justification has not been used directly. Instead, the Kullback–Leibler (KL) divergence from the approximated factorization to the full joint approximated factorization is used. By doing so, it can be shown [13] that the the differential entropy over the approximated factorization is minimized. Given maximum likelihood estimates, it can furthermore be shown [13] that the differential entropy difference is the same as the negative log–likelihood difference from equation 11.

The equality of the KL divergence and the negative log–likelihood difference under the assumption of a maximum likelihood estimate, implies that the previous approaches can be seen to have used the negative log–likelihood without justifying any model selection decisions. However, unless strong restrictions are imposed on the probabilistic model, using this metric will unnecessarily increase the complexity since any more complex model will always result in a fit with a lower negative log–likelihood. We can therefore conclude that justification is truly required in order for the resulting optimization algorithm to be scalable on decomposable problems.

If we do not use any justification, we can refrain from using the average negative log–likelihood but use the differential entropy instead. Computing the average negative log–likelihood is a time consuming task as it requires $|\boldsymbol{\mathcal{S}}|$ evaluations. However, for the normal pdf, the differential entropy over variables $Y\langle\boldsymbol{a}\rangle$ can be shown (see for instance [15]) to be $\frac{1}{2}(|\boldsymbol{a}|+\ln((2\pi)^{|\boldsymbol{a}|}(\det \boldsymbol{S}(\boldsymbol{a}))))$, where $\boldsymbol{S}(\boldsymbol{a})$ is the sample covariance matrix. However, in this section we *do* require justification. In such a case, we cannot suffice by only computing the average negative log–likelihood, since we also require the variance over the negative log–likelihood to compute the test statistic $\mathbb{T}$. Still, we shall see in section 3.2.3 that this remark can yet be of practical concern.

For more involved pdfs such as the normal mixture model, we cannot efficiently compute the differential entropy. Furthermore, neither can we guarantee a maximum likelihood estimate using the EM algorithm. To compute the negative log–likelihood difference, we thus have to evaluate the implied probability distribution for each sample.

3.1.2 Complexity penalization

In the previous section, we have seen that we can minimize the error of the fit and justify each minimization based upon a large enough increase in the negative log–likelihood difference. However, even though one fit may indeed be better than the other, this does not mean that we are interested in using it from either a generalization or a computational effectiveness point of view. To enforce this, a penalty term that increasingly penalizes more complex models can be introduced. The metric M that should be minimized can then be formalized as follows:

$$M(\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})|\boldsymbol{\mathcal{S}}) = \text{Error}(\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})|\boldsymbol{\mathcal{S}}) + \text{Complexity}(\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})|\boldsymbol{\mathcal{S}}) \tag{15}$$

The use of a complexity term is often termed *regularization* in data modeling fields. Regularization is used to prevent the model from overfitting the data. Classicly, the regularization equation has a *regularization parameter* λ as a factor times the regularization term. Setting $\lambda = 0$ then reduces the metric to an unconstrained one with respect to error minimization. Here, we have used a general complexity term that could be of such a form. However, we leave the actual contents completely unspecified, which gives no restrictions on the complexity term. Furthermore, the metric from equation 15 is thereby completely transparent from the start.

To derive a metric of the form as we have in equation 15, we note that we are interested in the probability of a probability distribution implied by a model $\mathcal{M}$, given a sample vector $\boldsymbol{\mathcal{S}}$. The model that implies a probability distribution with the largest probability, is the most preferable one if we disregard complexity and generalization at the moment. The probability that we are interested in, can be written as follows using Bayes' Rule:

$$P(\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})|\boldsymbol{\mathcal{S}}) = \frac{P(\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}}))P(\boldsymbol{\mathcal{S}}|\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}}))}{P(\boldsymbol{\mathcal{S}})} \tag{16}$$

As the sample vector $\boldsymbol{\mathcal{S}}$ is fixed and thus $P(\boldsymbol{\mathcal{S}}) = 1$, we can discard $P(\boldsymbol{\mathcal{S}})$ from equation 16. The probability of $\boldsymbol{\mathcal{S}}$ given some probability distribution $\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})$ implied by a model $\mathcal{M}$, can be seen as the likelihood as defined in equation 8, meaning $P(\boldsymbol{\mathcal{S}}|\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})) = \mathfrak{L}(\boldsymbol{\mathcal{S}}|\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}}))$. In order to add a preference for simpler models, we want to set the probability at a probability distribution $P(\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}}))$ implied by a model $\mathcal{M}$ in such a way that it gets smaller if its complexity increases. The complexity of $\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})$ is given by the amount of parameters $\boldsymbol{\theta}$ that is required to define every pdf in the model $\mathcal{M}$. We can define the probability of some probability distribution $P(\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}}))$ implied by $\mathcal{M} = (\varsigma, \boldsymbol{\theta})$ by using a function $\vartheta(\cdot)$, $\vartheta(\cdot) \geq 0$, of the amount of parameters $|\boldsymbol{\theta}|$:

$$P(\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})) = \frac{\vartheta(\boldsymbol{\theta})}{\sum_{\varsigma' \in \mathfrak{C}_\varsigma} \vartheta(\boldsymbol{\theta'} \xleftarrow{fit} \varsigma')} \tag{17}$$

If we for instance take each probabilistic model to be equally likely, we set $\vartheta(\cdot) = 1$ and get:

$$P(\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})) = \frac{1}{|\mathfrak{C}_\varsigma|} \prod_{i=0}^{|\boldsymbol{S}|-1} \hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})(\boldsymbol{y^i}) \tag{18}$$

In order to compare one model to another, the normalization factor $\sum_{\varsigma' \in \mathfrak{C}_\varsigma} \vartheta(\boldsymbol{\theta'} \xleftarrow{fit} \varsigma')$ in equation 17 may be discarded since this factor is the same for every $\mathcal{M}$. As a result however, we no longer have a probability distribution but a metric. Using the metric, we note that under the assumption that $\vartheta(\cdot) = 1$, equation 18 becomes just the likelihood from equation 8.

Alternatively, we can let the probability at some probability distribution $P(\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}}))$ implied by a model $\mathcal{M}$ decrease exponentially with its complexity by setting $\vartheta(\boldsymbol{\theta}) = e^{-|\boldsymbol{\theta}|}$. Denoting the resulting metric by $\mathfrak{A}(\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})|\boldsymbol{S})$, we get:

$$\mathfrak{A}(\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})|\boldsymbol{S}) = e^{-|\boldsymbol{\theta}|} \prod_{i=0}^{|\boldsymbol{S}|-1} \hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})(\boldsymbol{y^i}) \tag{19}$$

The objective is to maximize the metric. Alternatively, we can take the negative logarithm of the metric and minimize it. Just as is the case for the likelihood, using the negative logarithm is computationally more convenient. This can directly be seen from the definition of the metric, since the exponentials disappear and the product of a large amount of values in $[0, 1]$ becomes a sum. By doing so, we get an instance of the general metric expression in equation 15:

$$\underbrace{-\ln(\mathfrak{A}(\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})|\boldsymbol{S}))}_{M(\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})|\boldsymbol{S})} = \underbrace{-\ln(\mathfrak{L}(\boldsymbol{S}|\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})))}_{\text{Error}(\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})|\boldsymbol{S})} + \underbrace{|\boldsymbol{\theta}|}_{\text{Complexity}(\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})|\boldsymbol{S})} \tag{20}$$

In the metric that we have arrived at, the negative log–likelihood is the error to minimize and the complexity term is the amount of parameters in the probabilistic model. The resulting metric is similar to the Akaike Information Criterion (AIC) [1]. The AIC equals 2 times the expression in equation 20. This constant factor does not influence the result when computing the difference and checking whether it is larger than 0. So basically, we have arrived at the AIC metric. The AIC metric elegantly favors simpler models. If two probability distributions have the same likelihood, the one with the least amount of parameters is favored.

At this point, we turn back to our note on regularization theory. Even though the AIC metric favors simpler models by increasingly penalizing more complex ones, the amount of penalization is fixed. By using a regularization parameter λ times the AIC metric, the regularization could be increased. However, this would only be by a constant factor, assuming that λ is a free parameter $\lambda \in \mathbb{R}$. Other than by increasing some regularization parameter, we can also use a different metric that penalizes the complexity differently. The above derivation is a very general one. To demonstrate its transparent use, we propose to set $\vartheta(\boldsymbol{\theta}) = e^{-\lambda \ln(|\boldsymbol{S}|)|\boldsymbol{\theta}|}$. We have now introduced a parameter λ that has the same semantics as the regularization parameter. The resulting metric that we arrive at, can be written as follows:

$$\underbrace{-\ln(\mathfrak{B}(\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})|\boldsymbol{S}))}_{M(\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})|\boldsymbol{S})} = \underbrace{-\ln(\mathfrak{L}(\boldsymbol{S}|\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})))}_{\text{Error}(\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})|\boldsymbol{S})} + \underbrace{\lambda \ln(|\boldsymbol{S}|)|\boldsymbol{\theta}|}_{\text{Complexity}(\hat{P}_{\mathcal{M}}(\boldsymbol{\mathcal{Y}})|\boldsymbol{S})} \tag{21}$$

The metric in equation 21 is commonly known as the *Bayesian Information Criterion* (BIC) [29]. This metric has been proposed before in iterated density estimation approaches [24, 19, 27]. The BIC metric has empirically been observed to give good results. It penalizes more complex models more heavily than does the AIC metric, but in such a way that the most important structural building blocks are found. The AIC metric is less efficient because it tends to let such building blocks be merged as well, which gives an unnecessarily complex model structure.

Before we conclude this section, we note that there is a correspondence between the metrics in equations 20 and 21 and the general hypothesis test in the previous section. In both cases, the negative log–likelihood has to be computed. In the case of the hypothesis test, the average negative log–likelihood difference is compared to a critical value. In the case of the penalization metrics, the negative log–likelihood difference is compared to a value as well. In this case, the value is based on the difference in amount of parameters. The two approaches thus result in a similar test for candidacy that can be written as:

$$\ln(\mathfrak{L}(\boldsymbol{\mathcal{S}}|\hat{P}_{\mathcal{M}^1}(\boldsymbol{\mathcal{Y}}))) - \ln(\mathfrak{L}(\boldsymbol{\mathcal{S}}|\hat{P}_{\mathcal{M}^0}(\boldsymbol{\mathcal{Y}}))) \geq \Delta \tag{22}$$

For the metrics, Δ is equal to the parameter difference in the case of the AIC metric, or $\lambda\ln(|\boldsymbol{\mathcal{S}}|)$ times the parameter difference in the case of the BIC metric. For the general statistical hypothesis test, Δ is equal to the critical value times the amount of samples $|\boldsymbol{\mathcal{S}}|$. This is an important difference between the two approaches. Beyond a certain size of the sample vector, the hypothesis test does no longer depend on $|\boldsymbol{\mathcal{S}}|$ since the negative log–likelihood is divided by it before testing it against the critical value and the critical value doesn't change significantly anymore above a certain size of $|\boldsymbol{\mathcal{S}}|$. However, the size of the sample vector is not factored out of the equation for the penalization metric. This means that if the *average* negative log–likelihood difference is some value $\varepsilon \geq 0$, then the difference itself becomes $|\boldsymbol{\mathcal{S}}|\varepsilon$. This expression can grow to any value, given a large enough sample vector. For the BIC metric, Δ also grows as the size of the sample vector increases, but logarithmically. Empirically, this has been observed to be effective in practice. The setting of Δ in the case of the metrics is done in a desirable way that is not the case for the hypothesis test. Given enough samples, more complex models are allowed. The amount of parameters that a more complex model conveys is thereby thus in some way justified as there are enough samples to estimate the parameters properly.

3.2 Factorization selection

In section 2.2, we observed that in order to obtain a valid factorization, we have to make sure that we do not introduce any cycles into the factorization graph. This is the most general and only necessary condition on the graph, both in the unconditional and the conditional case. In addition, other constraints can be imposed such as that the graph must be a chain, a tree or that it may have no interactions of an order above κ. Such additional constrains are especially useful if the decisions that regard introducing dependencies are not primarily based on the intrinsic data. In such a case, overly complex models are easily introduced if these additional constraints are absent. To be able to use a general graph search algorithm effectively, we either have to justify the introduction of more complex models or we have to penalize them. In this way, we avoid introducing unrequired complex models that take up a lot of computation time.

The idea of such a general graph search algorithm is to maintain a set of edges or arcs that may still be added to the factorization graph without introducing cycles. From this set, the next edge or arc is selected according to certain criteria and the set is filtered again until either the set is empty or no edge or arc meets the criteria of being added anymore. If a scoring metric can be assigned to the factorization graph, we can add the arc or edge that increases the metric the most. When the set of addable edges or arcs is empty or no edge or arc can be added anymore so that the scoring metric is further increased, the algorithm terminates. This becomes more attractive if the metric is decomposable so that the metric increase can be computed based upon the involved variables only instead of upon the entire graph. This approach was first used in evolutionary optimization by Pelikan, Goldberg and Cantú–Paz [26] and subsequently by Mühlenbein and Mahnig [24] as well as Bosman and Thierens [8, 10, 11], all using different scoring metrics.

In the general case, we leave the presence of such a metric to specific implementation details. This brings us to define the general incremental factorization search algorithm. As the notion of conditional dependence and unconditional dependence are intrinsically different, as are the corresponding graph types, we will specify the general algorithm for both cases separately. Even though the unconditional case can be written as a special version of the conditional case, the importance of specifying different algorithms can also be seen from the fact that computing the conditional pdf can be very difficult in contrast to the multivariate joint pdf.

3.2.1 An incremental general factorization search algorithm

We start with the unconditional version of the general graph search. In this case, there is at most one edge between every pair of nodes. Initially this means that a triangle (eg. lower or upper) of the edge adjacency matrix without the diagonal is set to $\boldsymbol{true}$. If an edge is set to $\boldsymbol{true}$, it means that it may still be added to the graph as it does not introduce any cycles. Subsequently, the edge selection process is iterated. Each iteration, a single edge is selected to be added to the graph. This edge is to be chosen from the set of allowed edges $\{(i,j) \mid a[i,j] \wedge (i,j) \in \boldsymbol{\mathcal{L}}^2\}$. If no selection can be made at some point, the iterated edge selection process is halted. If an edge is selected however, it is added to the graph and the set of edges that are still allowed to be added is updated. This is done by traversing the connected components belonging to the nodes that are incident to the edge that is to be added. Each edge between vertices of these opposite connected components is marked as $\boldsymbol{false}$. If there are no more edges addable, the iterated selection stops. The actual factorization is determined afterwards. In the case of conditional dependencies, we can directly make use of the definition in equation 5. In the case of unconditional dependencies, we can do this indirectly by using equation 3. However, we will end up with the requirement of specifying a conditional pdf for each element of the so defined factorization. Therefore, we note that in theory we may very well write unconditional dependencies using $(\pi, \boldsymbol{\omega})$, but in practice we shall use a different structure.

The result of the general factorization search algorithm in the unconditional case is the node vector $\boldsymbol{\nu}$. This is also the case in the ECGA by Harik [19]. The resulting factorization in this case corresponds to the vertices in the connected components of the factorization graph. If we now leave the edge selection procedure to be defined exterior to the general factorization search algorithm and write separate functions for adding an edge to the graph and finding the vertices in the connected components, we have a general factorization search algorithm for the case of unconditional dependencies. However, the algorithm in this case can be simplified. To see this, let $\boldsymbol{\nu}_0$ and $\boldsymbol{\nu}_1$ be two connected components in the graph. If we only observe the variables within these components, the probability distribution is given by the factorized distribution $P(\{Y_v|v \in \boldsymbol{\nu}_0\})P(\{Y_v|v \in \boldsymbol{\nu}_1\})$. Any edge (u,v) with $u \in \boldsymbol{\nu}_0 \wedge v \in \boldsymbol{\nu}_1$ will now result in testing the factorized distribution $P(\{Y_v|v \in (\boldsymbol{\nu}_0 \sqcup \boldsymbol{\nu}_1)\})$ against the former factorized distribution. Hence, if the decision is made that the edge should be added to the graph, all other edges between $\boldsymbol{\nu}_0$ and $\boldsymbol{\nu}_1$ become unallowed. Moreover, if the decision is made that the edge should *not* be added to the graph, all other edges between $\boldsymbol{\nu}_0$ and $\boldsymbol{\nu}_1$ do not have to be tested anymore. In other words, the edges that connect a connected component are not important, as the vertices inside a single connected component make up a joint pdf in the resulting factorization. A graphical impression of this is depicted in figure 5. This implies that we can simplify the general algorithm in the case of unconditional dependencies by first placing every variable in a singleton vector. Subsequently, we find 2 vectors that are to be spliced. If no such combination of vectors can be found, the algorithm terminates. Otherwise, the splice operation is executed and the process is iterated. As each splice operation reduces the amount of available vectors by 1, the amount of splice operations in this algorithm is clearly bounded by $l-1$. This iterated splicing operation in using marginal product models was first proposed by Harik [19]. Concluding, we thus arrive at the following efficient general graph search algorithm for the case of unconditional dependencies:

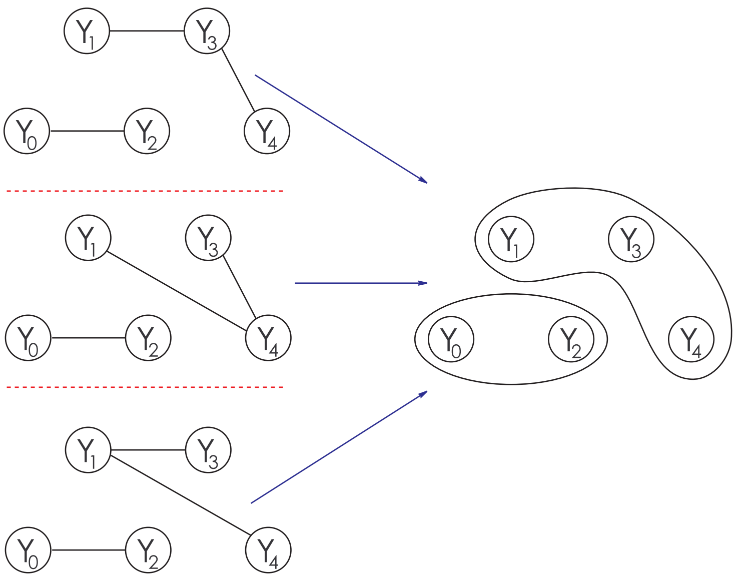

Figure 5: Simplifying the general case of unconditional dependencies.

UNCONDITIONALFACTORIZATIONGRAPHSEARCH()
1 $\boldsymbol{\nu} \leftarrow$ ***new vector*** of (***vector*** of integer)
2 $|\boldsymbol{\nu}| \leftarrow l$
3 ***for*** $i \leftarrow 0$ ***to*** $l-1$ ***do***
  3.1 $\boldsymbol{\nu}_i \leftarrow (i)$
4 ***while*** $|\boldsymbol{\nu}| > 1$ ***do***
  4.1 $(c_0, c_1) \leftarrow$ FINDNODEVECTORS($\boldsymbol{\nu}$)
  4.2 ***if*** $c_0 < 0$ ***then***
    4.2.1 ***breakwhile***
  4.3 ***for*** $i \leftarrow 0$ ***to*** $|\boldsymbol{\nu}_{c_1}|$ ***do***
    4.3.1 $(\boldsymbol{\nu}_{c_0})_{|\boldsymbol{\nu}_{c_0}|} \leftarrow (\boldsymbol{\nu}_{c_1})_i$
  4.4 $\boldsymbol{\nu}_{c_1} \leftarrow \boldsymbol{\nu}_{|\boldsymbol{\nu}|}$
  4.5 $|\boldsymbol{\nu}| \leftarrow |\boldsymbol{\nu}| - 1$
5 ***return***($\boldsymbol{\nu}$)

The general factorization search algorithm in the conditional case is essentially the same as the unsimplified algorithm in the unconditional case. Initially however, for every candidate arc (i, j) we also have an arc (j, i) in the graph now. The amount of arcs initially is therefore twice as large as the amount of edges in the unsimplified unconditional case. The algorithm that finds an arc to add from the set of allowed arcs, is named FINDARC. Furthermore, we now require a cycle detection algorithm, which we specify as ADDARC. This algorithm returns the amount of arcs that have become unallowed because of the addition of an arc. In order to define this algorithm, we require the vector of predecessor nodes v^p and the vector of successor nodes v^s. Algorithm ADDARC is given in appendix D. The resulting factorization can now be specified using $(\pi, \boldsymbol{\omega})$. The parent variable indices in the vector function π can directly be computed from the graph. As the graph contains no cycles, computing a topological sort gives the required ordering $\boldsymbol{\omega}$:

CONDITIONALFACTORIZATIONGRAPHSEARCH()
1 $a \leftarrow$ ***new array*** of ***boolean*** in 2 dimensions with size $l \times l$
2 $v^p, v^s \leftarrow 2$ ***new arrays*** of (***vector*** of ***integer***) with size l
3 ***for*** $i \leftarrow 0$ ***to*** $l - 1$ ***do***
  3.1 ***for*** $j \leftarrow 0$ ***to*** $l - 1$ ***do***
    3.1.1 $a[i, j] \leftarrow$ ***true***
  3.2 $a[i, i] \leftarrow$ ***false***
4 $\gamma \leftarrow l^2 - l$
5 ***while*** $\gamma > 0$ ***do***
  5.1 $(v_0, v_1) \leftarrow$ FINDARC(a, v^p, v^s)
  5.2 ***if*** $v_0 < 0$ ***then***
    5.2.1 ***breakwhile***
  5.3 $\gamma \leftarrow \gamma -$ ADDARC$(\kappa,\ v_0,\ v_1,\ a,\ v^p,\ v^s)$
6 ***return***(TOPOLOGICALSORT(v^p))

We now have a detailed outline of an incremental general factorization search algorithm for both the conditional and the unconditional case. In order to complete the search algorithms, we require to specify how to find two vectors to splice or how to find an arc to add to the graph. A general algorithm for this is dependent on the type of information we wish to guide model selection by. If we have a statistical hypothesis test through which we can decide whether or not two vectors should be spliced or an arc should be added to the graph, we can randomly order the possible splices or arcs to still be added to the graph and test them in this order until an object is found for which the test evaluates to ***true***. If every test fails, the resulting graph has been found. We can formalize this algorithm for the unconditional case as follows:

FINDNODEVECTORSBYTESTING($\boldsymbol{\nu}$)
1 $splices \leftarrow$ ***new array*** of (***integer***, ***integer***) with size $\frac{1}{2}|\boldsymbol{\nu}|(|\boldsymbol{\nu}| - 1)$
2 $k \leftarrow 0$
3 ***for*** $i \leftarrow 0$ ***to*** $|\boldsymbol{\nu}| - 1$ ***do***
  3.1 ***for*** $j \leftarrow i + 1$ ***to*** $|\boldsymbol{\nu}| - 1$ ***do***
    3.1.1 $splices[k] \leftarrow (i, j)$
    3.1.2 $k \leftarrow k + 1$
4 ***for*** $i \leftarrow 0$ ***to*** $\lceil \frac{1}{2}k \rceil$ ***do***
  4.1 $r_1 \leftarrow$ RANDOMNUMBER(k)
  4.2 $r_2 \leftarrow$ RANDOMNUMBER(k)
  4.3 $(v_0, v_1) \leftarrow splices[r_1]$
  4.4 $splices[r_1] \leftarrow splices[r_2]$
  4.5 $splices[r_2] \leftarrow (v_0, v_1)$
5 ***for*** $i \leftarrow 0$ ***to*** $k - 1$ ***do***
  5.1 ***if*** PERFORMSPLICEVECTORSTEST$(splices[i], \boldsymbol{\nu})$ ***then***
    5.1.1 ***return***$(splices[i])$
6 ***return***$((-1, -1))$

In order to derive the conditional variant of the above algorithm, we require only slight changes. These changes involve using the predecessor nodes vector v^p and successor nodes vector v^s instead of the node vector $\boldsymbol{\nu}$ as well as making the for loops regard variables 0 up to and including $l - 1$:

FindArcByTesting(a, v^p, v^s)

1 $arcs \leftarrow$ ***new array*** of (***integer***, ***integer***) with size $l^2 - l$
2 $k \leftarrow 0$
3 ***for*** $i \leftarrow 0$ ***to*** $l - 1$ ***do***
  3.1 ***for*** $j \leftarrow 0$ ***to*** $l - 1$ ***do***
    3.1.1 ***if*** $a[i, j]$ ***then***
      3.1.1.1 $arcs[k] \leftarrow (i, j)$
      3.1.1.2 $k \leftarrow k + 1$
4 ***for*** $i \leftarrow 0$ ***to*** $\lceil \frac{1}{2} k \rceil$ ***do***
  4.1 $r_1 \leftarrow$ RandomNumber(k)
  4.2 $r_2 \leftarrow$ RandomNumber(k)
  4.3 $(v_0, v_1) \leftarrow arcs[r_1]$
  4.4 $arcs[r_1] \leftarrow arcs[r_2]$
  4.5 $arcs[r_2] \leftarrow (v_0, v_1)$
5 ***for*** $i \leftarrow 0$ ***to*** $k - 1$ ***do***
  5.1 ***if*** PerformAddArcTest($arcs[i], v^p, v^s$) ***then***
    5.1.1 ***return***($arcs[i]$)
6 ***return***($(-1, -1)$)

If instead of a statistical hypothesis test we have a metric for how well a factorization fits as a model, we can try to optimize this metric in a greedy fashion by performing the splice or by adding the arc that increases this metric the most. If there is no such possible splice or arc left, the resulting graph has been found. This approach is derived from the Bayesian approach we mentioned earlier. We can formalize the unconditional variant of the algorithm as follows:

FindNodeVectorsByMetric($\boldsymbol{\nu}$)

1 $\delta_{\max} \leftarrow 0$
2 $i_{\max} \leftarrow -1$
3 $j_{\max} \leftarrow -1$
4 ***for*** $i \leftarrow 0$ ***to*** $|\boldsymbol{\nu}| - 1$ ***do***
  4.1 ***for*** $j \leftarrow i + 1$ ***to*** $|\boldsymbol{\nu}| - 1$ ***do***
    4.1.1 $\delta \leftarrow$ ComputeMetricSpliceVectors($i, j, \boldsymbol{\nu}$)
    4.1.2 ***if*** $\delta > \delta_{\max}$ ***then***
      4.1.2.1 $\delta_{\max} \leftarrow \delta$
      4.1.2.2 $i_{\max} \leftarrow i$
      4.1.2.3 $j_{\max} \leftarrow j$
5 ***return***($(i_{\max}, j_{\max})$)

The conditional variant of the above algorithm is quite the same. Instead of the the node vector $\boldsymbol{\nu}$, the predecessor nodes vector v^p and successor nodes vector v^s are required. The for loops go over every combination of two variables (i, j) and only compute the metric change if the arc (i, j) is still allowed to be added to the graph. This results in the following algorithm:

```
FindArcByMetric( a, v^p, v^s )
1   δ_max ← 0
2   i_max ← −1
3   j_max ← −1
4   for i ← 0 to l − 1 do
    4.1   for j ← 0 to l − 1 do
          4.1.1   if a[i, j] then
                  4.1.1.1   δ ← ComputeMetricAddArc(i, j, v^p, v^s)
                  4.1.1.2   if δ > δ_max then
                            4.1.1.2.1   δ_max ← δ
                            4.1.1.2.1   i_max ← i
                            4.1.1.2.1   j_max ← j
5   return((i_max, j_max))
```

The only ingredient missing from our general factorization search algorithms, is either a metric or a test. In most previous approaches [5, 7, 8, 12, 19, 24, 26], a metric was used in combination with a search algorithm. In a selection of these cases, a *general* factorization search algorithm was used [8, 12, 24, 26]. The use of a test to find a factorization has been used in a single case using discrete random variables [28] and in previous work on continuous random variables [13].

3.2.2 Negative log–likelihood statistical hypothesis testing

Let $\mathfrak{f}_0$ be the current factorization and $\mathfrak{f}_1$ be the factorization that results when the arc is added to the graph or when the two vectors are spliced. Given these two factorizations, we want to have a test that tells us which of these two factorizations significantly better fits the given sample points. To this end, we specify the following hypothesis that is derived from the MSTH:

Factorization Selection Test Hypothesis (FSTH)
A probability distribution based on factorization $\mathfrak{f}^1$ cannot better describe the given samples than can a probability distribution based on factorization $\mathfrak{f}^0$.

If the FSTH is rejected, the test returns that the arc (v_0, v_1) should be added to the graph or that the two node vectors $\boldsymbol{\nu}_{c_0}$ and $\boldsymbol{\nu}_{c_1}$ should be spliced. A statistical hypothesis testing metric for selecting a factorization is readily available since in section 3.1.1 a general statistical hypothesis test on the basis of the negative log–likelihood difference was given for *any* probabilistic model. Combining this test with the factorization search algorithms from section 3.2.1, we now have a factorization selection method based on statistical hypothesis tests.

Regarding efficiency, we note that computing the negative log–likelihood is decomposable over the factorization. This means that if we test to alter the model over some arc or by merging some sets, we only have to compute the negative log–likelihood difference for the involved variables. In the case of undirected dependencies, the negative log–likelihood can be written as:

$$-\ln(\mathfrak{L}(\boldsymbol{\mathcal{S}}|\hat{P}_{\boldsymbol{\nu}}(\boldsymbol{\mathcal{Y}}))) = -\sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \ln\left(\prod_{j=0}^{|\boldsymbol{\nu}|-1} \hat{P}(\boldsymbol{\mathcal{Y}}\langle\boldsymbol{\nu}_j\rangle)(y^i\langle\boldsymbol{\nu}_j\rangle)\right) = -\sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1}\sum_{j=0}^{|\boldsymbol{\nu}|-1} \ln(\hat{P}(\boldsymbol{\mathcal{Y}}\langle\boldsymbol{\nu}_j\rangle)(y^i\langle\boldsymbol{\nu}_j\rangle)) \tag{23}$$

Now we assume that $\boldsymbol{\nu}^{\mathbf{0}}$ is the node vector for the base model and that $\boldsymbol{\nu}^{\mathbf{1}}$ is the node vector for the new model. Furthermore, we assume that they differ only in vectors $\boldsymbol{c}^{\mathbf{0}}$ and $\boldsymbol{c}^{\mathbf{1}}$ such that $\boldsymbol{\nu}^{\mathbf{0}} - (\boldsymbol{c}^{\mathbf{0}}, \boldsymbol{c}^{\mathbf{1}}) = \boldsymbol{\nu}^{\mathbf{1}} - (\boldsymbol{c}^{\mathbf{0}} \sqcup \boldsymbol{c}^{\mathbf{1}})$. The negative log–likelihood difference then becomes:

$$\ln(\mathfrak{L}(\boldsymbol{\mathcal{S}}|\hat{P}_{\boldsymbol{\nu}^1}(\boldsymbol{\mathcal{Y}}))) - \ln(\mathfrak{L}(\boldsymbol{\mathcal{S}}|\hat{P}_{\boldsymbol{\nu}^0}(\boldsymbol{\mathcal{Y}}))) = \tag{24}$$

$$\sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \ln(\hat{P}_{\boldsymbol{\nu}^1}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)) - \sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \ln(\hat{P}_{\boldsymbol{\nu}^0}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)) =$$

$$\sum_{i=0}^{|\mathcal{S}|-1}\left(\ln(\hat{P}(\mathcal{Y}\langle \boldsymbol{c^0} \sqcup \boldsymbol{c^1}\rangle)(y^i\langle \boldsymbol{c^0} \sqcup \boldsymbol{c^1}\rangle)) + \sum_{\boldsymbol{c}\in\boldsymbol{\nu}^1-(\boldsymbol{c^0}\sqcup\boldsymbol{c^1})} \ln(\hat{P}(\mathcal{Y}\langle \boldsymbol{c}\rangle)(y^i\langle \boldsymbol{c}\rangle))\right) -$$

$$\sum_{i=0}^{|\mathcal{S}|-1}\left(\ln(\hat{P}(\mathcal{Y}\langle \boldsymbol{c^0}\rangle)(y^i\langle \boldsymbol{c^0}\rangle)) + \ln(\hat{P}(\mathcal{Y}\langle \boldsymbol{c^1}\rangle)(y^i\langle \boldsymbol{c^1}\rangle)) + \sum_{\boldsymbol{c}\in\boldsymbol{\nu}^0-(\boldsymbol{c^0},\boldsymbol{c^1})} \ln(\hat{P}(\mathcal{Y}\langle \boldsymbol{c}\rangle)(y^i\langle \boldsymbol{c}\rangle))\right) =$$

$$\sum_{i=0}^{|\mathcal{S}|-1} \ln(\hat{P}(\mathcal{Y}\langle \boldsymbol{c^0} \sqcup \boldsymbol{c^1}\rangle)(y^i\langle \boldsymbol{c^0} \sqcup \boldsymbol{c^1}\rangle)) - \ln(\hat{P}(\mathcal{Y}\langle \boldsymbol{c^0}\rangle)(y^i\langle \boldsymbol{c^0}\rangle)) - \ln(\hat{P}(\mathcal{Y}\langle \boldsymbol{c^1}\rangle)(y^i\langle \boldsymbol{c^1}\rangle))$$

This implies that we only have to compute the sum over the variables that are involved in a splice operation. In the case of conditional probabilities, assume we add arc (v_0, v_1) to the graph, meaning that Y_{v_1} is made conditionally dependent on Y_{v_0}. In an analogous way, we find that given conditional factorizations $(\pi^0, \boldsymbol{\omega}^0)$ and $(\pi^1, \boldsymbol{\omega}^1)$ such that $\pi^1(v_1) = (v_0) \sqcup \pi^0(v_1)$, the negative log–likelihood difference becomes:

$$\ln(\mathfrak{L}(\mathcal{S}|\hat{P}_{(\pi^1,\boldsymbol{\omega}^1)}(\mathcal{Y}))) - \ln(\mathfrak{L}(\mathcal{S}|\hat{P}_{(\pi^0,\boldsymbol{\omega}^0)}(\mathcal{Y}))) = \tag{25}$$

$$\sum_{i=0}^{|\mathcal{S}|-1} \ln(\hat{P}(Y_{v_1}|\mathcal{Y}\langle\pi^1(v_1)\rangle)(y^i\langle(v_1) \sqcup \pi^1(v_1)\rangle)) - \ln(\hat{P}(Y_{v_1}|\mathcal{Y}\langle\pi^0(v_1)\rangle)(y^i\langle(v_1) \sqcup \pi^0(v_1)\rangle))$$

Again, we only have to compute a part of the entire sum of the negative log–likelihood because the parts in the factorization that do not change, cancel out in the difference.

