# Estimating the structure of probabilistic graphical models through a Gaussian copula with discrete marginals

Bogdan Mazoure

A Thesis Submitted to McGill University in Partial Fulfillment of the Requirements for the Degree of Master of Science

Department of Mathematics and Statistics

McGill University

Montreal,Quebec

December 2018

## ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor Prof. Johanna Neslehova for continuous support throughout my Master's study, for her patience, motivation, and expertise. This research was made possible by research grants to my supervisor from the Natural Sciences and Engineering Research Council of Canada. I would like to thank my colleagues in the Department who greatly contributed to the work through discussions. Last but not the least, I would like to thank my parents, my sister, my girlfriend Kayla Branson and my friends for supporting me throughout writing this thesis and in my life in general.

## ABSTRACT

The purpose of this thesis is to investigate parameter estimation in a multivariate Gaussian copula model with discrete marginal distributions. Connections between Gaussian copula models and probabilistic graphical models are analysed. Building upon theoretical results in Nešlehová (2007) and Popovic et al. (2018), three novel algorithms are proposed. The first method relies on multi-armed bandits, the second uses a variational upper bound approximation and the third leverages recent advances in deep generative models (Kingma and Welling, 2013). All three methods rely on a novel full rank decomposition of square real matrices which is shown to have desirable theoretical properties. The performance of all three algorithms is compared to existing algorithms on two and three dimensional simulated datasets.

# ABRÉGÉ

La recherche qui fait l'objet de ce mémoire a pour but d'explorer l'estimation des paramètres de dépendance d'un modèle de copule Gaussienne pour données discrètes. Les liens entre les modèles de copule Gaussienne et les modèles graphiques sont analysés. En s'appuyant sur les travaux antérieurs de Nešlehová (2007) et Popovic et al. (2018), trois nouveaux algorithmes sont proposés. La première méthode se base sur les bandit manchots, la deuxième utilises une approximation à la limite variationnelle supérieure et la troisième apprend le modèle de copule à l'aide d'un modèle génératif profond (Kingma and Welling, 2013). Les trois algorithmes utilisent une nouvelle factorization de matrice et il est ensuite démontré que cette décomposition a des propriétés théoriques intéressantes. Finalement, la performance des méthodes est comparée à celle des algorithmes existants sur des données deux et trois dimensionnelles simulées.

TABLE OF CONTENTS

LIST OF TABLES

Table page

LIST OF FIGURES

# Chapter 1
# Introduction

## 1.1 Introduction

Recent breakthroughs in deep statistical learning caused a drastic increase in the volume of processed data. For example, the cloud infrastructure allows organizations to store information taken from social networks, finance sector and search history with little to no loss (Abaker et al., 2015). On the other hand, the emergence of fast, flexible and scalable learning algorithms such as logistic regression, random forests, Bayesian networks and later convolutional neural networks makes the task of data filtering (e.g., imputation of missing values or removal of outliers) easier and more efficient to perform. We can informally split all regression and classification methods in two categories: blackbox and whitebox methods. Whitebox approaches are models with interpretable outputs and parameters whose primary goal is inferring the underlying structure of a given process. Such methods include but are not limited to linear and logistic regression, random forests and Bayesian networks. Blackbox algorithms are models in which everything except the input and output variables, as well as a small set of hyperparameters is known; their mechanism is usually very complex and does not yield any meaningful insight (Wei Koh and Liang, 2017). We can list support vector machines (SVMs) and artificial neural networks as the most widely known blackbox algorithms.

In practice, fitting a whitebox model to a high-dimensional dataset tends to be time-consuming due to large numbers of covariates. Moreover, approaches such as logistic regression are prone to overfitting and require explicit regularization. Using lasso ($\ell_1$), ridge ($\ell_2$) or ElasticNet (Zou and Hastie, 2005) (combined lasso and ridge) penalties will yield a sparse solution of regression coefficients. Another widely used feature selection method is based on removing correlated covariates from the design matrix (Hall, 1999). In fact, knowing the dependence structure of features is hence crucial for efficient filtering.

Bayesian networks form an appealing class of probabilistic models: they are fast to learn, can handle both discrete and continuous data and, most importantly, are arguably one of the most interpretable learning algorithms due to their explicit representation of conditional independence. Most of the literature on the estimation of parameters in Bayesian networks assumes a known conditional dependency structure represented through a directed acyclic graph. In this thesis, we will focus on estimating dependencies between random variables, which reduces to finding the most suitable connectivity structure (to be formally defined in the next chapter) of the dependence graph. The main objective of the thesis is, given a design matrix in which every row represents a $d-$dimensional discrete observation, to estimate the adjacency matrix of the corresponding dependence graph. That is, if two columns of the design matrix are dependent in the probability sense, the vertices corresponding to these columns should be connected with an edge in the dependence graph. We use the fact that for samples from the multivariate Gaussian distribution, correlation is equivalent to dependence and leverage this property to represent the dependence

between random variables with a latent Gaussian copula model. In this setting, estimating the adjacency of the dependence graph is equivalent to estimating an invertible covariance matrix for the latent Gaussian copula function.

In the second chapter, we provide background information on methods currently used to estimate the dependence structure between random variables and discuss their limitations when applied to discrete data. We argue that models based on the multivariate Gaussian distribution can be used to represent the dependence structure between random variables. For instance, the graphical lasso known as glasso (Friedman et al., 2008) has traditionally been used to provide a sparse estimate of the precision matrix in Gaussian graphical models. If the marginal distribution functions of the data are discrete, glasso does not explicitly account for the presence of ties in the observations, which might lead to bias and numerical stability issues. To solve the dependence graph reconstruction problem in the discrete case, we introduce the reinforcement learning and approximate Bayesian inference frameworks and argue for their use in the newly proposed methods.

In the third chapter, we propose three novel algorithms based on state-of-the-art machine learning techniques in order to estimate the copula parameters. The first approach makes use of multi-armed bandits and frames the estimation of the covariance matrix problem as a multi-lever game. The second approach uses Hölder's variational upper bound to estimate the latent Gaussian integral associated with the marginal probability mass functions and proceeds similarly to coordinate ascent to estimate the precision matrix. The third algorithm relies on auto-encoding variational Bayes neural networks to automatically learn the marginal probability mass

functions as well as the parameters of the latent model. Moreover, we suggest a full-rank decomposition of the Gaussian covariance matrix into the product of two learnable vectors and show that the proposed parameterization is always invertible. This property guarantees the existence of a precision matrix and hence of a dependence graph associated to the design matrix of interest.

In the fourth chapter, we conduct simulation studies to compare the performance of our methods among them and to existing approaches. We show that while all proposed methods manage to recover the correct graph structure, the number of hyperparameters which require tuning is quite high for both the multi-armed bandit and the variational Hölder algorithms. The third proposed method, however, achieved comparable performance to state-of-the-art algorithms while being easy to tune, fast and requiring a number of learnable parameters linear in the dimension of the data.

In the fifth chapter, we discuss the results of the simulated studies and argue that the variational Bayes approach performed the best among the three proposed algorithms. We provide insight into the interpretation of the latent Gaussian copula model and draw parallels with recent work on stochastic computational graphs. Finally, we conclude by suggesting an additional modification that could be examined to improve the performance and the interpretability of our suggested approaches.

# Chapter 2
# Background

## 2.1 Dependence modeling with graphical models

Probabilistic graphical models (PGMs) are widely used in statistics and computer science (Lauritzen, 1996). They represent joint probability distributions through specification of conditional independence.

A graphical model is a graph $\mathcal{G} = (V, E)$ with a set of vertices $V$ and set of edges $E = \{e : e = (v_i, v_j) \forall v_i, v_j \in V\}$. Along with $\mathcal{G}$, a PGM also has a set of marginal probability distributions defined on subsets of $V$. That is, a graphical model based on a graph $\mathcal{G}$ induces a joint probability distribution on vertices $v_1, ..., v_d$.

If $\mathcal{G}$ is a directed acyclic graph (DAG), that is a graph without cycles, then the PGM defines a so-called *Bayesian network*. On the other hand, if it is composed of undirected edges, $\mathcal{G}$ is called a *Markov random field*. While both representations have common properties, they are used in different situations (see Lauritzen (1996) and Højsgaard et al. (2012)). In both models, the joint density function factors along the edges of the graph due to the Markov assumption.

A parent vertex of $w$ in a DAG is any vertex $v \in V$ such that $\exists\ e \in E : e = (v, w)$, that is from $v$ to $w$. Parent vertices in Markov random fields are vertices which are connected to the target vertex by an undirected edge. Furthermore, a vertex $v'$ is said to be adjacent to the vertex $v \in V$ if and only if $\exists e \in E : e = (v, v')$ or $e = (v', v)$. Finally, a *clique* is a subset $C$ of vertices of a graph $\mathcal{G} = (V, E)$ such that

any two $c, c' \in C$ are adjacent.

Let $\mathcal{G} = (V, E)$ be an undirected graph. We say that for any $A, B \subseteq V$ and any $S \subseteq V$ where $S \cap (A \cup B) = \emptyset$, $S$ $d$-separates $A$ and $B$ if every path from any node in $A$ to any node in $B$ passes by a node in $S$. Finally, we allow the random variables to be indexed by sets of corresponding vertices. That is, if $I \subseteq V$, then $Y_I = \{Y_i : i \in I\}$.

We now present what is known as the Markov property for graphical models.

**Definition 2.1.1.** *Let $\mathcal{G} = (V, E)$ be either a directed or undirected graph which, together with a set of random variables $Y = \{Y_{v_1}, ..., Y_{v_d}\}$ indexed by $V$, forms a tuple $\mathcal{B}$. The following statements define the pairwise, local and global Markov property over graphs, respectively:*

1. *If for any two non-adjacent vertices $v_i, v_j \in V : Y_{v_i} \perp Y_{v_j} | Y_{V \setminus \{v_i, v_j\}}$, then $\mathcal{B}$ is said to satisfy the pairwise Markov property;*
2. *If for any vertex $v_i \in V : Y_{v_i} \perp Y_{V \setminus Parents(v_i)} | Parents(v_i)$, then $\mathcal{B}$ is said to satisfy the local Markov property;*
3. *If for any subsets of vertices $S, S'$ and $T \subseteq V$ such that $T$ d-separates $S$ and $S'$,*

$$Y_S \perp Y_{S'} | Y_T, \tag{2.1}$$

*then $\mathcal{B}$ is said to satisfy the global Markov property.*

When the tuple $\mathcal{B}$ satisfies either the local or the global Markov property, it is known as a graphical model. If the graph is directed, the model is known as a Bayesian network and if the graph is undirected, it is known as Markov random field.

**Definition 2.1.2.** *Given a DAG $\mathcal{G} = (V, E)$, a multivariate random vector indexed by $V$ is said to form a Bayesian network with respect to $\mathcal{G}$ if it satistfies the local Markov property with respect to $\mathcal{G}$.*

If $Y = (Y_1, ..., Y_d)$ is a Bayesian network with respect to $\mathcal{G}$, then the joint density factorizes, viz

$$P[Y_{v_1} = y_{v_1}, ..., Y_{v_d} = y_{v_d}] = \prod_{i=1}^{d} P[Y_{v_i} = y_{v_i} | Y_{\text{Parents}(v_i)}]. \tag{2.2}$$

**Definition 2.1.3.** *Given an undirected graph $\mathcal{G} = (V, E)$, a multivariate random vector indexed by $V$ is said to form a Markov random field with respect to $\mathcal{G}$ if it satisfies the global Markov property with respect to $\mathcal{G}$.*

By the Hammersley-Clifford theorem, if $Y$ is a Markov random field and its density is strictly positive, then the joint distribution factorizes according to the cliques in the graph:

$$P[Y_{v_1} = y_{v_1}, ..., Y_{v_d} = y_{v_d}] = \prod_{k=1}^{K} \psi_k(y_{c_k}), \tag{2.3}$$

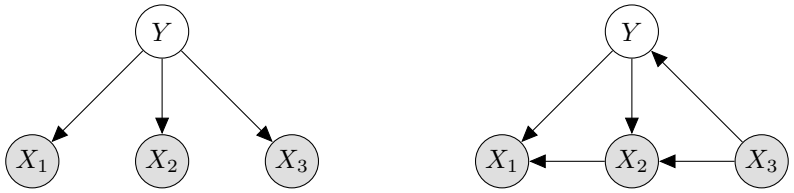
where $c_1, ..., c_k \subseteq V$ are the cliques of $\mathcal{G}$ and $\psi_k(y_{c_k})$ is referred to as the potential of $y_{c_k}$ indexed by the clique $c_k$.