3.2.3 Complexity penalization

We noted at the end of the previous subsection that the negative log–likelihood is decomposable over a factorization. This fact can be used to efficiently compute the AIC and BIC metrics as introduced in section 3.1.2. We only have to compute the results over the variables for which the dependencies are different in one factorization as compared to another. In the unconditional case, each element in the node vector is mutually exclusive with respect to any other element in the node vector. This implies that for each element in the unconditional factorization, the amount of parameters is just the sum of the amount of parameters over all elements in the factorization:

$$|\boldsymbol{\theta} \xleftarrow{fit} \boldsymbol{\nu}| = \sum_{i=0}^{|\boldsymbol{\nu}|-1} |\boldsymbol{\theta} \xleftarrow{fit} \boldsymbol{\nu}_i| \tag{26}$$

Therefore, in a similar manner as in equation 24, we find for the AIC metric that:

$$\ln(\mathfrak{A}(\hat{P}_{\boldsymbol{\nu}^1}(\mathcal{Y})|\mathcal{S})) - \ln(\mathfrak{A}(\hat{P}_{\boldsymbol{\nu}^0}(\mathcal{Y})|\mathcal{S})) = \tag{27}$$

$$\sum_{i=0}^{|\mathcal{S}|-1} \ln(\hat{P}(\mathcal{Y}\langle \boldsymbol{c_0} \sqcup \boldsymbol{c_1}\rangle)(y^i\langle \boldsymbol{c_0} \sqcup \boldsymbol{c_1}\rangle)) - \ln(\hat{P}(\mathcal{Y}\langle \boldsymbol{c_0}\rangle)(y^i\langle \boldsymbol{c_0}\rangle)) - \ln(\hat{P}(\mathcal{Y}\langle \boldsymbol{c_1}\rangle)(y^i\langle \boldsymbol{c_1}\rangle)) +$$

$$|\boldsymbol{\theta} \xleftarrow{fit} \boldsymbol{c_0}| + |\boldsymbol{\theta} \xleftarrow{fit} \boldsymbol{c_1}| - |\boldsymbol{\theta} \xleftarrow{fit} \boldsymbol{c_0} \sqcup \boldsymbol{c_1}|$$

In the conditional case however, we do not have that the amount of parameters $|\boldsymbol{\theta} \xleftarrow{fit} (\pi, \omega)|$ can be computed in a decomposable manner over the factorziation. Assume for instance that we use a multivariate normal pdf for each element in the factorization. If we add an arc $Y_3 \to Y_0$ to the factorization that underlies the probability distribution $P(Y_0|Y_1Y_2)P(Y_1)P(Y_2)P(Y_3)$, the first factor in the probability distribution becomes $P(Y_0|Y_1Y_2Y_3)$. If computing the amount of parameters would be decomposable, we would be able to compute the increase in amount of parameters by computing the difference in amount of implied parameters of $P(Y_0|Y_1Y_2Y_3)$ and $P(Y_0|Y_1Y_2)$. Since each conditional pdf uses a multivariate joint pdf over all the involved variables, the difference would be 5. However, the parameters in the factor $P(Y_0|Y_1Y_2Y_3)$ are now not only present in this factor. The factors themselves overlap, which is not the case for the unconditional dependencies. Because of this overlap, for instance the full covariance matrix index $(3,3)$ is already required, regardless of the fact that Y_3 is added as a conditional to the first factor. Therefore, the amount of additional parameters is not equal to 5, but less. However, the amount of additional computational resources that are required, *do* increase similarly as in the unconditional case with respect to joint pdfs. The reason for this is that the covariance matrix will have to be inverted. Furthermore, if we do not take a normal pdf, but a normal mixture pdf, we estimate the parameters anew using the EM algorithm anyway since we cannot suffice with a single $l \times l$ covariance matrix in which the required entries are stored. Also, since applying the EM algorithm is a computationally resourcefull task and its performance degrades with increasing dimension, we do not want to apply it directly to the full joint distribution to find all involved parameters, but apply it anew as the factorization is built during the search algorithm. Concluding, decomposing the amount of parameters over the conditional factorization is a rational operation as it penalizes the additional computational resources. The resulting metric difference can be written as:

$$\ln(\mathfrak{A}(\hat{P}_{(\pi^1,\boldsymbol{\omega}^1)}(\boldsymbol{\mathcal{Y}})|\boldsymbol{\mathcal{S}})) - \ln(\mathfrak{A}(\hat{P}_{(\pi^0,\boldsymbol{\omega}^0)}(\boldsymbol{\mathcal{Y}})|\boldsymbol{\mathcal{S}})) = \tag{28}$$

$$\sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \ln(\hat{P}(Y_{v_1}|\boldsymbol{\mathcal{Y}}\langle\pi^1(v_1)\rangle)(y^i\langle(v_1) \sqcup \pi^1(v_1)\rangle)) - \ln(\hat{P}(Y_{v_1}|\boldsymbol{\mathcal{Y}}\langle\pi^0(v_1)\rangle)(y^i\langle(v_1) \sqcup \pi^0(v_1)\rangle)) +$$

$$|\boldsymbol{\theta} \xleftarrow{fit} ((\pi^0(v_1)),(v_1))| - |\boldsymbol{\theta} \xleftarrow{fit} ((\pi^1(v_1)),(v_1))|$$

Thus, adding a penalty term in the way proposed, results in an efficiently computable decomposable metric. If we can derive the differential entropy analytically to find an efficiently computable expression, we can use the entropy over the estimated model instead of the negative log–likelihood. This is for instance the case for the normal pdf. As a result, the computations for model selection based on minimizing the metric can be done efficiently when using a normal pdf.

3.3 Factorization mixture selection

So far, we have discussed finding relations between variables based on a sample vector and using a factorization to build a useful probabilistic model. However, the structure of the sample vector may be highly non–linear. This non–linearity can force us to use probabilistic models of a high complexity to retain some of this non–linearity. However, especially using relatively simple pdfs such as the normal pdf, the non–linear interactions cannot be captured even with higher order models. For instance, a two dimensional sample vector in which the samples are aligned in a V shape cannot be fit adequately by a product of one dimensional normal pdfs, nor by a single two dimensional normal pdf. However, if we would identify each line in the V shape as a single cluster, we could adequately model the sample vector using a sum of two dimensional normal pdfs.

The key issue is the notion of *clusters*, which are possibly overlapping subvectors of the original sample vector and are optionally augmented with additional data. When we do not allow the subsets to overlap, we classify each point in the sample vector to be explicitly part of some cluster or class. This instance of clustering is called *partitioning*. The use of clusters allows us to efficiently break up non–linear interactions so that we can use simple models to get an adequate representation of the sample vector. Furthermore, computationally efficient clustering algorithms exist that provide useful results.

There are exact algorithms for partitioning [20], but the running times for these algorithms are of no practical use in our case. What we require, is a fast approximate assessment of clusters that we can fit well using some pdf. Within these clusters, we can infer what factorization is best and get a probability distribution over the cluster. As all of this is part of the larger algorithm that attempts to infer the structure of the search space and use it in optimization, the clustering should be effective in representation as well as in computation time requirements.

Each cluster is processed separately in order to have a probability distribution fit over it. As such, we have a *mixture* of probability distributions. By assigning each of these probability distributions a weight in $[0,1]$ and letting the sum over all these weights equal 1, the result is again a probability distribution. We write these weights as β_i. These weights are commonly known as *mixing coefficients*. We write $\boldsymbol{\mathfrak{K}} = (\boldsymbol{\mathfrak{K}}^{\mathbf{0}}, \boldsymbol{\mathfrak{K}}^{\mathbf{1}}, \ldots, \boldsymbol{\mathfrak{K}}^{|\boldsymbol{\mathfrak{K}}|-\mathbf{1}})$ for the vector of clusters. Each cluster contains a variable amount of sample vectors, which we denote so that $\boldsymbol{\mathfrak{K}}^{\boldsymbol{i}} = (\boldsymbol{\mathfrak{K}}^{\boldsymbol{i}}_{\mathbf{0}}, \boldsymbol{\mathfrak{K}}^{\boldsymbol{i}}_{\mathbf{1}}, \ldots, \boldsymbol{\mathfrak{K}}^{\boldsymbol{i}}_{|\boldsymbol{\mathfrak{K}}^{\boldsymbol{i}}|-\mathbf{1}})$. Furthermore we have for each j that $\boldsymbol{\mathfrak{K}}^{\boldsymbol{i}}_{\boldsymbol{j}} \in \boldsymbol{\mathcal{S}}$, so $\boldsymbol{\mathfrak{K}}^{\boldsymbol{i}}_{\boldsymbol{j}} = ((\boldsymbol{\mathfrak{K}}^{\boldsymbol{i}}_{\boldsymbol{j}})_0, (\boldsymbol{\mathfrak{K}}^{\boldsymbol{i}}_{\boldsymbol{j}})_1, \ldots, (\boldsymbol{\mathfrak{K}}^{\boldsymbol{i}}_{\boldsymbol{j}})_{l-1})$. In our modeling, we assume that each probability distribution in the mixture is factorized as discussed in section 2.2. We assume to have k clusters, so $|\boldsymbol{\mathfrak{K}}| = k$ and let $\boldsymbol{\mathcal{K}} = (0, 1, \ldots, k-1)$. We can now write the resulting probability distribution as:

$$\hat{P}_{\mathfrak{f}\langle\boldsymbol{\mathcal{K}}\rangle}(\boldsymbol{\mathcal{Y}}) = \sum_{i=0}^{|\boldsymbol{\mathfrak{K}}|-1} \beta_i \hat{P}^i_{\mathfrak{f}_i}(\boldsymbol{\mathcal{Y}}) \tag{29}$$

The reason why we require to have an i in the superscript in $\hat{P}^i_{\mathfrak{f}_i}(\boldsymbol{\mathcal{Y}})$, is to be able to indicate that the i–th probability distribution is based upon the i–th cluster of samples. To determine the β_i and the $\hat{P}^i_{\mathfrak{f}_i}(\boldsymbol{\mathcal{Y}})$ in equation 29, two approaches can be taken. If we cluster the sample vector into possibly overlapping subvectors, we can for instance determine the mixing coefficients as the proportional subvector size. If we do not explicitly cluster the sample vector, but allow for each sample a probability of assigning it to a certain cluster, we can mathematically derive a way of finding the maximum expectation of this probability over all samples. By doing so, the clusters can be seen as copies of the original sample vector augmented with likelihood values for each sample that indicate the probability that the sample belongs to a certain cluster. As such, the resulting clusters can be seen as *fuzzy* clusters since there is no clear border between the clusters as is the case in partitioning. In practical applications, the first approach leads to the use of partitioning algorithms and the second approach leads to expectation maximization (EM) algorithms. If we use such an EM algorithm however, we do not get any clusters to which we can apply our factorization selection procedures for constructing the final probability distribution. Furthermore, we have noted that we only use the normal pdf. If we identify a single cluster equal to the entire sample vector and fit a normal mixture model to the cluster, we also have to use the EM algorithm as we shall see in section 4.3.2. This gives an approach that is similar to identifying fuzzy clusters using the EM algorithm and subsequently using the result as the probability distribution. Therefore, we only focus on partitioning algorithms so as to be able to apply factorization selection to the identified partitions.

3.3.1 Partitioning algorithms

One of the most fundamental aspects in partioning, is the distance function. The distance of a point to a cluster determines to which cluster the point will actually be assigned. Often, the *Euclidean* distance is used. However, if we are to cluster two variables that lie on a different scale, the Euclidean distance might not be the best choice. Optically, if we would present the sample vector in a rectangular graph where the axes are rescaled to fit within the rectangle, the clusters we would observe are not the clusters as they appear in the data based upon the Euclidean distance. These clusters will be found if both axes are of the same scale. One axis might become a lot longer in this case and clusters that might have been obvious in the rectangular graph, may not be so obvious at all anymore. Using the Euclidean distance is in this case still useful, but *after* the sample data has been rescaled in every dimension. Rescaling on the other hand is sensitive to outliers. For two $|\boldsymbol{a}|$ dimensional points $y^i\langle\boldsymbol{a}\rangle$ and $y^j\langle\boldsymbol{a}\rangle$, the Euclidean distance $d_E(y^i\langle\boldsymbol{a}\rangle, y^j\langle\boldsymbol{a}\rangle)$ between them is given by:

$$d_E(y^i\langle\boldsymbol{a}\rangle, y^j\langle\boldsymbol{a}\rangle) = \sqrt{(y^i\langle\boldsymbol{a}\rangle - y^j\langle\boldsymbol{a}\rangle)^T(y^i\langle\boldsymbol{a}\rangle - y^j\langle\boldsymbol{a}\rangle)} = \sqrt{\sum_{k=0}^{|\boldsymbol{a}|-1}(y^i_{\boldsymbol{a}_k} - y^j_{\boldsymbol{a}_k})^2} \tag{30}$$

Scaling can be introduced into the Euclidean distance metric by computing the sample variance in each dimension separately:

$$d_{ES}(y^i\langle \boldsymbol{a}\rangle, y^j\langle \boldsymbol{a}\rangle) = \sqrt{(y^i\langle \boldsymbol{a}\rangle - y^j\langle \boldsymbol{a}\rangle)^T(\boldsymbol{V}(\boldsymbol{a}))^{-1}(y^i\langle \boldsymbol{a}\rangle - y^j\langle \boldsymbol{a}\rangle)} = \sqrt{\sum_{k=0}^{|\boldsymbol{a}|-1} \frac{(y^i_{\boldsymbol{a}_k} - y^j_{\boldsymbol{a}_k})^2}{s^2_{\boldsymbol{a}_k}}} \tag{31}$$

$$\text{where } \boldsymbol{V}(\boldsymbol{a}) = \begin{bmatrix} s^2_{\boldsymbol{a}_0} & 0 & \cdots & 0 \\ 0 & s^2_{\boldsymbol{a}_1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & s^2_{\boldsymbol{a}_{|\boldsymbol{a}|-1}} \end{bmatrix}$$

If we have a cluster, we can use its centroid as a reference point. The distance between a point and the cluster is then the scaled Euclidean distance between the cluster centroid and the point. A problem with this approach is that the distance to a cluster has ellipsoid shaped isolines that are aligned along with the axes. Therefore, even samples with linear interaction through correlation will likely be broken up into multiple axes–aligned ellipsoid shaped clusters. To overcome this problem, the *Mahalanobis* distance metric [23] can be used. Whereas the Euclidean distance metric can be seen as the result of fitting a normal pdf with zero covariances, the Mahalanobis distance metric can be seen as the result of fitting a normal pdf with a full covariance matrix. The clusters can therefore potentially be shaped better to the form of the data. Let $\overline{\mathfrak{K}^i}\langle \boldsymbol{a}\rangle$ be the centroid of cluster i and let $\boldsymbol{S}(\boldsymbol{\mathfrak{K}^i}, \boldsymbol{a})$ be the sample covariance matrix of the points in cluster i over the features selected by $\boldsymbol{a}$:

$$\overline{\mathfrak{K}^i}\langle \boldsymbol{a}\rangle = \frac{1}{|\boldsymbol{\mathfrak{K}^i}|} \sum_{j=0}^{|\boldsymbol{\mathfrak{K}^i}|-1} \boldsymbol{\mathfrak{K}^i_j} \tag{32}$$

$$\boldsymbol{S}(\boldsymbol{\mathfrak{K}^i}, \boldsymbol{a}) = \frac{1}{|\boldsymbol{\mathfrak{K}^i}|} \sum_{j=0}^{|\boldsymbol{\mathfrak{K}^i}|-1} (\boldsymbol{\mathfrak{K}^i_j} - \overline{\mathfrak{K}^i}\langle \boldsymbol{a}\rangle)(\boldsymbol{\mathfrak{K}^i_j} - \overline{\mathfrak{K}^i}\langle \boldsymbol{a}\rangle)^T \tag{33}$$

The Mahalanobis distance between a point $y^i\langle \boldsymbol{a}\rangle$ and cluster j can now be written as:

$$d_M(y^i\langle \boldsymbol{a}\rangle, j) = \sqrt{(y^i\langle \boldsymbol{a}\rangle - \overline{\mathfrak{K}^j}\langle \boldsymbol{a}\rangle)^T(\boldsymbol{S}(\boldsymbol{\mathfrak{K}^i}, \boldsymbol{a}))^{-1}(y^i\langle \boldsymbol{a}\rangle - \overline{\mathfrak{K}^j}\langle \boldsymbol{a}\rangle)} \tag{34}$$

It can be shown that the isolines of the Mahalanobis distance are ellipsoids that are centered about the cluster centroid and can be aligned along *any* direction. In the special case where the samples are uncorrelated, these ellipsoids are aligned along the axes and the Mahalanobis distance becomes equivalent to the Euclidean distance. The Mahalanobis distance overcomes two limitations of the scaled Euclidean distance. On the one hand, it corrects for correlation between the different features. On the other hand, it can provide curved as well as linear decision boundaries. The disadvantage of the Mahalanobis distance is that the covariance matrices have to be determined, which becomes less reliable if the amount of samples in a cluster becomes smaller. Furthermore, the memory and computation time requirements are $\mathcal{O}(l^2)$ instead of $\mathcal{O}(l)$.

The mixture coefficients β_i can be determined in different ways. One of the two most common ways, is to use a probabilistic approach and set β_i to the ratio of the size of the i–th cluster to the total size of all clusters:

$$\forall_{i \in \boldsymbol{\mathcal{K}}} \left\langle \beta_i = \frac{|\boldsymbol{\mathfrak{K}^i}|}{\sum_{j=0}^{|\mathfrak{K}|-1} |\boldsymbol{\mathfrak{K}^j}|} \right\rangle \tag{35}$$

As selection will drive the optimization process toward the more promising regions of the search space, the clusters will be concentrated on different peaks in the cost landscape. Such *niches* will however not remain stable using equation 35 if the peaks are not equally high in case of maximization, since selection will prefer the larger peak. In the case of equal peaks, the search may still converge to concentrate on a single peak because of finite statistical instability. To this end, the other common way to determine the mixing coefficients can be used. In this case, the mixing coefficients are determined as the average cost of the niche divided by the sum of the average costs over every niche:

$$\forall_{i \in \mathcal{K}} \left\langle \beta_i = \frac{\frac{1}{|\mathfrak{K}^i|}\sum_{q=0}^{|\mathfrak{K}^i|-1} C(\mathfrak{K}^i_q)}{\sum_{j=0}^{|\mathfrak{K}|-1} \frac{1}{|\mathfrak{K}^j|}\sum_{q=0}^{|\mathfrak{K}^j|-1} C(\mathfrak{K}^j_q)} \right\rangle \tag{36}$$

Clustering for niching in stochastic optimization was first introduced in the field of genetic algorithms [33]. Clustering and the mixing coefficients in equations 35 and 36 were later used in the first approach [25] that applied clustering to iterated density estimation evolutionary algorithms. However, the formula in equation 36 doesn't work if the objective is to minimize and the values are below zero. Therefore, it is reasonable to assume that the optimum value will be far from the initial sample cost average. If we compute the average fitness for niching with respect to this initial sample cost average $\overline{C_I}$ and take the absolute value to account for maximization as well as minimization, we get:

$$\forall_{i \in \mathcal{K}} \left\langle \beta_i = \frac{\left|\frac{1}{|\mathfrak{K}^i|}\sum_{q=0}^{|\mathfrak{K}^i|-1} C(\mathfrak{K}^i_q) - \overline{C_I}\right|}{\sum_{j=0}^{|\mathfrak{K}|-1} \left|\frac{1}{|\mathfrak{K}^j|}\sum_{q=0}^{|\mathfrak{K}^i|-1} C(\mathfrak{K}^i_q) - \overline{C_I}\right|} \right\rangle \tag{37}$$

Before we continue on the actual clustering algorithms, we place a final note on the use of distance measures. In general, clusters should unite sample points that constitute promising regions in the search space. These regions do not necessarily have to correspond to the Euclidean space in which they are coded. This coding space is called the *genotypic* space. The decoded space in which parameters have their true meaning and directly contribute to the cost function, is called the *phenotypic* space. It should be noted that clustering in phenotypic space is in general more desirable as it is more likely to cluster the samples the most effective way. For most continuous problems, the two spaces are identical. We now focus on the partioning algorithms themselves.

The leader algorithm

The leader algorithm is one of the fastest clustering algorithms. The use of it can thus be beneficial if the amount of overhead that is introduced by factorization mixture selection methods is desired to remain small. Furthermore, there is no need to specify in advance how many clusters there should be. The first sample to make a new cluster is appointed to be its leader. The leader algorithm goes over the sample vector exactly once. For each sample it encounters, it finds the first cluster that has a leader being closer to the sample than a given threshold $\mathfrak{T}_d$. If no such cluster can be found, a new cluster is created containing only this single sample.

Unless the order in which the clusters are inspected is randomized, the first clusters are always quite a lot larger than the later ones. However, the asymptotic running time for finding the first cluster with a leader closer than $\mathfrak{T}_d$ is the same as going over all clusters and finding the one with the smallest distance. Therefore, we prefer to find the nearest cluster based on the leader of the cluster. If it is closer than $\mathfrak{T}_d$, we assign the sample to the cluster. Another problem that arises is that if we have two clusters to which all samples are equally close, one cluster will be assigned all samples if we go over the clusters in a deterministic order. To overcome this problem, the order in which the clusters are scanned should be randomized as well.

One of the major drawbacks of the algorithm is that it is not invariant given the sequence of the input samples. Most clustering algorithms have this property, but not as strongly as does the leader algorithm. Therefore, to be sure that the ordering of the sample vector is not subject to large repeating sequences of samples, which results in undesired fixed density estimations, we propose to randomize the ordering of the samples as input to the leader algorithm. This can be done by allocating an array of indices that is randomly accessed in the algorithm. In order to use the scaled Euclidean distance measure, we first have to compute the sample variance s_i in each dimension. The resulting algorithm can be formalized as follows:

EUCLIDEANLEADERCLUSTERING($\mathfrak{T}_d$)
1 $L \leftarrow$ ***new array*** of (***vector*** of ***real***) with size $|\boldsymbol{S}|$
2 $\boldsymbol{\mathfrak{K}} \leftarrow ()$
3 $O^s, O^c \leftarrow 2$ ***new arrays*** of ***integer*** with size $|\boldsymbol{S}|$
4 ***for*** $i \leftarrow 0$ ***to*** $l-1$ ***do***
    4.1 $\overline{Y_i} \leftarrow \frac{\sum_{j=0}^{|\boldsymbol{S}|-1} y_i^j}{|\boldsymbol{S}|}$
    4.2 $s_i \leftarrow \frac{\sum_{j=0}^{|\boldsymbol{S}|-1} (y_i^j - \overline{Y_i})^2}{|\boldsymbol{S}|}$
5 ***for*** $i \leftarrow 0$ ***to*** $|\boldsymbol{S}|-1$ ***do***
    5.1 $O^s[i] \leftarrow i$
6 ***for*** $i \leftarrow 0$ ***to*** $|\boldsymbol{S}|-1$ ***do***
    6.1 $q^s \leftarrow$ RANDOM($|\boldsymbol{S}| - i$)
    6.2 ***for*** $j \leftarrow 0$ ***to*** $|\boldsymbol{\mathfrak{K}}|-1$ ***do***
        6.2.1 $O^c[j] \leftarrow j$
    6.3 $d_{\min} \leftarrow \mathfrak{T}_d$
    6.4 ***for*** $j \leftarrow 0$ ***to*** $|\boldsymbol{\mathfrak{K}}|-1$ ***do***
        6.4.1 $q^c \leftarrow$ RANDOM($|\boldsymbol{\mathfrak{K}}| - j$)
        6.4.2 ***if*** $d_{ES}(\boldsymbol{y}^{\boldsymbol{O^s[q^s]}}, L[O^c[q^c]]) < d_{\min}$ ***then***
            6.4.2.1 $d_{\min} \leftarrow d_{ES}(\boldsymbol{y}^{\boldsymbol{O^s[q^s]}}, L[O^c[q^c]])$
            6.4.2.2 $c_{\min} \leftarrow O^c[q^c]$
        6.4.3 $O^c[q^c] \leftarrow O^c[|\boldsymbol{\mathfrak{K}}| - j - 1]$
    6.5 ***if*** $d_{\min} < \mathfrak{T}_d$ ***then***
        6.5.1 $\boldsymbol{\mathfrak{K}}^{\boldsymbol{c}_{\min}}_{|\boldsymbol{\mathfrak{K}}^{\boldsymbol{c}_{\min}}|} \leftarrow \boldsymbol{y}^{\boldsymbol{O^s[q^s]}}$
    6.6 ***else then***
        6.6.1 $L[|\boldsymbol{\mathfrak{K}}|] \leftarrow \boldsymbol{y}^{\boldsymbol{O^s[q^s]}}$
        6.6.2 $\boldsymbol{\mathfrak{K}}^{|\boldsymbol{\mathfrak{K}}|} \leftarrow (\boldsymbol{y}^{\boldsymbol{O^s[q^s]}})$
    6.7 $O^s[q^s] \leftarrow O^s[|\boldsymbol{S}| - i - 1]$
7 ***return***$(\boldsymbol{\mathfrak{K}})$

In order to allow for more flexible clusters, we may want to use the Mahalanobis distance measure. This can however not be achieved by substituting the Mahalanobis distance measure for the Euclidean distance measure in the above algorithm. The reason for this is that in order to evaluate the Mahalanobis distance, we require to know the sample covariance matrix of the samples that belong to a cluster as well as their sample mean. For the Euclidean measure, it suffices to have a single point serving as a centroid.

To use the Mahalanobis distance, we can determine the required parameters as the clustering algorithm progresses through the samples. As a point is added to a cluster, its sample covariance matrix and its sample mean are updated. However, the covariance matrix can only be computed and used in the Mahalanobis distance if we have at least two samples. Furthermore, based on only a few samples, the Mahalanobis distance may give unstable results, leading to unnatural clusters. Therefore, we propose that a cluster must first grow to some minimum size $\mathfrak{T}_s$ before the Mahalanobis distance can be used for it.

The leader aspect in the Euclidean variant of the algorithm lies within the fact that the first sample to define a new cluster is taken to be the leader of the cluster. It serves as a centroid for the Euclidean distance. In the variant that results when we use the Mahalanobis distance, the sample mean takes the role of the leader for some cluster as soon as the size of that cluster has reached $\mathfrak{T}_s$. The sample mean of a cluster changes whenever a sample is added to that cluster. Therefore, the term leader is somewhat misleading in the resulting algorithm:

MahalanobisLeaderClustering($\mathfrak{T}_d$, $\mathfrak{T}_s$)

1 $L \leftarrow$ ***new array*** of (***vector*** of ***real***) with size $|\boldsymbol{S}|$

2 $\mathfrak{K} \leftarrow ()$

3 $O^s, O^c \leftarrow 2$ ***new arrays*** of ***integer*** with size $|\boldsymbol{S}|$

4 ***for*** $i \leftarrow 0$ ***to*** $l-1$ ***do***

  4.1 $\overline{Y_i} \leftarrow \frac{\sum_{j=0}^{|\boldsymbol{S}|-1} y_i^j}{|\boldsymbol{S}|}$

  4.2 $s_i \leftarrow \frac{\sum_{j=0}^{|\boldsymbol{S}|-1} (y_i^j - \overline{Y_i})^2}{|\boldsymbol{S}|}$

5 ***for*** $i \leftarrow 0$ ***to*** $|\boldsymbol{S}|-1$ ***do***

  5.1 $O^s[i] \leftarrow i$

6 ***for*** $i \leftarrow 0$ ***to*** $|\boldsymbol{S}|-1$ ***do***

  6.1 $q^s \leftarrow$ Random($|\boldsymbol{S}| - i$)

  6.2 ***for*** $j \leftarrow 0$ ***to*** $|\mathfrak{K}|-1$ ***do***

    6.2.1 $O^c[j] \leftarrow j$

  6.3 $d_{\min} \leftarrow \mathfrak{T}_d$

  6.4 ***for*** $j \leftarrow 0$ ***to*** $|\mathfrak{K}|-1$ ***do***

    6.4.1 $q^c \leftarrow$ Random($|\mathfrak{K}| - j$)

    6.4.2 ***if*** $|\mathfrak{K}^{O^c[q^c]}| \geq \mathfrak{T}_s$ ***then***

      6.4.2.1 $d \leftarrow d_M(\boldsymbol{y}^{\boldsymbol{O^s[q^s]}}, O^c[q^c])$

    6.4.3 ***else***

      6.4.3.1 $d \leftarrow d_{ES}(\boldsymbol{y}^{\boldsymbol{O^s[q^s]}}, L[O^c[q^c]])$

    6.4.4 ***if*** $d < d_{\min}$ ***then***

      6.4.4.1 $d_{\min} \leftarrow d$

      6.4.4.2 $c_{\min} \leftarrow O^c[q^c]$

    6.4.5 $O^c[q^c] \leftarrow O^c[|\mathfrak{K}| - j - 1]$

  6.5 ***if*** $d_{\min} < \mathfrak{T}_d$ ***then***

    6.5.1 ***for*** $j \leftarrow 0$ ***to*** $l-1$ ***do***

      6.5.1.1 ***for*** $q \leftarrow j$ ***to*** $l-1$ ***do***

        6.5.1.1.1 $\boldsymbol{S}(\mathfrak{K}^{\boldsymbol{c}_{\min}}, \boldsymbol{\mathcal{L}})(j,q) \leftarrow \boldsymbol{S}(\mathfrak{K}^{\boldsymbol{c}_{\min}}, \boldsymbol{\mathcal{L}})(j,q) + (\overline{\mathfrak{K}^{c_{\min}}})_j (\overline{\mathfrak{K}^{c_{\min}}})_q$

        6.5.1.1.2 $\boldsymbol{S}(\mathfrak{K}^{\boldsymbol{c}_{\min}}, \boldsymbol{\mathcal{L}})(j,q) \leftarrow \frac{|\mathfrak{K}^{\boldsymbol{c}_{\min}}| \boldsymbol{S}(\mathfrak{K}^{\boldsymbol{c}_{\min}}, \boldsymbol{\mathcal{L}})(j,q) + y_j^{O^s[q^s]} y_q^{O^s[q^s]}}{|\mathfrak{K}^{\boldsymbol{c}_{\min}}| + 1}$

    6.5.2 $\overline{\mathfrak{K}^{c_{\min}}} \leftarrow \frac{\overline{\mathfrak{K}^{c_{\min}}} |\mathfrak{K}^{\boldsymbol{c}_{\min}}| + \boldsymbol{y}^{\boldsymbol{O^s[q^s]}}}{|\mathfrak{K}^{\boldsymbol{c}_{\min}}| + 1}$

    6.5.3 ***for*** $j \leftarrow 0$ ***to*** $l-1$ ***do***

      6.5.3.1 ***for*** $q \leftarrow j$ ***to*** $l-1$ ***do***

        6.5.3.1.1 $\boldsymbol{S}(\mathfrak{K}^{\boldsymbol{c}_{\min}}, \boldsymbol{\mathcal{L}})(j,q) \leftarrow \boldsymbol{S}(\mathfrak{K}^{\boldsymbol{c}_{\min}}, \boldsymbol{\mathcal{L}})(j,q) - (\overline{\mathfrak{K}^{c_{\min}}})_j (\overline{\mathfrak{K}^{c_{\min}}})_q$

        6.5.3.1.2 $\boldsymbol{S}(\mathfrak{K}^{\boldsymbol{c}_{\min}}, \boldsymbol{\mathcal{L}})(q,j) \leftarrow \boldsymbol{S}(\mathfrak{K}^{\boldsymbol{c}_{\min}}, \boldsymbol{\mathcal{L}})(j,q)$

    6.5.4 $\mathfrak{K}^{\boldsymbol{c}_{\min}}_{|\mathfrak{K}_{\boldsymbol{c}_{\min}}|} \leftarrow \boldsymbol{y}^{\boldsymbol{O^s[q^s]}}$

  6.6 ***else then***

    6.6.1 $\overline{\mathfrak{K}^{|\mathfrak{K}|}} \leftarrow \boldsymbol{y}^{\boldsymbol{O^s[q^s]}}$

    6.6.2 ***for*** $j \leftarrow 0$ ***to*** $|\boldsymbol{S}|-1$ ***do***

      6.6.2.1 ***for*** $q \leftarrow 0$ ***to*** $|\boldsymbol{S}|-1$ ***do***

        6.6.2.1.1 $\boldsymbol{S}(\mathfrak{K}^{|\mathfrak{K}|}, \boldsymbol{\mathcal{L}})(j,q) \leftarrow 0$

    6.6.3 $L[|\mathfrak{K}|] \leftarrow \boldsymbol{y}^{\boldsymbol{O^s[q^s]}}$

    6.6.4 $\mathfrak{K}^{|\mathfrak{K}|} \leftarrow (\boldsymbol{y}^{\boldsymbol{O^s[q^s]}})$

  6.7 $O^s[q^s] \leftarrow O^s[|\boldsymbol{S}| - i - 1]$

7 ***return***$(\mathfrak{K})$

The k–means algorithm

An algorithm that is invariant of the permutation of the sample vector, is the *joining* algorithm. This algorithm initially generates a cluster for each sample and then iteratively joins clusters until no clusters are close to each other enough anymore. The algorithm is however not really suitable for large sample vectors, since $|\boldsymbol{S}|^2$ distances must be computed and examined.

A faster algorithm that is less subject to the input ordering as is the leader algorithm but not independent as is the joining algorithm, is the k–means clustering algorithm. This algorithm constructs exactly k clusters. As we a priori mostly have little notion of how many clusters are appropriate to fit to the data, the adaptive k–means algorithm is usually preferred. Based on distance thresholds, the number of clusters k is determined by the algorithm itself, just as is done by the leader algorithm. The amount of clusters influences the running time of the subsequent distribution estimation, so a bound on k might be preferred. We restrict ourselves to the non–adaptive version of the clustering algorithm and elaborate on the value of k in section 3.3.2.

The k–means algorithm computes k clusters. A difference with respect to the leader algorithm, is that it uses the cluster centroids. The general procedure is as follows. First, k clusters are picked at random. This can be done by partioning the sample vector at random into k subvectors. The resulting centroids are however expected to lie close to each other. Therefore, the initial clusters are usually taken to consist of a single sample, which is chosen at random from the sample vector. Subsequently, the algorithm iterates until the means do not change to within a significance of ε anymore. One iteration consists of assigning each point to the nearest cluster based on the distance to the cluster centroid. If multiple clusters are equally likely to be chosen, one of them is picked at random. To ensure this, the clusters are scanned in a random order for each new sample point. Once all of the points have been assigned, the means of the clusters are recomputed.

An advantage over the leader algorithm, is that the result of the k–means algorithm is less dependent on the input ordering. A disadvantage is that the algorithm takes N_I times as long as the leader algorithm where N_I is the amount of iterations in the k–means clustering algorithm. To avoid spending large computation efforts, a maximum of $\mathfrak{T}_i$ iterations is allowed:

EuclideanKMeansClustering(k, ε, $\mathfrak{T}_i$)
1 $O \leftarrow$ ***new array*** of ***integer*** with size k
2 $\boldsymbol{\mathfrak{K}} \leftarrow ()$
3 ***for*** $i \leftarrow 0$ ***to*** $l-1$ ***do***
  3.1 $\overline{Y_i} \leftarrow \frac{\sum_{j=0}^{|\boldsymbol{S}|-1} y_i^j}{|\boldsymbol{S}|}$
  3.2 $s_i \leftarrow \frac{\sum_{j=0}^{|\boldsymbol{S}|-1} (y_i^j - \overline{Y_i})^2}{|\boldsymbol{S}|}$
4 ***for*** $i \leftarrow 0$ ***to*** $k-1$ ***do***
  4.1 $q \leftarrow$ Random($|\boldsymbol{S}|$)
  4.2 $\overline{\mathfrak{K}^i} \leftarrow \boldsymbol{y^q}$
  4.3 $\boldsymbol{\mathfrak{K}^i} \leftarrow ()$
  4.4 $O[i] \leftarrow i$
5 $first \leftarrow$ ***true***
6 $ready \leftarrow$ ***false***
7 $t \leftarrow 0$
8 ***while*** $\neg ready$ ***do***
  8.1 ***for*** $i \leftarrow 0$ ***to*** $k-1$ ***do***
    8.1.1 $\boldsymbol{\mathfrak{K}^i} \leftarrow ()$
  8.2 ***for*** $i \leftarrow 0$ ***to*** $|\boldsymbol{S}|-1$ ***do***
    8.2.1 $q \leftarrow$ Random(k)
    8.2.2 $d_{\min} \leftarrow d_{ES}(\boldsymbol{y^i}, \overline{\mathfrak{K}^{O[q]}})$
    8.2.3 $c_{\min} \leftarrow O[q]$
    8.2.4 $O[q] \leftrightarrow O[k-1]$
    8.2.5 ***for*** $j \leftarrow 0$ ***to*** $k-2$ ***do***
      8.2.5.1 $q \leftarrow$ Random($k-j-1$)
      8.2.5.2 ***if*** $d(\boldsymbol{y^i}, \overline{\mathfrak{K}^{O[q]}}) < d_{\min}$ ***then***
        8.2.5.2.1 $d_{\min} \leftarrow d(\boldsymbol{y^i}, \overline{\mathfrak{K}^{O[q]}})$
        8.2.5.2.2 $c_{\min} \leftarrow O[q]$
      8.2.5.3 $O[q] \leftrightarrow O[k-j-2]$
    8.2.6 $(\boldsymbol{\mathfrak{K}^{c_{\min}}})_{|\boldsymbol{\mathfrak{K}^{c_{\min}}}|} \leftarrow \boldsymbol{y^i}$

EUCLIDEANKMEANSCLUSTERING(k, ε, $\mathfrak{T}_i$)
8.3 $ready \leftarrow$ ***true***
8.4 ***for*** $i \leftarrow 0$ ***to*** $k-1$ ***do***
  8.4.1 $a \leftarrow \frac{\sum_{j=0}^{|\boldsymbol{\mathfrak{K}^i}|-1} \boldsymbol{\mathfrak{K}^i_j}}{|\boldsymbol{\mathfrak{K}^i}|}$
  8.4.2 ***if*** $|a - \overline{\mathfrak{K}^i}| \geq \varepsilon$ ***then***
    8.4.4 $ready \leftarrow$ ***false***
  8.4.3 $\overline{\mathfrak{K}^i} \leftarrow a$
8.5 ***if*** $first$ ***then***
  8.5.1 $first \leftarrow$ ***false***
  8.5.2 $ready \leftarrow$ ***false***
8.6 $t \leftarrow t+1$
8.7 ***if*** $t = \mathfrak{T}_i$ ***then***
  8.7.1 $ready \leftarrow$ ***true***
9 ***return***($\boldsymbol{\mathfrak{K}}$)

Just as for the leader algorithm, we can alter the k–means clustering algorithm to make use of the Mahalanobis distance measure. If a cluster has grown to some minimum size, the Mahalanobis distance may be used. Otherwise, the Euclidean distance is used. In subsequent generations, the Mahalanobis distance is always used as the covariance values are then kept fixed in any single loop. Again, the maximum amount of iterations that the algorithm is allowed, is given by $\mathfrak{T}_i$. The resulting algorithm can be formalized as follows:

MAHALANOBISKMEANSCLUSTERING(k, ε, $\mathfrak{T}_s$, $\mathfrak{T}_i$)
1 $\boldsymbol{a} \leftarrow$ ***new vector*** of ***real*** with size l
2 $\boldsymbol{B} \leftarrow$ ***new matrix*** of ***real*** with size $l \times l$
3 $O \leftarrow$ ***new array*** of ***integer*** with size k
4 $\boldsymbol{\mathfrak{K}} \leftarrow ()$
5 ***for*** $i \leftarrow 0$ ***to*** $l-1$ ***do***
  5.1 $\overline{Y_i} \leftarrow \frac{\sum_{j=0}^{|\boldsymbol{\mathcal{S}}|-1} y_i^j}{|\boldsymbol{\mathcal{S}}|}$
  5.2 $s_i \leftarrow \frac{\sum_{j=0}^{|\boldsymbol{\mathcal{S}}|-1} (y_i^j - \overline{Y_i})^2}{|\boldsymbol{\mathcal{S}}|}$
6 ***for*** $i \leftarrow 0$ ***to*** $k-1$ ***do***
  6.1 $q \leftarrow$ RANDOM($|\boldsymbol{\mathcal{S}}|$)
  6.2 $\overline{\mathfrak{K}^i} \leftarrow \boldsymbol{y^q}$
  6.3 ***for*** $j \leftarrow 0$ ***to*** $|\boldsymbol{\mathcal{S}}|-1$ ***do***
    6.3.1 ***for*** $q \leftarrow 0$ ***to*** $|\boldsymbol{\mathcal{S}}|-1$ ***do***
      6.3.1.1 $\boldsymbol{S}(\boldsymbol{\mathfrak{K}^i}, \boldsymbol{\mathcal{L}})(j,q) \leftarrow 0$
  6.4 $\boldsymbol{\mathfrak{K}^i} \leftarrow ()$
  6.5 $O[i] \leftarrow i$
7 $first \leftarrow$ ***true***
8 $ready \leftarrow$ ***false***
9 $t \leftarrow 0$
10 ***while*** $\neg ready$ ***do***
  10.1 ***for*** $i \leftarrow 0$ ***to*** $k-1$ ***do***
    10.1.1 $\boldsymbol{\mathfrak{K}^i} \leftarrow ()$

MahalanobisKMeansClustering(k, ε, $\mathfrak{T}_s$, $\mathfrak{T}_i$)

10.2 ***for*** $i \leftarrow 0$ ***to*** $|\boldsymbol{S}| - 1$ ***do***
10.2.1 $q \leftarrow$ Random(k)
10.2.2 ***if*** $\neg first \vee |\boldsymbol{\mathfrak{K}}^{\boldsymbol{O[q]}}| \geq \mathfrak{T}_s$ ***then***
10.2.2.1 $d_{\min} \leftarrow d_M(\boldsymbol{y^i}, O[q])$
10.2.3 ***else***
10.2.3.1 $d_{\min} \leftarrow d_{ES}(\boldsymbol{y^i}, \overline{\mathfrak{K}^{O[q]}})$
10.2.4 $c_{\min} \leftarrow O[q]$
10.2.5 $O[q] \leftrightarrow O[k-1]$
10.2.6 ***for*** $j \leftarrow 0$ ***to*** $k-2$ ***do***
10.2.6.1 $q \leftarrow$ Random($k - j - 1$)
10.2.6.2 ***if*** $\neg first \vee |\boldsymbol{\mathfrak{K}}^{\boldsymbol{O[q]}}| \geq \mathfrak{T}_s$ ***then***
10.2.6.2.1 $d \leftarrow d_M(\boldsymbol{y^i}, O[q])$
10.2.6.3 ***else***
10.2.6.3.1 $d \leftarrow d_{ES}(\boldsymbol{y^i}, \overline{\mathfrak{K}^{O[q]}})$
10.2.6.4 ***if*** $d < d_{\min}$ ***then***
10.2.6.2.1 $d_{\min} \leftarrow d$
10.2.6.2.2 $c_{\min} \leftarrow O[q]$
10.2.6.5 $O[q] \leftrightarrow O[k-j-2]$
10.2.7 $\boldsymbol{\mathfrak{K}}^{\boldsymbol{c_{\min}}}_{|\boldsymbol{\mathfrak{K}}_{\boldsymbol{c_{\min}}}|} \leftarrow \boldsymbol{y^i}$
10.2.8 ***if*** $first \wedge |\boldsymbol{\mathfrak{K}}^{\boldsymbol{c_{\min}}}| = \mathfrak{T}_s$ ***then***
10.2.8.1 ***for*** $j \leftarrow 0$ ***to*** $l-1$ ***do***
10.2.8.1.1 ***for*** $q \leftarrow j$ ***to*** $l-1$ ***do***
10.2.8.1.1.1 $\boldsymbol{S}(\boldsymbol{\mathfrak{K}}^{\boldsymbol{c_{\min}}}, \boldsymbol{\mathcal{L}})(j,q) \leftarrow \boldsymbol{S}(\boldsymbol{\mathfrak{K}}^{\boldsymbol{c_{\min}}}, \boldsymbol{\mathcal{L}})(j,q) + \left(\overline{\mathfrak{K}^{c_{\min}}}\right)_j \left(\overline{\mathfrak{K}^{c_{\min}}}\right)_q$
10.2.8.1.1.2 $\boldsymbol{S}(\boldsymbol{\mathfrak{K}}^{\boldsymbol{c_{\min}}}, \boldsymbol{\mathcal{L}})(j,q) \leftarrow \frac{|\boldsymbol{\mathfrak{K}}^{\boldsymbol{c_{\min}}}| \boldsymbol{S}(\boldsymbol{\mathfrak{K}}^{\boldsymbol{c_{\min}}}, \boldsymbol{\mathcal{L}})(j,q) + y^i_j y^i_q}{|\boldsymbol{\mathfrak{K}}^{\boldsymbol{c_{\min}}}| + 1}$
10.2.8.2 $\overline{\boldsymbol{\mathfrak{K}}^{\boldsymbol{c_{\min}}}} \leftarrow \frac{\overline{\mathfrak{K}^{c_{\min}}} |\boldsymbol{\mathfrak{K}}^{\boldsymbol{c_{\min}}}| + y^i}{|\boldsymbol{\mathfrak{K}}^{\boldsymbol{c_{\min}}}| + 1}$
10.2.8.3 ***for*** $j \leftarrow 0$ ***to*** $l-1$ ***do***
10.2.8.3.1 ***for*** $q \leftarrow j$ ***to*** $l-1$ ***do***
10.2.8.3.1.1 $\boldsymbol{S}(\boldsymbol{\mathfrak{K}}^{\boldsymbol{c_{\min}}}, \boldsymbol{\mathcal{L}})(j,q) \leftarrow \boldsymbol{S}(\boldsymbol{\mathfrak{K}}^{\boldsymbol{c_{\min}}}, \boldsymbol{\mathcal{L}})(j,q) - \left(\overline{\mathfrak{K}^{c_{\min}}}\right)_j \left(\overline{\mathfrak{K}^{c_{\min}}}\right)_q$
10.2.8.3.1.2 $\boldsymbol{S}(\boldsymbol{\mathfrak{K}}^{\boldsymbol{c_{\min}}}, \boldsymbol{\mathcal{L}})(q,j) \leftarrow \boldsymbol{S}(\boldsymbol{\mathfrak{K}}^{\boldsymbol{c_{\min}}}, \boldsymbol{\mathcal{L}})(j,q)$
10.3 $ready \leftarrow$ ***true***
10.4 ***for*** $i \leftarrow 0$ ***to*** $k-1$ ***do***
10.4.1 $\boldsymbol{a} \leftarrow \frac{\sum_{j=0}^{|\mathfrak{K}^{\boldsymbol{i}}|-1} \boldsymbol{\mathfrak{K}}^{\boldsymbol{i}}_{\boldsymbol{j}}}{|\boldsymbol{\mathfrak{K}}^{\boldsymbol{i}}|}$
10.4.2 ***if*** $\exists_{j \in \boldsymbol{\mathcal{L}}} \langle |\boldsymbol{a}_j - (\overline{\mathfrak{K}^i})_j| \geq \varepsilon \rangle$ ***then***
10.4.2.1 $ready \leftarrow$ ***false***
10.4.3 $\overline{\mathfrak{K}^i} \leftarrow \boldsymbol{a}$
10.4.4 $\boldsymbol{B} \leftarrow \frac{\sum_{j=0}^{|\mathfrak{K}^{\boldsymbol{i}}|-1} (\boldsymbol{\mathfrak{K}}^{\boldsymbol{i}}_{\boldsymbol{j}} - \overline{\mathfrak{K}^i})(\boldsymbol{\mathfrak{K}}^{\boldsymbol{i}}_{\boldsymbol{j}} - \overline{\mathfrak{K}^i})^T}{|\boldsymbol{\mathfrak{K}}^{\boldsymbol{i}}|}$
10.4.5 ***if*** $\exists_{(j,q) \in \boldsymbol{\mathcal{L}}^2} \langle |\boldsymbol{B}(j,q) - (\boldsymbol{S}(\boldsymbol{\mathfrak{K}}^{\boldsymbol{i}}, \boldsymbol{\mathcal{L}})(j,q)| \geq \varepsilon \rangle$ ***then***
10.4.5.1 $ready \leftarrow$ ***false***
10.4.6 $\boldsymbol{S}(\boldsymbol{\mathfrak{K}}^{\boldsymbol{i}}, \boldsymbol{\mathcal{L}}) \leftarrow \boldsymbol{B}$
10.5 ***if*** $first$ ***then***
10.5.1 $first \leftarrow$ ***false***
10.5.2 $ready \leftarrow$ ***false***
10.6 $t \leftarrow t + 1$
10.7 ***if*** $t = \mathfrak{T}_i$ ***then***
10.7.1 $ready \leftarrow$ ***true***
11 ***return***$(\boldsymbol{\mathfrak{K}})$

3.3.2 Setting parameters

On the one hand, we now have clustering algorithms on the basis of which we can define a mixture of factorizations. However, on the other hand, we have introduced new parameters that determine the amount of mixtures that will be introduced. For the k–means clustering algorithms, this amount equals exactly k. For the leader algorithms, this amount depends on the value for the distance threshold $\mathfrak{T}_d$. In order to perform factorization mixture selection, we clearly have to determine what value to use for these parameters.

Setting the parameters requires some experience. To automate this, we can apply the general approaches for the detection of dependence between variables from section 3.2. If we for instance take the k parameter for the k–means clustering algorithms, we can increment the value of k and compute the negative log–likelihood value. If based on this value the probability distribution is significantly better than for a smaller value of k, this value is accepted and the search continues. Alternatively, a scoring metric can be increasingly penalized as the amount of clusters increases. A similar scheme can be used for the distance threshold $\mathfrak{T}_d$.

It should be noted that using such methods requires a vast amount of computation time. Factorization selection for a single cluster can already take up a significant amount of computation time. Over k clusters, we thus require k times as much computation time. It therefore seems by far the most efficient choice to fix k or $\mathfrak{T}_d$ on beforehand.

3.3.3 Examples

In this section, we present some examples of the partioning algorithms that are described in the previous sections. We apply each of the the clustering algorithms from section 3.3 to a set of sample vectors and observe the results. We use sample vectors that can be displayed graphically to gain better insights into the workings of the algorithms. Sample vectors of a higher dimensionality than 3 cannot be visualized in an intuitive manner. Such sample vectors will appear when we apply the iterated density estimation to continuous optimization in section 6. We go over the sample vectors and apply the four clustering algorithms that we discussed in section 3.3.

We use 8 sample vectors. The first 7 of these are two dimensional and consist of various structures. The first sample vector is sampled from the uniform distribution. The second sample vector is sampled from a multiple of local uniform distributions, creating clusters. The third sample vector is a filled circle with a higher concentration of samples in the center and can be seen to have been sampled from a cone. The fourth sample vector is an outlined circle, which is difficult for most approaches. Even though the space can be described by a single parameter that is the angle around the center, this structure is hardly ever detected by any probability density estimation procedure. The fifth sample vector is a line that exemplifies full correlation between the two variables. The sixth and seventh sample vectors are sample vectors that express dependency, but in such a way that it usually cannot be accounted for by a single factorization. The last sample vector shows different behavior in different regions of the plane. In one region, the density is uniform, whereas in the other it is fully correlated. For any probabilistic model to efficiently describe this sample vector, it should be clustered into at least two sample vectors that are processed individually.

We have applied the four clustering algorithms from section 3.3 to each sample vector. For the k–means clustering algorithms, we used $k = 10$. The results are shown in figures 6 through to 13. Each figure shows, from left to right, the result of Euclidean leader clustering, Mahalanobis leader clustering, Euclidean k–means clustering and Mahalanobis k–means clustering. The distance threshold for the Euclidean leader clustering algorithm was $\mathfrak{T}_d = 1.0$. For the Mahalanobis leader clustering algorithm, the distance threshold was $\mathfrak{T}_d = 2.5$ and the minimum cluster size threshold was $\mathfrak{T}_s = 10$. For both k–means clustering algorithms, we used $k = 10, \varepsilon = 0.001, \mathfrak{T}_i = 10$. All of the settings were chosen so that a visually intuitive amount of clusters was obtained.

If we compare the leader results to the k–means results, it is clear that the cluster boundaries of the k–means results are strict in the sense that there is no overlap between the different clusters. Our randomized approach to the leader algorithm proves to be effective in the sense that there

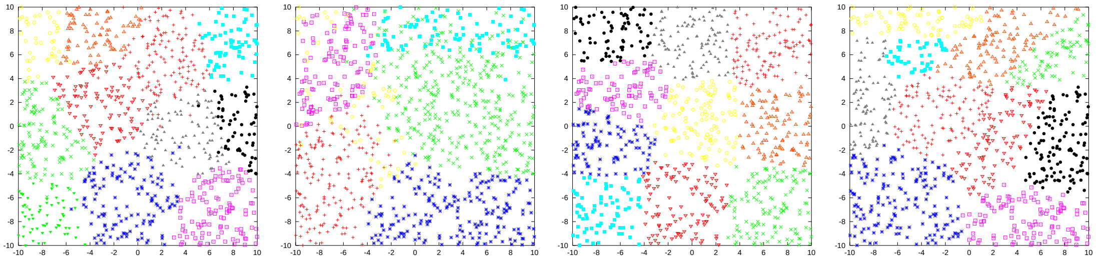

Figure 6: Cluster results for sample vector 0.

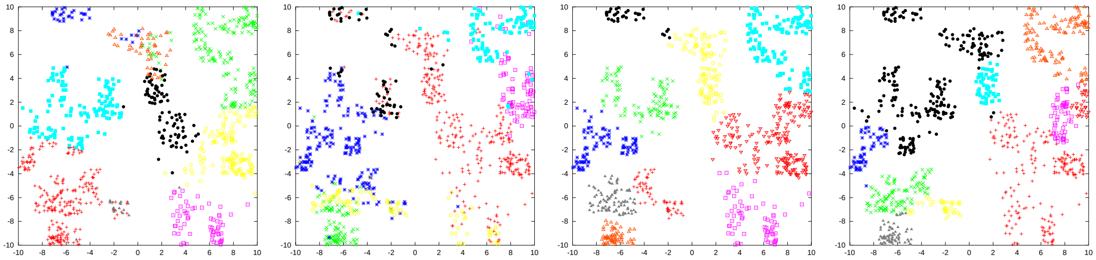

Figure 7: Cluster results for sample vector 1.

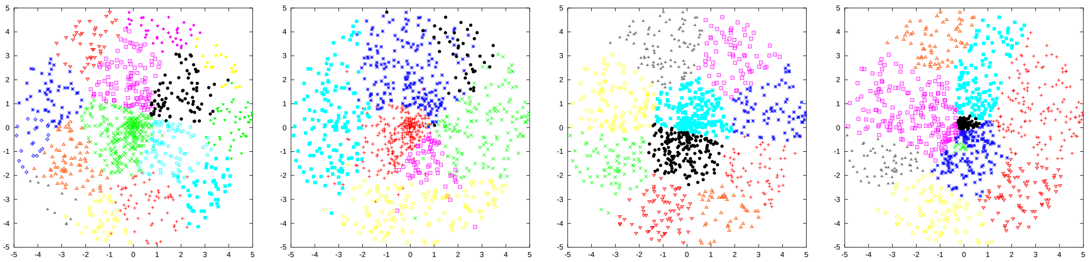

Figure 8: Cluster results for sample vector 2.

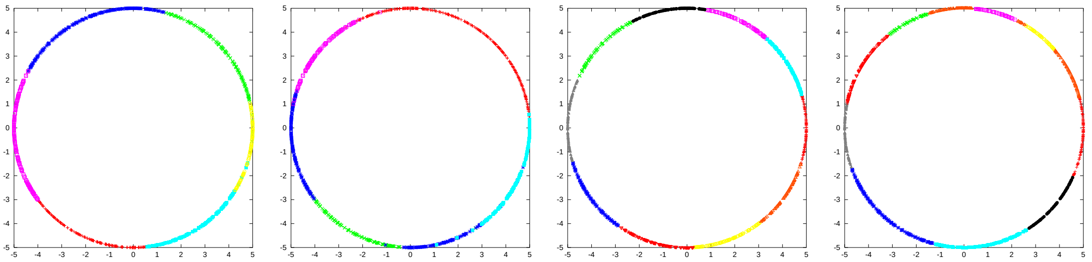

Figure 9: Cluster results for sample vector 3.

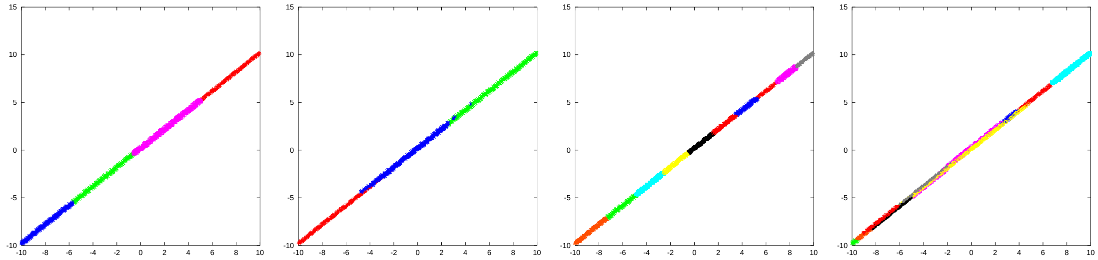

Figure 10: Cluster results for sample vector 4.

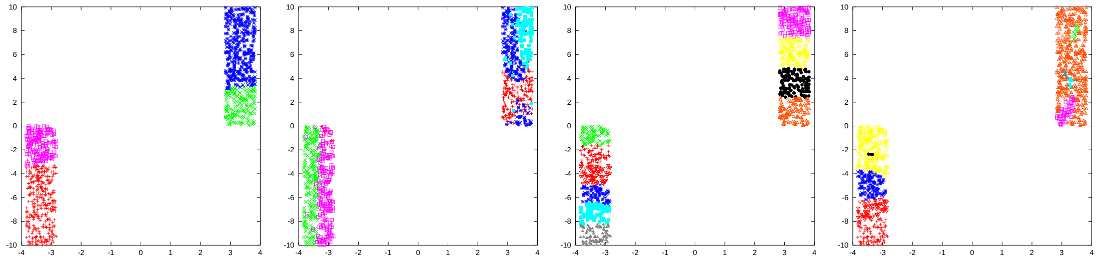

Figure 11: Cluster results for sample vector 5.

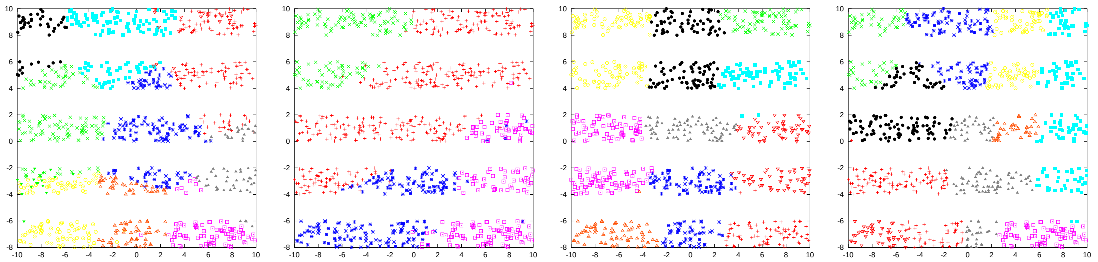

Figure 12: Cluster results for sample vector 6.

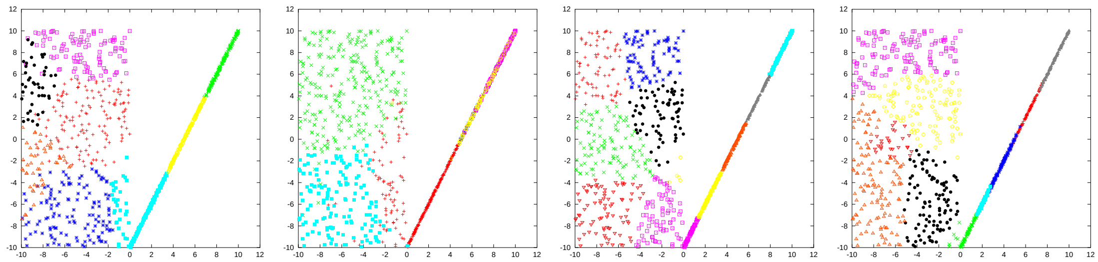

Figure 13: Cluster results for sample vector 7.

are no extremely large clusters in any of its results and that the boundaries are not disturbingly overlapping. This means that the resulting densities that are estimated on both of the clustering results, will not differ much. Especially for our application in which a correct density estimation is not as important as is a good approximation, the results of the leader algorithm seem at least as promising as those of the k–means clustering algorithm. Since the running time of the leader algorithm is smaller than that of the k–means algorithm, the leader algorithm seems preferable.

If we compare the results between the use of the Euclidean distance measure and the Mahalanobis distance measure, it is not directly clear which of these approaches leads to more desirable results. One of the main drawbacks of using the Mahalanobis distance measure, is that small clusters are more likely to appear. Such clusters can for instance lie completely inside another cluster since the normal pdf that is based upon it, is sharply peaked. It seems that the k–means algorithm in this case leads to somewhat better formed clusters with less overlap. The true advantage of the Mahalanobis distance in that it can better describe linear relations, is unfortunately not exploited in a useful way using the described clustering algorithms as becomes clear from figure 12 for instance. In this figure, the horizontal bars of sample points are not separated by the clusters even though this is possible by using the Mahalanobis distance. This means that the resulting estimated density will not be effective. As the Mahalanobis distance measure takes more computational effort, it seems that the Euclidean distance measure is preferable.

Given the computational speed of the leader algorithm and its acceptable two–dimensional results we have seen so far, this algorithm seems the most promising to use to use for factorization mixture selection based on clustering. The experiments in section 6 will bring the use of clustering within the $\mathbb{IDEA}$ framework into practice, which will indicate the actual computational usefulness of the proposed clustering algorithms.

4 Model fitting and sampling

Once a factorization has been selected, a pdf has to be fit for each element in the factorization. If we have used statistical hypothesis tests directly through for instance the correlation coefficient, we yet have to find such a fit. Then again, in the case of using the Kullback–Leibler divergence, the required parameters have been observed to already have been computed [8]. In general, we assume that our model structure has been selected but that the pdf information is not available yet. In this section, we concentrate on computing the pdf information. In general, whereas finding a model structure, is called *model selection*, estimating a pdf for each element in the structure as we do in this section, is called *model fitting*.

4.1 Observing pdf characteristics

In this article, we have described algorithms that use conditional densities and unconditional densities. We have also shown that the conditional densities are a more general case than are the unconditional densities as the latter can be written as a special case of the former.

From equation 3 we have that we only require unconditional densities. However, simplifying the actual conditional density can lead to expressions that can be computed more efficiently. We shall give the conditional density for each pdf that we discuss. First however, we discuss some pdfs in general. From these, we select the pdfs to actually work with. In subsequent sections we go into more detail on the selected pdfs.

We restrict ourselves to the use of the normal pdf and variants thereof. One of the main reasons for this, is that the normal pdf has relatively simple analytical properties. Furthermore, the central limit theorem tells us that given enough samples, the approximation to the normal pdf is often quite good for several components that represent some source.

Based on the normal pdf, we distinguish three different pdfs. These are the single multivariate normal pdf, the normal kernels pdf and the normal mixture pdf. Both the multivariate normal pdf as well as the normal kernels pdf have been proposed previously in the $\mathbb{IDEA}$ framework [8, 10, 11, 12]. The single multivariate normal pdf is defined as follows:

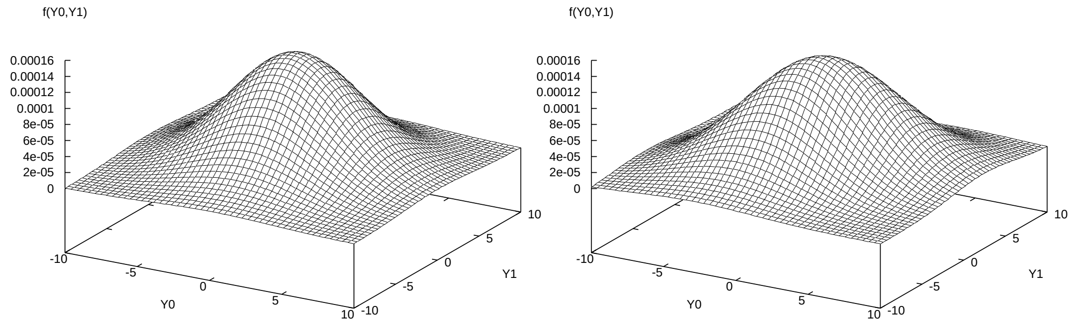

Figure 14: Fitting sample vectors 0 (left) and 1 (right) with a joint normal pdf.

$$f_{\mathcal{N}}(y\langle \boldsymbol{a}\rangle, \mu\langle \boldsymbol{a}\rangle, \boldsymbol{\Sigma}(\boldsymbol{a})) = \frac{(2\pi)^{-\frac{|\boldsymbol{a}|}{2}}}{(\det\ \boldsymbol{\Sigma}(\boldsymbol{a}))^{\frac{1}{2}}} e^{-\frac{1}{2}(y\langle \boldsymbol{a}\rangle - \mu\langle \boldsymbol{a}\rangle)^T(\boldsymbol{\Sigma}(\boldsymbol{a}))^{-1}(y\langle \boldsymbol{a}\rangle - \mu\langle \boldsymbol{a}\rangle)} \tag{38}$$

For sample vectors 0 and 1 that are given in section 3.3.3, we have fitted a maximum likelihood estimate of the multivariate normal pdf. The results can be seen in figure 14.