Bayesian graphical models are often used to learn the posterior distribution of some parameter of interest $\theta$. To fully specify a Bayesian network, it is required to provide a graph $\mathcal{G}$ which encodes the dependence structure of the joint probability mass function of $d$ discrete random variables, $P[Y_{v_1} = y_{v_1}, Y_{v_2} = y_{v_2}, ..., Y_{v_d} = y_{v_d}]$, as well as one conditional marginal per random vertex, $P[Y_{v_i} = y_{v_i} | Y_{\text{Parents}(v_i)}]$. The

main reason to ever consider dealing with such models is mostly a reduced parameter space. For example, suppose that we wish to represent a 3-dimensional joint Bernoulli distribution of $X_1, X_2, X_3$. We would need $2^3 - 1 = 7$ parameters to deduce the corresponding probabilities since we cannot assume independence. In general, for a $d-$dimensional probability function with binary outcomes, we must fit $2^d - 1$ parameters. Now, suppose that we have a Bayesian network. The number of parameters is at most $2^d - 1$ but is usually much less. The graphical model factorizes the multivariate density by removing parameters due to conditional independence assumptions. For example, the network shown in Fig. 2–1a with 4 nodes has 7 parameters. In general, Bayesian networks with $d+1$ binary nodes can be defined with only $2d+1$ parameters as opposed to $2^d-1$ if no assumptions about the independence structure are made.

One of the most famous examples of a Bayesian PGM is a naive Bayesian network (Mccallum and Nigam, 1998). Very often used for classification tasks in high-dimensional problems, naive Bayes classifiers assume the following setup. Given a tuple $(X, Y)$ where $X$ is a design or feature matrix and $Y$ the labels or categories of the corresponding observations, we are interested in finding $P[Y = k|X] = \frac{P[X|Y=k]P[Y=k]}{P[X]}$. If $k \in \{0, 1\}$, the task is binary classification and can be solved through alternative methods such as logistic regression. However, if we view every column in $X$ (i.e. every covariate) as a node in $\mathcal{G}$, then the naive Bayes assumption stipulates that the features $X_1, ..., X_d$ are conditionally independent of each other given the outcome $Y$. This drastically reduces the dimensionality of the parameter space and allows fast inference with a smaller chance of overfitting.

To represent graphical models, we make use of the following graph notation: shaded




(a) Naive Bayesian network (b) Bayesian network

Figure 2–1: The graphical model specified by the set of of covariates $\{X_1, X_2, X_3\}$ and the response $Y$ under **a)** the naive Bayes assumption and **b)** no naive Bayes assumption.

nodes represent observations (i.e. evidence), blank nodes represent latent (i.e. unobserved) random variables, edges between two nodes indicate that both random variables are not independent of each other and, finally, a plate around a set of nodes provides a shorthand notation for $N$ identically distributed nodes.

In this case, Fig.2–1a shows a network assuming conditional independence of covariates or features on the outcome variable $Y$. If $Y, X_1, ..., X_d$ are Bernoulli random variables, then we require the following parameters to be learned: $P[Y = 1], P[X_3 = 1|Y = 1], P[X_3 = 1|Y = 0], P[X_2 = 1|Y = 1], P[X_2 = 1|Y = 0], P[X_1 = 1|Y = 1], P[X_1 = 1|Y = 0]$ for $d = 3$. For a Bayesian network, we typically have at most $2^d - 1$ parameters (fully connected joint) and at least $d$ parameters. Thus, we are required to fit $O(d)$ parameters for $d$ features under the Naive Bayes assumption, $2^d - 1$ parameters with no conditional independence assumptions and anywhere in between with a Bayesian network. Since the connectivity of the graph directly affects the cardinality of the parameter space and hence the time complexity of the algorithm, estimating the conditional independence structure of $d$ random variables is of great importance.

It is often more time and space efficient to reconstruct an undirected graphical model from data as opposed to a directed graphical model. Suppose that we wish to estimate the connectivity of a dependence graph consisting of $d$ vertices. In an undirected graph, each edge is either present or not, resulting in $2^{\binom{d}{2}}$ distinct models, each of which should be assessed with a goodness-of-fit criterion based on penalized likelihood such as AIC, BIC or DIC. In a directed graph, each edge between nodes $v, v' \in V$ can be either absent, going from $v$ to $v'$ or from $v'$ to $v$, resulting in $3^{\binom{d}{2}}$ possible graphs. As the number of vertices increases, we obtain an exponentially large search space for which no efficient (that is, polynomial time) algorithm exists. In this work, we deal with undirected dependence structures between observed random variables.

In order to transpose the Markov property on undirected networks, we need to mention results related to conditional independence in probabilistic graphical models.

In contrast with Bayesian networks, undirected PGMs are defined over subsets of nodes called cliques. As mentioned earlier, estimating a directed graphical model from data is much more time-consuming than estimating an undirected graphical model. The search space for the second problem is much smaller than the first one and, if we were to use a heuristic approach and examine the most promising solutions only, is expected to be more accurate.

### 2.2 Copula functions

Copulas are a powerful tool used in multiple disciplines in order to model dependence between random variables. This section deals with the general formulation of copulas and proceeds to focus on a particular copula function: the multivariate

Gaussian copula. It is important to mention that all results in this section that are related to copulas have been borrowed from (Nelsen, 2007).

We begin by citing a fundamental result in probability theory.

**Theorem 2.2.1.** *Let $X$ be a random variable with a continuous distribution function $F$.*

*Then $U = F(X) \sim Uniform(0,1)$, called the probability transform. Conversely, if $U \sim Uniform(0,1)$ and $F$ an arbitrary distribution, then $Y = F_X^{-1}(U)$ has distribution $F$. The later is called the quantile transform.*

Theorem 2.2.1 is widely used in sampling techniques, where we are interested in obtaining i.i.d. draws from a target continuous distribution $Y$ with a cdf $F$. We would then proceed by sampling $U_1, ..., U_n \sim U(0,1)$ and inverting the cdf in order to obtain $F^{-1}(U_1), ..., F^{-1}(U_n)$, a random sample from $F$.

We are now ready to provide a definition for copula functions.

**Definition 2.2.2.** *A copula $C$ is a $d-$variate joint distribution function on the real hypercube $[0,1]^d$ with standard uniform margins.*

A fundamental result linking distribution functions to copulas is due to Abe Sklar (1959).

**Theorem 2.2.3** (Sklar)**.** *Let $(X_1, ..., X_d)$ be a random vector with a joint distribution function $F$ and marginal distribution functions $\{F_j\}_{j=1}^d$. Then, there exists a copula $C$ such that for all $x_1, ..., x_d \in \mathbb{R}$*

$$F(x_1, ..., x_d) = C(F_1(x_1), ..., F_d(x_d)).$$

*If the marginals are continuous, then $C$ is unique. Conversely, for any univariate distribution functions $\{F_j\}_{j=1}^d$ and arbitrary copula $C$, the function given by $C(F_1(x_1), ..., F_d(x_d))$ is a joint distribution function with marginals $F_1, ..., F_d$.*

Consider a random vector $X = (X_1, ..., X_d)$ with a set of corresponding marginal distribution functions $\{F_j\}_{j=1}^d$ and joint distribution function $F$. Then, by Sklar's theorem there exists a copula $C$ such that

$$F(x_1, ..., x_d) = C(F_1(x_1), ..., F_d(x_d)) \tag{2.4}$$

When the margins are continuous, we can find the copula by applying Theorem 2.2.1 to the joint distribution function, which yields

$$C(u_1, ..., u_d) = F(F_1^{-1}(u_1), ..., F_d^{-1}(u_d)) \tag{2.5}$$

Broadly speaking, copula functions rely on the probability and quantile transforms in order to model the dependence structure between random variables with arbitrary (continuous and discrete) distributions.

An important type of copulas is the copula of the multivariate Gaussian distribution. Consider the simplest (bivariate) case, where $C_R = \Phi_2(\Phi_1^{-1}(u_1), \Phi_1^{-1}(u_2))$ is parameterized by a correlation coefficient $R \in [-1, 1]$. Varying $R$ between these limits yields a dependence pattern for $X_1, X_2$, as shown in Fig. 2–2.

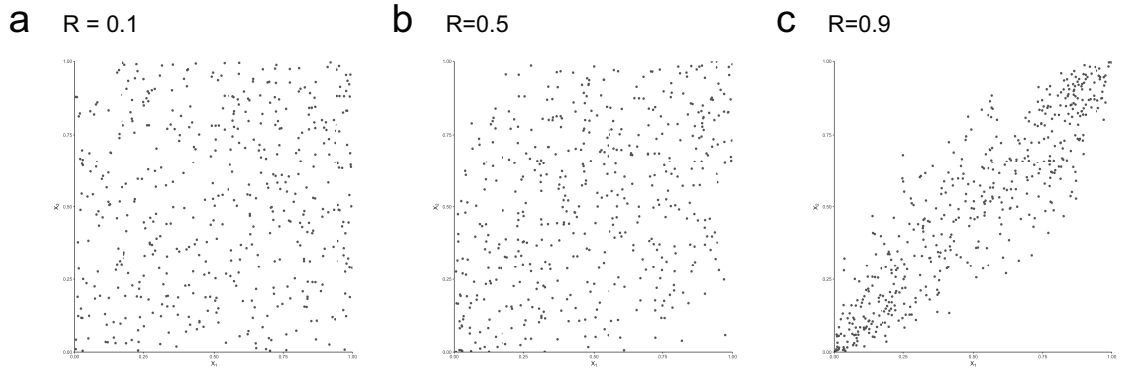



Figure 2–2: Comparison of 500 samples from Gaussian copulas with different correlation parameters.

Definition 2.2.2 has a number of side results, some of which are mentioned below.

**Corollary 2.2.3.1.** *Let $C$ be a $d-$dimensional copula. Then,*

1. $C(u_1, ..., u_{i-1}, 0, u_{i+1}, ..., u_d) = 0,\ i = 1, ..., d;$
2. $C(1, ..., 1, u_i, 1, ..., 1) = u_i,\ i = 1, ..., d;$
3. $\underbrace{\max\left\{1 - d + \sum_{j=1}^{d} u_j, 0\right\}}_{W(u_1,...,u_d)} \leq C(u_1, ..., u_d) \leq \underbrace{\min\left\{u_1, ..., u_d\right\}}_{M(u_1,...,u_d)}$, *where $W$ and $M$ are known as the lower and upper Fréchet-Hoeffding bounds, respectively.*
4. $c(u_1, ..., u_d) = \frac{d^d}{du_1 ... du_d} C(u_1, ..., u_d)$ *is the copula density function (if it exists).*

## 2.3 Likelihood of a copula function's parameters

In this section, we discuss the computation of the likelihood function for the parameters of a copula.

Consider a $d-$variate distribution function $F$ of a random vector $X = (X_1, ..., X_d)$ with marginals $\{F_j\}_{j=1}^d$ parameterized by $\psi = \{\psi_j\}_{j=1}^d$ and a copula $C_\theta$ parameterized by $\theta$ using Sklar's theorem, that is $F = C_\theta(F_1, ..., F_d)$. Furthermore, if $F$ is continuous

with a density function $f$, then

$$f(x_1, ..., x_d) = c_\theta(F_1(x_1), ..., F_d(x_d)) \prod_{j=1}^{d} f_j(x_j) \tag{2.6}$$

and the log-likelihood of the marginal and copula parameters at the point $(x_1, ..., x_d)$ will be of the form

$$\ell_1(\psi, \theta) = \sum_{j=1}^{d} \log f_j(x_j) + \log c_\theta(F_1(x_1), ..., F_d(x_d)) \tag{2.7}$$

In the case when the marginal distribution functions have unknown parameters, we want to estimate both the copula and the marginal parameters from a sample using the maximum likelihood principle. Consider a random sample $X \in \mathbb{R}^{n \times d}$ from the distribution function $F$. To estimate the marginal distribution functions $F_1, ..., F_j$, we can use the approach from Joe (1997) with sequential maximization of the likelihood with respect to $\psi$ and $\theta$. Using (2.7) together with the method proposed by Genest et al. (1998), the maximization can be done sequentially. In the first step, the marginal parameters $\psi$ are estimated by maximizing the marginal likelihoods. In the second step, the copula parameter $\theta$ is estimated.