The normal kernels pdf is obtained by placing a multivariate normal pdf over each sample point. The covariance matrices of these normal kernels are fixed to have non–zero entries on the diagonal and zero entries off the diagonal. The entries on the diagonal are the individual variances of the normal pdf in each dimension. We denote such a variance in dimension j by $\mathfrak{s}_j$. The so constructed matrix implies that the normal kernels can be scaled in each dimension separately, but that each multivariate normal pdf cannot be aligned to any other axes than the standard axes. We denote the resulting matrix over variables $Y\langle \boldsymbol{a}\rangle$ by $\boldsymbol{\mathfrak{S}}(\boldsymbol{a})$:

$$\boldsymbol{\mathfrak{S}}(\boldsymbol{a}) = \begin{bmatrix} \mathfrak{s}^2_{\boldsymbol{a}_0} & 0 & \dots & 0 \\ 0 & \mathfrak{s}^2_{\boldsymbol{a}_1} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mathfrak{s}^2_{\boldsymbol{a}_{|\boldsymbol{a}|-1}} \end{bmatrix} \tag{39}$$

As a result, the multivariate normal kernels pdf can be formalized as follows:

$$f_{\mathcal{N}_K}(y\langle \boldsymbol{a}\rangle, \boldsymbol{\mathcal{S}}, \boldsymbol{\mathfrak{S}}(\boldsymbol{a})) = \frac{1}{|\boldsymbol{\mathcal{S}}|}\sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} f_{\mathcal{N}}(y\langle \boldsymbol{a}\rangle, y^i\langle \boldsymbol{a}\rangle, \boldsymbol{\mathfrak{S}}(\boldsymbol{a})) \tag{40}$$

In the definition of the normal kernels pdf, the standard deviation values $\mathfrak{s}_i$ are determined externally. Previously [12], a linear scaling of the standard deviations with respect to the range was proposed. One interesting property of the normal kernels pdf, is that by increasing the standard deviation values $\mathfrak{s}_i$, we allow for less detail in the final density estimation. Furthermore, an advantage over the use of the normal pdf, is that we have a better way of modeling clusters. A resulting fit over the two sample vectors can be seen in figure 15.

If we take M normal pdfs instead of $|\boldsymbol{\mathcal{S}}|$ and fit them to the sample data as well as possible while allowing for full covariance matrices, we have a normal mixture pdf. Note that we are fitting such a mixture over a *subset* of the variables instead of over all variables as we did in section 3.3. It may be clear that fitting such a normal mixture pdf over a subset of the available variables, can be done in the same way as described earlier in section 3.3, namely through the use of the

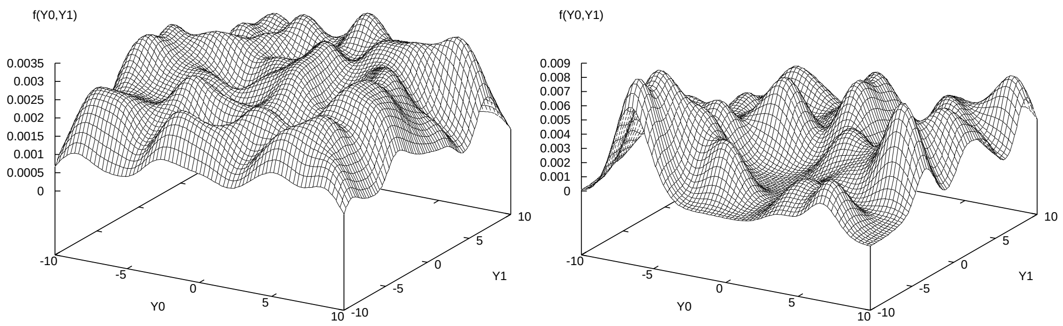

Figure 15: Fitting sample vectors 0 (left) and 1 (right) with a joint normal kernels pdf with $\mathfrak{s}_0 = \mathfrak{s}_1 = 1.0$.

EM algorithm or the adaptive EM algorithm. An advantage of the normal mixture pdf over the normal pdf and the normal kernels pdf, is that it is not as global and cluster insensitive as is the normal pdf and neither over–sensitive to clusters as is the normal kernels pdf. Therefore, the normal mixture pdf can be seen as a trade–off. The normal mixture pdf requires a vector of vectors of means as well as a vector of covariance matrices since we have multiple multivariate normal pdfs. We assume to have w such normal components in the mixture and let $\boldsymbol{\mathcal{W}} = (0, 1, \ldots, w-1)$. The vector of vectors of means is $\mu\langle \boldsymbol{a}\rangle\langle\boldsymbol{\mathcal{W}}\rangle$ and the vector of covariance matrices is $\boldsymbol{\Sigma}(\boldsymbol{a})\langle\boldsymbol{\mathcal{W}}\rangle$. We define $\mu^i\langle\boldsymbol{a}\rangle$ to be the mean vector of the i–th component $\mu^i\langle\boldsymbol{a}\rangle = (\mu\langle\boldsymbol{a}\rangle\langle\boldsymbol{\mathcal{W}}\rangle)_i$ and $\boldsymbol{\Sigma^i}(\boldsymbol{a})$ to be the covariance matrix of the i–th component $\boldsymbol{\Sigma^i}(\boldsymbol{a}) = (\boldsymbol{\Sigma}(\boldsymbol{a})\langle\boldsymbol{\mathcal{W}}\rangle)_i$. The normal mixture distribution can be formalized as in equation 41. A resulting fit over the two sample vectors can be seen in figure 16.

$$f_{\mathcal{N}_M}(y\langle\boldsymbol{a}\rangle, \alpha\langle\boldsymbol{\mathcal{W}}\rangle, \mu\langle\boldsymbol{a}\rangle\langle\boldsymbol{\mathcal{W}}\rangle, \boldsymbol{\Sigma}(\boldsymbol{a})\langle\boldsymbol{\mathcal{W}}\rangle) = \sum_{i=0}^{w-1} \alpha_i f_{\mathcal{N}}(y\langle\boldsymbol{a}\rangle, \mu^i\langle\boldsymbol{a}\rangle, \boldsymbol{\Sigma^i}(\boldsymbol{a})) \tag{41}$$

In equation 41, $\alpha\langle\boldsymbol{\mathcal{W}}\rangle$ is a vector of *mixing coefficients*. The same condition on the α mixing coefficients holds as in the case of the β mixing coefficients for the factorization mixture as discussed in section 3.3, namely that $\sum_{i=0}^{w-1} \alpha_i = 1$ and $\alpha_i \geq 0$ for $0 \leq i < w$.

The normal pdf has been used successfully on a variety of test functions [10, 11]. The normal pdf however strongly generalizes the data. This aspect becomes clear from figure 14 where clearly the clustered set is underfit. The normal kernels pdf overcomes this aspect. However, the costs for this are quite high. Firstly, evaluating the function takes $|\boldsymbol{\mathcal{S}}|$ times more time than the single normal pdf. This scales with the amount of available samples. However, getting more samples should imply that we get a better description of the source from which the samples were drawn and not that we necessarily get a much more costly pdf. Sampling from the distribution takes much more time as well. In order to solve higher dimensional problems, we require increasingly more samples. As the complexity of the normal kernels pdf increases with this amount and since the pdf is used very often throughout the optimization algorithm, the normal kernels pdf is not a very efficient way of reducing the strong generalization of the normal pdf. Secondly, it is not straightforward what values to use for the variances of each kernel. It has been observed [10] that the performance of the resulting algorithm strongly depends on the variances. This renders the normal kernels pdf hard to handle, even though it has been observed [10] that better results can be obtained with it on epistatic problems than with the normal pdf.

We point out that in terms of modeling, the normal mixture pdf generalizes both the normal

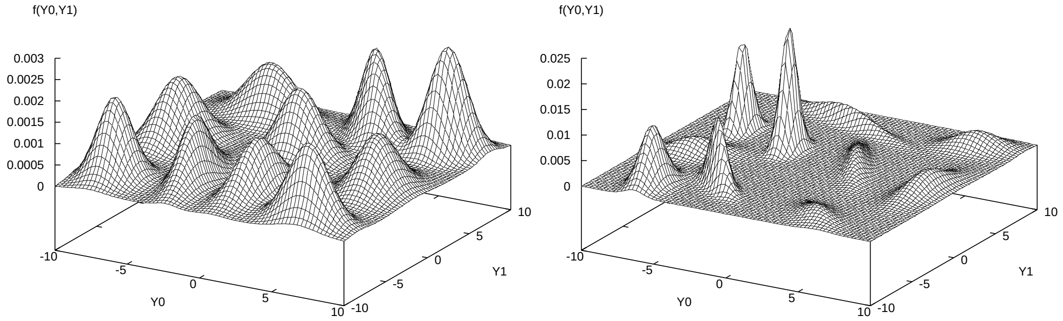

Figure 16: Fitting sample vectors 0 (left) and 1 (right) with a joint normal mixture pdf using the EM algorithm with 10 components and $\varepsilon = 10^{-3}$.

kernels pdf as well as the single normal pdf. If we have $w = 1$, we have a single normal pdf. On the other hand, if we have $w = |\mathcal{S}|$ and $\boldsymbol{\Sigma}^i(\boldsymbol{a}) = \mathfrak{S}(\boldsymbol{a})$, we get the normal kernels pdf.

We conclude that for simplicity and efficiency, it is beneficial to continue to use the single normal pdf. We refrain from using the normal kernels pdf any further since it is very sensitive to parameter settings and a good fit is hard to find. On the other hand, for the normal mixture pdf, the EM algorithm gives us an automated procedure to obtain a fit. Therefore, we continue to regard only the normal pdf and the normal mixture pdf. In the next subsections, we present some important derivations of the pdfs in order for them to be used in the $\mathbb{IDEA}$ framework. For each pdf, we first present its multivariate conditional version in which a single variable is conditioned on a multiple of others and show how we can sample from it as well as from the multivariate joint pdf. Subsequently, we show how the parameters $\boldsymbol{\theta}$ for the pdf can be estimated.

4.2 The normal pdf

The definition of the multivariate normal joint pdf, given a vector of variables $y\langle\boldsymbol{a}\rangle$, a mean vector $\mu\langle\boldsymbol{a}\rangle$ and a covariance matrix $\boldsymbol{\Sigma}(\boldsymbol{a})$, is given in equation 38. Using this definition, we can directly account for unconditional dependencies as these result in multivariate joint pdfs. Therefore, we can evaluate a sample in a marginal product model that is fit with normal pdfs. However, in the case of conditional dependencies, we must be able to *evaluate* the multivariate normal conditional pdf. Furthermore, to complete the resulting algorithms, we need to be able to sample from the multivariate normal joint pdf as well as from the multivariate normal conditional pdf. In this section, we first show how to perform the actual sampling. Once this is known, we know how to sample from both the special case multivariate conditional normal pdf as well as the multivariate joint normal pdf. However, given a model structure, we still have to estimate the parameters of the normal pdfs, which is the actual model fitting. How to best estimate the parameters, is what we show second in this section.

4.2.1 Sampling

We first want to know how to sample from the special case multivariate conditional normal pdf. Usually, only sampling from a one dimensional normal pdf is available. However, it was pointed out in earlier work [8] that the multivariate normal conditional pdf in which a single variable $y_{\boldsymbol{a}_0}$ is conditioned on $|\boldsymbol{a}| - 1$ other variables $y\langle\boldsymbol{a} - \boldsymbol{a}_0\rangle$, is exactly a one dimensional normal pdf and can be written as follows:

$$f_{\mathcal{N}}((y_{\boldsymbol{a}_0}|y\langle\boldsymbol{a}-\boldsymbol{a}_0\rangle),\mu\langle\boldsymbol{a}\rangle,\boldsymbol{\Sigma}(\boldsymbol{a})) = f_{\mathcal{N}}(y_{\boldsymbol{a}_0},\tilde{\mu},\tilde{\sigma}^2) \tag{42}$$

$$\text{where } \begin{cases} \tilde{\sigma}^2 &= \frac{1}{\boldsymbol{\Sigma}(\boldsymbol{a})^{-1}(0,0)} \\ \tilde{\mu} &= \frac{\mu_{\boldsymbol{a}_0}\boldsymbol{\Sigma}(\boldsymbol{a})^{-1}(0,0)-\sum_{i=1}^{|\boldsymbol{a}|-1}(y_{a_i}-\mu_{a_i})\boldsymbol{\Sigma}(\boldsymbol{a})^{-1}(i,0)}{\boldsymbol{\Sigma}(\boldsymbol{a})^{-1}(0,0)} \end{cases}$$

If we use a factorization based on conditional dependencies, we can directly use the pdf in equation 42 to draw more samples from the resulting probability distribution. In the case of unconditional dependencies however, we have to draw samples from a product of multivariate *joint* normal pdfs. We have assumed that only sampling from a one dimensional normal pdf is available. Therefore, if we have to sample from a multivariate normal pdf, we use equation 3 to write the multivariate pdf as a product of multivariate conditional pdfs for which we know that they can be written as one dimensional normal pdfs using equation 42. This procedure is known as sampling by *simulation*.

The multivariate differential entropy for the multivariate joint normal pdf can be derived analytically. It can be shown (see for instance [15]) that it equals:

$$h(f_{\mathcal{N}}(y\langle\boldsymbol{a}\rangle,\mu\langle\boldsymbol{a}\rangle,\boldsymbol{\Sigma}(\boldsymbol{a}))) = \frac{1}{2}(|\boldsymbol{a}| + \ln((2\pi)^{|\boldsymbol{a}|}(\det\boldsymbol{\Sigma}(\boldsymbol{a})))) \tag{43}$$

4.2.2 Parameter estimation

From equations 38 and 42 it follows that we have to estimate the values for $\mu\langle\boldsymbol{a}\rangle$ and $\boldsymbol{\Sigma}(\boldsymbol{a})$. It can be shown that a maximum likelihood fit of the normal pdf is found by using the sample average $\overline{Y}\langle\boldsymbol{a}\rangle$ and sample covariance matrix $\boldsymbol{S}(\boldsymbol{a})$ for them respectively:

$$\overline{Y}\langle\boldsymbol{a}\rangle = \frac{1}{|\boldsymbol{\mathcal{S}}|}\sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} y^i\langle\boldsymbol{a}\rangle \tag{44}$$

$$\boldsymbol{S}(\boldsymbol{a}) = \frac{1}{|\boldsymbol{\mathcal{S}}|}\sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1}(y^i\langle\boldsymbol{a}\rangle - \overline{Y}\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle - \overline{Y}\langle\boldsymbol{a}\rangle)^T \tag{45}$$

4.3 The normal mixture pdf

The definition of a multivariate normal mixture pdf is given in equation 41. Just as in the case of a single normal pdf, we can thereby directly account for unconditional dependencies. In this section, we firstly derive an expression for the multivariate conditional normal mixture pdf and subsequently note how to sample from it. Also, we show how to sample from the multivariate joint normal mixture pdf. Secondly, we show how the EM algorithm can be used to estimate the parameters of a multivariate normal mixture pdf.

4.3.1 Sampling

To sample from a model that incorporates conditional dependencies, we must first derive the multivariate conditional normal mixture pdf in which a single variable $y_{\boldsymbol{a}_0}$ is conditioned on multiple others $y\langle\boldsymbol{a}-\boldsymbol{a}_0\rangle$. To this end, we note that any expression that contains only the variables $y\langle\boldsymbol{a}-\boldsymbol{a}_0\rangle$ is a constant, since these variables are *given* in the conditional expression. Using equation 2, we have:

$$f_{\mathcal{N}_M}((y_{\boldsymbol{a}_0}|y\langle\boldsymbol{a}-\boldsymbol{a}_0\rangle),\alpha\langle\boldsymbol{\mathcal{W}}\rangle,\mu\langle\boldsymbol{a}\rangle\langle\boldsymbol{\mathcal{W}}\rangle,\boldsymbol{\Sigma}(\boldsymbol{a})\langle\boldsymbol{\mathcal{W}}\rangle) \tag{46}$$

$$= \frac{f_{\mathcal{N}_M}(y\langle\boldsymbol{a}\rangle,\alpha\langle\boldsymbol{\mathcal{W}}\rangle,\mu\langle\boldsymbol{a}\rangle\langle\boldsymbol{\mathcal{W}}\rangle,\boldsymbol{\Sigma}(\boldsymbol{a})\langle\boldsymbol{\mathcal{W}}\rangle)}{f_{\mathcal{N}_M}(y\langle\boldsymbol{a}-\boldsymbol{a}_0\rangle,\alpha\langle\boldsymbol{\mathcal{W}}\rangle,\mu\langle\boldsymbol{a}-\boldsymbol{a}_0\rangle\langle\boldsymbol{\mathcal{W}}\rangle,\boldsymbol{\Sigma}(\boldsymbol{a}-\boldsymbol{a}_0)\langle\boldsymbol{\mathcal{W}}\rangle)}$$

The expression in the denominator of equation 46 depends only on variables $y\langle \boldsymbol{a} - \boldsymbol{a}_0\rangle$. As the values for these variables are *given*, we define a constant $c_0 = f_{\mathcal{N}_M}(y\langle \boldsymbol{a} - \boldsymbol{a}_0\rangle, \alpha\langle \boldsymbol{\mathcal{W}}\rangle, \mu\langle \boldsymbol{a} - \boldsymbol{a}_0\rangle\langle \boldsymbol{\mathcal{W}}\rangle, \boldsymbol{\Sigma}(\boldsymbol{a}-\boldsymbol{a}_0)\langle \boldsymbol{\mathcal{W}}\rangle)$. Using equation 2 on the nominator of equation 46, we continue to write:

$$= \frac{1}{c_0}\sum_{i=0}^{w-1} \alpha_i f_{\mathcal{N}}(y\langle \boldsymbol{a}\rangle, \mu^i\langle \boldsymbol{a}\rangle, \boldsymbol{\Sigma^i}(\boldsymbol{a})) \tag{47}$$

$$= \frac{1}{c_0}\sum_{i=0}^{w-1} \alpha_i f_{\mathcal{N}}((y_{\boldsymbol{a}_0}|\langle \boldsymbol{a} - \boldsymbol{a}_0\rangle), \mu^i\langle \boldsymbol{a}\rangle, \boldsymbol{\Sigma^i}(\boldsymbol{a})) f_{\mathcal{N}}(y\langle \boldsymbol{a} - \boldsymbol{a}_0\rangle, \mu^i\langle \boldsymbol{a} - \boldsymbol{a}_0\rangle, \boldsymbol{\Sigma^i}(\boldsymbol{a} - \boldsymbol{a}_0))$$

At this point, another constant appears as the last factor in equation 47 again only depends on variables $y\langle \boldsymbol{a} - \boldsymbol{a}_0\rangle$. This time however, the constant is additionally dependent on i. Therefore, we cannot move this factor outside the summation. The same is the case for the α_i factors. We introduce a "constant" that is a function of i by $c_1(i) = \alpha_i f_{\mathcal{N}}(y\langle \boldsymbol{a} - \boldsymbol{a}_0\rangle, \mu^i\langle \boldsymbol{a} - \boldsymbol{a}_0\rangle, \boldsymbol{\Sigma^i}(\boldsymbol{a} - \boldsymbol{a}_0))$. If we additionally use equation 42 on the multivariate conditional normal pdf factor that has appeared, we get the result for the special case multivariate conditional normal mixture pdf:

$$= \sum_{i=0}^{w-1} \frac{c_1(i)}{c_0} f_{\mathcal{N}}((y_{\boldsymbol{a}_0}|\langle \boldsymbol{a} - \boldsymbol{a}_0\rangle), \mu^i\langle \boldsymbol{a}\rangle, \boldsymbol{\Sigma^i}(\boldsymbol{a})) \tag{48}$$

$$= \sum_{i=0}^{w-1} \frac{\alpha_i f_{\mathcal{N}}(y\langle \boldsymbol{a} - \boldsymbol{a}_0\rangle, \mu^i\langle \boldsymbol{a} - \boldsymbol{a}_0\rangle, \boldsymbol{\Sigma^i}(\boldsymbol{a} - \boldsymbol{a}_0))}{f_{\mathcal{N}_M}(y\langle \boldsymbol{a} - \boldsymbol{a}_0\rangle, \alpha\langle \boldsymbol{\mathcal{W}}\rangle, \mu\langle \boldsymbol{a} - \boldsymbol{a}_0\rangle\langle \boldsymbol{\mathcal{W}}\rangle, \boldsymbol{\Sigma}(\boldsymbol{a} - \boldsymbol{a}_0)\langle \boldsymbol{\mathcal{W}}\rangle)} f_{\mathcal{N}}(y_{\boldsymbol{a}_0}, \tilde{\mu^i}, (\tilde{\sigma^i})^2)$$

$$\text{where } \begin{cases} (\tilde{\sigma^i})^2 &= \frac{1}{\boldsymbol{\Sigma^i}(\boldsymbol{a})^{-1}(0,0)} \\ \tilde{\mu^i} &= \frac{\mu^i_{\boldsymbol{a}_0}\boldsymbol{\Sigma^i}(\boldsymbol{a})^{-1}(0,0) - \sum_{i=1}^{|\boldsymbol{a}|-1}(y_{a_i} - \mu^i_{a_i})\boldsymbol{\Sigma^i}(\boldsymbol{a})^{-1}(i,0)}{\boldsymbol{\Sigma^i}(\boldsymbol{a})^{-1}(0,0)} \end{cases}$$

The resulting expression is thus a weighted sum of one dimensional normal pdfs. Therefore, in order to sample from the multivariate conditional normal mixture pdf, we first have to compute the weights $c_1(i)/c_0$. We then select the i–th normal pdf with probability $c_1(i)/c_0$. Since the complete expression is a pdf and each element in the sum is a weight times a normal pdf, the weights have to sum to 1. As the nominator is a component of the mixture in the denominator, $c_0 = \sum_{i=0}^{w-1} c_1(i)$, this can be seen to be the case. This implies that c_0 does not have to be computed separately, but that only the $c_1(i)$ have to be computed.

Next to being able to sample from the required multivariate conditional normal mixture pdf, we also have to be able to sample from the multivariate joint normal mixture pdf. This is quite simple compared to the conditional case. The joint pdf in equation 41 is a sum of weighted multivariate joint normal pdfs. Given the constraint that the α_i have to sum to 1 and may not be negative, we can select the i–th multivariate normal pdf to sample from with probability α_i. How to sample from a single multivariate normal pdf has already been discussed in section 4.2.

4.3.2 Parameter estimation

Estimating the parameters of each multivariate joint normal pdf, given w, is not straightforward. To this end, the EM algorithm has been proposed [16]. This algorithm does not guarantee a maximum likelihood fit, but at least it is a general approach to finding an approximation.

Even though we get an approximation by using the EM algorithm, the choice of w still has to be made. This is somewhat of a similar choice as in the case of selecting the amount of clusters in factorization mixture selection. An adaptive version of the EM algorithm can be constructed so as to prevent the user from having to specify the amount of normal pdfs to use in a fit over the data. Instead, the user must provide a threshold value in the same fashion as for the leader algorithms. Just as we did in the case of the k–means clustering algorithms however, we restrict ourselves to the non–adaptive algorithm.

The EM algorithm is an iterative parameter estimation procedure. Given a certain instance for the parameters, which is termed the *old* parameters, a single iteration consists of finding *new* estimates for these parameters. This recomputation of the parameters is iteratively repeated until no changes occur anymore within a certain precision or until a maximum of iterations has been reached. In appendix C, the update equations for a single iteration of the EM algorithm for a multivariate normal mixture model are derived given a vector of indices $\boldsymbol{a}$. The resulting equations concern the mixing coefficients α_j, the mean vectors $\mu^j\langle\boldsymbol{a}\rangle$ and the covariance matrices $\boldsymbol{\Sigma^j}(\boldsymbol{a})$ for each component $j \in \boldsymbol{\mathcal{W}}$ in the mixture:

$$\alpha_j^{\text{new}} = \frac{1}{|\boldsymbol{\mathcal{S}}|}\sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \frac{\alpha_j^{\text{old}}\{\hat{P}^j(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}}{\{\hat{P}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}} \tag{49}$$

$$\{\mu^j\langle\boldsymbol{a}\rangle\}^{\text{new}} = \frac{\sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \frac{\alpha_j^{\text{old}}\{\hat{P}^j(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}}{\{\hat{P}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}} y^i\langle\boldsymbol{a}\rangle}{\sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \frac{\alpha_j^{\text{old}}\{\hat{P}^j(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}}{\{\hat{P}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}}} \tag{50}$$

$$\{\boldsymbol{\Sigma^j}(\boldsymbol{a})\}^{\text{new}} = \frac{\sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \frac{\alpha_j^{\text{old}}\{\hat{P}^j_{\mathfrak{f}_j}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}}{\{\hat{P}_{\mathfrak{f}}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}}(y^i\langle\boldsymbol{a}\rangle - \{\mu^j\langle\boldsymbol{a}\rangle\}^{\text{new}})(y^i\langle\boldsymbol{a}\rangle - \{\mu^j\langle\boldsymbol{a}\rangle\}^{\text{new}})^T}{\sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \frac{\alpha_j^{\text{old}}\{\hat{P}^j_{\mathfrak{f}_j}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}}{\{\hat{P}_{\mathfrak{f}}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}}} \tag{51}$$

Even though the EM algorithm is derived so as to obtain a maximum likelihood estimate, the resulting estimation may not be one of maximum likelihood. The minimization problem contains many local minima and the algorithm may get stuck in one of them. Since the EM algorithm is essentially a gradient algorithm, this can easily happen if the problem has many local optima. To tackle this issue, an evolutionary algorithm could for instance be applied. However, this yields a chicken and egg problem since within the evolutionary algorithms that learn structure, tools such as the EM algorithm are used again.

We close this section by presenting pseudo–code for the EM algorithm for a normal mixture with full covariance matrices. The algorithm is based upon the update equations given above as derived in appendix C. The algorithm is defined with respect to an indices vector $\boldsymbol{a}$. These indices define the variables for which a multivariate joint normal mixture pdf must be estimated. Just as was the case for the k–means algorithm, we require an a priori specified amount of components k. We also use a convergence significance level ε as well as a maximum amount of iterations $\mathfrak{T}_i$:

NormalMixtureEMAlgorithm(k, ε, $\mathfrak{T}_i$, $\boldsymbol{a}$)
1 $a^o, a^n \leftarrow 2$ ***new arrays*** of ***real*** with size k
2 $m^o, m^n \leftarrow 2$ ***new arrays*** of (***vector*** of ***real*** with size $|\boldsymbol{a}|$) with size k
3 $S^o, S^n \leftarrow 2$ ***new arrays*** of (***matrix*** of ***real*** with size $|\boldsymbol{a}| \times |\boldsymbol{a}|$) with size k
4 $p \leftarrow$ ***new array*** of ***real*** in 2 dimensions with size $k \times |\boldsymbol{\mathcal{S}}|$
5 $p^m \leftarrow$ ***new array*** of ***real*** with size $|\boldsymbol{\mathcal{S}}|$
6 $d \leftarrow$ ***new array*** of ***real*** with size k
7 $r, min \leftarrow$ ***new array*** of ***real*** with size $|\boldsymbol{a}|$
8 ***for*** $i \leftarrow 0$ ***to*** $|\boldsymbol{a}| - 1$ ***do***
  8.1 $min[i] \leftarrow \min\{\boldsymbol{y}_{\boldsymbol{a}_i} \mid \boldsymbol{y} \in \boldsymbol{\mathcal{S}}\}$
  8.2 $max \leftarrow \max\{\boldsymbol{y}_{\boldsymbol{a}_i} \mid \boldsymbol{y} \in \boldsymbol{\mathcal{S}}\}$
  8.3 $r[i] \leftarrow max - min[i]$
9 ***for*** $k \leftarrow 0$ ***to*** $k - 1$ ***do***
  9.1 $a^o[k] \leftarrow \frac{1}{k}$
  9.2 ***for*** $i \leftarrow 0$ ***to*** $|\boldsymbol{a}| - 1$ ***do***
    9.3 $m^o[k]_i \leftarrow$ Random01() $\cdot\, r[i] + min[i]$
    9.3 ***for*** $j \leftarrow 0$ ***to*** $|\boldsymbol{a}| - 1$ ***do***
      9.3.1 $S^o[k]_{ij} \leftarrow 0$
    9.4 $S^o[k]_{ii} \leftarrow \frac{1}{5}r[i]$

10 $ready \leftarrow$ ***false***

11 $t \leftarrow 0$

12 ***while*** $\neg ready$ ***do***

  12.1 ***for*** $i \leftarrow 0$ ***to*** $|\boldsymbol{S}| - 1$ ***do***

    12.1.1 $p^m[i] \leftarrow 0$

    12.1.2 ***for*** $k \leftarrow 0$ ***to*** $k - 1$ ***do***

      12.1.2.1 $p[k,i] \leftarrow f_{\mathcal{N}}(y^i\langle \boldsymbol{a} \rangle, m^o[k], S^o[k])$

      12.1.2.2 $p^m[i] \leftarrow p^m[i] + a^o[k]p[k,i]$

  12.2 ***for*** $k \leftarrow 0$ ***to*** $k - 1$ ***do***

    12.2.1 $d[k] \leftarrow 0$

    12.2.2 ***for*** $i \leftarrow 0$ ***to*** $|\boldsymbol{S}| - 1$ ***do***

      12.2.2.1 $d[k] \leftarrow d[k] + \frac{a^o[k]p[k,i]}{p^m[i]}$

    12.2.3 $a^n[k] \leftarrow \frac{d[k]}{|\boldsymbol{S}|}$

    12.2.4 ***for*** $i \leftarrow 0$ ***to*** $|\boldsymbol{a}| - 1$ ***do***

      12.2.4.1 $m^n[k]_i \leftarrow 0$

    12.2.5 ***for*** $i \leftarrow 0$ ***to*** $|\boldsymbol{S}| - 1$ ***do***

      12.2.5.1 $m^n[k] \leftarrow m^n[k] + \frac{a^o[k]p[k,i]y^i\langle \boldsymbol{a} \rangle}{p^m[i]}$

    12.2.6 $m^n[k] \leftarrow \frac{m^n[k]}{d[k]}$

    12.2.7 ***for*** $i \leftarrow 0$ ***to*** $|\boldsymbol{a}| - 1$ ***do***

      12.2.7.1 ***for*** $j \leftarrow 0$ ***to*** $|\boldsymbol{a}| - 1$ ***do***

        12.2.7.1.1 $S^n[k]_{ij} \leftarrow 0$

    12.2.8 ***for*** $i \leftarrow 0$ ***to*** $|\boldsymbol{S}| - 1$ ***do***

      12.2.8.1 $S^n[k] \leftarrow S^n[k] + \frac{a^o[k]p[k,i](y^i\langle \boldsymbol{a} \rangle - m^n[k])(y^i\langle \boldsymbol{a} \rangle - m^n[k])^T}{p^m[i]}$

    12.2.9 $S^n[k] \leftarrow \frac{S^n[k]}{d[k]}$

  12.3 $ready \leftarrow$ ***true***

  12.4 ***for*** $k \leftarrow 0$ ***to*** $k - 1$ ***do***

    12.4.1 ***if*** $|a^n[k] - a^o[k]| \geq \varepsilon$ ***then***

      12.4.1.1 $ready \leftarrow$ ***false***

    12.4.2 $a^o[k] \leftarrow a^n[k]$

    12.4.3 ***for*** $i \leftarrow 0$ ***to*** $|\boldsymbol{a}| - 1$ ***do***

      12.4.3.1 ***if*** $|m^n[k]_i - m^o[k]_i| \geq \varepsilon$ ***then***

        12.4.3.1.1 $ready \leftarrow$ ***false***

      12.4.3.2 $m^o[k]_i \leftarrow m^n[k]_i$

    12.4.4 ***for*** $i \leftarrow 0$ ***to*** $|\boldsymbol{a}| - 1$ ***do***

      12.4.4.1 ***for*** $j \leftarrow 0$ ***to*** $|\boldsymbol{a}| - 1$ ***do***

        12.4.4.1.1 ***if*** $|S^n[k]_{ij} - S^o[k]_{ij}| \geq \varepsilon$ ***then***

          12.4.4.1.1.1 $ready \leftarrow$ ***false***

        12.4.4.1.2 $S^o[k]_{ij} \leftarrow S^n[k]_{ij}$

  12.5 $t \leftarrow t + 1$

  12.6 ***if*** $t = \mathfrak{T}_i$ ***then***

    12.6.1 $ready \leftarrow$ ***true***

13 ***for*** $k \leftarrow 0$ ***to*** $k - 1$ ***do***

  13.1 $\alpha_k \leftarrow a^n[k]$

  13.2 $\mu^k\langle \boldsymbol{a} \rangle \leftarrow m^n[k]$

  13.3 $\boldsymbol{\Sigma^k}(\boldsymbol{a}) \leftarrow S^n[k]$

5 IDEAs

Our aim is to apply the use of probabilistic models to continuous optimization. So far, we have formalized useful techniques and algorithms to find such models. However, we have not yet elaborated on how these probabilistic models can be used. In this section, we present the IDEA framework for Iterated Density Estimation Evolutionary Algorithms. In this framework, probabilistic models are built and used in optimization with an evolutionary algorithm. In section 5.1 we first go over the framework itself. Second, in section 5.2 we elaborate on the algorithmic instances of the framework that were used in earlier work. Third and finally, we summarize what techniques we will apply for the first time to get new algorithmic instances of the IDEA framework.

5.1 The IDEA framework

We write the cost function of our continuous optimization problem as $C(y\langle\boldsymbol{\mathcal{L}}\rangle)$ and without loss of generality, we assume that we want to *minimize* $C(y\langle\boldsymbol{\mathcal{L}}\rangle)$. For every problem variable y_i, we introduce a continuous random variable Y_i and get $\boldsymbol{\mathcal{Y}} = Y\langle\boldsymbol{\mathcal{L}}\rangle$. Without any prior information on $C(y\langle\boldsymbol{\mathcal{L}}\rangle)$, we might as well assume a uniform distribution over $\boldsymbol{\mathcal{Y}}$. Therefore, we generate an initial (population) vector of n samples at random. Now we let $P^{\theta}(\boldsymbol{\mathcal{Y}})$ be a probability distribution that is uniform over all vectors $y\langle\boldsymbol{\mathcal{L}}\rangle$ with $C(y\langle\boldsymbol{\mathcal{L}}\rangle) \leq \theta$. Sampling from $P^{\theta}(\boldsymbol{\mathcal{Y}})$ gives more samples that evaluate to a value below θ. Moreover, if we know $\theta^* = \min_{y\langle\boldsymbol{\mathcal{L}}\rangle}\{C(y\langle\boldsymbol{\mathcal{L}}\rangle)\}$, a single sample gives an optimal solution. To use this in an iterated algorithm, we select $\lfloor \tau n \rfloor$ samples in each iteration t and let θ_t be the worst selected sample cost. We then estimate the distribution of the selected samples and thereby find $\hat{P}^{\theta_t}_{\varsigma}(\boldsymbol{\mathcal{Y}})$ as an approximation to the true distribution $P^{\theta_t}(\boldsymbol{\mathcal{Y}})$. New samples can then be drawn from $\hat{P}^{\theta_t}_{\varsigma}(\boldsymbol{\mathcal{Y}})$ and be used to replace some of the current samples. This rationale has led to the definition of the IDEA framework [8]. The largest difference between other similar approaches and the IDEA, is that the IDEA has mostly been used to focus on continuous optimization problems [8, 10, 11, 12, 13]. The definition of the IDEA framework is the following:

IDEA($n, \tau, m, sel(), rep(), ter(), sea(), est(), sam()$)	
Initialize an empty vector of samples	$\boldsymbol{\mathcal{P}} \leftarrow ()$
Add and evaluate n random samples	***for*** $i \leftarrow 0$ ***to*** $n-1$ ***do***
	$\boldsymbol{\mathcal{P}} \leftarrow \boldsymbol{\mathcal{P}} \sqcup$ NewRandomVector()
	$c[\boldsymbol{\mathcal{P}}_i] \leftarrow C(\boldsymbol{\mathcal{P}}_i)$
Initialize the iteration counter	$t \leftarrow 0$
Iterate until termination	***while*** $\neg ter()$ ***do***
Select $\lfloor \tau n \rfloor$ samples	$(y^0\langle\boldsymbol{\mathcal{L}}\rangle, y^1\langle\boldsymbol{\mathcal{L}}\rangle, \ldots, y^{\lfloor \tau n \rfloor - 1}\langle\boldsymbol{\mathcal{L}}\rangle) \leftarrow sel()$
Set θ_t to the worst selected cost	$\theta_t \leftarrow c[y^k\langle\boldsymbol{\mathcal{L}}\rangle]$ such that
	$\forall_{i \in \boldsymbol{\mathcal{N}}_\tau}\langle c[y^i\langle\boldsymbol{\mathcal{L}}\rangle] \leq c[y^k\langle\boldsymbol{\mathcal{L}}\rangle]\rangle$
Search for a structure ς	$\varsigma \leftarrow sea()$
Estimate the parameters $\boldsymbol{\theta} \xleftarrow{fit} \varsigma$	$\boldsymbol{\theta} \leftarrow est()$
Create an empty vector of new samples	$\boldsymbol{\mathcal{O}} \leftarrow ()$
Sample m new samples from $\hat{P}_{\varsigma}(\boldsymbol{\mathcal{Y}})$	***for*** $i \leftarrow 0$ ***to*** $m-1$ ***do***
	$\boldsymbol{\mathcal{O}} \leftarrow \boldsymbol{\mathcal{O}} \sqcup sam()$
Replace a part of $\boldsymbol{\mathcal{P}}$ with a part of $\boldsymbol{\mathcal{O}}$	$rep()$
Evaluate the new samples in $\boldsymbol{\mathcal{P}}$	***for each*** unevaluated $\boldsymbol{\mathcal{P}}_i$ ***do***
	$c[\boldsymbol{\mathcal{P}}_i] \leftarrow C(\boldsymbol{\mathcal{P}}_i)$
Update the iteration counter	$t \leftarrow t+1$
Denote the required iterations by t_{end}	$t_{\text{end}} \leftarrow t$

In the IDEA framework, we have that $\boldsymbol{\mathcal{N}}_{\boldsymbol{\tau}} = (0, 1, \ldots, \lfloor \tau n \rfloor - 1)$, $\tau \in [\frac{1}{n}, 1]$, $sel()$ is the selection operator, $rep()$ replaces a subset of $\boldsymbol{\mathcal{P}}$ with a subset of $\boldsymbol{\mathcal{O}}$, $ter()$ is the termination condition, $sea()$ is a model structure search algorithm, $est()$ estimates the model parameters and $sam()$ generates a single sample using the estimated pdfs. The evolutionary algorithm characteristic of the IDEA lies in the fact that a population of individuals is used from which individuals are selected to generate new offspring with. Using these offspring along with the parent individuals and the current population, a new population is constructed.

If we set m to $(n - \lfloor \tau n \rfloor)$, $sel()$ to selection by taking the best $\lfloor \tau n \rfloor$ vectors and $rep()$ to replacing the worst $(n - \lfloor \tau n \rfloor)$ vectors by the new samples, we have that $\theta_{t+1} = \theta_t - \varepsilon$ with $\varepsilon \geq 0$. This assures that the search for θ^* is conveyed through a monotonically decreasing series $\theta_0 \geq \theta_1 \geq \ldots \geq \theta_{t_{\text{end}}}$. We call an IDEA with m, $sel()$ and $rep()$ so chosen, a *monotonic* IDEA.

5.2 Previous IDEAs

All of the prior algorithms that use continuous probabilistic models, have used a single factorization for the probabilistic model structure ς. The first few approaches all used the univariate factorization [18, 30, 31]. Simultaneously, the IDEA framework was introduced [8] alongside the work of Larrañaga, Etxeberria, Lozano and Peña [22]. Both of the works propose to use the Kullback–Leibler divergence between the estimated distribution and the full joint estimated distribution which leads to minimizing the entropy over all possible factorizations. This approach is inspired by the MIMIC algorithm [7], in which only a certain constrained class of factorizations is allowed. In addition, in the work by Larrañaga *et al.*, edge exclusions tests and a continuous version of the Bayesian Dirichlet metric for Gaussian networks (BGe) are used to learn a factorization that is not constrained. On the other hand, in the work by Bosman and Thierens [8], a general factorization is allowed in which each variable is allowed to interact with a maximum of κ other variables. The work by Bosman and Thierens has been tested and has proven to be an effective approach [10, 11].

Apart from the recent work by Larrañaga *et al.* and by Bosman and Thierens, there have been no attempts to use higher order probabilistic model structures for continuous optimization. Most of the approaches so far have solely used the normal pdf as the building blocks of the probability distribution over $\boldsymbol{\mathcal{Y}}$. The only exception to this, has been the work by Gallagher, Fream and Downs [18], in which a normal mixture model is used. However, the model structure is fixed to the univariate factorization. Bosman and Thierens have described the use of the normal kernels pdf [12]. This approach has however proven to be hard to handle [10]. On the other hand, using the normal kernels pdf has given results on certain highly epistatic problems that could not be obtained by using only a single normal pdf. Therefore, a normal mixture pdf was suggested for future research. Finally, Bosman and Thierens also investigated the use of a histogram distribution [8]. Using this pdf however does not scale up [10] and has therefore been abandoned.

5.3 New IDEAs

The novelty of the algorithms that we propose in this paper, lies in the complexity of the probabilistic models. First of all, for the first time, we use a mixture of factorizations instead of a single factorization by finding a factorization for each cluster that we construct first. By effectively removing non–linear interactions between the problem variables in this way, previously shown to be effective techniques regarding the normal pdf and the learning of factorizations can be used to obtain effective probability distribution estimations.

The learning of factorizations is different as well. We no longer need additional constraints on the factorization as we use the metrics to penalize more complex models. Therefore, we no longer have a variety of factorization search algorithms with different constraints.

Finally, we have shown for the first time how the normal mixture pdf can be used given any factorization and thus in an IDEA instance in which any factorization is allowed. As the normal mixture pdf can account for additional non–linearities that may still remain within the clusters, the new algorithms are thus more flexible and effectively use more complex probabilistic models.

The required techniques to get the new optimization algorithms, have been described in previous sections in this paper. Using those techniques, we are now ready to apply our new IDEAs to continuous optimization problems, which is done in section 6.

5.4 Methodological comparison with Evolution Strategies

Approaches such as the IDEA that build and use probabilistic models in evolutionary optimization were first proposed as an improvement over GAs [4, 7]. The IDEA framework itself has been used to focus on continuous models for problems with real valued variables. This has resulted in a new line of continuous evolutionary algorithms. However, for continuous optimization, evolutionary algorithms have been proposed almost as long ago as the original simple binary GA [2]. Currently, we can identify two variants of this continuous evolutionary algorithm, namely Evolutionary Programming (EP) and Evolution Strategies (ES). Generally, we can state that these algorithms also make use of normal pdfs. One might therefore say that there already exist algorithms that operate in a similar fashion as the IDEA. For this reason, it is important to question the relevance of a new approach such as the IDEA instances presented so far.

In this section, we describe how ES differs from IDEA and point out that there are indeed fundamental differences that markedly distinguish the approaches from one another. To this end, we first give a brief algorithmical introduction to Evolution Strategies in section 5.4.1. Subsequently, we describe the actual differences with our IDEA approach in section 5.4.2.

5.4.1 Brief introduction to Evolution Strategies

In this section, we give a birdseye overview of Evolution Strategies. Our goal is not to go give a detailed description of these approaches. For a detailed introduction, the interested reader is for instance referred to the work by Bäck and Schwefel [3].

At a top level, we can distinguish two ES variants based on the selection scheme that is used. These variants are commonly referred to as the (μ, λ)–ES and the $(\mu + \lambda)$–ES. In both cases, μ denotes the amount of parents and λ the amount of offspring that is produced by recombination and mutation. In the case of the (μ, λ)–ES, the μ parents are selected from the λ offspring ($\lambda \geq \mu$). In the case of the $(\mu + \lambda)$–ES, the μ parents are selected from both the parents of the previous generation as well as the λ offspring ($\lambda \geq 1$).

In ES, the parents are recombined and subsequently mutated as is the case for the simple GAs. However, the mutation operator has always been the most important one for ES. This operator samples from normal pdfs and makes the ES appear similar to the IDEA with normal pdfs.

With each variable, a normal pdf is associated. Each normal pdf can either be allowed to have identical standard deviations in each dimension, to be aligned with the axes of the search space without necessarily identical standard deviations or can allowed to be any arbitrary l–dimensional normal pdf. This comes down to a covariance matrix with respectively either similar entries on the diagonal and zero entries off the diagonal, non–similar entries on the diagonal and zero entries off the diagonal or an arbitrary symmetric covariance matrix. The variables that code the matrix are incorporated into the genome and are subject to mutation and recombination themselves.

The mutation of the entries in the covariance matrix is performed by multiplication with an exponential normally distributed value. After the values in the covariances matrix have been updated, they are used to mutate the values for the problem variables. These values are updated by adding a normally distributed l–dimensional variable to the current sample.

An ES individual contains real values for the l dimensions of the problem. These values represent the mean of a normal pdf that is used to sample offspring from after recombination. The covariance matrix is also stored in the genome (in either restricted or unrestricted form). These real values are mutated according to a fixed strategy. For a covariance matrix with zero entries off the diagonal, the ES genome can be specified by:

$$(y_0, y_1, \ldots, y_{l-1}, \sigma_0, \sigma_1, \ldots, \sigma_{l-1}) \tag{52}$$

Mutation can then be specified by using a normally distributed general control parameter s that is sampled for each sample to be mutated anew (note that the other two normally distributed variables are sampled anew for each dimension):

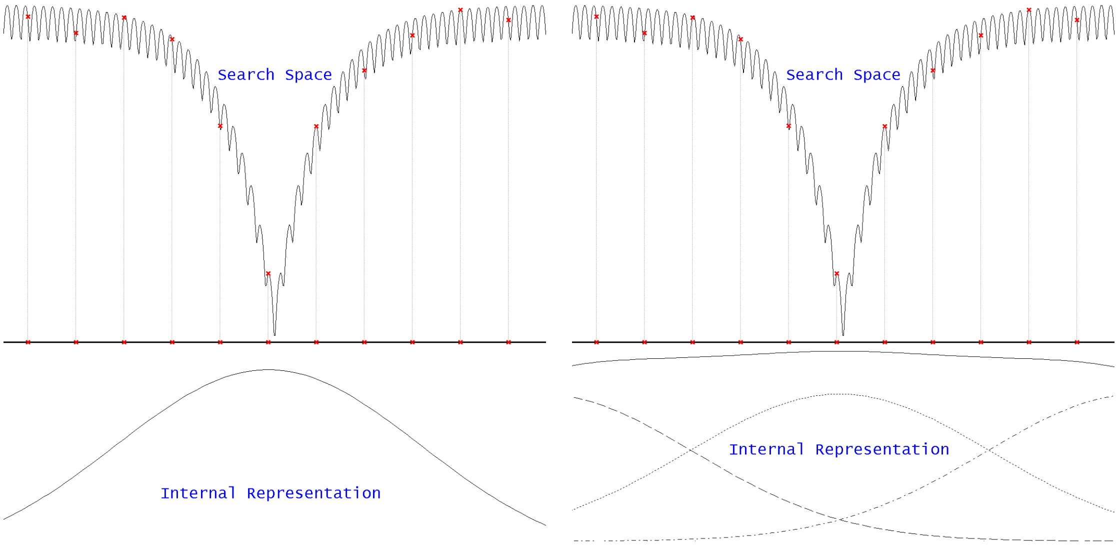

Figure 17: Operational sketch of IDEA for normal pdf (left) and normal mixture pdf (right).

$$\begin{aligned} s &= e^{\mathcal{N}(0,\tau)} \\ \sigma_i' &= \sigma_i e^{\mathcal{N}(0,\tau')} s \\ y_i' &= y_i + \mathcal{N}(0,\sigma_i') \end{aligned} \tag{53}$$

For τ and τ', the usual rule of thumb is to set $\tau = 1/\sqrt{2\sqrt{n}}$ and $\tau' = 1/\sqrt{2n}$. Recombination in ES is generally done in one of two possible ways, being *discrete* and *intermediate* crossover. By using discrete crossover, the values themselves are swapped without being altered. By using intermediate crossover, the values of the parents are averaged. Recombination is applied both to the problem values as well as the covariance matrix values.

5.4.2 ES and IDEA: Similarities and differences

One top level aspect that becomes clear is that a monotonic IDEA can be seen as a $(\mu + \lambda)$ algorithm. In evolutionary computation, such algorithms are said to be *elitist*. Since many algorithms in the field of EAs share such a selection strategy, this is not the most important thing to note about the two continuous evolutionary approaches.

In figure 17, a landscape is plotted in which we desire to locate the unique minimum. We assume that we have a set of samples available as indicated by crosses in the image. The IDEA approach is then to fit a probability distribution over these samples as well as possible to get a good representation of the samples. In figure 17 this is shown for a normal pdf and a normal mixture pdf. Once such a probability distribution has been estimated, λ offspring are generated by sampling from the probability distribution. As selection favors better solutions, the probability distribution is meant to model the promising regions of the search space. Note that this is a global approach to optimization as it attempts to globally capture the structure of the landscape and refines the model as the process is iterated.

We now turn our attention to the ES approach. In figure 18, a sketch is given of the mutation operation. If we set $\mu = 1$, we have only a single solution to recombine and mutate. By doing so, we are modeling a normal pdf just as is the case for the IDEA approach. However, its use is now different. If we only focus on sampling new solutions, we indeed get the same result as with the IDEA, since the λ offspring will be sampled from the normal pdf that is modeled. However, the way in which the normal pdf is found in the next generation is different. In the IDEA approach,

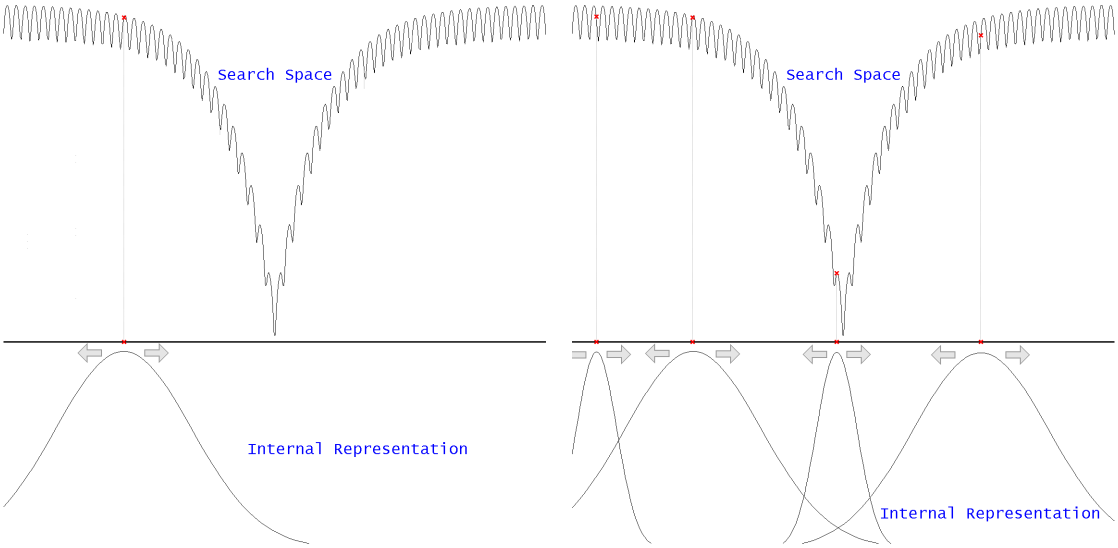

Figure 18: Operational sketch of mutation–based ES for $(1, \lambda)$ (left) and (μ, λ) (right).

the best $\lfloor \tau n \rfloor$ samples are selected. Based on these samples, a maximum likelihood normal pdf is fit. For the ES approach however, the single parent is selected from either the offspring or the offspring and the single parent of the previous generation. The result is a single point in the search space, since selection is based upon the function value only. This means that we can regard the ES approach with $\mu = 1$ as moving the normal pdf in some direction. The direction as well as the length of the step in which we move the normal pdf is subject to evolution itself. In other words, the ES in this case can be seen as evolving to find local gradient information on how to best traverse the search space.

In the case of $\mu > 1$ for ES, we get the representation as depicted on the righthandside of figure 18. Regarding only mutation for the moment again, generating the offspring involves sampling from the normal pdfs that lie centered around the μ parent samples. Therefore, this can be seen as the equivalent of the normal mixture pdf or a clustered normal pdf in the ID$\mathbb{E}$A approach. However, the way in which the μ parents make up the mixture distribution by evolution of the covariance matrices, is similar to the case in which $\mu = 1$. Again, we can therefore see the movement of the μ parent samples as moving the normal pdfs in a certain direction with a certain stepsize. Finding this direction and stepsize is subject to evolution. ES mutation can therefore also be taken as a more local based approach to finding and using the structure of the landscape.

Recombination in ES quite drastically changes the view on its operational semantics. If we for instance regard intermediate crossover, we are drawing lines between the μ parent samples in the l–dimensional space and place the new samples in the middle of these lines. It becomes clear that by doing so, the search is directed more to use the global structure of the search space to find promising regions of the search space. This strongly aids the ES to efficiently tackle a great amount of continuous optimization problems. Recombination allows for global exploration of the search space, while mutation attempts to use more local landscape information. Moreover, it should be noted that the initial span of the normal pdfs in ES is usually taken to be quite large. This improves the initial global exploration of the landscape further.

The ID$\mathbb{E}$A approach is a global procedure that attempts to use the structure of the problem landscape to explore the most promising regions. On the other hand, mutation based ES approaches are local procedures that use evolution to explore the inside of promising regions. By adding recombination, additional means of globally searching the landscape are introduced, but it is not as evident as for the ID$\mathbb{E}$A to what extent this helps to use global structure.

Concluding, the two approaches are fundamentally different. This becomes clear for instance on problems in which local information is important, such as Rosenbrock's function, which has a narrow valley. The bottom of this valley has a unique minimum. However, the gradient along the bottom of the valley is very small. This means that an $\mathbb{IDEA}$ based approach will quite easily find the valley itself, but will by no means be able to traverse the valley to find the global minimum unless points were sampled near the global minimum. The reason for this is that density estimation converges on a certain part of the valley since samples are only available in that part of the search space. On the other hand, once the ES is inside the valley, it can adapt its mutation direction and stepsize to follow the valley to its minimum in a gradient descent fashion. Even though this is a time consuming process, the ES is not as likely to prematurely converge on such a problem as is the $\mathbb{IDEA}$ approach.

6 Experiments

The continuous function optimization problems we used for testing are the following:

Name	**Definition**	**Domain**
Griewank	$\frac{1}{4000}\sum_{i=0}^{l-1}(y_i-100)^2-\prod_{i=0}^{l-1}\cos\left(\frac{y_i-100}{\sqrt{i+1}}\right)+1$	$[-5,5]^l$
Michalewicz	$-\sum_{i=0}^{l-1}\sin(y_i)\sin^{20}\left(\frac{(i+1)y_i^2}{\pi}\right)$	$[0,\pi]^l$
Rosenbrock	$\sum_{i=0}^{l-2}100(y_{i+1}-y_i^2)^2+(1-y_i)^2$	$[-5.12,5.12]^l$

All of the above test functions should be *minimized.* We tested a large variety of $\mathbb{IDEA}$ variants for $l=5$. In sections 6.1, 6.2 and 6.3 we go over the results for the individual functions. In all our testing, we used a *monotonic* $\mathbb{IDEA}$. We used the rule of thumb by Mühlenbein and Mahnig [24] for FDA and set τ to 0.3. Furthermore, for variants that use clustering, we let n increase from 250 to 5000 in steps of 250. For variants that do not use clustering, we increase the population size in steps of 25. The reason for this is that by introducing clusters, we also need to increase the population size by a larger amount to allow a significant change in cluster size. We allowed each run a maximum of $1\cdot10^7$ evaluations for all non–clustered approaches. Because of time restrictions, we only allowed $2\frac{1}{2}\cdot10^6$ evaluations for the clustered approaches, with the exception of the Euclidean leader algorithms on Griewank's function. If all of the solutions differed by less than $5\cdot10^{-7}$, termination was enforced also. Note that this implies a maximum precision of 6 decimal digits. All results were averaged over 10 runs.

We tested both the normal pdf as well as the normal mixture pdf. For the normal pdf, we searched for both unconditional as well as conditional factorizations with both the AIC metric as well as the BIC metric. For the normal mixture pdf, it was empirically observed that inference with respect to finding a factorization is extremely computationally expensive. Therefore, we have fixed the factorization structure to the univariate one for the normal mixture pdf. However, we did test various amounts of mixtures in the normal mixture pdf, namely $w\in\{2,5,10\}$. We applied clustering using both the leader algorithm as well as the k–means clustering algorithm. In the case of the leader algorithm, we applied both the scaled Euclidean distance as well as the Mahalanobis distance. For the k–means clustering algorithm, we only used the scaled Euclidean distance. We used $k\in\{2,5,10\}$ for the k–means clustering algorithm and $\mathfrak{T}_d\in\{3\frac{1}{2},5\}$ for the variants of the leader algorithm. This gives on average 7 and 2 clusters respectively for the scaled Euclidean distance and 11 and 3 clusters respectively for the Mahalanobis distance. The factorization mixture coefficients β_i are determined by using equation 35.

We present the best obtained average best results over the 10 independent runs and sort all of the $\mathbb{IDEA}$ instances by this index primarily. We also show the population size n for the best obtained result, the average amount of evaluations and the *Relative Run Time* $\mathbb{RT}$. Let $\mathrm{FT}(x)$ be the time to perform x random function evaluations and TRT be the *Total Run Time* on the same processing system. Then, $\mathbb{RT}(x)=\mathrm{TRT}/\mathrm{FT}(x)$. We determined $\mathbb{RT}$ as $\mathbb{RT}(10^6)$. The $\mathbb{RT}$ index is a processing system independent fair comparison metric of the total required time. We sort the results by this index secondarily instead of the amount of function evaluations, as it *truely* reflects the required amount of time.

6.1 Griewank's function

In figure 19, a summary of the results on Griewank's function is given. The unique minimum is 0, which is found by quite a few of the tested approaches within the allowed precision.

Quite a lot of approaches without clustering are able to find the unique minimum. This gives us the impression that Griewank's function is not extremely epistatic or non–linear. On the other hand, the amount of required function evaluations is very large when compared to clustering approaches or normal mixture approaches. The non–clustered normal mixture pdf with $w = 10$ for instance, requires only 51832 evaluations on average to minimize the function, whereas the approaches that only use a single normal pdf require at least 945971 evaluations on average. However, the $\mathbb{RT}$ index points out an important fact. It can be seen from the table that the running time that is required for the normal mixture pdf, is quite a lot larger. Even though the single normal pdf requires 20 times more evaluations, it runs 2 to 3 times as fast as the normal mixture pdf in which only the univariate factorization is used. We should note at this point that if the evaluation time for a single solution would have been significantly larger, then the normal mixture pdf would have been preferable.

It we take a look at the different clustering approaches, we first note that the successful entries in the table use the normal mixture pdf. It seems that we should mostly contribute the success of these approaches to the use of the normal mixture pdf instead of the use of clustering. If we take a look at clustering approaches in combination with only a normal pdf, it seems that the Mahalanobis distance results in no superior useful clustering algorithms within the ID$\mathbb{E}$A framework. What is very interesting however, is that the k–means clustering approaches seem to perform significantly worse than the leader approach, unless clustering is combined with the normal mixture pdf. The main reason for this is the difference in allowed amount of function evaluations. Otherwise, there does not seem to be a large difference between the two approaches. The only reason why they differ in their outcome, is that the amount of clusters differs. If we then regard the Euclidean leader algorithm with the BIC metric, $\mathfrak{T}_d = 3\frac{1}{2}$ and unconditional dependencies as well as the k–means algorithm with $k = 5$, we see that the running time of the leader algorithm is only slightly larger than that of the k–means algorithm, even though the population size is equal and the amount of evaluations is 3 times as large. This makes the leader algorithm preferable here.

It is hard to say whether conditional or unconditional dependencies are to be preferred on the basis of the presented results. The same holds for the search metric. The reason for this is that the amount of dimensions l is very small. The choice of the metric and the type of dependencies becomes truely significant if l goes up. Still, it seems that for expressional power, the conditional dependencies should be preferred.

6.2 Michalewicz's function

Observing the results on the highly epistatic Michalewicz function in figure 20, it becomes clear immediately that clustering is required. The non–clustering approaches only work well if the normal mixture pdf is used, which is in a way comparable to using clustering approaches. There is no significant difference between conditional and unconditional dependencies, but in the case of conditional dependencies, the required amount of function evaluations is often lower.

There seems to be a slight preference for the Mahalanobis distance, but the Euclidean distance seems to work quite well. A more significant difference is present between the leader clustering approach and the k–means clustering approach. The k–means approaches all seem to be inferior to the leader approaches. The only explanation for this can be the adaptiveness of the leader algorithm because of the threshold $\mathfrak{T}_d$. This adaptiveness allows the leader algorithm to be flexible in the amount of required clusters, which in turn results in a better performance.

We note that the normal mixture pdf requires on average quite a lot less evaluations than does any other approach using only the normal pdf. However, the computation time that is required

Clustering	Distance	$\mathfrak{f}$	Metric	pdf	n	$\overline{C}$	$\overline{\text{evals}}$	$\mathbb{R}$T
—	—	$(\pi, \boldsymbol{\omega})$	AIC	normal	175	0.000000	945971	2.21
—	—	$\boldsymbol{\nu}$	AIC	normal	175	0.000000	1039070	2.55
—	—	$(\pi, \boldsymbol{\omega})$	BIC	normal	150	0.000000	1403463	3.19
—	—	$\boldsymbol{\nu}$	BIC	normal	200	0.000000	1492826	3.48
—	—	univ.	—	mix 10	1100	0.000000	51832	6.61
k–means 2	Euclidean	univ.	—	mix 10	1750	0.000000	72552	8.86
Leader $3\frac{1}{2}$	Mahalanobis	univ.	—	mix 5	3250	0.000000	156311	9.97
Leader $3\frac{1}{2}$	Mahalanobis	univ.	—	mix 10	2250	0.000000	89718	10.30
k–means 5	Euclidean	univ.	—	mix 10	2500	0.000000	98805	11.43
Leader 5	Euclidean	$(\pi, \boldsymbol{\omega})$	AIC	normal	250	0.000000	4522165	11.65
k–means 5	Euclidean	univ.	—	mix 5	3250	0.000000	172812	11.81
—	—	univ.	—	mix 5	1600	0.000000	163024	13.01
Leader 5	Euclidean	univ.	—	mix 10	2250	0.000000	106660	13.65
Leader 5	Mahalanobis	univ.	—	mix 10	2250	0.000000	110206	14.40
Leader 5	Euclidean	univ.	—	mix 5	2500	0.000000	226102	16.42
k–means 2	Euclidean	univ.	—	mix 5	3000	0.000000	228857	16.75
k–means 10	Euclidean	univ.	—	mix 10	3750	0.000000	156386	17.58
Leader $3\frac{1}{2}$	Euclidean	univ.	—	mix 10	4750	0.000000	199321	23.74
Leader 5	Mahalanobis	univ.	—	mix 5	3250	0.000000	353185	26.94
Leader $3\frac{1}{2}$	Euclidean	univ.	—	mix 5	4750	0.000047	360632	22.05
k–means 10	Euclidean	univ.	—	mix 5	5000	0.000072	274139	17.00
Leader 5	Euclidean	$\boldsymbol{\nu}$	AIC	normal	500	0.000636	7477291	18.96
Leader 5	Euclidean	$\boldsymbol{\nu}$	BIC	normal	250	0.000905	3706986	10.29
Leader 5	Euclidean	univ.	—	mix 2	4000	0.001232	3239155	106.93
Leader 5	Euclidean	$(\pi, \boldsymbol{\omega})$	BIC	normal	250	0.001278	6631367	16.59
Leader $3\frac{1}{2}$	Mahalanobis	univ.	—	mix 2	4750	0.004618	698637	23.12
—	—	univ.	—	mix 2	2750	0.005451	4818328	123.82
Leader $3\frac{1}{2}$	Euclidean	$\boldsymbol{\nu}$	BIC	normal	1000	0.005755	6620613	18.40
Leader 5	Mahalanobis	$\boldsymbol{\nu}$	BIC	normal	250	0.005956	1628976	7.51
k–means 5	Euclidean	univ.	—	mix 2	3000	0.006179	653522	24.00
Leader $3\frac{1}{2}$	Euclidean	$(\pi, \boldsymbol{\omega})$	BIC	normal	750	0.006705	7268071	19.47
Leader 5	Mahalanobis	univ.	—	mix 2	2000	0.007339	1826277	54.79
Leader $3\frac{1}{2}$	Euclidean	$\boldsymbol{\nu}$	AIC	normal	1000	0.007477	2775053	18.38
Leader $3\frac{1}{2}$	Euclidean	$(\pi, \boldsymbol{\omega})$	AIC	normal	1250	0.007640	8992514	25.56
k–means 2	Euclidean	univ.	—	mix 2	1750	0.007930	979638	25.97
Leader 5	Mahalanobis	$\boldsymbol{\nu}$	AIC	normal	250	0.009353	1338268	5.81
Leader 5	Mahalanobis	$(\pi, \boldsymbol{\omega})$	BIC	normal	250	0.009428	2499626	10.28
Leader $3\frac{1}{2}$	Mahalanobis	$(\pi, \boldsymbol{\omega})$	AIC	normal	750	0.010497	2127631	10.52
Leader $3\frac{1}{2}$	Euclidean	univ.	—	mix 2	5000	0.011837	751763	21.90
Leader $3\frac{1}{2}$	Mahalanobis	$\boldsymbol{\nu}$	BIC	normal	750	0.011987	2007309	9.70
k–means 2	Euclidean	$\boldsymbol{\nu}$	AIC	normal	250	0.012384	736524	2.48
k–means 2	Euclidean	$(\pi, \boldsymbol{\omega})$	BIC	normal	500	0.012742	2381377	9.06
Leader 5	Mahalanobis	$(\pi, \boldsymbol{\omega})$	AIC	normal	250	0.012981	1793646	7.61
Leader $3\frac{1}{2}$	Mahalanobis	$(\pi, \boldsymbol{\omega})$	BIC	normal	500	0.013007	877386	4.11
k–means 2	Euclidean	$(\pi, \boldsymbol{\omega})$	AIC	normal	250	0.013202	1172454	3.89
k–means 10	Euclidean	univ.	—	mix 2	4750	0.013400	958065	31.46
k–means 2	Euclidean	$\boldsymbol{\nu}$	BIC	normal	250	0.013514	793724	2.79
k–means 10	Euclidean	$(\pi, \boldsymbol{\omega})$	AIC	normal	2500	0.013601	1718042	19.35
Leader $3\frac{1}{2}$	Mahalanobis	$\boldsymbol{\nu}$	AIC	normal	750	0.014672	2296346	11.27
k–means 10	Euclidean	$\boldsymbol{\nu}$	BIC	normal	2250	0.015987	703016	17.55
k–means 10	Euclidean	$\boldsymbol{\nu}$	AIC	normal	5000	0.015921	1662724	18.00
k–means 10	Euclidean	$(\pi, \boldsymbol{\omega})$	BIC	normal	2500	0.016360	2054672	23.03
k–means 5	Euclidean	$\boldsymbol{\nu}$	AIC	normal	1000	0.016410	1094122	6.40
k–means 5	Euclidean	$\boldsymbol{\nu}$	BIC	normal	1000	0.019069	2361092	17.05
k–means 5	Euclidean	$(\pi, \boldsymbol{\omega})$	BIC	normal	1000	0.020145	2236489	14.47
k–means 5	Euclidean	$(\pi, \boldsymbol{\omega})$	AIC	normal	250	0.101171	1725182	6.63

Figure 19: Results on Griewank's Function, sorted on $\overline{C}$ (primary) and $\mathbb{R}$T (secondary).

to estimate the parameters of the normal mixture, significantly increases the running time.