The non-parametric estimator known as the empirical cumulative distribution function (eCDF) can be used to estimate the marginals:

$$\hat{F}_j(x) = \frac{1}{n+1} \sum_{i=1}^{n} \mathbb{I}[X_{ij} \leq x] \tag{2.8}$$

where $\mathbb{I}$ is the indicator function. Using the eCDF has the advantage of not requiring the estimation of any marginal parameter. The copula parameter can the be

estimated by maximizing the pseudo log-likelihood

$$\ell_n(\theta) = \log c_\theta(\hat{F}_1(x_1), ..., \hat{F}_d(x_d)) \ . \tag{2.9}$$

To estimate the copula parameter $\theta$, we must solve the following optimization problem:

$$\hat{\theta} = \max_\theta \ell_n(\theta) \tag{2.10}$$

which can be found through traditional optimization algorithms such as Newton-Raphson.

To illustrate the approach, suppose that we have samples from a $d-$variate Gaussian copula model with unknown marginals $F_1, ..., F_d$. The copula is assumed to be parameterized by the correlation matrix $R \in \mathbb{R}^{d \times d}$. We can estimate $F_j$ with the non-parametric eCDF $\hat{F}_j$ for $j = 1, ..., d$. We will be left with the following likelihood function

$$\ell_n(R) = \sum_{i=1}^{n} \log c_R(\hat{u}_{i1}, ..., \hat{u}_{id}) \ , \tag{2.11}$$

with $\hat{u}_{ij} = \hat{F}_j(x_{ij}),\ 1 \leq i \leq n$ and $1 \leq j \leq d$. The likelihood function $\ell_n(R)$ for the parameter $R$ can be maximized to yield a maximum likelihood estimate of $R$.

### 2.4 Discrete marginals

Suppose we wish to use the probability integral transform on a discrete random variable $X$ with cdf $F$. That is, we want to compute $Y = F(X)$. Denote the left limit of $F$ as $F(x^-) = \lim_{x \uparrow t} F(t)$. By definition of a discrete CDF, $P[X = x] = P[x^- \leq X \leq x] = F(x) - F(x^-) > 0$ for at least one $x$. While using a copula model to link discrete marginals still makes sense (i.e. $F = C(F_1, ..., F_d)$ is a valid distribution

function), the copula of $F$ in this case is not unique. The non-uniqueness of the copula of $F$ has been discussed by Genest and Nešlehová (2007) who have shown that this issue invalidates inference procedures developed for copula models with continuous marginal distribution functions.

Consider a copula model for $F$, i.e. for a parametric copula family $C \in \{C_\theta : \theta \in \Theta\}$ and distributional functions $F_j \in \{F_{\psi_j} : \psi_j \in \Psi\}, 1 \leq j \leq d$ belonging to some discrete parametric family, we investigate the properties of $F = C(F_1, ..., F_d)$. A sequential approach can be used, where we first estimate the marginal and then the copula parameters. Let's take as an example a bivariate discrete dataset, a random sample from $\{X_1, X_2\}$ with a joint distribution function $F$ and unknown marginal CDFs $F_1$ and $F_2$, respectively. We model the dependence between $X_1$ and $X_2$ with a copula $C$ with parameter $\theta$. Note that after estimating the true (unknown) marginal distribution functions with their empirical CDFs $\hat{F}_1$ and $\hat{F}_2$, the log likelihood of the sample can be rearranged as:

$$\begin{aligned}
\ell_n(\theta) &= \sum_{i=1}^{n} \log P[X_1 = x_{i1}, X_2 = x_{i2}|\theta] \\
&= \sum_{i=1}^{n} \log P[x_{i1} < X_1 \leq x_{i1}, x_{i2} < X_2 \leq x_{i2}|\theta] \\
&= \sum_{i=1}^{n} \log \Big( F(x_{i1}, x_{i2}) - F(x_{i1}^-, x_{i2}) - F(x_{i1}, x_{i2}^-) + F(x_{i1}^-, x_{i2}^-) \Big) \qquad (2.12) \\
&= \sum_{i=1}^{n} \log \Big( C(F_1(x_{i1}), F_2(x_{i2})) - C(F_1(x_{i1}^-), F_2(x_{i2})) \\
&\quad - C(F_1(x_{i1}), F_2(x_{i2}^-)) + C(F_1(x_{i1}^-), F_2(x_{i2}^-)) \Big)
\end{aligned}$$

We can then conduct rank-based estimation by replacing $F_1$ with $\hat{F}_1$ and $F_2$ with $\hat{F}_2$ in (2.12), respectively. The properties of this rank-based estimator when $F_1, F_2$ are discrete were investigated in Ery (2016) but the optimization becomes cumbersome for $d > 2$. Moreover, if the copula does not have a closed form, evaluating the expression in (2.12) can lead to negative arguments in the logarithm function and thus numerical instability.

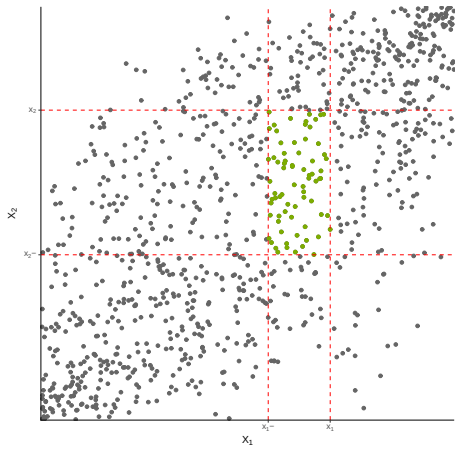



Figure 2–3: Illustration to Eq. 2.12 representing the volume of the copula $C$ contained between the left and right limits of $X_1$ and $X_2$.

Another treatment of discrete marginals often proposed in the literature (Trivedi and Zimmer, 2006; Popovic et al., 2018) is *jittering* binned observations with additive standard uniform noise. Formally, for any integer-valued random variable $X$ with distribution function $F$, we define $\tilde{X} = g(X, \varepsilon)$, where $\varepsilon \sim f_\varepsilon$, $g$ is often picked to be $g(x, y) = x + y$ and $\varepsilon \sim \mathrm{U}(0, 1)$. Jittering is a common technique of removing ties and stands behind the introduction of randomized quantile (also known as Dunn-Smyth) residuals (Chambers, John M and Cleveland, William S and Kleiner,

Beat and Tukey, 2017; Dunn and Smyth, 1996; Nešlehová, 2007). By uniformly selecting points in each bin $(F(x_i^-), F(x_i)], 1 \leq i \leq n$, ties in the discrete observations $x_i$ are broken randomly. Such an approach, however, fails to preserve the copula structure, as seen in Fig 2–4. Two random variables $X_1 \sim$ Bernoulli(0.3) and $X_2 \sim$ Bernoulli(0.6) are generated from a Gaussian copula with parameter $\rho = 0.7$. Samples from the original copula are shown on the left, while samples from $(\tilde{F}_1(\tilde{X}_1), \tilde{F}_2(\tilde{X}_2))$ are presented on the right. The green area corresponds to the joint probability $P[F_1(X_1) \leq 0.7, F_2(X_2) \leq 0.4]$. We can easily see that the introduction of uniform noise breaks ties but at the same time has impact on the underlying dependence structure.

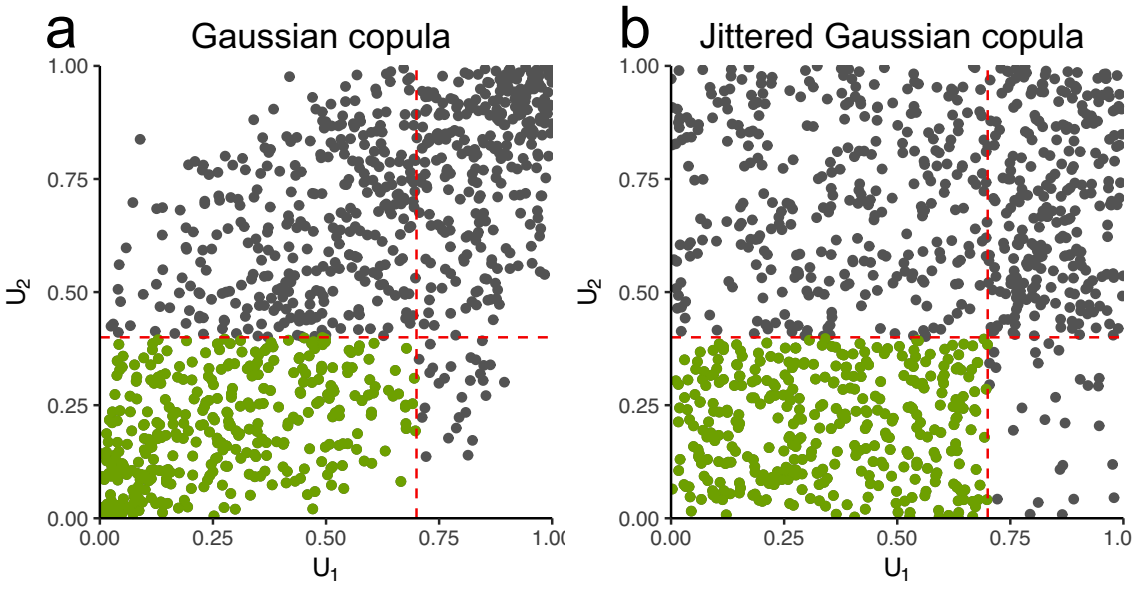



Figure 2–4: Comparison of samples from **a)** the true Gaussian copula and **b)** Gaussian copula obtained through jittering Bernoulli observations.

The copula $C$ is not unique in the discrete case. On the other hand, the copula of the jittered data is one admissible copula, the so-called multilinear (checkerboard) copula (Li et al., 1997). Since the checkerboard copula is not the copula $C_\theta$ that we are typically interested in, maximizing (2.11) leads to bias, as explained and illustrated in Genest and Nešlehová (2007).

# Chapter 3
# Methods

## 3.1 Copulas and probabilistic graphical models

In machine learning, one is often concerned with computing the marginal and conditional densities in probabilistic graphical models, estimating the parameters (e.g. if the random variable $X$ indexed by vertex $v$ has a parametric distribution $f_\theta$, estimate $\theta$) as well as model selection (structural learning).

Structural learning consists in estimating the connectivity of an undirected graph $\mathcal{G} = (V, E)$ with vertices indexing random variables grouped in the vector $X = (X_v, v \in V)$ (Elidan, 2013). For each pair $(X_{v_i}, X_{v_j})$, we would like to know whether $(v_i, v_j)$ is an edge in the graph.

The most widely used model in structure learning is the multivariate Gaussian distribution which gives rise to Gaussian Markov fields. The reason for its popularity is due to the fact that if $(X, Y)$ are jointly distributed as a bivariate Gaussian, then $X \perp Y \iff \text{Corr}(X, Y) = 0$. More generally, $X_{v_i} \perp X_{v_j} | X_{V \setminus \{v_i, v_j\}} \iff \Omega_{ij} = 0$, where $\Sigma_{ij} = \text{Cov}(X_{v_i}, X_{v_j})$ and $\Omega = \Sigma^{-1}$ is the inverse of the covariance matrix. $\Omega$ is also known as the *precision* matrix and measures how tightly the data is grouped around the mean vector.

To gain an intuition about the problem, consider three random variables $X, Y, Z$ which are jointly Gaussian with precision matrix

$$\Omega = \begin{array}{c} \\ x \\ y \\ z \end{array} \begin{array}{c} \begin{array}{ccc} x & y & z \end{array} \\ \begin{pmatrix} \omega_{xx} & \omega_{xy} & \omega_{xz} \\ \omega_{yx} & \omega_{yy} & 0 \\ \omega_{zx} & 0 & \omega_{zz} \end{pmatrix} \end{array} \tag{3.1}$$

The graph encoding this dependency structure is presented below:

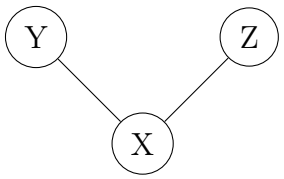



Figure 3–1: The graph representing a multivariate Gaussian distribution with precision matrix $\Omega$ in (3.1).

The structure depicted in Figure 3–1 encodes the conditional independence $Y \perp Z|X$. It can be seen that estimating the undirected graph above boils down to estimating $\Omega$. This can be done by inversion of the covariance matrix $\Sigma$ for $n$ samples of a random vector $X$. However, when $d > n$, the matrix $\Sigma$ is not of full rank and hence not invertible.

Generally speaking, the Gaussian copula can be abstractly thought of as the nonparanormal distribution (Liu et al., 2009).