The main thing to note is that whereas for Griewank's function we were still able to get good results even with only the normal pdf, clustering becomes much more important here.

6.3 Rosenbrock's function

Rosenbrock's function has proven to be a real challenge for any continuous optimization algorithm. It has a valley along which the quality of solutions is much better than that of the solutions in its neighborhood. Furthermore, this valley has a unique minimum of 0 itself. The gradient along the bottom of the valley is only very slight. Any gradient approach is therefore doomed to follow the long road along the bottom of the valley. For a density estimation algorithm, capturing the valley in a probabilistic model is difficult, even if all of the points within the valley are known. The reason for this is that the valley is non–linear in the coding space. Therefore, we expect that to get any reasonable results, we require to use clustering.

In figure 21, the results on Rosenbrock's function for $l = 5$ are given. It is clear from this table that any approach that does not use clustering, does not give useful results. Using 10 clusters allows us to effectively process the non–linearity of the valley and optimize the problem using only very few evaluations. We note that using a mixture of normal pdfs in this case does not help the optimization process at all. It even results in more evaluations than when using a single normal pdf in some cases. The main important thing is the non–linearity of the valley and its slight gradient. Because of the fact that only the univariate factorization is used for the normal mixture pdf, the non–linearity cannot be captured effectively. The problem with density estimation in this continuous search space is that it doesn't use the gradient information that can be extremely useful to find the minimum. Given the continuity of the search spaces, it is even *unwise* not to exploit this information in some way, which is demonstrated by the results in figure 21. Quite a large amount of clusters is namely required to effectively optimize the problem.

Note that the amount of dimensions is only very small. If the amount of dimensions goes up, the amount of clusters will have to increase accordingly. However, by doing so, the amount of samples will have to increase as well to ensure a large enough cluster size. Therefore, the sole use of density estimation on search spaces such as the one defined by Rosenbrock's function, is not effective enough to compete with gradient approaches. However, a hybrid combination of both approaches will most likely result in very effective continuous optimization techniques.

We close this section by noting that the k–means clustering algorithm now outperforms the leader algorithm. The main reason for this is that clustering is extremely important for this function. With $k = 10$, we have the largest amount of clusters that we tested. If we would have used a threshold lower than $\mathfrak{T}_d = 3\frac{1}{2}$, we would most likely have gotten better results with the leader algorithm as well. Finally, it is clear that the Euclidean distance measure significantly outperforms the Mahalanobis distance measure.

7 Discussion

The use of the normal mixture pdf in combination with the EM algorithm as proposed in this paper, requires a vast amount of computation time. For highly epistatic search spaces however, it has shown to be effective with respect to the required amount of evaluations as well as optimization performance. Since the EM algorithm gives a general approach to finding the involved parameters, the normal mixture pdf is preferable over the normal kernels pdf that was proposed earlier [12].

The use of clustering in combination with the normal pdf also results in a normal mixture pdf. However, the results in this paper show different behavior of the two techniques on different landscapes. On non–linear landscapes, clustering is the most effective way to estimate the non–linearity, unless higher order factorizations are allowed for the normal mixture pdf. However, allowing such factorizations results in vast additional amounts of computation time, which is undesirable.

Clustering	**Distance**	$\mathfrak{f}$	**Metric**	**pdf**	**n**	$\overline{C}$	$\overline{\text{evals}}$	$\mathbb{R}$T
Leader 5	Mahalanobis	$(\pi, \boldsymbol{\omega})$	AIC	normal	750	-4.687658	95364	0.32
Leader $3\frac{1}{2}$	Euclidean	univ.	—	mix 2	1500	-4.687658	38416	0.37
Leader 5	Mahalanobis	$\boldsymbol{\nu}$	BIC	normal	1000	-4.687658	117979	0.38
Leader 5	Mahalanobis	$(\pi, \boldsymbol{\omega})$	BIC	normal	1000	-4.687658	122711	0.41
Leader $3\frac{1}{2}$	Euclidean	$(\pi, \boldsymbol{\omega})$	BIC	normal	4000	-4.687658	180813	0.42
Leader 5	Mahalanobis	$\boldsymbol{\nu}$	AIC	normal	1000	-4.687658	131561	0.43
Leader 5	Mahalanobis	univ.	—	mix 5	1000	-4.687658	22118	0.45
Leader $3\frac{1}{2}$	Euclidean	$\boldsymbol{\nu}$	BIC	normal	3500	-4.687658	193759	0.45
Leader $3\frac{1}{2}$	Euclidean	$\boldsymbol{\nu}$	AIC	normal	3750	-4.687658	194792	0.46
Leader $3\frac{1}{2}$	Euclidean	$(\pi, \boldsymbol{\omega})$	AIC	normal	4000	-4.687658	201120	0.47
Leader 5	Euclidean	$\boldsymbol{\nu}$	BIC	normal	2250	-4.687658	221314	0.47
k–means 2	Euclidean	univ.	—	mix 2	2000	-4.687658	49634	0.52
Leader $3\frac{1}{2}$	Mahalanobis	$(\pi, \boldsymbol{\omega})$	BIC	normal	3500	-4.687658	129114	0.54
Leader 5	Mahalanobis	univ.	—	mix 2	2000	-4.687658	51035	0.55
Leader 5	Euclidean	univ.	—	mix 2	2250	-4.687658	58789	0.58
k–means 5	Euclidean	univ.	—	mix 5	1250	-4.687658	28954	0.59
Leader $3\frac{1}{2}$	Mahalanobis	$\boldsymbol{\nu}$	BIC	normal	3750	-4.687658	139646	0.59
—	—	univ.	—	mix 5	1300	-4.687658	28903	0.61
Leader 5	Euclidean	$(\pi, \boldsymbol{\omega})$	BIC	normal	1500	-4.687658	301297	0.63
k–means 5	Euclidean	univ.	—	mix 2	2500	-4.687658	63128	0.67
Leader $3\frac{1}{2}$	Mahalanobis	$\boldsymbol{\nu}$	AIC	normal	4250	-4.687658	155654	0.68
k–means 2	Euclidean	univ.	—	mix 10	750	-4.687658	16596	0.70
k–means 10	Euclidean	univ.	—	mix 5	1750	-4.687658	43894	0.88
Leader $3\frac{1}{2}$	Mahalanobis	univ.	—	mix 2	3500	-4.687658	86528	0.92
Leader 5	Euclidean	$\boldsymbol{\nu}$	AIC	normal	3250	-4.687658	423172	0.92
Leader 5	Euclidean	univ.	—	mix 10	1000	-4.687658	21680	0.96
k–means 2	Euclidean	univ.	—	mix 5	2000	-4.687658	45606	0.96
Leader $3\frac{1}{2}$	Mahalanobis	univ.	—	mix 5	2250	-4.687658	52288	1.04
Leader 5	Euclidean	$(\pi, \boldsymbol{\omega})$	AIC	normal	2000	-4.687658	512489	1.08
k–means 10	Euclidean	univ.	—	mix 10	1250	-4.687658	30925	1.10
Leader $3\frac{1}{2}$	Mahalanobis	univ.	—	mix 10	1500	-4.687658	34475	1.25
Leader $3\frac{1}{2}$	Euclidean	univ.	—	mix 5	3000	-4.687658	70232	1.33
k–means 10	Euclidean	univ.	—	mix 2	4250	-4.687658	115106	1.34
Leader 5	Euclidean	univ.	—	mix 5	3000	-4.687658	68131	1.38
—	—	univ.	—	mix 2	5000	-4.687658	135938	1.45
k–means 5	Euclidean	univ.	—	mix 10	1750	-4.687658	40522	1.51
—	—	univ.	—	mix 10	1800	-4.687658	39756	1.61
Leader $3\frac{1}{2}$	Euclidean	univ.	—	mix 10	2000	-4.687658	45431	1.68
Leader 5	Mahalanobis	univ.	—	mix 10	2500	-4.687658	58751	2.18
Leader $3\frac{1}{2}$	Mahalanobis	$(\pi, \boldsymbol{\omega})$	AIC	normal	4250	-4.682400	258326	1.14
k–means 2	Euclidean	$(\pi, \boldsymbol{\omega})$	AIC	normal	2500	-4.661225	114406	0.48
k–means 5	Euclidean	$(\pi, \boldsymbol{\omega})$	BIC	normal	5000	-4.660579	825985	2.63
k–means 2	Euclidean	$\boldsymbol{\nu}$	BIC	normal	4750	-4.659790	543562	1.44
—	—	$\boldsymbol{\nu}$	BIC	normal	325	-4.659300	1032504	2.13
—	—	$(\pi, \boldsymbol{\omega})$	AIC	normal	1100	-4.657363	4064810	8.20
k–means 2	Euclidean	$(\pi, \boldsymbol{\omega})$	BIC	normal	3500	-4.656492	615331	1.54
k–means 2	Euclidean	$\boldsymbol{\nu}$	AIC	normal	3750	-4.656120	646464	1.66
—	—	$(\pi, \boldsymbol{\omega})$	BIC	normal	1700	-4.654513	4088736	8.26
—	—	$\boldsymbol{\nu}$	AIC	normal	175	-4.654248	13693	0.03
k–means 10	Euclidean	$\boldsymbol{\nu}$	BIC	normal	2000	-4.650213	377993	1.48
k–means 5	Euclidean	$(\pi, \boldsymbol{\omega})$	AIC	normal	4000	-4.643722	773225	2.35
k–means 5	Euclidean	$\boldsymbol{\nu}$	AIC	normal	2750	-4.639595	443323	1.38
k–means 10	Euclidean	$\boldsymbol{\nu}$	AIC	normal	3500	-4.638577	856142	3.31
k–means 10	Euclidean	$(\pi, \boldsymbol{\omega})$	BIC	normal	4250	-4.637136	551834	2.35
k–means 5	Euclidean	$\boldsymbol{\nu}$	BIC	normal	3750	-4.635958	499408	1.59
k–means 10	Euclidean	$(\pi, \boldsymbol{\omega})$	AIC	normal	2250	-4.622931	1262262	4.52

Figure 20: Results on Michalewicz's Function, sorted on $\overline{C}$ (primary) and $\mathbb{R}$T (secondary).

Clustering	Distance	$\mathfrak{f}$	Metric	pdf	n	$\overline{C}$	$\overline{\text{evals}}$	$\mathbb{R}$T
k–means 10	Euclidean	$(\pi, \boldsymbol{\omega})$	AIC	normal	2500	0.000000	67506	3.03
k–means 10	Euclidean	$(\pi, \boldsymbol{\omega})$	BIC	normal	3000	0.000000	82575	3.65
k–means 10	Euclidean	$\boldsymbol{\nu}$	AIC	normal	4250	0.000000	157886	6.30
k–means 10	Euclidean	$\boldsymbol{\nu}$	BIC	normal	2750	0.001550	168627	5.35
Leader $3\frac{1}{2}$	Euclidean	$(\pi, \boldsymbol{\omega})$	AIC	normal	4750	0.002011	166477	1.81
Leader $3\frac{1}{2}$	Euclidean	$\boldsymbol{\nu}$	AIC	normal	5000	0.033155	773470	8.84
Leader $3\frac{1}{2}$	Euclidean	$(\pi, \boldsymbol{\omega})$	BIC	normal	5000	0.050435	646558	7.36
k–means 10	Euclidean	univ.	—	mix 5	3500	0.062745	309262	31.00
Leader 5	Euclidean	univ.	—	mix 10	4000	0.065469	341870	72.56
Leader $3\frac{1}{2}$	Euclidean	$\boldsymbol{\nu}$	BIC	normal	5000	0.074347	742398	8.52
k–means 5	Euclidean	$(\pi, \boldsymbol{\omega})$	BIC	normal	1250	0.078283	53482	1.18
k–means 5	Euclidean	$\boldsymbol{\nu}$	AIC	normal	4750	0.079437	904017	15.03
k–means 5	Euclidean	$(\pi, \boldsymbol{\omega})$	AIC	normal	4750	0.083104	615903	10.53
Leader 5	Euclidean	univ.	—	mix 5	2250	0.085702	191173	19.92
Leader 5	Euclidean	univ.	—	mix 2	3500	0.092902	278625	13.86
k–means 10	Euclidean	univ.	—	mix 10	4250	0.106256	610238	89.13
k–means 10	Euclidean	univ.	—	mix 2	2000	0.125456	272568	13.00
Leader 5	Euclidean	$\boldsymbol{\nu}$	AIC	normal	3750	0.128021	478400	4.81
Leader $3\frac{1}{2}$	Euclidean	univ.	—	mix 2	3500	0.143837	621458	20.36
k–means 5	Euclidean	$\boldsymbol{\nu}$	BIC	normal	4750	0.160205	1246595	19.71
k–means 5	Euclidean	univ.	—	mix 5	4750	0.160414	531505	45.24
Leader $3\frac{1}{2}$	Euclidean	univ.	—	mix 10	4500	0.172262	472424	79.29
Leader $3\frac{1}{2}$	Euclidean	univ.	—	mix 5	3000	0.189304	286372	25.40
k–means 2	Euclidean	$\boldsymbol{\nu}$	BIC	normal	5000	0.189420	1621587	19.09
Leader 5	Mahalanobis	univ.	—	mix 5	2250	0.191094	261108	27.56
Leader 5	Euclidean	$(\pi, \boldsymbol{\omega})$	AIC	normal	4500	0.209030	1361006	13.73
Leader $3\frac{1}{2}$	Mahalanobis	$\boldsymbol{\nu}$	AIC	normal	2000	0.209074	808100	20.10
Leader $3\frac{1}{2}$	Mahalanobis	$(\pi, \boldsymbol{\omega})$	AIC	normal	2500	0.247123	1765538	43.24
k–means 5	Euclidean	univ.	—	mix 10	5000	0.259675	551594	86.79
Leader 5	Mahalanobis	univ.	—	mix 2	4250	0.260017	457718	24.28
Leader 5	Mahalanobis	univ.	—	mix 10	4500	0.261190	735926	145.22
Leader 5	Euclidean	$\boldsymbol{\nu}$	BIC	normal	2250	0.285851	1338881	13.80
—	—	univ.	—	mix 5	450	0.288066	37074	3.52
Leader 5	Euclidean	$(\pi, \boldsymbol{\omega})$	BIC	normal	5000	0.304032	1427719	14.91
Leader $3\frac{1}{2}$	Mahalanobis	$\boldsymbol{\nu}$	BIC	normal	2500	0.325525	918273	23.56
—	—	univ.	—	mix 10	1400	0.345850	133443	25.19
Leader 5	Mahalanobis	$(\pi, \boldsymbol{\omega})$	BIC	normal	3000	0.363407	2252646	49.28
k–means 5	Euclidean	univ.	—	mix 2	4250	0.366888	600938	24.30
Leader $3\frac{1}{2}$	Mahalanobis	univ.	—	mix 10	3500	0.367077	555588	100.59
Leader $3\frac{1}{2}$	Mahalanobis	univ.	—	mix 2	2750	0.396008	378320	20.99
k–means 2	Euclidean	univ.	—	mix 2	5000	0.421393	1166456	45.95
Leader $3\frac{1}{2}$	Mahalanobis	$(\pi, \boldsymbol{\omega})$	BIC	normal	1750	0.445489	1915536	46.29
Leader $3\frac{1}{2}$	Mahalanobis	univ.	—	mix 5	2500	0.459225	406324	40.36
k–means 2	Euclidean	univ.	—	mix 10	3250	0.481794	291164	52.68
k–means 2	Euclidean	univ.	—	mix 5	2250	0.495617	180535	17.21
Leader 5	Mahalanobis	$\boldsymbol{\nu}$	AIC	normal	3250	0.605696	1506264	32.43
Leader 5	Mahalanobis	$(\pi, \boldsymbol{\omega})$	AIC	normal	5000	0.615485	2316097	53.52
Leader 5	Mahalanobis	$\boldsymbol{\nu}$	BIC	normal	4250	0.774399	1978082	43.80
k–means 2	Euclidean	$(\pi, \boldsymbol{\omega})$	AIC	normal	4750	0.923545	1057429	12.74
—	—	univ.	—	mix 2	975	1.083728	92292	5.01
k–means 2	Euclidean	$(\pi, \boldsymbol{\omega})$	BIC	normal	1250	1.281874	79419	1.53
k–means 2	Euclidean	$\boldsymbol{\nu}$	AIC	normal	3250	1.298608	919056	10.63
—	—	$(\pi, \boldsymbol{\omega})$	BIC	normal	2750	1.859228	2370574	19.30
—	—	$(\pi, \boldsymbol{\omega})$	AIC	normal	675	1.867434	79619	0.72
—	—	$\boldsymbol{\nu}$	BIC	normal	4000	1.906433	1592727	13.52
—	—	$\boldsymbol{\nu}$	AIC	normal	900	1.971504	135492	1.10

Figure 21: Results on Rosenbrock's Function, sorted on $\overline{C}$ (primary) and $\mathbb{R}$T (secondary).

Given the low computational requirements on the factorized normal pdf, introducing clustering is the most cost effective way to improve the performance of IDEAs on a variety of non–linear and epistatic continuous optimization functions. To further improve the effectiveness of the algorithms, local gradient information must be taken into account. By only estimating probability distributions and by generating more samples from them, gradient information is ignored. On true differential continuous search spaces however, this information is a very important guide to find local minima or maximima. Implicitly, approaches such as Evolution Strategies make use of such gradient information, be it completely local or on a less detailed scale of information. Furthermore, many classical gradient search techniques exist that have proven to be very effective. Therefore, to improve the optimization performance of IDEA approaches for differential continuous search spaces, a local search method that uses gradient information should be incorporated. Combining the global approach of the IDEA and the local gradient search will likely be very effective.

For the experiments in this paper, we have set the β_i mixing coefficients of the factorization mixture to the proportional cluster size. To increase the optimization pressure, the coefficients should be set to the proportional average cluster fitness. Preliminary results have shown that this is effective for many different cost functions.

In this paper, we have introduced general mixtures of factorizations. To learn them, we have applied clustering and subsequent factorization learning in each of the clusters separately. The amount of clusters has so far been determined either directly by means of the k–means clustering algorithm or indirectly through a distance threshold in the leader algorithms. Even though the latter is a form of adaptive clustering, it is not really adaptive in the sense that new clusters are introduced as they are required to estimate the probability distribution better. If we would have such a mechanism that is effective with respect to computational requirements, clustering in density estimation for evolutionary optimization in IDEAs, will become even more valuable.

From the results in this paper, we find that the Mahalanobis distance for clustering does not introduce a significant optimization performance increase of the use of the Euclidean distance measure. Since the Euclidean distance measure can be evaluated faster, this measure should be preferred. Furthermore, the use of the simple and fast leader clustering algorithm gives results that are comparable to those of the k–means clustering algorithm. Therefore, the leader algorithm is preferable. Since we have only tested low dimensional problems, the use of unconditional factorizations versus conditional factorizations can hardly be commented upon. However, it seems that using conditional factorizations often leads to superior results.

The use of either the AIC or the BIC metric to search for a factorization should be tested on problems of a higher dimensionality. Moreover, finding a good factorization becomes more important for problems in which the problem structure is made up of true building blocks. On such problems, the *linkage* between the problem variables has to be exploited in the best way possible to efficiently process the generated solutions and traverse the search space [9]. Such problems occur both in differential continuous optimization as well as in spaces such as the one that is defined by using *random keys*. Random keys are continuous values that are used to encode permutations. Therefore, we are actually dealing with a discrete space in which problem structure plays a subtle different role than is the case in conventional continuous differential problems.

8 Conclusions

We have expanded the use of continuous probabilistic models for any factorization from a single normal pdf to mixtures of normal pdfs. To learn normal mixture pdfs, we have used the EM algorithm. Next to the normal mixture pdf, we have shown how efficient clustering algorithms can be used to break the non–linearity of the search space. By treating each cluster as a separate sample vector, we can again achieve an ormal mixture pdf. The two approaches to reaching the mixture however, is fundamentally different. Furthermore, the EM algorithm has been found to be computationally more expensive. For non–linear search spaces, clustering is a cost effective way to break this non–linearity in order for the promising regions of the search space to be explored efficiently. The resulting probabilistic model structure is then a mixture of factorizations.

We have clarified the use of a set of intuitive search metrics and have shown how these fit in with the basic notion of likelihood, which is a measure of how well a probability distribution fits a vector of samples. Starting from there, we have shown how well known measures such as the AIC and BIC metric can be derived. It becomes clear that these metrics are quite similar and only differ by means of the amount of regularization that is applied to ensure a good generalization performance of the estimated probability distribution. In subsequent research, the use of the AIC and BIC metric in higher dimensional search spaces and non–continuous search spaces is to be investigated to get a notion of their performance quality.

Even though nice optimization results have been achieved for varying epistatic and non–linear differential continuous search spaces, the use of gradient information to help guide the search towards promising regions of the search space will most likely result in even more effective continuous optimization algorithms. By only using the estimation of and sampling from probability distributions, as is done at this point, the local gradient information is ignored. In true differential continuous search spaces however, it is a waste to ignore such important information.

Summarizing, in this paper we have extended the set of available tools for IDEAs to perform continuous optimization with. This extension consists of new and more complex tools that we have shown to allow for a better performance of IDEAs.

References

[1] H. Akaike. Information theory and an extension of the maximum likelihood principle. In P.N. Petrov and F. Csaki, editors, *Proceedings of the 2nd International Symposium on Information Theory*, pages 267–281. Akademia Kiado, 1973.

[2] T. Bäck and H.-P. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Journal of Evolutionary Computation*, 1(1):1–23, 1993.

[3] T. Bäck and H-P. Schwefel. Evolution strategies I: Variants and their computational implementation. In G. Winter, J. Priaux, M. Galn, and P. Cuesta, editors, *Genetic Algorithms in Engineering and Computer Science, Proceedings of the First Short Course EUROGEN'95*, pages 111–126. Wiley, 1995.

[4] S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. In A. Prieditis and S. Russell, editors, *Proceedings of the twelfth International Conference on Machine Learning*, pages 38–46. Morgan Kauffman publishers, 1995.

[5] S. Baluja and S. Davies. Using optimal dependency–trees for combinatorial optimization: Learning the structure of the search space. In D.H. Fisher, editor, *Proceedings of the 1997 International Conference on Machine Learning*. Morgan Kauffman publishers, 1997.

[6] J. Bilmes. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. ICSI Technical Report TR-97-021. http://www.icsi.berkeley.edu/ bilmes/papers/em.ps.gz, 1997.

[7] J.S. De Bonet, C. Isbell, and P. Viola. MIMIC: Finding optima by estimating probability densities. *Advances in Neural Information Processing*, 9, 1996.

[8] P.A.N. Bosman and D. Thierens. An algorithmic framework for density estimation based evolutionary algorithms. Utrecht University Technical Report UU–CS–1999–46. ftp://ftp.cs.uu.nl/pub/RUU/CS/techreps/CS-1999/1999-46.ps.gz, 1999.

[9] P.A.N. Bosman and D. Thierens. Linkage information processing in distribution estimation algorithms. In W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, and R.E. Smith, editors, *Proceedings of the GECCO–1999 Genetic and Evolutionary Computation Conference*, pages 60–67. Morgan Kaufmann Publishers, 1999.

[10] P.A.N. Bosman and D. Thierens. Continuous iterated density estimation evolutionary algorithms within the IDEA framework. In M. Pelikan, H. Mühlenbein, and A.O. Rodriquez, editors, *Proceedings of the Optimization by Building and Using Probabilistic Models OBUPM Workshop at the Genetic and Evolutionary Computation Conference GECCO–2000*. Morgan Kaufmann Publishers, 2000.

[11] P.A.N. Bosman and D. Thierens. Expanding from discrete to continuous estimation of distribution algorithms: The IDEA. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature – PPSN VI*, pages 767–776. Springer, 2000.

[12] P.A.N. Bosman and D. Thierens. IDEAs based on the normal kernels probability density function. Utrecht University Technical Report UU–CS–2000–11. ftp://ftp.cs.uu.nl/pub/RUU/CS/techreps/CS-2000/2000-11.ps.gz, 2000.

[13] P.A.N. Bosman and D. Thierens. Negative log–likelihood and statistical hypothesis testing as the basis of model selection in IDEAs. In A. Feelders, editor, *Proceedings of the Tenth Dutch–Netherlands Conference on Machine Learning*. Tilburg University, 2000.

[14] S. Brandt. *Statistical And Computation Methods In Data Analysis*. North-Holland, 1970.

[15] T.M. Cover and J.A. Thomas. *Elements of Information Theory*. John Wiley & Sons Inc., 1991.

[16] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistic Society*, Series B 39:1–38, 1977.

[17] W. Feller. *An Introduction To Probability Theory And Its Applications, Volume 1*. Wiley, 1968.

[18] M. Gallagher, M. Fream, and T. Downs. Real–valued evolutionary optimization using a flexible probability density estimator. In W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, and R.E. Smith, editors, *Proceedings of the GECCO–1999 Genetic and Evolutionary Computation Conference*, pages 840–846. Morgan Kaufmann Publishers, 1999.

[19] G. Harik. Linkage learning via probabilistic modeling in the ECGA. IlliGAL Technical Report 99010. ftp://ftp-illigal.ge.uiuc.edu/pub/papers/IlliGALs/ 99010.ps.Z, 1999.

[20] J.A. Hartigan. *Clustering Algorithms*. John Wiley & Sons, Inc., 1975.

[21] M.G. Kendall and A. Stuart. *The Advanced Theory Of Statistics, Volume 2, Inference And Relationship*. Charles Griffin & Company Limited, 1967.

[22] P. Larrañaga, R. Etxeberria, J.A. Lozano, and J.M. Peña. Optimization by learning and simulation of bayesian and gaussian networks. University of the Basque Country Technical Report EHU-KZAA-IK-4/99. http://www.sc.ehu.es/ccwbayes/ postscript/kzaa-ik-04-99.ps, 1999.

[23] P.C. Mahalanobis. On tests and measures of groups divergence I. *Journal of the Asiatic Society of Benagal*, 26:541, 1930.

[24] H. Mühlenbein and T. Mahnig. FDA – a scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary Computation*, 7:353–376, 1999.

[25] M. Pelikan and D.E. Goldberg. Genetic algorithms, clustering, and the breaking of symmetry. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature – PPSN VI*, pages 385–394. Springer, 2000.

[26] M. Pelikan, D.E. Goldberg, and E. Cantú-Paz. BOA: The bayesian optimization algorithm. In W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, and R.E. Smith, editors, *Proceedings of the GECCO–1999 Genetic and Evolutionary Computation Conference*, pages 525–532. Morgan Kaufmann Publishers, 1999.

[27] M. Pelikan, D.E. Goldberg, and K. Sastry. Bayesian optimization algorithm, decision graphs and occam's razor. Also available as IlliGAL Report 2000020. ftp://ftp-illigal.ge.uiuc.edu/pub/papers/IlliGALs/2000020.ps.Z, 2000.

[28] M. Pelikan and H. Mühlenbein. The bivariate marginal distribution algorithm. In R. Roy, T. Furuhashi, K. Chawdry, and K. Pravir, editors, *Advances in Soft Computing – Engineering Design and Manufacturing*. Springer–Verlag, 1999.

[29] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

[30] M. Sebag and A. Ducoulombier. Extending population–based incremental learning to continuous search spaces. In A.E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature – PPSN V*, pages 418–427. Springer, 1998.

[31] I. Servet, L. Trave-Massuyes, and D. Stern. Telephone network traffic overloading diagnosis and evolutionary computation technique. In J.K. Hao, E. Lutton, E. Ronald, M. Schoenauer, and D. Snyers, editors, *Proceedings of Artificial Evolution '97*, pages 137–144. Springer Verlag, LNCS 1363, 1997.

[32] K.S. Trivedi. *Probability & Statistics With Reliability, Queuing And Computer Science Applications*. Prentice-Hall, 1982.

[33] X. Yin and N. Germay. A fast genetic algorithm with sharing scheme using cluster analysis methods in multimodal function optimization. In *Proceedings of the 1993 International Conference on Artificial Neural Nets and Genetic Algorithms*, pages 450–457, 1993.

A Notation glossary

General	
$A = \{\bullet, \bullet, \ldots, \bullet\}$	A set A of size $\lvert A\rvert - 1$. The order of the elements in a set is *irrelevant*, so its contents cannot be enumerated.
$\lvert A\rvert$	The amount of elements in set A.
$\emptyset$	The empty set.
$a \in A = \begin{cases} \text{true} & \text{if } a \text{ is contained in } A \\ \text{false} & \text{otherwise} \end{cases}$	Element query.
$A \subseteq B = \forall_{a \in A}\langle a \in B\rangle$	Subset query.
$A \subset B = A \subseteq B \wedge \lvert A\rvert < \lvert B\rvert$	Proper subset query.
$A \cup B = \{x \mid x \in A \vee x \in B\}$	Union operation on sets, each element appears only once in the result, $\lvert A \cup B\rvert \leq \lvert A\rvert + \lvert B\rvert$.
$A \cap B = \{x \mid x \in A \wedge x \in B\}$	Intersection operation on sets, each element appears only once in the result, $(\lvert A \cap B\rvert \leq \lvert A\rvert) \wedge (\lvert A \cap B\rvert \leq \lvert B\rvert)$.
$A - B = \{x \mid x \in A \wedge x \notin B\}$	Difference operation on sets.
$\boldsymbol{a} = (\boldsymbol{a}_0, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_{\lvert\boldsymbol{a}\rvert-1})$	A vector $\boldsymbol{a}$ of length $\lvert\boldsymbol{a}\rvert$. The order of the elements in a vector is *relevant*.
$\lvert\boldsymbol{a}\rvert$	The amount of elements in vector $\boldsymbol{a}$.
$()$	The empty vector.
$a \in \boldsymbol{a} = \bigvee_{0 \leq i < \lvert\boldsymbol{a}\rvert}(a = \boldsymbol{a}_i)$	Element query.
$\boldsymbol{a} \sqsubseteq \boldsymbol{b} = \forall_{0 \leq i < \lvert\boldsymbol{a}\rvert}\langle \boldsymbol{a}_i \in \boldsymbol{b}\rangle$	Subvector query.
$\boldsymbol{a} \sqsubset \boldsymbol{b} = \boldsymbol{a} \sqsubseteq \boldsymbol{b} \wedge \lvert A\rvert < \lvert B\rvert$	Proper subvector query.
$a\langle\boldsymbol{j}\rangle = (a_{\boldsymbol{j}_0}, a_{\boldsymbol{j}_1}, \ldots, a_{\boldsymbol{j}_{\lvert\boldsymbol{j}\rvert-1}})$	Shorthand for a vector, $\lvert a\langle\boldsymbol{j}\rangle\rvert = \lvert\boldsymbol{j}\rvert$.
$\boldsymbol{a} \sqcup \boldsymbol{b} = (\boldsymbol{a}_0, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_{\lvert\boldsymbol{a}\rvert-1}, \boldsymbol{b}_0, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_{\lvert\boldsymbol{b}\rvert-1})$	Union/splice operation on vectors, each element appears exactly as often as it appears in the input, $\lvert\boldsymbol{a} \sqcup \boldsymbol{b}\rvert = \lvert\boldsymbol{a}\rvert + \lvert\boldsymbol{b}\rvert$.
$\boldsymbol{a} \sqcap \boldsymbol{b} = \boldsymbol{a}\langle\boldsymbol{j}\rangle$ s.t. $\forall_{0 \leq i < \lvert\boldsymbol{j}\rvert}\langle \boldsymbol{a}_{\boldsymbol{j}_i} \in \boldsymbol{b} \wedge (i > 1 \rightarrow \boldsymbol{j}_i > \boldsymbol{j}_{i-1})\rangle$ $\wedge\ \forall_{0 \leq i < \lvert\boldsymbol{a}\rvert}\langle \boldsymbol{a}_i \in \boldsymbol{b} \rightarrow \boldsymbol{a}_i \in \boldsymbol{a}\langle\boldsymbol{j}\rangle\rangle$	Intersection operation on vectors, each element of $\boldsymbol{a}$ is preserved if it appears in $\boldsymbol{b}$, $\lvert\boldsymbol{a} \sqcap \boldsymbol{b}\rvert \leq \lvert\boldsymbol{a}\rvert$.
$\boldsymbol{a} - \boldsymbol{b} = \boldsymbol{a}\langle\boldsymbol{j}\rangle$ s.t. $\forall_{0 \leq i < \lvert\boldsymbol{j}\rvert}\langle \boldsymbol{a}_{\boldsymbol{j}_i} \notin b\ \wedge (i > 1 \rightarrow \boldsymbol{j}_i > \boldsymbol{j}_{i-1})\rangle$ $\wedge\ \forall_{0 \leq i < \lvert\boldsymbol{a}\rvert}\langle \boldsymbol{a}_i \notin \boldsymbol{b} \rightarrow \boldsymbol{a}_i \in \boldsymbol{a}\langle\boldsymbol{j}\rangle\rangle$	Difference operation on vectors.
$dy\langle\boldsymbol{a}\rangle = \prod_{i=0}^{\lvert\boldsymbol{a}\rvert-1} dy_{\boldsymbol{a}_i}$	Shorthand for the multivariate derivative.