**Definition 3.1.1.** *Let $Z = (Z_1, ..., Z_d)$ be jointly distributed as $\mathcal{N}(0, \Sigma)$. The random vector $X = (X_1, ..., X_d)$ with margins $F_1, ..., F_d$ is said to follow a nonparanormal or*

*Gaussian copula model if there exists a set of monotonically increasing transformations* $g = \{g_j\}_{j=1}^{d}$ *such that* $X_j = g_j(Z_j)$ *for* $1 \leq j \leq d$.

In fact, choosing $g_j = F_j^{-1} \circ \Phi, 1 \leq j \leq d$ makes the nonparanormal distribution follow a Gaussian copula model. The conditional independence property of Gaussian graphical models is conserved through the transformation $g_j = F_j^{-1} \circ \Phi, 1 \leq j \leq d$ for continuous random variables $X_1, ..., X_d$ with distribution functions $F_1, ..., F_d$.

The *glasso* method (Friedman et al., 2008) provides a way to estimate the precision matrix through $L_1$ penalization of the likelihood function.

Consider the vector $X = (X_1, ..., X_n)$ jointly distributed as a multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$. If $\mu$ is known or estimated with an unbiased estimator $\hat{\mu}$, the likelihood for $\Omega$ will be

$$L(\Sigma^{-1}) \propto \log|\Sigma^{-1}| - \mathrm{Tr}((\hat{S}\Sigma^{-1})), \tag{3.2}$$

where $\hat{S}$ is the sample covariance matrix. Taking $\Omega = \Sigma^{-1}$ , we maximize the log-likelihood subject to $L_1$ penalization given by

$$f(\Omega) = \log|\Omega| - \mathrm{Tr}(\hat{S}\Omega) - \lambda||\Omega||_1, \tag{3.3}$$

for some positive definite matrix $\Omega$ and Lagrangian multiplier $\lambda$. Using the primal-dual formulation, the graphical lasso can be switched to minimization of an analogous objective function over all positive definite arguments. The main reason behind solving the dual over the primal is computational efficiency of modern optimizers, for which minimization is the standard form of an optimization problem.

The *glasso* method starts with the following matrices:

$$\Omega = \begin{bmatrix} \Omega_{11} & \omega_{12} \\ \omega_{21} & \omega_{22} \end{bmatrix}, \quad \hat{S} = \begin{bmatrix} S_{11} & s_{12} \\ s_{21} & s_{22} \end{bmatrix}, \quad W = \begin{bmatrix} W_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix}, \tag{3.4}$$

where $\Omega = \Sigma^{-1}$ and $W = \Omega^{-1}$.

We proceed as follows:

1. Take $W = \hat{S} + \lambda I$
2. For each row and column:
   (a) Solve the following quadratic programming problem:

$$\hat{\beta} = \min_{\beta} \left[ \frac{1}{2} \beta^T W_{11} \beta + \beta^T s_{12} + \lambda ||\beta||_1 \right], \tag{3.5}$$

   using results from previous iterations as starting points.

   (b) Update the weight $\hat{w}_{12} = -W_{11}\hat{\beta}$.

   (c) Save the coefficient $\hat{\beta}$ for this position.
3. For each column, compute

$$\begin{aligned} \hat{\omega}_{22} =& (\hat{s}_{22} + \lambda - \hat{\beta}^T \hat{w}_{12})^{-1}, \\ \hat{\omega}_{12} =& \hat{\beta}\hat{\omega}_{22} \quad . \end{aligned} \tag{3.6}$$

The above algorithm produces, after convergence, a sparse estimate of the matrix $\Omega$. Higher values of the hyperparameter $\lambda$ lead to weights being driven to zero, and hence to a sparse precision matrix and a disjoint graph.

We know that *glasso* works on variables that are jointly normally distributed. If we have a continuous random vector $(X_1, ..., X_d)$ with respective marginal CDFs

$F_1, ..., F_d$, we can then define the vector $h$ as

$$h_j(x_j) = \Phi_1^{-1}(F_j(x_j)), \tag{3.7}$$

We know by probability and quantile transforms that $h_j(x_j) \sim \mathcal{N}(0,1)$. However, the components of the vector $h(X)$ need *not* to be independent, which is what we use to build the graph. Once again, denote the precision matrix of $h(X)$ by $\Omega$. The conditional independence between continuous random variables holds in the nonparanormal case: $h(X_{v_i}) \perp h(X_{v_j})|h(X_{V\setminus\{v_i,v_j\}}) \iff \Omega_{ij} = 0$ and it then follows that $X_{v_i} \perp X_{v_j}|X_{V\setminus\{v_i,v_j\}} \iff \Omega_{ij} = 0$ (Elidan, 2013).

This property does not hold for a copula model with discrete marginals. To see this, consider a graphical model in which $(X, Y, Z)$ jointly follow a Gaussian copula model with Bernoulli marginals with parameters 0.89, 0.27 and 0.37, respectively, with correlation matrix

$$\Sigma = \begin{array}{c} \\ x \\ y \\ z \end{array} \begin{array}{c} \begin{array}{ccc} x & y & z \end{array} \\ \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \end{array} \tag{3.8}$$

and precision matrix

$$\Omega = \begin{array}{c} \\ x \\ y \\ z \end{array} \begin{array}{c} \begin{array}{ccc} x & y & z \end{array} \\ \begin{pmatrix} -1 & 1 & 1 \\ 1 & 0 & -1 \\ 1 & -1 & 0 \end{pmatrix} \end{array} \tag{3.9}$$

The joint conditional probability $P[Y = 0, Z = 0|X = 0]$ depends on the copula through the joint $P[X = 0, Y = 0, Z = 0]$, which we calculated to be approximately 0.0845 using bootstrap. If the conditional independence of the latent variables would imply the conditional independence of their respective discrete observations, then the joint should factor as $P[X = 0, Y = 0, Z = 0] = P[X = 0]P[Y = 0|X = 0]P[Z = 0|X = 0] \approx 0.097$. Since the observed joint density does not factorize along the graph the same way that the latent joint density did, we can conclude that this implication is only valid when no ties are present, i.e. the observations are continuous.
Therefore, in order to reconstruct the dependence graph of the random vector $(Z_1, ..., Z_d)$, we need to estimate $\Omega$ by combining (3.7) with rank-based inference in the proposed approaches.

Because the graphical lasso is meant to work on multivariate Gaussian distributions, any monotone increasing transformations of continuous data will yield correct estimates of $\Omega$. The algorithm has an implicit bias when dealing with discrete data which cannot be mapped to the multivariate Gaussian. It is still possible to use the glasso on either the original observed data or the latent variables. In this case, however, the estimated precision matrices will tend to be biased.

### 3.2 Reinforcement learning

Various tasks in statistical learning require finding an optimal behaviour in a given setting or, in other words, solving the reinforcement learning (known in statistical literature as decision theory) problem. A considerable part of this work relies on multi-armed bandits and is hence closely related to stochastic optimal control. The problem is posed as follows.

Let ($S$,$A$,$R$,$P$,$s_0$,$\gamma$) be a 6-tuple representing a *Markov decision process* (MDP) where $S$ is a finite set of states, $A$ a finite set of allowed actions, $R = R(s_t, a_t, s_{t+1}) : S \times A \times S \to \mathbb{R}$ is the (deterministic or stochastic) reward function, $P = P(s_{t+1} = s'|s_t = s, a_t = a), \forall a \in A, s, s' \in S$ are the environment transition probabilities and $s_0 \subseteq S$ is the set of initial states. The system is also known as the reinforcement learning *agent* and starts in one of the initial states $s_0$ at time $t = 0$. In subsequent time steps $t = 1, 2, 3, ...$, the agent acts according to a policy $\pi(a|s) = P[a_t|s_t]$ and moves from state $s_t$ to $s_{t+1}$, collecting a reward $R_{t+1} = R(s_t, a_t, s_{t+1})$. The environment is characterized by its set of initial states in which the agent starts, as well as the Markov transition model which encodes the mechanics of moving from one state to another. The transition probability matrix $P$ is a row stochastic matrix which describes the dynamics of the MDP.

In order to compare states, we introduce the state value function $V : S \to \mathbb{R}$ which gives the expected sum of discounted rewards in that state. Define $Q_\pi(s, a) : S \times A \to \mathbb{R}$ and $V_\pi(s) : S \to \mathbb{R}$ as:

$$\begin{aligned} Q_\pi(s,a) &= \mathbb{E}_{\pi,P}[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a] \\ &= \sum_{s' \in S} \sum_{k=0}^{\infty} \pi(a|s) P[S_{t+k+1} = s' | S_{t+k} = s, A_t = a] \gamma^k R_{t+k+1} \\ V_\pi(s) &= \mathbb{E}_{\pi,P}[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s] \\ &= \sum_{\substack{a \in A \\ s' \in S}} \sum_{k=0}^{\infty} \pi(a|s) P[S_{t+k+1} = s' | S_{t+k} = s, A_t = a] \gamma^k R_{t+k+1} \end{aligned} \tag{3.10}$$

The reinforcement learning problem is two-fold: (1) given a fixed policy $\pi$ we would like to obtain the correct state value function given by $V_\pi(s), \forall s \in S$ and (2) we wish to find the optimal policy $\pi^*$ which yields the highest $V(s)$ for all states of the MDP or, equivalently, $\pi^*(s) = \max_\pi V_\pi(s)$. The first task is known in the reinforcement learning literature as prediction and the second as control.

In order to find the value function for each state, we need to solve the Bellman equations (Bellman, 1952):

$$\begin{aligned} V_\pi(s) &= \mathbb{E}_{\pi,P}[R(s_{t+1}, \pi(a_{t+1}|s_{t+1})) + \gamma V_\pi(s_{t+1})] \text{ or} \\ V_\pi &= R + \gamma P V_\pi \ , \end{aligned} \tag{3.11}$$

for $V_\pi = \{V_\pi(s_1), ..., V_\pi(s_t)\}$ and $R = \{R(s_1, \pi(a_1|s_1)), ..., R(s_t, \pi(a_t|s_t))\}$. We can rewrite (3.11) as:

$$\begin{aligned} Q_\pi(s,a) &= \mathbb{E}_{\pi,P}[R(s_{t+1}, a_{t+1}) + \gamma Q_\pi(s_{t+1}, a)] \text{ or} \\ Q_\pi &= R + \gamma P Q_\pi \ , \end{aligned} \tag{3.12}$$

for $Q_\pi = \{Q_\pi(s_1, a_1), ..., Q_\pi(s_t, a_t)\}$. Here, we used the fact that if $G_t = R_t + \gamma R_{t+1} +$ ... at state $s_t$, then $G_t = R_t + \gamma G_{t+1}$ and $\pi, \mathbb{P}[G_t] = V_\pi(s_t)$. The Bellman equations then follow by applying the expectation operator on both sides.

Both (3.11) and (3.12) have the following direct solution obtained by matrix inversion: $V_\pi = (I - \gamma P)^{-1} R$. However it should be noted that this is only well-defined for finite-state MDPs, and further as this computation has $O(n^{2.4})$ complexity (Coppersmith and Winograd, 1987), it is only computationally feasible for small MDPs. Therefore, sample-based algorithms such as Temporal Difference (TD) (Sutton, 1988)

for prediction and SARSA or Q-learning (Rummery and Niranjan, 1994; Watkins and Dayan, 1992) for control are preferred for more general classes of environments.

One particular instance of MDPs with no explicit states is called a *multi-armed bandit* (MAB). The analogy is borrowed from the setting of the well-known casino machine: an agent has to choose between $K$ levers, each of which, when activated, returns a sample from its reward distribution. The prediction problem for MABs consists in learning a $Q$ value for each action (or lever) in order to approximate the reward distribution. In the MAB setting, the $Q$ value function for an arm $a$ pulled $n$ times is simply $Q_n(a) = \frac{1}{n}\sum_{t=1}^{n} R(a_t)$, that is the average of all observed rewards for that arm. Note that $R(a)$ is a distribution which can be degenerate at one point but does not have to be.

### 3.3 Variational inference

The most traditional and well-known Bayesian inference method for parameter estimation consists in sampling from the posterior distribution of a set of parameters conditioned on the evidence (i.e. observed data) and is known as *Markov Chain Monte Carlo*. Formally, suppose that we observe a sample $X_1, ..., X_n$ of independent and identically distributed $d-$dimensional random variables with probability density (mass for discrete) function $f_X$. We use continuous variables for convenience but the same logic applies to discrete data. If $f_X$ is parameterized by a parameter vector $\theta$, then $f_X \equiv f_X(x|\theta) = \lim_{dx\to 0} P[X \in (x, x+dx)|\theta]$ are equivalent. We refer to the joint distribution of the data for a given set of parameters as the *likelihood* and denote it as $\ell_n(\theta) = \prod_{i=1}^{n} f_X(x_i|\theta)$. We adopt notation used in Bayesian literature and use $P[X = x] = f_X(x)$ interchangeably. Recalling Bayes' rule, we have the following

expression which can be used to update the probability of observing a parameter vector $p(\theta|X_1 = x_1, ..., X_n = x_n) = p(\theta|x)$:

$$p(\theta|x) = \frac{\prod_{i=1}^{n} f_X(x_i|\theta)p(\theta)}{f_X(x)} = \frac{\ell_n(\theta)p(\theta)}{\int_{\tilde{\theta}} \ell_n(\tilde{\theta})p(\tilde{\theta})d\tilde{\theta}} \propto \ell_n(\theta)p(\theta) \ . \tag{3.13}$$

( 3.13) shows the *posterior* distribution $p(\theta|x)$ proportional to the product of the likelihood $\ell_n(\theta)$ and the parameter prior $p(\theta)$. Taking $p(\theta) = \delta_{\theta^*}$ at the true value $\theta^*$ recovers the frequentist hypothesis. The advantage of Bayesian methods consists in representing parameter uncertainty through updating the initial belief (i.e. prior) with the likelihood. As opposed to maximum likelihood estimation where one solves the optimization problem $\max_\theta \ell_n(\theta)$, Bayesian inference aims to obtain samples from the posterior distribution.

Sampling algorithms such as Gibbs or Metropolis-Hastings (Gelman, 2013) (outlined in Alg. 1) rely on Markov Chain Monte Carlo (MCMC) to produce sequences of independent draws from $p(\theta|x)$. They have long been considered as the traditional Bayesian methods and extensively studied throughout the statistical literature. One major downside of Metropolis-Hastings (from now on we will deal with Metropolis-Hastings since Gibbs sampling is a special subcase) is the time complexity: in order to achieve independent draws from the Markov chain, the sampler has to discard everything obtained in between consecutive accepted points $X_t$ and $X_{t+h}$ for $h \gg 0$. The parameter $h$ has to be sufficiently large to guarantee independence. Moreover, the Markov chain has to be run long enough in order to achieve the equilibrium (also known as stationary) distribution. Because during this *burn-in* period no samples are retrieved, recent work in the field focuses on improving MCMC convergence as

a mean to increase the algorithms' efficiency (Larjo and Lähdesmäki, 2015; Franzke and Kosko, 2015).

**Algorithm 1:** Metropolis-Hastings

**Input :** Observed values $x_1, ..., x_n$, iterations number $T$, proposal distribution $J_t(\theta^*|\theta^{(t-1)})$;

**Output:** Samples from the posterior $p(\theta|x)$;

Initialize $\theta^{(0)}$ s.t. $p(\theta^{(0)}|x) > 0$;

**for** $t = 1, 2, .., T$ **do**

  Sample $\theta^* \sim J_t(\theta^*|\theta^{(t-1)})$;

  $r \leftarrow \frac{p(\theta^*|x)}{p(\theta^{(t-1)}|x)}$;

  Sample $u \sim$Uniform(0,1);

  **if** $u < \min(r, 1)$ **then**

    $\theta^{(t)} \leftarrow \theta^*$;

  **else**

    $\theta^{(t)} \leftarrow \theta^{(t-1)}$;

  **end**

**end**

Because they do not make use of gradient updates, MCMC methods highly depend on the right choice of the proposal distribution $J_t$ in order to converge. It is, however, possible to make use of gradient descent coupled with approximate inference to achieve a much faster convergence. This idea of approximate Bayesian methods is known as *variational inference*.

In order to properly define variational methods, we first have to examine the concept of *divergence*.

**Definition 3.3.1.** *Let $S$ be the space of all probability density functions with some common support $R_X$. A divergence $D$ is a mapping $D : S \times S \to \mathbb{R}$ with the property that for any two probability densities $f, g \in S$, $D(f||g) \geq 0$ and $D(f||g) = 0 \iff f = g$ almost surely.*

Divergences can but need not to be symmetric and do not have to satisfy the triangle inequality. For this reason, they are usually seen as a weaker version of a distance. Definition 3.3.1 presents a very broad concept of divergence. Instead, we will focus on a commonly used divergence known as *Kullback-Leibler* (KL)(Kullback and Leibler, 1951). For $g, f \in S$,

$$D_{KL}(f||g) = \int_{x \in R_X} f(x) \log \frac{f(x)}{g(x)} dx = \mathop{\mathbb{E}}_{f}[\log f(X) - \log g(X)] \,. \tag{3.14}$$

One can easily show that the KL divergence is not symmetric: consider $X \in \{0, 1\}$, $f(0) = 0.5, f(1) = 0.5$ and $g(0) = 0.1, g(1) = 0.9$. Then, $D(f||g) = 0.5 \log 5 + 0.5 \log \frac{5}{9} = 0.51$ while $D(g||f) = -0.1 \log 5 + 0.9 \log \frac{9}{5} = 0.36$ which concludes the counterexample. Additionally, the KL divergence can be seen as the continuous extension to entropy, also called relative entropy. A closed-form expression for $D_{KL}(f||g)$ where $f \sim \mathcal{N}(\mu_1, \Sigma_1)$ and $g \sim \mathcal{N}(\mu_2, \Sigma_2)$ exists, which greatly improves gradient computations and is presented below.

$$D_{KL}(f||g) = \frac{1}{2}\left( \mathrm{Tr}(\Sigma_2^{-1}\Sigma_1) + (\mu_2 - \mu_1)^T \Sigma_2^{-1}(\mu_2 - \mu_1) - d + \log\left(\frac{\det(\Sigma_2)}{\det(\Sigma_1)}\right)\right), \tag{3.15}$$

where the trace operator is denoted $\mathrm{Tr}(A)$ and returns the sum of the main diagonal entries of the matrix $A$ and $d$ is the rank of $\Sigma_1, \Sigma_2$.

Variational Bayes methods rely on divergence minimization, be it the KL divergence

or any other metric with desirable properties. Renyi, $\alpha, \beta$ and $f-$divergences are borrowed from statistical physics and have become increasingly popular in machine learning applications (Li and Turner, 2016; Hernández-Lobato et al., 2016; Nowozin et al., 2016). A widely used byproduct of the KL divergence is known as the reverse KL divergence and is defined in (3.16).

$$D_{\underline{KL}} = \mathbb{E}_f[\log g(X) - \log f(X)] \tag{3.16}$$

To distinguish the KL divergence from the reverse KL divergence, we call the former the forward KL divergence. Fig. 3–2 illustrates the different sets of parameters obtained when minimizing KL (a) and reverse KL (b) divergences. We see that

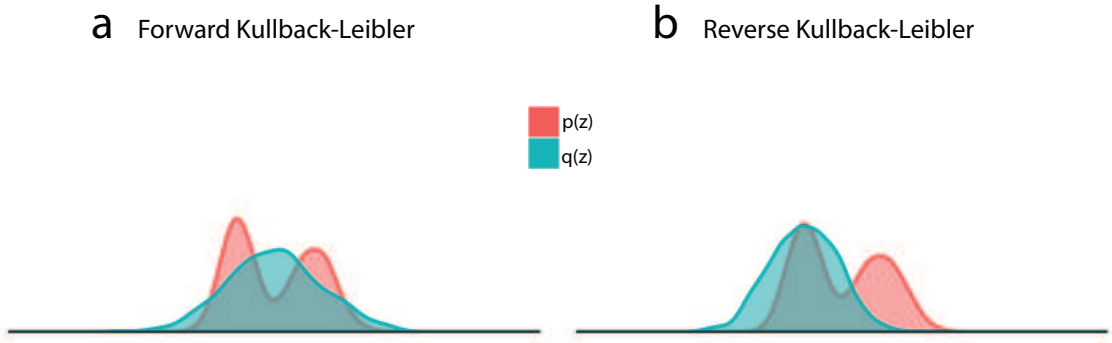



Figure 3–2: Fitting a unimodal Gaussian distribution to a mixture of two Gaussians has a different behaviour if using **a)** Kullback-Leibler or **b)** reverse Kullback-Leibler.

the forward KL divergence approximates both modes of the two-Gaussians mixture, while the reverse KL divergence focuses on a single mode.

In order to derive the classical variational Bayes algorithm, we first recall Jensen's inequality(Casella and Berger, 2002), presented below.

**Lemma 3.3.2.** *(Jensen's inequality) Let $f : \mathbb{R} \to \mathbb{R}$ be a convex function. If $X$ is a random variable with $\mathbb{E}_X[X] < \infty$, then $f(\mathbb{E}_X[X]) \leq \mathbb{E}_X[f(X)]$. Conversely, if $f$ is concave, then $f(\mathbb{E}_X[X]) \geq \mathbb{E}_X[f(X)]$.*

Recall that a real function $f$ is called convex if for any $x_1, x_2 \in A$ for a convex set $A \subseteq \mathbb{R}$ and any $r \in (0, 1)$, $f(rx_1 + (1 - r)x_2) \leq rf(x_1) + (1 - r)f(x_2)$. The function $f$ is concave if $-f$ is convex. Suppose that we are interested in maximizing the data probability $p(x)$. Following the convention in statistical literature, we can maximize the log evidence $\log p(x)$. Taking a latent random variable $z$ defined over a sample space $\mathcal{Z}$ with a *variational* density $q$, we obtain:

$$\begin{aligned}
\log p(x) &= \log \int_{z \in \mathcal{Z}} p(x, z) dz \\
&= \log \int_{z \in \mathcal{Z}} q(z) \frac{p(x, z)}{q(z)} dz \\
&= \log \mathbb{E}_q \left[ \frac{p(x, Z)}{q(Z)} \right] \\
&\geq \mathbb{E}_q[\log p(x, Z) - \log q(Z)] \\
&= \mathbb{E}_q[\log p(x|Z) + \log p(Z) - \log q(Z)] \\
&= \mathbb{E}_q[\log p(x|Z)] - D_{KL}(q||p)) \\
&= -\mathcal{L}(x)
\end{aligned} \tag{3.17}$$

The final expression for $\mathcal{L}$ in (3.17) is known as the *evidence lower bound* or ELBO. All derivations of the ELBO require the use of Jensen's inequality as a means to bound a random variable's distribution by its expectation.

The lower bound depends on a new density $q \equiv q_\phi$ which is parameterized by a new set of parameters $\phi$. When choosing the family of $q$, it is crucial to pick a tractable

(i.e. easy to sample from) distribution.

Since $\log p(x) \geq -\mathcal{L}(x)$, maximizing the ELBO or equivalently minimizing $\mathcal{L}$ should maximize the log probability of the data. Minimization of the objective function should be done with respect to the new parameters of the density $q$. The major advantage of variational inference (VI) methods over MCMC is faster convergence. Indeed, the lower bound is a differentiable function which can be used to find the optimal set of parameters $\phi$ and does not require burn-in time. The variational inference task then consists in finding a $q \in Q$, for $Q$ being the class of tractable parametric distributions, which minimizes the ELBO. Because $q_\phi$ comes from a parametric family, it is sufficient to find the optimal set of parameters $\phi$ with respect to $\mathcal{L}$. However, due to a number of reasons such as long computation time, it has been suggested to further factor the variational density $q$ into the product of independent densities parameterized by subgroups of latent variables. Let $z_j$ denote the $j^{\text{th}}$ latent variable subset such that $z = (z_1, ..., z_m)$. Note that each $z_j$ can but does not have to be a single variable. Then the variational density $q_\phi$ can be decomposed for any $z \in \mathcal{Z}$:

$$q_\phi(z) = \prod_{j=1}^{m} q_{\phi_j}(z_j) \tag{3.18}$$

This simplification (3.18) is known in Bayesian literature as the *mean-field variational family* (Blei et al., 2017) and allows to greatly speed up the optimization process by assuming independence between each subgroup of latent variables $z_j$.

It can be shown using the calculus of variations that the logarithm of variational distribution over a subset of latent variables $\log q_{\phi_j}(z_j)$ must be proportional to $\mathbb{E}_{z_{i\neq j}}[\log p(z_j | z_{i\neq j}, x)]$, where $z_{i\neq j} = (z_1, .., z_{j-1}, z_{j+1}, .., z_m)$ in order to minimize

$D_{\underline{KL}}(q(z_j)||p(z_j|z_{i\neq j}, x))$. Therefore, one can select the optimal form of the variational distribution $q$ in order to minimize the reverse Kullback-Leibler divergence.