Special symbols	
l	The amount of available random variables.
$\boldsymbol{\mathcal{L}} = (0, 1, \ldots, l-1)$	A vector containing l numbers, $\boldsymbol{\mathcal{L}}_i = i$.
k	The amount of components in the factorization mixture.
$\boldsymbol{\mathcal{K}} = (0, 1, \ldots, k-1)$	A vector containing k numbers, $\boldsymbol{\mathcal{K}}_i = i$.
w	The amount of components in the normal mixture pdf.
$\boldsymbol{\mathcal{W}} = (0, 1, \ldots, w-1)$	A vector containing w numbers, $\boldsymbol{\mathcal{W}}_i = i$.
$\boldsymbol{\mathcal{S}} = (\boldsymbol{y}^{\boldsymbol{0}}, \boldsymbol{y}^{\boldsymbol{1}}, \ldots, \boldsymbol{y}^{\lvert\boldsymbol{\mathcal{S}}\rvert-\boldsymbol{1}})$	The vector of sample points.
$\boldsymbol{y}^{\boldsymbol{i}} = (y_0^i, y_1^i, \ldots, y_{l-1}^i) = y^i\langle\boldsymbol{\mathcal{L}}\rangle$	The i–th sample point is an l dimensional vector, $(\boldsymbol{y}^{\boldsymbol{i}})_j = y_j^i$.
$\boldsymbol{\mathfrak{K}} = (\boldsymbol{\mathfrak{K}}^{\boldsymbol{0}}, \boldsymbol{\mathfrak{K}}^{\boldsymbol{1}}, \ldots, \boldsymbol{\mathfrak{K}}^{\lvert\boldsymbol{\mathfrak{K}}\rvert-\boldsymbol{1}})$	The vector of clusters.
$\boldsymbol{\mathfrak{K}}^{\boldsymbol{i}} = (\boldsymbol{\mathfrak{K}}_{\boldsymbol{0}}^{\boldsymbol{i}}, \boldsymbol{\mathfrak{K}}_{\boldsymbol{1}}^{\boldsymbol{i}}, \ldots, \boldsymbol{\mathfrak{K}}_{\lvert\boldsymbol{\mathfrak{K}}^{\boldsymbol{i}}\rvert-\boldsymbol{1}}^{\boldsymbol{i}})$	The i–th cluster contains $\lvert\boldsymbol{\mathfrak{K}}^{\boldsymbol{i}}\rvert - 1$ samples, $\forall_{0 \leq j < \lvert\boldsymbol{\mathfrak{K}}^{\boldsymbol{i}}\rvert}\langle \boldsymbol{\mathfrak{K}}_{\boldsymbol{j}}^{\boldsymbol{i}} \in \boldsymbol{\mathcal{S}}\rangle$.
$\boldsymbol{\mathcal{Y}} = (Y_0, Y_1, \ldots, Y_{l-1}) = Y\langle\boldsymbol{\mathcal{L}}\rangle$	A vector containing l continuous random variables, $\boldsymbol{\mathcal{Y}}_i = Y_i$.

B Statistical hypothesis testing

The testing of statistical hypothesis is a well understood topic. In this appendix, we restrict ourselves to the basics. We discuss important and frequently used existing tests along with their applicability in section B.1. In section B.2, we show how some of these tests can be used to define some dependency tests based upon correlation.

B.1 Some important statistical hypothesis tests and their distributions

A lot of statistical hypothesis tests are common because certain statistical expressions that are used to base hypotheses on, occur in many applications. Such tests have therefore been formalized over the past. The normalized pdfs that the expressions follow have been tabulated and are included in statistics text books. In this section, we go over some of these tests. We introduce the accompanying statistical expression, show how the density of the expression is formed and how the statistical hypothesis test can be formulated. The distributions are the most important. When we wish to test some property, the main idea is that some statistical expression is used that usually is not the same as the default expressions we shall introduce. The distribution however will be equal to the ones discussed here, so the hypothesis tests as explained here, can be used.

B.1.1 Z–Test

Assume that we have an a priori fixed hypothesized value $\mathfrak{h}$ for some random variable R that a hypothesis is based on. Assume furthermore that we know the actual standard deviation σ_R for random variable R. The definition of the standard deviation σ can be given with respect to the pdf P_R according to which the values for R are distributed and the mean μ_R of R:

$$\mu_R = E[R] = \int_{-\infty}^{\infty} y P_R(y) dy \tag{54}$$

$$\sigma_R^2 = E[(R-\mu)^2] = \int_{-\infty}^{\infty} (y-\mu)^2 P_R(y) dy \tag{55}$$

Given a vector $\boldsymbol{\mathcal{R}}$ of one–dimensional samples $\boldsymbol{\mathcal{R}} = (r_0, r_1, \ldots, r_{|\boldsymbol{\mathcal{R}}|-1})$ for random variable R, their sample mean $\overline{R}$ is defined as follows:

$$\overline{R} = \frac{\sum_{i=0}^{|\boldsymbol{\mathcal{R}}|-1} r_i}{|\boldsymbol{\mathcal{R}}|} \tag{56}$$

The Z–test is a test on the mean μ_R. The statistical expression that underlies this test, given some hypothesized value $\mathfrak{h}$, can be described as follows:

$$Z = \frac{\sqrt{|\boldsymbol{\mathcal{R}}|}(\overline{R} - \mathfrak{h})}{\sigma_R} \tag{57}$$

The *central limit* theorem states that if the samples $r_i \in \boldsymbol{\mathcal{R}}$ are all drawn from a pdf that has mean 0 and standard deviation σ, the distribution for the random variable $Y_{\mathcal{N}} = (\sqrt{|\boldsymbol{\mathcal{R}}|\overline{R}})/\sigma$ in the limit of $|\boldsymbol{\mathcal{R}}| \to \infty$ is the standard normal pdf $\mathcal{N}(0,1)$.

Using the central limit theorem, we have that if we transform the $r_i \in \boldsymbol{\mathcal{R}}$ so that we get $r'_i = r_i - \mu_R$, we have $\overline{R'} = \overline{R} - \mu_R$. By doing so, we have a random variable R' whose expectation is $E[R'] = 0$. Substituting this variable along with its samples in equation 57, we have according to the central limit theorem that the value of $Z' = (\sqrt{|\boldsymbol{\mathcal{R}}|}(\overline{R} - \mu_R))/\sigma_R$ is approximately distributed according to $\mathcal{N}(0,1)$. Furthermore, this approximation becomes more precise for increasing $|\boldsymbol{\mathcal{R}}|$.

In the Z–test, we assume that we know the actual standard deviation σ_R. However, we only have a hypothesized value $\mathfrak{h}$ for the actual mean μ_R. We can therefore hypothesize that $\mu_R = \mathfrak{h}$, or in other words that our hypothesized value is the actual mean. Given the sample set $\boldsymbol{\mathcal{R}}$, we can use the Z–test to validate this hypothesis. To show how this can be done, we first note that we may rewrite equation 57 as follows:

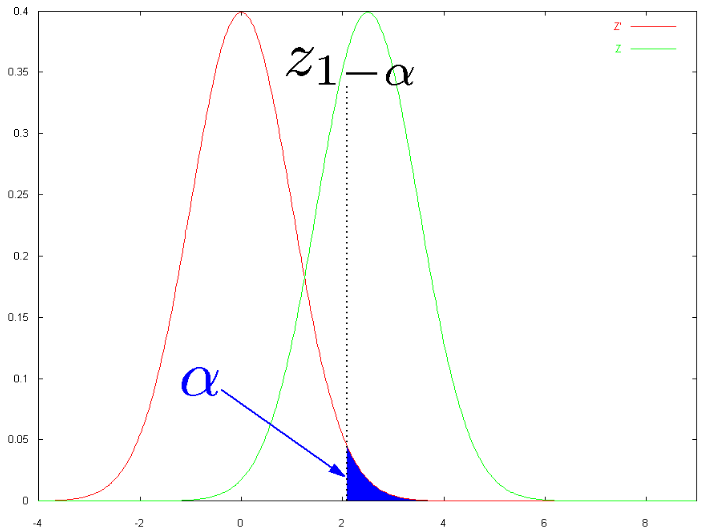

Figure 22: A right sided Z–test with $\mu_R = 2, \sigma_R = 2, \mathfrak{h} = 1, |\boldsymbol{\mathcal{R}}| = 25$.

$$Z = \frac{\sqrt{|\boldsymbol{\mathcal{R}}|}(\overline{R} - \mu_R + \mu_R - \mathfrak{h})}{\sigma_R} = Z' + \frac{\sqrt{|\boldsymbol{\mathcal{R}}|}(\mu_R - \mathfrak{h})}{\sigma_R} \tag{58}$$

If we now use the fact that if $Y_0 \sim \mathcal{N}(\mu, \sigma)$ and $Y_1 = aY_0 + b$, then $Y_1 \sim \mathcal{N}(a\mu + b, a\sigma)$, we can conclude that $Z \sim \mathcal{N}((\sqrt{|\boldsymbol{\mathcal{S}}|}(\mu_R - \mathfrak{h}))/\sigma_R, 1)$. We then observe that if our hypothesized value is correct, we have $\mu_R = \mathfrak{h}$ and thus $Z = Z' \sim \mathcal{N}(0, 1)$. If our hypothesized value is not correct however, the distribution for Z will deviate in its mean from Z'. The basic idea is now that the hypothesis regarding $\mathfrak{h}$ is merely an hypothesis and that its justification lies in rejecting the actual value, even though we might not know it. This means that we look at the distribution of Z' and observe the critical values $z_{\frac{\alpha}{2}}$ and $z_{1-\frac{\alpha}{2}}$ for which we take it to be significant that the sample mean differs from the actual mean and looks more like the hypothesized mean. A critical value c_x given some $x \in [0, 1]$ for some one–dimensional pdf $f(y)$ is defined as the input value for which the cumulative pdf equals x. This can be defined as follows:

$$\int_{-\infty}^{c_x} f(y)dy = x \tag{59}$$

We disregard the distribution of Z. Instead, we regard the standard distribution given by Z' and relate the actual instance value of Z to this distribution as a possible rejection of the hypothesis that $\mathfrak{h}$ is the actual mean value. This comes down to the notion of $\mu_R = \mathfrak{h}$ being the hypothesis H_1 we are testing against our basic hypothesis, which is often referred to as the null hypothesis H_0: $\mu_R \neq \mathfrak{h}$. Graphically this is depicted in figure 22.

Making a decision between the two distributions for Z and Z' is not errorfree as can be seen in figure 22. The area of overlap indicates the how well we can make a decision based on the set $\boldsymbol{\mathcal{R}}$. Clearly, if the mean of Z falls to the leftside of $z_{1-\alpha}$, we will not be able to decide very well whether to accept or reject the hypothesis given $\boldsymbol{\mathcal{R}}$. Note however that if the amount of samples $|\boldsymbol{\mathcal{R}}|$ goes up, the decision strength increases as the normalized distributions will be further apart since the mean of Z' remains 0, but the mean of Z moves further away from 0. The fact that the form of the distributions remains the same and that only the mean of Z changes is a result of the fact that Z is a *normalized* expression. The effect of more samples on the decision making is graphically depicted in figure 23.

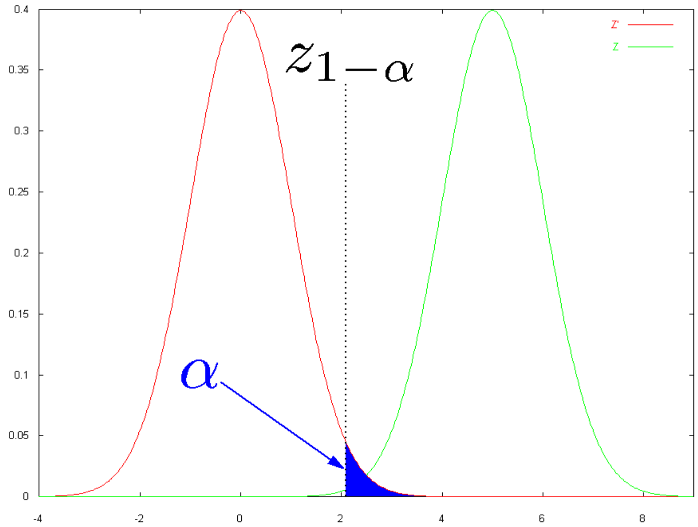

Figure 23: A right sided Z–test with $\mu_R = 2, \sigma_R = 2, \mathfrak{h} = 1, |\mathcal{R}| = 100$.

The hypothesis of the Z–test concerns the validity of $\mathfrak{h}$ being the correct mean. The probability α that such a hypothesis is not true, can be found using critical values. This test can be formalized by stating three hypotheses and noting for what critical value to accept them:

Hypothesis	**Accept if**
$H(\mu_R \neq \mathfrak{h})$	$\lvert Z\rvert > z_{1-\frac{\alpha}{2}}$
$H(\mu_R > \mathfrak{h})$	$Z > z_{1-\alpha}$
$H(\mu_R < \mathfrak{h})$	$Z < z_{\alpha}$

(60)

B.1.2 Student T–Test

In a typical case where we want to use a test for equality of means, the actual standard deviation σ_R will not be available. In such a case, we can use the unbiased sample standard deviation $\tilde{s}_R$:

$$\tilde{s}_R = \sqrt{\frac{\sum_{i=0}^{|\mathcal{R}|-1}(r_i - \overline{R})^2}{|\mathcal{R}| - 1}} \tag{61}$$

The reason why the denominator is not $|\mathcal{R}|$ but $|\mathcal{R}|-1$, is that the expected value of the actual sample variance does not equal σ_R^2, but $(\mathcal{R} - 1)/\mathcal{R}\sigma_R^2$. The actual sample variance is defined by by dividing by $|\mathcal{R}|$ instead of $|\mathcal{R}| - 1$ in equation 61:

$$s_R = \sqrt{\frac{\sum_{i=0}^{|\mathcal{R}|-1}(r_i - \overline{R})^2}{|\mathcal{R}|}} \tag{62}$$

The statistical expression that underlies the T–test, given some hypothesized value $\mathfrak{h}$, is the same as that for the Z–test, with σ_R substituted by $\tilde{s}_R$:

$$T = \frac{\sqrt{|\mathcal{R}|}(\overline{R} - \mathfrak{h})}{\tilde{s}_R} \tag{63}$$

The statistic T however cannot be shown to follow a normal distribution unless $|\mathcal{R}|$ goes to infinity. We call $\delta = |\mathcal{R}| - 1$ the amount of *degrees of freedom*. The reason for this is that if we

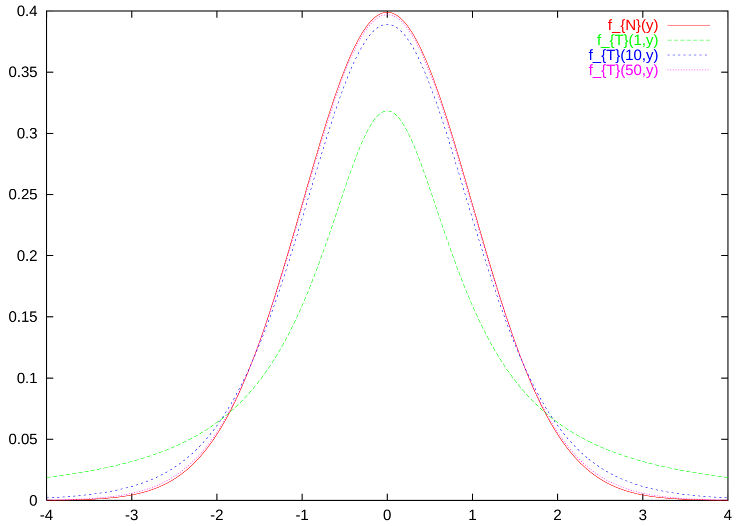

Figure 24: The difference between $f_{\mathcal{N}}(y, 0, 1)$ and $f_{\mathcal{T}}(\delta, y)$ for different values of δ.

are given $|\boldsymbol{\mathcal{R}}| - 1$ values $r_0, r_1, \ldots, r_{|\boldsymbol{\mathcal{R}}|-2}$, we are not yet constrained in any way with respect to the definition of the sample mean and the property that:

$$\sum_{i=0}^{|\boldsymbol{\mathcal{R}}|-1} r_i - \overline{R} = 0 \tag{64}$$

However, when the first $|\boldsymbol{\mathcal{R}}| - 1$ values are set, the value for $r_{|\boldsymbol{\mathcal{R}}|-1}$ cannot be chosen freely and must be fixed if equations 56 and 64 are to be obeyed. Therefore, we say that the T statistic has $\delta = |\boldsymbol{\mathcal{R}}| - 1$ degrees of freedom. It can be shown (see for instance [21]) that the T statistic has the following corresponding pdf with δ degrees of freedom:

$$f_{\mathcal{T}}(\delta, y) = \frac{\Gamma(\frac{\delta+1}{2})}{\Gamma(\frac{\delta}{2})\sqrt{\delta\pi}} \left(1 + \frac{y^2}{\delta}\right)^{-\frac{\delta+1}{2}} \tag{65}$$

In the definition of $f_{\mathcal{T}}$ we have used Euler's Gamma function $\Gamma(y)$:

$$\Gamma(y) = \int_0^\infty x^{y-1} e^{-x}\, dx, \quad y > 0 \tag{66}$$

To evaluate $\Gamma(y)$ efficiently, we can use a result by Feller [17], that states that as $y \to \infty$,

$$\Gamma(y) = y^{y-\frac{1}{2}} e^{-y} \sqrt{2\pi} \left(1 + \frac{1}{12y} + R_y\right), \text{ where } |R_y| < \frac{2}{3y^2} \tag{67}$$

The difference between the T pdf and the Z pdf is depicted in figure 24. In practice, for $\delta > 30$, $f_{\mathcal{N}}(y, 0, 1)$ can be used as an approximation to $f_{\mathcal{T}}(\delta, y)$ without any loss of significance.

It may be clear that this test is also a test on the mean. The accompanying hypotheses as well as the way in which we should use the T statistic are therefore of the exact same kind as in the case of the Z statistic. The test we have described, is called the Student T test for equality of means. We can formalize the Student T test by stating three hypothesis and noting for what critical value to accept them:

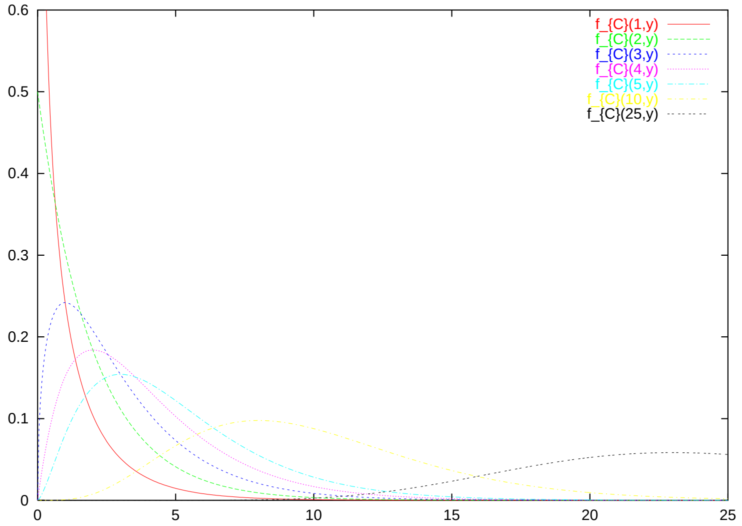

Figure 25: Plots of $f_C(\delta, y)$ for different values of δ.

Hypothesis	**Accept if**
$H(\mu_R \neq \mathfrak{h})$	$\|T\| > t_{1-\frac{\alpha}{2}}$
$H(\mu_R > \mathfrak{h})$	$T > t_{1-\alpha}$
$H(\mu_R < \mathfrak{h})$	$T < t_{\alpha}$

(68)

B.1.3 χ^2 Test

The notion of χ^2 was introduced by Pearson. The symbol χ^2 actually stands for a random variable, even though it appears to be a quadratic. The reason for this, is that the random variable χ^2 stands for the sum of squares of the samples for R in vector $\boldsymbol{\mathcal{R}}$ we have been working with so far:

$$\chi^2 = \sum_{i=0}^{|\boldsymbol{\mathcal{R}}|-1} r_i^2 \tag{69}$$

The χ^2 test is based on the assumption that R is a standard normally distributed variable with $\mu_R = 0$ and $\sigma_R = 1$. As there is no restriction on the variance as is the case for the T statistic, we have $\delta = |\boldsymbol{\mathcal{R}}|$ degrees of freedom in this case. It can be shown (see for instance [14]) that χ^2 follows the following pdf with δ degrees of freedom:

$$f_C(\delta, y) = \frac{y^{\frac{\delta}{2}-1} e^{-\frac{y}{2}}}{2^{\frac{\delta}{2}} \Gamma(\frac{\delta}{2})} \tag{70}$$

Example plots of the χ^2 pdf are depicted in figure 25. Now we turn our attention to the following test statistic:

$$C = \frac{(|\boldsymbol{\mathcal{R}}| - 1)\tilde{s}_R^2}{\mathfrak{h}^2} \tag{71}$$

Using equation 61, we can rewrite the C statistic to get:

$$C = \frac{\sum_{i=0}^{|\boldsymbol{\mathcal{R}}|-1} (r_i - \overline{R})^2}{\mathfrak{h}^2} \tag{72}$$

If we again set $r'_i = r_i - \overline{R}$, we have that $\overline{R'} = 0$. By dividing by the sum over r'_i by the actual variance σ_R^2, we get that the so achieved C' expression follows a χ^2 distribution from equation 70. The amount of degrees of freedom however is not equal to $|\boldsymbol{\mathcal{R}}|$ but $|\boldsymbol{\mathcal{R}}|-1$ as the statistic contains $\tilde{s}_R^2$. By substituting the actual variance by a hypothesized value $\mathfrak{h}^2$, we obtain again a similar situation to the previous two tests, but now concerning the variance parameter instead of the mean. This test is called the χ^2 test of variance.

Hypothesis	**Accept if**
$H(\sigma_R^2 \neq \mathfrak{h}^2)$	$C > c_{1-\frac{\alpha}{2}} \vee C < c_{\frac{\alpha}{2}}$
$H(\sigma_R^2 > \mathfrak{h}^2)$	$C > c_{1-\alpha}$
$H(\sigma_R^2 < \mathfrak{h}^2)$	$C < c_\alpha$

(73)

B.1.4 F–Test

The χ^2 test can be used to validate some hypothesized value for the standard deviation. If we are given two sets of samples $\boldsymbol{\mathcal{R}}^0$ and $\boldsymbol{\mathcal{R}}^1$ for respective random variables R^0 and R^1 however, we can test whether the variances of the distributions for the sets of samples are the same $H(\sigma_{R^0}^2 = \sigma_{R^1}^2)$. This leads to the F test. Firstly, observe the following definition:

$$C^i = \frac{(|\boldsymbol{\mathcal{R}}^i|-1)s_{R^i}^2}{\sigma_{R^i}^2},\ i \in \{0,1\} \tag{74}$$

It follows from the previous paragraph that C^0 and C^1 follow χ^2 distributions with $\delta^0 = |\boldsymbol{\mathcal{R}}^0|$ and $\delta^1 = |\boldsymbol{\mathcal{R}}^1|$ degrees of freedom respectively. We then define the quotient of the C^i multiplied by the inverse ratio of the degrees of freedom:

$$C_Q = \frac{\delta^1 C^0}{\delta^0 C^1} = \frac{s_{R^0}^2 \sigma_{R^1}^2}{s_{R^1}^2 \sigma_{R^0}^2} \tag{75}$$

It can be shown that if both R^0 and R^1 are normally distributed with arbitrary means and variances $\sigma_{R^0}^2$ and $\sigma_{R^1}^2$ (see for instance [32]), then C_Q is described by the following pdf with (δ^0, δ^1) degrees of freedom:

$$f_{\mathcal{F}}(\delta^0, \delta^1, y) = \left(\frac{\delta^0}{\delta^1}\right)^{\frac{\delta^0}{2}} \frac{\Gamma(\frac{\delta^0+\delta^1}{2})}{\Gamma(\frac{\delta^0}{2})\Gamma(\frac{\delta^1}{2})} \frac{y^{\frac{\delta^0}{2}-1}}{(1+\frac{y\delta^0}{\delta^1})^{\frac{\delta^0+\delta^1}{2}}} \tag{76}$$

Example plots of the F pdf are depicted in figure 26. If we now wish to test the hypothesis $H(\sigma_{R^0}^2 = \sigma_{R^1}^2)$, we can use C_Q as a statistic. Note that the hypothesis can be alternatively stated as $H(\sigma_{R^0}^2 = \mathfrak{h}^2 = \sigma_{R^1}^2)$ so that the $\sigma_{R^i} = \mathfrak{h}$ disappear in the resulting statistic because of their hypothesized equality:

$$F = \frac{s_{1R}^2}{s_{0R}^2} \tag{77}$$

The F statistic follows the F distribution with (δ^0, δ^1) degrees of freedom. The corresponding F test is called the F test for equality of variance.

Hypothesis	**Accept if**
$H(\sigma_{0R}^2 \neq \sigma_{1R}^2)$	$F > f_{1-\frac{\alpha}{2}} \vee F < f_{\frac{\alpha}{2}}$
$H(\sigma_{0R}^2 > \sigma_{1R}^2)$	$F > f_{1-\alpha}$
$H(\sigma_{0R}^2 < \sigma_{1R}^2)$	$F < f_\alpha$

(78)

Before concluding this paragraph, we note that there is an obvious correspondence between the F distribution and the χ^2 distribution as the F distribution is the ratio of two χ^2 distributions. We also point out however that the F distribution has a clear correspondence with the Student T distribution from equation 76. To be precise, if we set δ^1 to 1 and δ_2 to δ, the F distribution

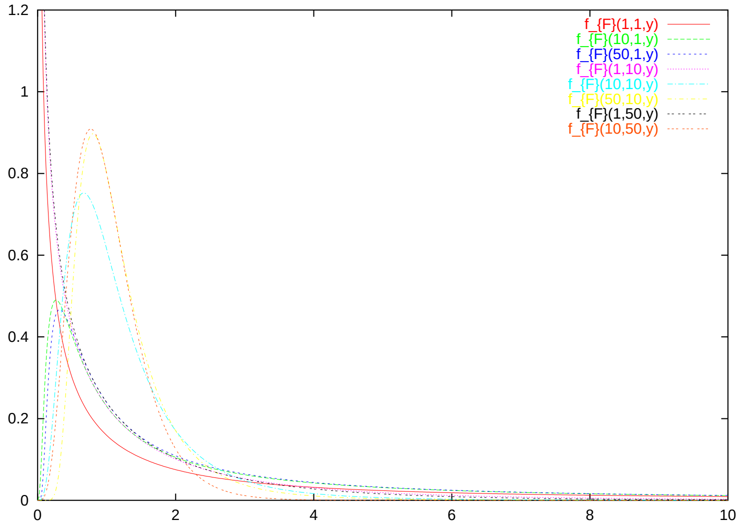

Figure 26: Plots of $f_{\mathcal{F}}(\delta^0, \delta^1, y)$ for different combinations of (δ^0, δ^1).

equals the T^2 distribution. This can be seen by first noting that the probability that $a < Y < b$ for some continuous random variable Y is given by:

$$P(a < Y < b) = \int_a^b f_Y(y)dy \tag{79}$$

In order to find the pdf for Y^2, we note that the probability that $0 \leq a < Y^2 < b$ is given by:

$$P(0 \leq a < Y^2 < b) = 2\int_{\sqrt{a}}^{\sqrt{b}} f_Y(y)dy \tag{80}$$

We cannot directly derive the density function for Y^2 from equation 80 as we have to transform $\sqrt{a}$ into a and $\sqrt{b}$ into b. To do this, we note that for any function $f(y)$ and its primitive function $F(y)$ such that $dF(y)/dy = f(y)$ and any function $g(y)$ with its corresponding inverse $g^{-1}(y)$ such that, at least on the interval $[a, b]$, $g^{-1}(g(y)) = g(g^{-1}(y)) = y$, we have that:

$$\int_a^b f(y)dy = F(t)\,\big|_a^b = F(b) - F(a) = \tag{81}$$

$$F(g^{-1}(g(b))) - F(g^{-1}(g(a))) = F(g^{-1}(y))\,\Big|_{g(a)}^{g(b)} = \int_{g(a)}^{g(b)} \left(\frac{dg^{-1}(y)}{dy}\right) f(g^{-1}(y))dy$$

If we now set $g(y) = y^2$ and thus $g^{-1}(y) = \sqrt{y}$ on the interval $[0, \infty\rangle$, we can combine equations 80 and 81 to get:

$$P(0 \leq a < Y^2 < b) = 2\int_{(\sqrt{a})^2}^{(\sqrt{b})^2} \frac{1}{2\sqrt{y}} f_Y(\sqrt{y})dy = \int_a^b \frac{1}{\sqrt{y}} f_Y(\sqrt{y})dy \tag{82}$$

Thus we find from equation 82 that the pdf for Y^2 equals $f_{Y^2} = \frac{1}{\sqrt{y}} f_Y(\sqrt{y})$. Using the fact that $\Gamma(\frac{1}{2}) = \sqrt{\pi}$, we can now show that the F distribution equals the T^2 distribution under the conditions that $\delta^1 = 1$ and $\delta_2 = \delta$:

$$f_{T^2}(\delta, y) = \frac{1}{\sqrt{y}} f_T(\delta, \sqrt{y}) = \frac{1}{\sqrt{y}} \frac{\Gamma(\frac{\delta+1}{2})}{\Gamma(\frac{\delta}{2})\sqrt{\delta\pi}} \left(1 + \frac{y}{\delta}\right)^{-\frac{\delta+1}{2}} = \tag{83}$$

$$\frac{1}{\sqrt{\delta}}\frac{\Gamma(\frac{1+\delta}{2})}{\sqrt{\pi}\Gamma(\frac{\delta}{2})}\frac{y^{-\frac{1}{2}}}{(1+\frac{y}{\delta})^{\frac{1+\delta}{2}}} = \left(\frac{1}{\delta}\right)^{\frac{1}{2}}\frac{\Gamma(\frac{1+\delta}{2})}{\Gamma(\frac{1}{2})\Gamma(\frac{\delta}{2})}\frac{(y)^{\frac{1}{2}-1}}{(1+\frac{y\cdot 1}{\delta})^{\frac{1+\delta}{2}}} = f_{\mathcal{F}}(1,\delta,y)$$

B.2 Unconditional dependency testing using correlation

In this section, we give an example of how we can test for unconditional dependencies by using the correlation measure. A test may be either *parametric* or *non–parametric*. In the case of a parametric test, the sample data is assumed to follow a certain distribution. Given this fact, a test can be derived. If this assumption is dropped, a test must be independent of whatever distribution the samples were drawn from. Such a *non–parametric* test is of course ultimately what we are looking for, but is mostly also more involved. Therefore, there is always a trade–off between being as precise as possible for every test and being faster and less precise. The amount in which the optimization process benefits from a certain test using as little effort as possible, determines its actual usage. The correlation coefficient is a measure of the *linear* dependence between two or more variables. As such, the test based upon the correlation measure is *parametric*. This section however serves mainly as an example. Furthermore, the correlation coefficient is a widely known measure. Finally, but not least most important, as using the correlation measure does *not* evaluate the estimated or underlying pdf, its computational running time requirements are always the same. In the case of using tests that do require an evaluation of the underlying pdf, its computation running time complexity can grow to become quite large.