**Variational auto-encoding networks**

With the growth of computational powers of personal devices, even a low-end smartphone can carry out a complicated simulation such as MCMC or multivariate regression. In particular, the development of Graphical Processing Units (GPUs) allowed for a surge in performance across high-dimensional classification and regression tasks on complex data types such as images, texts, audio and even video files. This idea of letting the statistical model select the relevant covariates (or features) is known as deep learning. Deep learning makes use of artificial neural networks (NNs) to approximate any real-valued function. For this reason, neural nets are sometimes called universal function approximators.

Let $X = \{X_1, ..., X_n\}$ be a random sample from some true data distribution $f_X$ of dimension $d$. Moreover, a column vector of responses $y = \{y_1, ..., y_n\}^T$ is associated with $X$. The classical regression setting assumes that the responses are a linear combination of covariates:

$$g(Y) = \beta_0 + \beta_1 X_1 + .. + \beta_d X_d + \varepsilon, \tag{3.19}$$

for column vectors $X_1, .., X_d$, link function $g$, coefficient vector $\beta = \{\beta_0, .., \beta_d\}$ and random independent noise $\varepsilon$ such that $\mathbb{E}[\varepsilon] = 0$.

If we follow best practices in the literature, we reformulate (3.19) as

$$Y = f(w^T X), \tag{3.20}$$

where we take $f = g^{-1}$ and $w = \beta$. The quantity $w^T x_i$ defines the coordinate of the point $p_i = (\hat{y}_i, x_i)$ in $\mathbb{R}^d$ which lies on the hyperplane $w^T X$. The distance $d(y_i, \hat{y}_i)$ provides an error signal of how far away the predicted value $\hat{y}_i$ lies from the true response $y_i$. Choosing $d$ to be the Euclidean distance corresponds to the traditional *mean squared error* minimization and can be solved iteratively. Recall Newton's method for root finding with updates:

$$\beta^{(t)} = \beta^{(t-1)} - \frac{f(\beta^{(t-1)})}{f'(\beta^{(t-1)})} \tag{3.21}$$

for first-order Newton methods and

$$\beta^{(t)} = \beta^{(t-1)} - \frac{f'(\beta^{(t-1)})}{f''(\beta^{(t-1)})} \tag{3.22}$$

for second-order methods. (3.22) is derived by second order Taylor series expansion of the objective function $f(\beta)$.

For example, the mean squared error $L_w(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ is the most commonly used loss function. A *loss function* $L : \mathbb{R} \to \mathbb{R}$ is a real-valued function which indicates the cost associated with observing a certain event or data point. Minimizing the loss function (or error) improves the goodness-of-fit of the model; selecting the most appropriate loss function for either classification or regression tasks can greatly impact both model performance and convergence properties. We denote the loss function with respect to parameters $w$ as $L_w$.

The simplest algorithm for parameter estimation in neural networks is known as *stochastic gradient descent* (SGD) and is presented in Alg. 2.

**Algorithm 2:** Stochastic gradient descent

**Input :** Observed values $x_1, .., x_n$, responses $y_1, .., y_n$ , iterations number $T$, learning rate $\alpha(t)$, loss function $L_w$, batch size $B$;

**Output:** Converged weights $w^{(T)}$;

Initialize $w^{(0)}$ according to some distribution;

**for** $t = 1, 2, .., T$ **do**

  **for** $b = 1, 2, .., n/B$ **do**

    $w^{(t)} \leftarrow w^{(t-1)} - \alpha(t)\frac{1}{B}\sum_{i \in b} \nabla_w L_w(y_i, \hat{y}_i)$;

  **end**

**end**

When $B = 1$, the method is known as *online* SGD and requires the true gradient update values in order for the algorithm to converge to a local optimum.

Using the Hessian matrix $H_w$ instead of the Jacobian (multivariate extension of the gradient vector) leads to the well-known Newton-Raphson iterative procedure. In the case when the vector of reconstructed responses $\hat{y}$ is itself fed into a linear model, the architecture is known as a multilayer neural network. Parameter updates consequently use the chain rule on each respective weight set.

Suppose that we have a multilayer neural network $M : \mathbb{R}^d \to V^K$ with input size $d$ and output size $K$. That is, all observations $X_1, .., X_n$ are of dimension $d$ and all responses $y_1, .., y_n$ are of dimension $K$. When the subspace $V \subseteq \mathbb{R}$, the problem is known as *regression*, when $V \subseteq \mathbb{Z}$ as *classification* and when $V^K = \mathbb{R}^d$ as *reconstruction*. In this work, we focus on the task of reconstructive latent learning. Reconstructive networks are also called *autoencoders* and can be seen as a nonlinear

version of Principal Component Analysis (PCA).

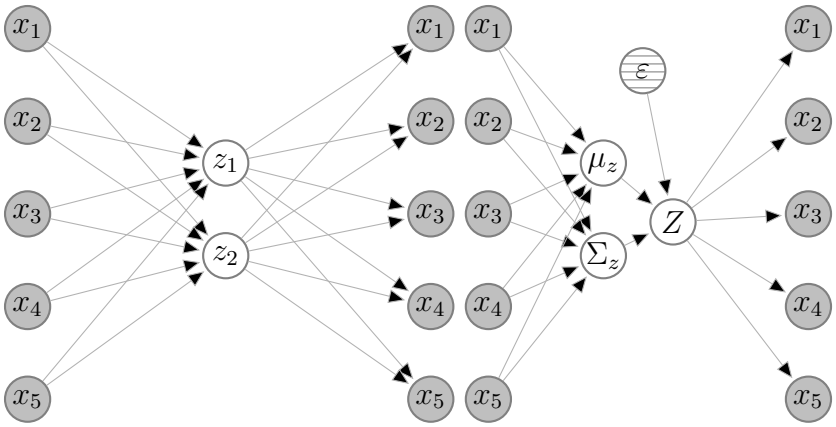



Figure 3–3: Graphical models of the standard autoencoder (left) and variational autoencoder (right) are shown. Note that VAEs incorporate a sampling mechanism within their latent space.

Fig. 3–3 compares architectures of autoencoding networks and variational autoencoding networks (Kingma and Welling, 2013) (VAEs). The most staggering difference between both is given by the shaded node $\varepsilon$; injecting stochastic noise allows for sampling new observations instead of reconstructing the input exactly. VAEs belong to the class of generative models, among with generative adversarial networks (GANs) and restricted Boltzmann machines (RBMs). Generative models map the data points received as input onto a lower-dimensional latent space (in most of the cases, a manifold), which is analogous to PCA, t-SNE and other dimensionality reduction techniques. By explicitly adding stochastic noise as a covariate, it is possible to generate samples from the distribution that approximates the true distribution of the input data while still being able to estimate the gradient with respect to the model's parameters using the score function trick (explained in the next chapter).

# Chapter 4
# Proposed algorithms

## 4.1 Decomposition of the covariance matrix $\Sigma$

Recall that, given a random sample $\mathbf{X}$ of dimension $n \times d$ from some distribution function $F$, our task consists in finding the most likely PGM representing the independence structure of each component. Working with a latent Gaussian copula model parameterized by a $d-$dimensional correlation matrix $R$, we aim to estimate $R$.

The correlation matrix $R$ for a collection of jointly Gaussian distributed random variables $X_1, ..., X_d$ can be defined as $R_{i,j} = \frac{\text{Cov}(X_i, X_j)}{\sqrt{\text{Var}(X_i)\text{Var}(X_j)}}$ or in matrix form:

$$\begin{bmatrix} 1 & R_{1,2} & R_{1,j} & R_{1,d-1} & R_{1,d} \\ R_{2,1} & 1 & & & \\ R_{i,1} & & 1 & & \\ R_{d-1,1} & & & 1 & \\ R_{d,1} & & & & 1 \end{bmatrix} \tag{4.1}$$

Note that $R = R^T$ and is constrained to have values between $-1$ and 1, respectively. In fact, if the observed random variables have unit variance, then $R = \Sigma$. This imposes $d$ additional constraints on the problem but allows to interchange $R$ with $\Sigma$. In all of the proposed methods, we instead work with the covariance matrix $\Sigma$, from

which we can obtain the correlation matrix $R = \left(\sqrt{\text{diag}(\Sigma)}\right)^{-1} \Sigma \left(\sqrt{\text{diag}(\Sigma)}\right)^{-1}$ and the precision matrix $\Omega = \Sigma^{-1}$.

We begin by parameterizing the covariance matrix $\Sigma \equiv u^T u + \exp(a) \circ I_d$ for $u, a \in \mathbb{R}^d$ and identity matrix $I_d \in \mathbb{R}^{d \times d}$. If we continue working with $\Sigma$, there are $\frac{d(d+1)}{2}$ free parameters and $2d$ equations given by the entries of $u$ and $a$. On the other hand, using the correlation matrix $R$ leaves us with $\frac{d^2-d}{2}$ parameters for $2d$ equations. Solving $\frac{d(d-1)}{2} \leq 2d$ shows that for $0 \leq d \leq 5$, the factorized form $u^T u + \exp(a) \circ I_d$ has enough degrees of freedom to represent any matrix.

By sampling both vectors $u$ and $a$ from a prior distribution, we induce a prior distribution over covariance matrices $\Sigma$. Define $s(A)$ as the sum operator applied to every row of the matrix $A \in \mathcal{M}_{d \times d}$ and resulting in a $d-$dimensional column vector. If the components of $u$ and $a$ are sampled independently from distribution functions $F_a$ and $F_u$ with support $R_a$ and $R_u$ respectively, then the matrix $\Sigma$ is diagonally dominant with probability

$$\begin{aligned} P[\exp(a) + \text{diag}(u^T u) \geq s(u^T u - \text{diag}(u^T u))] &= \prod_{i=1}^{d} P[\exp(a_i) + u_i^2 \geq \sum_{j \neq i} |u_i u_j|] \\ &= \prod_{i=1}^{d} P[\exp(a_i) + u_i^2 - \sum_{j \neq i} |u_i u_j| \geq 0] \\ &\geq \prod_{i=1}^{d} P[u_i^2 - \sum_{j \neq i} |u_i u_j| \geq 0] \end{aligned} \tag{4.2}$$

If we assume that the support of $u$ is contained in $(0, \infty)^d$, observe that the random quantity $\Delta_{ij} = u_i^2 - \sum_{j \neq i} |u_i u_j|$ is non-negative with probability

$$\begin{aligned} P[\Delta_{ij} > 0] &= P[u_i^2 > \sum_{j \neq i} u_i u_j] \\ &= P[u_i > \sum_{j \neq i} u_j] \\ &= P[u_i - \sum_{j \neq i} u_j > 0] \\ &:= p \, . \end{aligned} \tag{4.3}$$

It then follows that the probability that a matrix generated in the fashion described above is diagonally dominant is at least $p^d$.

Table 4–1 presents approximate values of $p$ for random variables defined over a positive support. We note that by carefully selecting the prior distribution over $u$, we can obtain a diagonally dominant matrix with larger or smaller probability.

For example, based on the table, a $2 \times 2$ matrix obtained from a $2-$dimensional vector $u$ where each component $u_i \sim \text{Uniform}(0, 1), 1 \leq i \leq 2$ will be diagonally dominant with probability of at least 0.029, as opposed to 0.009 if $u$ comes from a standard normal passed through the absolute value function.

If $\Sigma$ is strictly diagonally dominant, that is, $\sigma_{ii} > \sum_{j \neq i} |\sigma_{ij}| \; \forall i$, then it is non-singular according to Gershgorin circle theorem (Gerschgorin, 1931). It also has the following useful property:

**Lemma 4.1.1.** *Let $A \in \mathcal{M}_{d \times d}$ be a symmetric real-valued matrix. If $A$ is diagonally dominant and diag$(A) > 0$, then $A$ is positive semi-definite.*

| $d$ | Statistics $p_{\text{true}}$ | $p^d$ |
|---|---|---|
| | Gamma($\alpha = 3, \beta = 1$) | |
| 2 | 0.887 | 0.147 |
| 3 | 0.443 | 0.036 |
| 5 | 0.024 | 0.003 |
| 7 | 0 | 0 |
| 10 | 0 | 0 |
| | $\lvert\mathcal{N}(0,1)\rvert$ | |
| 2 | 0.904 | 0.105 |
| 3 | 0.532 | 0.042 |
| 5 | 0.11 | 0.004 |
| 7 | 0.011 | 0 |
| 10 | 0 | 0 |
| | Uniform$(0,1)$ | |
| 2 | 1 | 0.163 |
| 3 | 1 | 0.048 |
| 5 | 0.847 | 0.002 |
| 7 | 0.373 | 0 |
| 10 | 0.033 | 0 |

Table 4–1: Probability $p^d$ of observing the event $\cap_{i=1}^{d}\{\Delta_{ij} \geq 0\}$ for various positive random variables, as well as the true probability that the generated matrix is diagonally dominant ($p_{\text{true}}$).

Thus, we established that the vector decomposition of $\Sigma$ guarantees the generation of positive semi-definite matrices with probability at least $p^d$.

We go further and establish a tighter result.

**Theorem 4.1.2.** *(Vector decomposition of $\Sigma$) Let $a, u \in \mathbb{R}^d : u > 0$, $d > 1$ and $\Sigma = u^T u + \exp(a) \circ I_d$. Then $\Sigma$ is positive definite.*

*Proof.* A square real-valued matrix $\Sigma$ is positive definite if and only if all its eigenvalues are greater than 0. Equivalently, the characteristic equation $p_\Sigma(\lambda) = \det(\Sigma - \lambda I_d) = 0$ has only positive real roots.

Suppose that $\lambda = 0$ is a root of $p_\Sigma(\lambda)$. Then $\det(\Sigma) = 0$ must be satisfied. Using the Minkowski determinant theorem, we see that $\det(\Sigma) = \det(u^T u + \exp(a) \circ I_d) \geq \det(u^T u) + \det(\exp(a) \circ I_d) = \det(\exp(a) \circ I_d) > 0$, where $\det(u^T u) = 0$. To see the later, consider the system of equations $u^T u x = 0$. There exists a non-zero vector $x$ orthogonal to $u$ such that $ux = 0$ and hence $u^T u x = 0$ but $x \neq 0$, which implies that $u^T u$ is singular and hence $\det(u^T u) = 0$ and we are allowed to use Minkowski's theorem (on at least positive semi-definite matrices). On the other hand, $\exp(a) \circ I_d$ is simply the identity matrix scaled by the factor $\exp(a)$ and hence is full rank, implying $\det(\exp(a) \circ I_d) > 0$. Therefore, $\lambda = 0$ cannot be an eigenvalue of the system. Now, suppose that $\lambda < 0$. It follows that $\det(\Sigma + |\lambda| I_d) = 0$ should hold. We use Minkowski's inequality once again to obtain $\det(u^T u + (\exp(a) + |\lambda|) \circ I_d) \geq \det(u^T u) + \det(\exp(a) \circ I_d) + |\lambda|^d \det(I_d) = \det(\exp(a) \circ I_d) + |\lambda|^d > 0$ since $\lambda < 0$. Here, we define the addition between the vector $\exp(a)$ and the scalar $\lambda$ as $\exp(a) + \lambda \equiv \{\exp(a_1) + \lambda, ..., \exp(a_d) + \lambda\}$. Hence, $\lambda < 0$ is not an eigenvalue of $\Sigma$ either and $\Sigma$ is positive definite. □

An alternative proof uses the matrix determinant lemma as follows:

$$\begin{aligned} \det(\Sigma) &= \det(\exp(a) \circ I_d + u^T u) \\ &= \Big(1 + u\big(\exp(-a) \circ I_d\big) u^T\Big) \det(\exp(a) \circ I_d), \end{aligned} \tag{4.4}$$

where $\det(\exp(-a) \circ I_d) = \prod_{i=1}^{d} \exp(-a_i) > 0$ by properties of the exponential function. Hence, for $\det(\Sigma) \neq 0$, we must have $-1 \neq u\big(\exp(-a) \circ I_d\big)u^T$. In fact, note that since $\exp(-a) \circ I_d$ is positive definite, then $u\big(\exp(-a) \circ I_d\big)u^T > 0$ for any real vector $u$. This shows that $u\big(\exp(-a) \circ I_d\big)u^T > 0$ and hence $\det(\Sigma) \neq 0$, which makes the matrix invertible.

We obtain a recipe to randomly sample from the positive definite cone by generating two positive real vectors $a$ and $u$. Note that this parameterization allows to model both positive and negative covariance patterns. For example, taking $u = \{-1, 1\}$ forces $u^T u$ to have off-diagonal entries of $-1$.

The inverse matrix $\Sigma^{-1}$ can be computed using the Sherman-Morrison formula:

$$\begin{aligned} \Sigma^{-1} &= \Big(u^T u + \exp(a) \circ I_d\Big)^{-1} \\ &= \exp(-a) \circ I_d - \frac{\big(\exp(-a) \circ I_d\big)u^T u\big(\exp(-a) \circ I_d\big)}{1 + u \exp(-a) \circ I_d u^T}, \end{aligned} \tag{4.5}$$

where we use $\big(\exp(a) \circ I_d\big)^{-1} = \exp(-a) \circ I_d$. The inversion is well-defined since $\Sigma$ is non-singular as a result of being positive definite.

If we wish to compute the inverse correlation matrix $R^{-1}$, we can use the relation $R = \Big(\sqrt{\text{diag}(\Sigma)}\Big)^{-1} \Sigma \Big(\sqrt{\text{diag}(\Sigma)}\Big)^{-1}$ and hence:

$$\begin{aligned} R^{-1} &= \left[\Big(\sqrt{\text{diag}(\Sigma)}\Big)^{-1} \Sigma \Big(\sqrt{\text{diag}(\Sigma)}\Big)^{-1}\right]^{-1} \\ &= \sqrt{\text{diag}(\Sigma)}\Sigma^{-1}\sqrt{\text{diag}(\Sigma)}, \end{aligned} \tag{4.6}$$

because $\text{diag}(\Sigma)^T = \text{diag}(\Sigma)$.

To conclude this section, note how in order to fit a covariance matrix $\Sigma$ to a dataset, it is sufficient to independently sample two real vectors $a, u$ from some distribution function as starting points of the iterative procedure. Using either stochastic gradient descent (SGD) or second-order optimization methods such as Newton-Raphson, we can then update $a$ and $u$ to be as close to the real $\Sigma^*$ as possible. Finally, the dependence structure between random variables encoded by the precision matrix $\Omega$ can be approximated through $\Sigma^{-1}$ using the shorthand formula presented in (4.5). Note that under the suggested decomposition, $\Sigma$ has $d^2$ learnable entries. The vector $\exp(a)$ can handle the diagonal entries which by properties of variance should be nonnegative, leaving $\frac{d^2-d}{2}$ entries to be learned through a $d-$dimensional real vector $u$. Solving $\frac{d^2-d}{2} \leq d$ we obtain that for $d \leq 3$ the system has infinitely many solutions, meaning that it spans the sets $\mathcal{M}_{1\times1}, \mathcal{M}_{2\times2}$ and $\mathcal{M}_{3\times3}$ intersected with the positive definite cone, while in larger dimensions the decomposition becomes rather restrictive and is equivalent to clipping any $A \in \mathcal{M}_{d\times d}$ to the form $u^T u + \exp(a) \circ I_d$.

## 4.2 Multi-armed bandits for estimation of dependence structure between discrete random variables

In this section, we propose a simple algorithm based on multi-armed bandits borrowed from the reinforcement learning literature.

As argued in the previous section, both vectors $u$ and $a$ give us only $2d$ free parameters to interact with. If we see the system as a $4d$-armed bandit, each entry in $u$ and $a$ can be thought of having an up and a down lever. At timestep $t$, the former will increase the value of that entry by some amount $\xi(t)$ while the later will decrease the value of the entry by the same amount. The task consists in $(a)$ estimating the

value of each lever and ($b$) find the policy which maximizes the expected returns (i.e. expected sum of future rewards). In order to do both at the same time, we make use of the off-policy control method known as Q-learning (Watkins and Dayan, 1992). Algorithm 3 outlines the main steps of the classical Q-learning bandit algorithm.

**Algorithm 3:** $\varepsilon-$greedy discrete MAB

**Input:** Exploration coefficient $\varepsilon \in (0,1)$, vectors $u, a \in \mathbb{R}^d$, matrix of observations $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, set of actions $\mathcal{A}$, multi-armed bandit $\mathcal{B}$;
**Output:** Q-values $Q_u^+, Q_u^-, Q_a^+, Q_a^-$ for all actions;
**for** $a$ *in* $1, .., d$ **do**
  $Q_a^+(a), N_a^+ \leftarrow 0$;
  $Q_a^-(a), N_a^- \leftarrow 0$;
  $Q_u^+(a), N_u^+ \leftarrow 0$;
  $Q_u^-(a), N_u^- \leftarrow 0$;
**end**
**for** $t$ *in* $1, .., \infty$ **do**
  $p_1 \rightarrow \text{Uniform}(0,1)$;
  $A_t \leftarrow \begin{cases} \lfloor \frac{a}{4d} \rfloor, a \sim \text{Uniform}(0,1) & p_1 < \varepsilon \\ \arg\max_a Q_a^+(a) \cup Q_a^-(a) \cup Q_u^+(a) \cup Q_u^-(a) & p_1 \geq \varepsilon \end{cases}$;
  **if** $p_1 \geq \varepsilon$ **then**
    $i \leftarrow \arg\max_Q Q_a^+(a) \cup Q_a^-(a) \cup Q_u^+(a) \cup Q_u^-(a)$;
  **end**
  $R_t \leftarrow \mathcal{B}(A_t, u, a)$,$N(A_t) \leftarrow N(A_t) + 1$;
  $\begin{cases} Q_a^+(A_t) \leftarrow Q_a^+(A_t) + \frac{1}{N(A_t)_a^+}[R_t - Q_a^+(A_t)] & i = 0 \\ Q_a^-(A_t) \leftarrow Q_a^-(A_t) + \frac{1}{N(A_t)_a^-}[R_t - Q_a^-(A_t)] & i = 1 \\ Q_u^+(A_t) \leftarrow Q_u^+(A_t) + \frac{1}{N(A_t)_u^+}[R_t - Q_u^+(A_t)] & i = 2 \\ Q_u^-(A_t) \leftarrow Q_u^-(A_t) + \frac{1}{N(A_t)_u^-}[R_t - Q_u^-(A_t)] & i = 3 \end{cases}$;
**end**

The output of the method is a list of Q-values for each action, as well as converged $u$ and $a$ vectors. We use the $\varepsilon-$greedy policy $\pi_\varepsilon$ which takes optimal actions according to the Q-values in $100(1-\varepsilon)\%$ of cases and takes a random action otherwise.

To be able to use a bandit algorithm, we need a reward signal which provides feedback to the agent. We suggest to use the copula likelihood as the reward function. If we use (2.12), then

$$R_t = \ell_n(R), \tag{4.7}$$

where $R$ is the copula parameter. The two dimensional likelihood formula can be extended to any arbitrary size through the inclusion-exclusion principle. This entails that $Q(a) = \frac{R_1+R_2+\ldots+R_n}{n}$ is the average estimated likelihood.

Here is a short outline of the method:

- Sample an action $a_0$ uniformly;

- Depending on the action, perform one of the following:
  - $u_{a_0} \leftarrow u_{a_0} + \xi(0)$;
  - $u_{a_0} \leftarrow u_{a_0} - \xi(0)$;
  - $a_{a_0} \leftarrow a_{a_0} + \xi(0)$;
  - $a_{a_0} \leftarrow a_{a_0} - \xi(0)$;
- Recompute $\Sigma_{a_0}$ and the corresponding likelihood $\ell_n(R)$;
- The agent receives the likelihood as a reward and updates the corresponding Q-values.

Eventually, the process stabilizes when some actions have much larger values than others, meaning that a local optimum has been reached. Recent advances in curriculum learning (Graves et al., 2017) have suggested that instead of (4.7), the reward

function should be the progress made by the system:

$$R_t = \ell_n(R_{a_t}) - \ell_n(R_{a_{t-1}}), \tag{4.8}$$

i.e. the difference of likelihoods obtained at consecutive timesteps.

The hyperparameter $\varepsilon$ can be tuned by $K-$fold cross-validation but can usually be set to 0.1 to achieve good performance. Alternative exploration strategies such as Upper Confidence Bound (UCB), Softmax Policy and Thompson sampling can be used but for the sake of simplicity, we restrict all further mentions of the MAB algorithm to the $\varepsilon-$greedy policy. The step size $\xi$ can be set to a constant value (e.g. $\xi(t) = 0.1$) or to a Robbins-Monro sequence, where $\sum_{t=0}^{\infty} \xi(t) = \infty$ but $\sum_{t=0}^{\infty} \xi^2(t) < \infty$ to guarantee convergence with probability one.