B.2.1 Correlation coefficient

An introduction to statistics is most likely to visit the field of correlation and regression given two–dimensional observations. Such an introduction presents *linear* regression so as to investigate the linear relationship between two random variables. Even though such analysis is a crude measure of relationship, it is also a fundamental one. Furthermore, the resulting expressions can be evaluated efficiently. To start off, we first recall the *mean* μ_{Y_j} and the *sample mean* $\overline{Y_j}$ over random variable Y_j and refer to equations 54 and 56 respectively. Second, we recall the measure of how much the actual samples vary around this sample average, which is called the *sample variance* $s^2_{Y_j}$ over random variable Y_j and has been defined in equation 62. Its exact counterpart is the *variance*, which has been specified in equation 55.

The average of the product of the deviations from the sample averages for two random variables Y_j and Y_k over all samples, is the *covariance*. This measure $\sigma_{Y_jY_k}$ is useful for examining linear dependencies. If Y_k tends to be smaller than μ_{Y_k} whenever Y_j is larger than μ_{Y_j}, the covariance is negative. The covariance is positive in an analogous case. The sample equivalent of covariance is written as $s_{Y_jY_k}$. Their definitions are the following:

$$\sigma_{Y_jY_k} = E[(Y_j-\mu_{Y_j})(Y_k-\mu_{Y_k})] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(y_j-\mu_{Y_j})(y_k-\mu_{Y_k})f(y_j,y_k)dy_jy_k \tag{84}$$

$$s_{Y_jY_k} = \frac{\sum_{i=0}^{|\boldsymbol{S}|-1}(y^i_j-\overline{Y_j})(y^i_k-\overline{Y_k})}{|\boldsymbol{S}|} = \frac{\sum_{i=0}^{|\boldsymbol{S}|-1}y^i_jy^i_k}{|\boldsymbol{S}|} - \overline{Y_j}\,\overline{Y_k} \tag{85}$$

Note that $\sigma_{Y_jY_k} = \sigma_{Y_kY_j}$ and $s_{Y_jY_k} = s_{Y_kY_j}$. Also note that $\sigma_{Y_jY_j} = \sigma^2_{Y_j}$ and $s_{Y_jY_j} = s^2_{Y_j}$. We now turn our attention to linear regression. If we have an exact linear regression for Y_j on Y_k, the expected value for Y_j given a value for Y_k, is a linear equation. This can be formalized by writing:

$$E[Y_j|Y_k] = a_{jk} + b_{Y_jY_k}Y_k \tag{86}$$

Since $E[E[Y_j|Y_k]] = E[Y_j]$, taking expectation with respect to Y_k in equation 86 leads to:

$$\mu_{Y_j} = a_{jk} + b_{Y_jY_k}\mu_{Y_k} \tag{87}$$

If we now combine equation 86 with equation 87, and use sample approximations to the expectation expressions, we can derive the following:

$$E[Y_j|Y_k] - \mu_{Y_j} = b_{Y_jY_k}(Y_k - \mu_{Y_k}) \tag{88}$$

$$\Rightarrow$$

$$E[(E[Y_j|Y_k] - \mu_{Y_j})(Y_k - \mu_{Y_k})] = E[b_{Y_jY_k}(Y_k - \mu_{Y_k})^2] \quad \Rightarrow \quad b_{Y_jY_k} = \frac{\sigma_{Y_jY_k}}{\sigma^2_{Y_k}}$$

Analogously, we have that $b_{Y_kY_j} = \sigma_{Y_kY_j}/\sigma^2_{Y_j}$. The values $b_{Y_jY_k}$ and $b_{Y_kY_j}$ are indicators of the linear dependence that exists between variables Y_j and Y_k. The square root of their product is called the *correlation coefficient*:

$$\rho_{Y_jY_k} = \sqrt{b_{Y_jY_k}b_{Y_kY_j}} = \frac{\sigma_{Y_jY_k}}{\sigma_{Y_j}\sigma_{Y_k}} \tag{89}$$

Note that $\rho_{Y_jY_k} = \rho_{Y_kY_j}$. Furthermore, $-1 \leq \rho_{Y_jY_k} \leq 1$. Now, if Y_j and Y_k are independent, we have that $P(Y_j, Y_k) = P(Y_j)P(Y_k)$ and thus that $E[Y_jY_k] = E[Y_j]E[Y_k]$, meaning that $\sigma_{Y_jY_k} = \sigma_{Y_kY_j} = 0$ and thus that $\rho_{Y_jY_k} = \rho_{Y_kY_j} = 0$. However, if $\rho_{Y_jY_k} = \rho_{Y_kY_j} = 0$, we do not have in general that Y_j and Y_k are independent. This can only be stated if Y_j and Y_k are jointly normally distributed. Furthermore, the correlation coefficient is very sensitive to *outliers*. At this point, we should remind ourselves again that ρ_{jk} is a coefficient of *linear* dependence only. For these reasons, it is questionable whether or not ρ_{jk} should be used as a measure of unconditional dependence and what its strength in expressing such dependence is.

Having defined the correlation coefficient, we must still construct a test for the FSTH in the case of unconditional dependencies. We first continue to consider the case of two isolated variables Y_j and Y_k. Observing the definition of $\rho_{Y_jY_k}$, we define our hypothesis as $H(\rho_{Y_jY_k} = 0)$, representing the case in which Y_j and Y_k are not unconditionally dependent. Now observe the following statistic:

$$T_{Y_jY_k} = \sqrt{\frac{(|\boldsymbol{S}| - 2)}{\frac{1}{r^2_{Y_jY_k}} - 1}} \tag{90}$$

In the above statistic, $r_{Y_jY_k}$ is the sample correlation coefficient, which is equivalent to equation 89 with the covariance and standard deviation expressions substituted by their sample equivalents. Kendall [21] has shown that the statistic in equation 90 follows a Student T distribution with $(|\boldsymbol{S}| - 2)$ degrees of freedom. This implies that we can compute $T_{Y_jY_k}$ for a given sample set $\boldsymbol{S}$ and test whether the value is larger than the critical value of the Student T distribution at the confidence factor of $\frac{\alpha}{2}$, or in other words whether $T_{Y_jY_k} > t_{1-\frac{\alpha}{2}}$. If this is the case, the hypothesis is rejected $H(\rho_{Y_jY_k} = 0)$. Note that we have to test for half the significance value of α because of the fact that $r^2_{Y_jY_k}$ is symmetric. To violate the hypothesis we can either have $r_{Y_jY_k} < 0$ or $r_{Y_jY_k} > 0$. The FSTH in this case thus equals the null hypothesis $H(\rho_{Y_jY_k} \neq 0)$ we just described.

We have now arrived at the point where we can use the statistic and the accompanying test based on correlation for the use of unconditional factorization selection. However, we have only discussed the case of two variables. In the general case, we require to be able to perform a test for the unconditional dependence between $|\boldsymbol{a}|$ variables. A notion of *multiple correlation* exists, but it is a measure of the linear dependence of a single variable on a set of other variables. However, if a linear dependence of a single variable on a set of other variables is strong, the dependence of any other involved variable on the remainder of the variables will also be strong. So if we have a notion of multiple correlation, we can start out by finding the strongest correlation we can find between sets of variables, where we define the correlation between set S_0 and set S_1 as the maximum correlation of any $Y \in S_0$ with the variables in S_1. A rationale for such a decision lies in the fact that the variables in S_0 have already been found to be strongly correlated, just as is the case for the variables in S_1. If any of the variables in S_0 have a strong correlation with all of the variables in S_1, there must be a strong mutual correlation between the variables in both sets.

As we will have a test for validating whether the correlation is zero and we thus assume that there is no relationship, we actually test whether the strongest correlation that is found, is zero. If it is not, the merge is performed, otherwise the search for sets to merge stops. Kendall [21] warns for problems that may arise in using such a methodology as small lower order coefficients do not imply that higher order coefficients are also small. This means that linear dependence between multiple variables may come out only if a test is portrayed for higher order correlations. A method to perform such tests is to find the strongest correlation between a variable and all possible subsets of other variables, but such an approach leads to $\sum_{i=1}^{|\boldsymbol{S}|-1}(|\boldsymbol{S}|-1)!/(i!(|\boldsymbol{S}|-1-i)!)$ combinations, which is intractable for increasing $|\boldsymbol{S}|$.

In order to use our correlation approach, we thus need the notion of *multiple correlation*. To this end, we first define the *covariance matrix* $\boldsymbol{\Sigma}(\boldsymbol{a})$ to be the matrix such that $\boldsymbol{\Sigma}(\boldsymbol{a})(i,j) = \sigma_{Y_{\boldsymbol{a}_i}Y_{\boldsymbol{a}_j}}$. Furthermore, let $\boldsymbol{\Sigma}(\boldsymbol{a})_{\overline{ij}}$ be the matrix that is obtained by removing row i and column j from $\boldsymbol{\Sigma}(\boldsymbol{a})$. We can then define a cofactor $\boldsymbol{\Sigma}(\boldsymbol{a})_{ij}$:

$$\boldsymbol{\Sigma}(\boldsymbol{a})_{ij} = (-1)^{i+j}\det(\boldsymbol{\Sigma}(\boldsymbol{a})_{\overline{ij}}) \tag{91}$$

We now write the partial regression coefficient of $y_{\boldsymbol{a}_0}$ on $y_{\boldsymbol{a}_j}$ with the other $|\boldsymbol{a}|-2$ variables held fixed, by $b_{Y_{\boldsymbol{a}_0}Y_{\boldsymbol{a}_j}|Y\langle\boldsymbol{a}\rangle-\{Y_{\boldsymbol{a}_0},Y_{\boldsymbol{a}_j}\}}$. Kendall [21] shows that:

$$\frac{E[Y_{\boldsymbol{a}_0}|Y\langle\boldsymbol{a}\rangle - Y_{\boldsymbol{a}_0}]}{\sigma_{\boldsymbol{a}_0}} = -\sum_{j=1}^{l-1}\frac{\boldsymbol{\Sigma}(\boldsymbol{a})_{0j}}{\boldsymbol{\Sigma}(\boldsymbol{a})_{00}}\frac{Y_{\boldsymbol{a}_j}}{\sigma_{\boldsymbol{a}_j}} \tag{92}$$

Furthermore, using partial regression coefficients, we have that:

$$E[Y_{\boldsymbol{a}_0}|Y\langle\boldsymbol{a}\rangle - Y_{\boldsymbol{a}_0}] = \sum_{j=1}^{l-1} b_{Y_{\boldsymbol{a}_0}Y_{\boldsymbol{a}_j}|Y\langle\boldsymbol{a}\rangle-Y_{\boldsymbol{a}_0}Y_{\boldsymbol{a}_j}}Y_{\boldsymbol{a}_j} \tag{93}$$

Combining equations 92 and 93, we get a definition for the partial regression coefficients:

$$b_{Y_{\boldsymbol{a}_0}Y_{\boldsymbol{a}_j}|Y\langle\boldsymbol{a}\rangle-\{Y_{\boldsymbol{a}_0},Y_{\boldsymbol{a}_j}\}} = -\frac{\sigma_{Y_{\boldsymbol{a}_0}}}{\sigma_{Y_{\boldsymbol{a}_1}}}\frac{\boldsymbol{\Sigma}(\boldsymbol{a})_{0j}}{\boldsymbol{\Sigma}(\boldsymbol{a})_{00}} \tag{94}$$

The error or residual of order $l-1$ can now be defined by

$$Y_{\boldsymbol{a}_0|\boldsymbol{a}-\boldsymbol{a}_0} = Y_{\boldsymbol{a}_0} - E[Y_{\boldsymbol{a}_0}|Y\langle\boldsymbol{a}\rangle - Y_{\boldsymbol{a}_0}] \tag{95}$$

The mean of equation 95 is zero and its variance is defined to be:

$$\sigma^2_{Y_{\boldsymbol{a}_0}|Y\langle\boldsymbol{a}\rangle-Y_{\boldsymbol{a}_0}} = E[(Y_{\boldsymbol{a}_0} - E[Y_{\boldsymbol{a}_0}|Y\langle\boldsymbol{a}\rangle - Y_{\boldsymbol{a}_0}])^2] \tag{96}$$

Using the equations so far, we are now able to find that:

$$\sigma^2_{Y_{\boldsymbol{a}_0}|Y\langle\boldsymbol{a}\rangle-Y_{\boldsymbol{a}_0}} = \sigma^2_{Y_{\boldsymbol{a}_0}} - \sum_{j=1}^{l-1} b_{Y_{\boldsymbol{a}_0}Y_{\boldsymbol{a}_j}|Y\langle\boldsymbol{a}\rangle-\{Y_{\boldsymbol{a}_0},Y_{\boldsymbol{a}_j}\}}\sigma_{Y_{\boldsymbol{a}_0}Y_{\boldsymbol{a}_j}} \tag{97}$$

The *multiple correlation coefficient* $\boldsymbol{R}_{Y_{\boldsymbol{a}}(Y\langle\boldsymbol{a}\rangle-Y_{\boldsymbol{a}_0})}$ of $Y_{\boldsymbol{a}_0}$ on $Y_{\boldsymbol{a}_1}, Y_{\boldsymbol{a}_2}, \ldots, Y_{\boldsymbol{a}_{|\boldsymbol{a}|-1}}$ is a measure of the linear dependence. It is defined by:

$$\boldsymbol{R}_{Y_{\boldsymbol{a}_0}(Y\langle\boldsymbol{a}\rangle-Y_{\boldsymbol{a}_0})} = \sqrt{1 - \frac{\sigma^2_{Y_{\boldsymbol{a}_0}|Y\langle\boldsymbol{a}\rangle-Y_{\boldsymbol{a}_0}}}{\sigma^2_{Y_{\boldsymbol{a}_0}}}} \tag{98}$$

The sample analogue of equation 98 is written $R_{Y_{\boldsymbol{a}}(Y\langle\boldsymbol{a}\rangle-Y_{\boldsymbol{a}_0})}$. The multiple correlation coefficient is truly an extension of the ordinary correlation coefficient. If $l=2$, we have that $\boldsymbol{R}_{Y_{\boldsymbol{a}_0}(Y_{\boldsymbol{a}_1})} = \rho_{Y_{\boldsymbol{a}_0}Y_{\boldsymbol{a}_1}}$ and $R_{Y_{\boldsymbol{a}_0}(Y_{\boldsymbol{a}_1})} = r_{Y_{\boldsymbol{a}_0}Y_{\boldsymbol{a}_1}}$. In order to use this multivariate statistic, we require a test for the FSTH. To this end, observe the following ratio:

$$F_{Y_{\boldsymbol{a}_0}(Y\langle\boldsymbol{a}\rangle-Y_{\boldsymbol{a}_0})} = \frac{R^2_{Y_{\boldsymbol{a}_0}(Y\langle\boldsymbol{a}\rangle-Y_{\boldsymbol{a}_0})}(|\boldsymbol{S}|-|\boldsymbol{a}|)}{(1-R^2_{Y_{\boldsymbol{a}_0}(Y\langle\boldsymbol{a}\rangle-Y_{\boldsymbol{a}_0})})(|\boldsymbol{a}|-1)} \tag{99}$$

Kendall [21] has shown that $F_{Y_{\boldsymbol{a}_0}(Y\langle \boldsymbol{a}\rangle - Y_{\boldsymbol{a}_0})}$ follows an F distribution with $(|\boldsymbol{a}|-1, |\boldsymbol{\mathcal{S}}|-|\boldsymbol{a}|)$ degrees of freedom. This implies that we can compute the statistic and test whether the value is larger than the critical value $f_{1-\frac{\alpha}{2}}$ of the F distribution with the specified degrees of freedom. If this is so, the null hypothesis, which corresponds to the hypothesis that there is *no* dependence, is rejected. The FSTH thus corresponds to $H(R_{Y_{\boldsymbol{a}_0}(Y\langle \boldsymbol{a}\rangle - Y_{\boldsymbol{a}_0})} \neq 0)$. Note that the resulting hypothesis test in the multivariate case is an F test, whereas in the case of two variables we had a Students T test. However, if we use the correspondence between the multiple correlation coefficient and the ordinary correlation coefficient in the test expression in equation 99, we get:

$$T^2_{Y_{\boldsymbol{a}_j} Y_{\boldsymbol{a}_k}} = \frac{R^2_{Y_{\boldsymbol{a}_j}(Y_{\boldsymbol{a}_k})}(|\boldsymbol{\mathcal{S}}|-2)}{(1-R^2_{Y_{\boldsymbol{a}_j}(Y_{\boldsymbol{a}_k})})(2-1)} = \frac{|\boldsymbol{\mathcal{S}}|-2}{\frac{1}{r^2_{Y_{\boldsymbol{a}_j} Y_{\boldsymbol{a}_k}}}-1} \tag{100}$$

Thus, the tests are equivalent in the sense that the F test is the same as using a T^2 test in the case of two variables. This agrees with the correspondence between the F distribution and the Students T distribution as was shown in section B.1.4.

B.2.2 Spearmans rank correlation coefficient

In some cases, the use of the correlation measure as defined by equations 89 and 98 are unfit to serve as an indicator of linear dependence. This is for instance the case when we have *outliers*. If we have a strong outlier, the correlation measure will tend to find a linear relation as the remainder of the points seem more clustered into a single point relative to the outlier. This problem can be overcome if we focus on the relations *between* the points instead of on the locations *themselves*.

Another problem with using the ordinary correlation coefficient is that testing the hypothesis of no correlation is based on the assumption that the joint density of the variables is normal. Therefore, it is desirable to have a *nonparametric* test as an alternative in case the assumption on the underlying distribution of the variables is not satisfied. To this end, the values to actually compute the correlation coefficients with, have to be altered so that they reflect the relationships between the samples instead of the samples themselves. This leads to the definition of a *rank* as used in order statistics. We define the rank $\text{rank}(y^i_j)$ of a sample value y^i_j to be:

$$\text{rank}(y^i_j) = |\{y^i_k | ((y^i_k < y^i_j) \vee (y^i_k = y^i_j \wedge k \leq j)) \wedge k \in \{0, 1, \ldots, \boldsymbol{\mathcal{S}}\}\}| \tag{101}$$

The rank of a sample value in a certain dimension is thus defined as the amount of samples that is smaller than that value in that dimension, increased by the amount of samples that have the same value in that dimension. The latter is done to break ties. An equivalent definition of the rank of the i–th sample is its position in an ascending sorting that preserves the input ordering in case of ties. Kendall [21] points out that the use of ranks is invariant under the use of the hypothesis of independence as described in the case of the ordinary correlation measure. The simplest measure is then to use the ranks of the variables instead of the variables themselves in equations 89 and 98. This means that every occurrence of any $y^i_j, i \in \boldsymbol{\mathcal{L}}, j \in \boldsymbol{\mathcal{S}}$ is replaced by $\text{rank}(y^i_j)$. The resulting measure is called *Spearman's rank correlation coefficient.* There are other uses of ranks in correlation coefficients. The one described is however effective as well simple. The measure was named after Spearman, who invented it.

Since the formulae for the ordinary correlation coefficient are independent of the actual values, the transformation using the ranks directly makes it possible to use the tests and methodology as described for the ordinary correlation coefficient.

C The EM algorithm for the normal mixture pdf

The EM algorithm [16] is a general approach to compute maximum likelihood estimates. To achieve these estimates, it uses an iterative procedure. The reason why we require the EM algorithm, is that the maximum likelihood estimate cannot always be analytically determined. This is for instance the case for a normal mixture pdf. We discuss the EM algorithm in a somewhat general setting and give an instance for it using the normal pdf in the multivariate mixture model.

The EM algorithm uses the difference in negative log–likelihood between iterations to derive the parameters. The negative log–likelihood can be seen as an error that we wish to minimize. We let E^{old} be the error of the current mixture model and let E^{new} be the error of the new mixture model. The error difference then equals:

$$E^{\text{new}} - E^{\text{old}} = -\sum_{i=0}^{|\boldsymbol{S}|-1} \ln(\{\hat{P}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{new}}) + \sum_{i=0}^{|\boldsymbol{S}|-1} \ln(\{\hat{P}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}) = \tag{102}$$

$$-\sum_{i=0}^{|\boldsymbol{S}|-1} \ln\left(\frac{\{\hat{P}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{new}}}{\{\hat{P}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}}\right)$$

Starting from equation 102, update equations can be derived for the parameters involved in the probability distribution. As we assume that the probability distribution is a mixture that consists of w components, by writing α_i for the mixing coefficients, we can rewrite equation 102 as:

$$E^{\text{new}} - E^{\text{old}} = -\sum_{i=0}^{|\boldsymbol{S}|-1} \ln\left(\sum_{j=0}^{w-1} \varrho_j \frac{\alpha_j^{\text{new}}\{\hat{P}^j(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{new}}}{\varrho_j\{\hat{P}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}}\right) \tag{103}$$

$$\text{where } \varrho_j = \frac{\alpha_j^{\text{old}}\{\hat{P}^j(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}}{\{\hat{P}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}}$$

Jensen's inequality states that:

$$\left(\boldsymbol{\lambda} \in \mathbb{R}^{|\boldsymbol{\lambda}|} \wedge \forall_{\lambda\in\boldsymbol{\lambda}}\langle\lambda \geq 0\rangle \wedge \sum_{i=0}^{|\boldsymbol{\lambda}|-1} \boldsymbol{\lambda}_i = 1\right) \Rightarrow \ln\left(\sum_{i=0}^{|\boldsymbol{\lambda}|-1} \boldsymbol{\lambda}_i\gamma_i\right) \leq \sum_{i=0}^{|\boldsymbol{\lambda}|-1} \boldsymbol{\lambda}_i \ln(\gamma_i) \tag{104}$$

If we now observe equation 103, we note that $\varrho\langle\boldsymbol{\mathcal{W}}\rangle$ can be used for $\boldsymbol{\lambda}$ in equation 104, since $\{\hat{P}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}} = \sum_{j=0}^{w-1} \alpha_j^{\text{old}}\{\hat{P}^j(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}$. By using Jensen's inequality, we get:

$$E^{\text{new}} - E^{\text{old}} \leq -\sum_{i=0}^{|\boldsymbol{S}|-1}\sum_{j=0}^{w-1} \frac{\alpha_j^{\text{old}}\{\hat{P}^j(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}}{\{\hat{P}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}} \ln\left(\frac{\alpha_j^{\text{new}}\{\hat{P}^j(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{new}}}{\alpha_j^{\text{old}}\{\hat{P}^j(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}}\right) \tag{105}$$

In order to minimize the error, we seek to minimize the righthandside of equation 105 so that the decrease in the error $-(E^{\text{new}} - E^{\text{old}})$ is the largest possible. The old parameters are known and all of the terms that contain only these parameters are therefore constant. The righthandside of equation 105 can be written as:

$$-\sum_{i=0}^{|\boldsymbol{S}|-1}\sum_{j=0}^{w-1} \frac{\alpha_j^{\text{old}}\{\hat{P}^j(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}}{\{\hat{P}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}} \ln(\alpha_j^{\text{new}}\{\hat{P}^j(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{new}}) + \tag{106}$$

$$\sum_{i=0}^{|\boldsymbol{S}|-1}\sum_{j=0}^{w-1} \frac{\alpha_j^{\text{old}}\{\hat{P}^j(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}}{\{\hat{P}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}} \ln(\alpha_j^{\text{old}}\{\hat{P}^j(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}})$$

As we want to minimize the expression in equation 106, the second pair of sums can be omitted to this end since they involve *only* old values. If we now take the pdfs in the mixture to be of a certain form, we can derive an actual instance of the EM algorithm. We write $\boldsymbol{\Sigma}(\boldsymbol{a})$ for a symmetric covariance matrix over variables $Y\langle\boldsymbol{a}\rangle$. By taking each pdf in the mixture to be a multivariate normal pdf as described by equation 38, the expression to minimize, becomes:

$$E = -\sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \sum_{j=0}^{w-1} \frac{\alpha_j^{\text{old}} \{\hat{P}^j(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}}{\{\hat{P}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}} \times \tag{107}$$

$$\Bigg(\ln(\alpha_j^{\text{new}}) - \frac{|\boldsymbol{a}|}{2}\ln(2\pi) - \frac{1}{2}\ln(\det \{\boldsymbol{\Sigma^j}(\boldsymbol{a})\}^{\text{new}}) -$$

$$\frac{1}{2}(y^i\langle\boldsymbol{a}\rangle - \{\mu^j\langle\boldsymbol{a}\rangle\}^{\text{new}})^T(\{\boldsymbol{\Sigma^j}(\boldsymbol{a})\}^{\text{new}})^{-1}(y^i\langle\boldsymbol{a}\rangle - \{\mu^j\langle\boldsymbol{a}\rangle\}^{\text{new}})\Bigg)$$

In order to obtain expressions for the new parameters, we compute the partial derivative of E with respect to each parameter individually and equate it to 0. The tricky part in the mixing parameters is that they must continue to sum up to 1. By introducing a Lagrange multiplier λ, we can minimize:

$$E' = E + \lambda\left(-1 + \sum_{j=0}^{w-1} \alpha_j^{\text{new}}\right) \tag{108}$$

Subsequently, using the fact that $\sum_{j=0}^{w-1}(\alpha_j^{\text{old}}\{\hat{P}^j(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}})/\{\hat{P}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}} = 1$ as well as $\sum_{j=0}^{w-1} \alpha_j^{\text{new}} = 1$, we can derive:

$$\frac{\partial E'}{\partial \alpha_j^{\text{new}}} = 0 \iff \lambda - \sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \frac{\alpha_j^{\text{old}} \{\hat{P}^j(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}}{\{\hat{P}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}} \frac{1}{\alpha_j^{\text{new}}} = 0 \tag{109}$$

$$\iff$$

$$\sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \frac{\alpha_j^{\text{old}} \{\hat{P}^j(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}}{\{\hat{P}_{\mathfrak{f}}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}} = \lambda\alpha_j^{\text{new}}$$

$$\iff$$

$$\sum_{j=0}^{w-1} \sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \frac{\alpha_j^{\text{old}} \{\hat{P}^j(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}}{\{\hat{P}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}} = \sum_{j=0}^{w-1} \lambda\alpha_j^{\text{new}}$$

$$\iff$$

$$\sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \sum_{j=0}^{w-1} \frac{\alpha_j^{\text{old}} \{\hat{P}^j(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}}{\{\hat{P}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}} = \lambda\sum_{j=0}^{w-1} \alpha_j^{\text{new}}$$

$$\iff$$

$$\lambda = |\boldsymbol{\mathcal{S}}|$$

As a result, we have directly from the second derivation step in equation 109 that:

$$\alpha_j^{\text{new}} = \frac{1}{|\boldsymbol{\mathcal{S}}|} \sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \frac{\alpha_j^{\text{old}} \{\hat{P}^j(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}}{\{\hat{P}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\text{old}}} \tag{110}$$

For the mean parameters $\mu^j\langle\boldsymbol{a}\rangle$ of the j–th normal pdf in the mixture, we require to compute the partial derivative of E with respect to $\mu^j\langle\boldsymbol{a}\rangle$ and equate the result to 0:

$$\frac{\partial E}{\partial \{\mu^j\langle\boldsymbol{a}\rangle\}^{\text{new}}} = 0 \tag{111}$$

$$\Longleftrightarrow$$

$$-\sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \frac{\alpha_j^{\mathrm{old}}\{\hat{P}^j(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\mathrm{old}}}{\{\hat{P}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\mathrm{old}}} \left(\frac{1}{2}(\{\boldsymbol{\Sigma^j}(\boldsymbol{a})\}^{\mathrm{new}})^{-1}2(y^i\langle\boldsymbol{a}\rangle - \{\mu^j\langle\boldsymbol{a}\rangle\}^{\mathrm{new}})(-1)\right) = 0$$

$$\Longleftrightarrow$$

$$(\{\boldsymbol{\Sigma^j}(\boldsymbol{a})\}^{\mathrm{new}})^{-1}\{\mu^j\langle\boldsymbol{a}\rangle\}^{\mathrm{new}} \sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \frac{\alpha_j^{\mathrm{old}}\{\hat{P}^j(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\mathrm{old}}}{\{\hat{P}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\mathrm{old}}} =$$

$$(\{\boldsymbol{\Sigma^j}(\boldsymbol{a})\}^{\mathrm{new}})^{-1} \sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \frac{\alpha_j^{\mathrm{old}}\{\hat{P}^j(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\mathrm{old}}}{\{\hat{P}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\mathrm{old}}} y^i\langle\boldsymbol{a}\rangle$$

$$\Longleftrightarrow$$

$$\{\mu^j\langle\boldsymbol{a}\rangle\}^{\mathrm{new}} = \frac{\sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \frac{\alpha_j^{\mathrm{old}}\{\hat{P}^j(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\mathrm{old}}}{\{\hat{P}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\mathrm{old}}} y^i\langle\boldsymbol{a}\rangle}{\sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \frac{\alpha_j^{\mathrm{old}}\{\hat{P}^j(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\mathrm{old}}}{\{\hat{P}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\mathrm{old}}}}$$

Lastly, we want to find the new value for the covariance matrix for each model. The derivation of this is quite involved and requires the derivative of a function of a matrix with respect to elements in that matrix. An extensive description can be found elsewhere [6]. The unique solution is given by:

$$\frac{\partial E}{\partial\{\boldsymbol{\Sigma^j}(\boldsymbol{a})\}^{\mathrm{new}}} = 0 \tag{112}$$

$$\Longleftrightarrow$$

$$\{\boldsymbol{\Sigma^j}(\boldsymbol{a})\}^{\mathrm{new}} = \frac{\sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \frac{\alpha_j^{\mathrm{old}}\{\hat{P}^j_{\mathfrak{f}_j}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\mathrm{old}}}{\{\hat{P}_{\mathfrak{f}}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\mathrm{old}}}(y^i\langle\boldsymbol{a}\rangle - \{\mu^j\langle\boldsymbol{a}\rangle\}^{\mathrm{new}})(y^i\langle\boldsymbol{a}\rangle - \{\mu^j\langle\boldsymbol{a}\rangle\}^{\mathrm{new}})^T}{\sum_{i=0}^{|\boldsymbol{\mathcal{S}}|-1} \frac{\alpha_j^{\mathrm{old}}\{\hat{P}^j_{\mathfrak{f}_j}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\mathrm{old}}}{\{\hat{P}_{\mathfrak{f}}(Y\langle\boldsymbol{a}\rangle)(y^i\langle\boldsymbol{a}\rangle)\}^{\mathrm{old}}}}$$

D Additional algorithms

```
AddArc(κ, v_0, v_1, a[·,·], v^p[·], v^s[·])
1   m ← new array of boolean with size l
2   n^s, n^p ← 2 new vectors of integer
3   S ← new stack of integer
4   δ ← 0
5   v^s[v_0]_{|v^s[v_0]|} ← v_1
6   v^p[v_1]_{|v^p[v_1]|} ← v_0
7   if a[v_0, v_1] then
    7.1   a[v_0, v_1] ← false
    7.2   δ ← δ + 1
8   for i ← 0 to l − 1 do
    8.1 m[i] ← false
9   push(S, v_1)
10  while ¬empty(S) do
    10.1   v ← pop(S)
    10.2   if ¬m[v] then
           10.2.1   m[v] ← true
           10.2.2   n^s_{|n^s|} ← v
           10.2.3   for k ← 0 to |v^s[v]| do
                    10.2.3.1   push(S, v)
11  push(S, v_0)
12  while ¬empty(S) do
    12.1   v ← pop(S)
    12.2   if ¬m[v] then
           12.2.1   m[v] ← true
           12.2.2   n^p_{|n^p|} ← v
           12.2.3   for k ← 0 to |v^p[v]| do
                    12.2.3.1   push(S, v)
13  for i ← 0 to |n^s| do
    13.1   for j ← 0 to |n^p| do
           13.1.1   if a[n^s_i, n^p_j] then
                    13.1.1.1   a[n^s_i, n^p_j] ← false
                    13.1.1.2   δ ← δ + 1
14  if |v^p[v_1]| = κ then
    14.1   for i ← 0 to l − 1 do
           14.1.1   if a[i, v_1] then
                    14.1.1.1   a[i, v_1] ← false
                    14.1.1.2   δ ← δ + 1
15  return(δ)
```

```
TopologicalSort(v^p[·])
1   m ← new array of boolean with size l
2   for i ← 0 to l − 1 do
    2.1   m[i] ← false
    2.2   π(i) ← v^p[i]
3   z ← l − 1
4   for i ← 0 to l − 1 do
    4.1   if ¬m[i] then
          4.1.1   z ← TopologicalSortVisit(i,m,v^p,z)
5   return((π, ω))
```

$\textsc{TopologicalSortVisit}(i, m[\cdot], v^p[\cdot], z)$

1 $m[i] \leftarrow \boldsymbol{true}$

2 $\boldsymbol{for}\ j \leftarrow 0\ \boldsymbol{to}\ |v^p[i]| - 1\ \boldsymbol{do}$

  2.1 $\boldsymbol{if}\ \neg m[v^p[i]_j]\ \boldsymbol{then}$

    2.1.1 $z \leftarrow \textsc{TopologicalSortVisit}(v^p[i]_j, m, v^p, z)$

3 $\boldsymbol{\omega}_z \leftarrow i$

4 $\boldsymbol{return}(z - 1)$