As a concluding remark, the suggested $\varepsilon-$greedy discrete MAB approach can be seen as a combinatorial optimization method closely related to simulated annealing and genetic algorithms which do not use the gradient of the objective function but prefer to rely on a quality heuristic.

### 4.3 Full-rank variational autoencoder for discrete data

We are interested in modelling the distribution of the data, that is $P[X_{i1} = x_{i1}, ..., X_{id} = x_{id}]$. Suppose that the random vector $X = (X_1, ..., X_d)$ has been generated by the random vector $Z$ which follows a $d-$variate Gaussian distribution. Because $X$ is discrete, its probability mass function can be recovered by truncating the vector $Z$. The latent Gaussian problem consists in estimating the parameters of the multivariate Gaussian based on samples of the vector $X$.

To do so, we assume a latent Gaussian model which is learned by an encoder-decoder stochastic network, also known as auto-encoding variational Bayes and presented in Fig. 4–1.

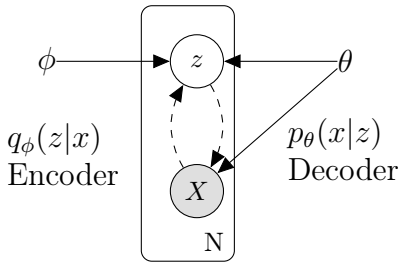



Figure 4–1: Full-rank auto-encoding variational Bayes model. Dashed lines show neural networks, solid show parameterization.

We choose the variational density $q_\phi(z|x) = \mathcal{N}(\mu(x), \Sigma(x))$ as a multivariate Gaussian distribution where parameters $\mu(x) \in \mathbb{R}^d$ and $\Sigma(x) \in \mathbb{R}^{d\times d}, \Sigma(x) \succ 0$ are computed by neural networks as defined in Eq. 4.9. Theorem 4.1.2 guarantees that the covariance matrix will always lie in the positive-definite cone by fitting two vectors $u(x), a(x) \in \mathbb{R}^d$ such that $\Sigma(x) = u(x)^T u(x) + \exp(a(x)) \circ I_d$.

The following equation fully specifies a minimal architecture for a full-rank VAE:

$$\text{Encoder} \begin{cases} \mu(x_i) = f_\mu(w_\mu^T x_i) \\ a(x_i) = f_a(w_a^T x_i) \\ u(x_i) = f_u(w_u^T x_i) \\ \Sigma(x) = u(x_i)^T u(x_i) + \exp(a(x_i)) \circ I_d \\ q_\phi(\tilde{z}|x_i) = \mathcal{N}(\mu(x_i), \Sigma(x_i)) \end{cases} \qquad \text{Decoder} \begin{cases} \theta(z_i) = f_\theta(w_\theta^T z_i) \\ p_\theta(\tilde{x}|z_i) = \delta(z_i - \theta(z_i)) \end{cases} \tag{4.9}$$

Note how the decoder is in fact a deterministic pass through the network and has no sampling involved, unlike the encoder. This rejoins the theory of Popovic et al. (2018) who decomposed an arbitrary probability mass function into a latent Gaussian model $p(x) = \int_{R(z)} p(x,z)dz = \int_{R(z)} p(x|z)dP(z)$ where $p(x|z) = \delta(z \in (z_{i,\min}, z_{i,\max}))$ for $(z_{i,\min}, z_{i,\max}) = \left(\Phi^{-1}(F_X(x_i^-)), \Phi^{-1}(F_X(x_i))\right)$ and $F_X(x_i^-) = \sup_{t<x_i} F_X(t)$ which is exactly our model for $p_\theta(\tilde{x}|z_i)$.

To fit the parameters of interest, we optimize the ELBO with a slight modification to the prior over the latent vector $p(z)$:

$$\mathcal{L} = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)||p(z)), \tag{4.10}$$

where the true unknown conditional densities $p_\theta$ and $q_\phi$ are learned by the encoder and decoder networks. Furthermore, we make use of the empirical Bayes paradigm and set the prior over the latent Gaussian distribution $p(z) \sim \mathcal{N}(\bar{x}_n, s_n)$ where $\bar{x}_n = \frac{1}{n}\sum_{i=1}^n x_i$ estimates the mean vector and $s_n$ estimates the population covariance matrix through the classical estimator $s_n = \frac{1}{n-1}\sum_{i=1}^n (x_i - \bar{x})(x_i - \bar{x})^T$. While Kingma and Welling (2013) originally used a multivariate Gaussian prior with identity covariance $p(z) \sim \mathcal{N}(0_d, I_d)$, the model failed to capture dependence within the latent vector $z$ due to being pushed closer to an independent Gaussian by the KL term. Our prior, on the other hand, allows the VAE to learn a vector decomposition of $\Sigma$ which is always invertible and hence easily estimate the dependence structure of the inputs. Using the sample covariance matrix and performing modifications on it as to make $\Sigma$ invertible is precisely the idea of glasso (Friedman et al., 2008). The major difference between the VAE approach and glasso lies in the regularization:

glasso imposes a $L_1$ regularization on the entries of the matrix, while our algorithm aims to factorize $\Sigma$ into a product of vectors.

In theory, we aim to solve the following optimization problem:

$$\hat{\Sigma} = \max_{\Sigma \in S_d} \mathcal{L}, \tag{4.11}$$

where $S$ is the set of all $d \times d$ real, positive semi-definite symmetric matrices. We instead restrict ourselves to solving

$$\hat{\Sigma} = \max_{\Sigma \in S_d'} \mathcal{L}, \tag{4.12}$$

for $S_d' = \{M : M = u^T u + \exp(a) \circ I_d, (u, a) \in \mathbb{R}^d \times \mathbb{R}^d\}$. It was shown earlier that $S_d' \subseteq S_d$ for all positive $d$, where equality holds for $d = 1, 2, 3$. This implies that we solve a more specific problem by imposing a structural constraint on $\Sigma$.

Implementation-wise, the lower bound presented in (4.10) is differentiable and the gradient $\nabla_\theta \mathcal{L}$ is computed and applied automatically when using the `Pyro` and `Pytorch` libraries in Python. These two libraries (one can also use `Tensorflow` and `Edward` Python packages to achieve an identical result) work greatly together: we do not have to specify an explicit bound, but rather "observe" our realized dataset and "sample" a latent vector from any implemented random variable. `Pyro` then computes the ELBO $\mathcal{L}$, as well as all gradients $\nabla \mathcal{L}$ using the score function trick, viz (Williams, 1992)

$$\nabla_\theta \mathbb{E}_X[f(x;\theta)] = \mathbb{E}_X[\nabla_\theta f(x;\theta)] = \mathbb{E}_X[f(x;\theta) \nabla_\theta \log f(x;\theta)], \tag{4.13}$$

for any given (not necessarily differentiable) probability density function parameterized by the parameter vector $\theta$. The estimator above has multiple useful properties, for instance it is unbiased.

Finally, we can approximate both the lowerbound $\mathcal{L}$ and its gradients with a minibatch. First, sample a subset $\boldsymbol{X}^M$ of size $M$ from the available data. Then, evaluate

$$\hat{\mathcal{L}}^M(\theta, \phi; \boldsymbol{X}^M, z) = \frac{1}{M} \sum_{x_i \in \boldsymbol{X}^M} [\log p_\theta(x_i|z)] - D_{KL}(q_\phi(z|x_i)||p(z)), \tag{4.14}$$

where the latent noise $z \sim q_\phi(z|x_i)$ is taken to be multivariate Gaussian and can be sampled using the reparameterization trick: sample $\varepsilon \sim \mathcal{N}(0, I)$ and then $z = \mu + L\varepsilon$ where $L^T L = \Sigma$. In turn, $\Sigma = u^T u + \exp(a) \circ I_d$, where only $u$ and $a$ are updated by gradient backpropagation.

**Algorithm 4:** Full-rank VAE

**Input:** Matrix of observations $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, minibatch size $N$, learning rate $\alpha$;

**Output:** Learned $a$ and $v$ vectors, estimated $q_\phi$ and $p_\theta$ densities;

Initialize $\theta, \phi$;

**for** *t in* $1, .., \infty$ *(until convergence)* **do**

$\boldsymbol{X}^M \leftarrow$ Sample $M$ points from $\boldsymbol{X}$;

$z \leftarrow$ Sample $M$ points from the latent distribution $q_\phi$;

$g \leftarrow \nabla_{\theta,\phi} \hat{\mathcal{L}}^M(\theta, \phi; \boldsymbol{X}^M, z)$ (Estimate the gradients using minibatch);

$\theta, \phi \leftarrow \theta, \phi - \alpha g$ (Update the parameters);

**end**

More sophisticated gradient descent methods such as Adam (Kingma and Ba, 2015) or RMSProp (Hinton et al., 2012) can be used for faster convergence but are not

necessary. After convergence of the algorithm, we have access to the encoder network $q_\phi$ which can be used to estimate $\mu$ and $\Sigma$ from data and to the decoder network $p_\theta$ which can generate samples from the learned multivariate distribution (i.e. copula model with discrete marginals). Using the Sherman-Morrison inversion formula, we can recover $\Omega = \Sigma^{-1}$ as a function of the data and build the corresponding graphical model.

### 4.4 Variational Hölder upper bound minimization

While the classical formulation of the ELBO involves the maximization of the evidence $p(x)$, recent work on the topic (Bouchard and Lakshminarayanan, 2015) suggests the feasibility of minimizing a variational upper bound. Before describing the proposed algorithm, we recall Hölder's inequality.

**Theorem 4.4.1** (Hölder's inequality)**.** *Let $(X, \Sigma, \mu)$ define a measurable space. Let $p, q, \in (1, \infty]$ such that $\frac{1}{p} + \frac{1}{q} = 1$. Then, for all measurable functions $f$ and $g$ on $X$,*

$$||fg||_1 \leq ||f||_p ||g||_q, \tag{4.15}$$

*where $||f||_r$ denotes the $L_r$ norm, viz $\sqrt[r]{\int_X |f|^r dx}$.*

We recall the approach used by Popovic et al. (2018) and assume a latent Gaussian model $Z_i, 1 \leq i \leq n$ with $d-$dimensional discrete observations $X_i, 1 \leq i \leq n$:

$$P[X_1 = x_{i1}, ..., X_d = x_{id}] = \int_{B_i} \phi_d(z_i; R) dz_i, \tag{4.16}$$

where $B_i = \cap_j [\Phi^{-1}(F_{ij}(x_{ij}^-)), \Phi^{-1}(F_{ij}(x_{ij}))]$. Integrating out the latent variable yields the probability mass of the vector $(x_{i1}, ..., x_{id})$.

To see this, consider a discrete random variable $X$. If, for example, $X$ has a high

probability of taking zero values and $P[X = 0] = 0.5$, then solving $\int_{-\infty}^{0} \phi_1(z_1)dz_1$ recovers the probability mass. In general, (Nikoloulopoulos, 2013) provides a neat formulation for the discrete marginal Gaussian copula problem. Consider the following setup: let $X = (X_1, ..., X_d)$ be a discrete random vector generated by a multivariate Gaussian copula $C$ with correlation matrix R that we wish to estimate and marginal distribution functions $\{F_j\}_{j=1}^{d}$. Then,

$$\begin{aligned} P[X_1 = x_1, ..., X_d = x_d] &= P[x_1 - 1 \leq X_1 \leq x_1, ..., x_d - 1 \leq X_d \leq x_d] \\ &= P[z_1^- \leq X_1 \leq z_1, ..., z_d^- \leq X_d \leq z_1] \\ &= \int_{z_1^-}^{z_1} ... \int_{z_d^-}^{z_d} \phi_d(z_1, ..., z_d; R)dz_1...dz_d, \end{aligned} \tag{4.17}$$

which is the integral that we aim to evaluate.

Maximizing the log evidence $p(x)$ then boils down to estimating the corresponding truncated Gaussian integral. A recent paper (Bouchard and Lakshminarayanan, 2015) suggests to view the Gaussian integral with orthogonal truncations (that is, integral over a rectangle region) as a norm of the random variable $Z$ taking values in a Hilbert space $\mathcal{Z}$. Then, we wish to approximate

$$I^* = \int \gamma_1(Z)\gamma_2(Z)d\nu(Z) = ||\gamma_1\gamma_2||_1, \tag{4.18}$$

where $\gamma = \gamma_1\gamma_2$ is the density function of $Z$ and $\nu$ is the Lebesgue measure if we work over the real space. For $\Psi : \mathcal{Z} \to \mathbb{R}^+$ and $\alpha = (\alpha_1, \alpha_2), \alpha_1, \alpha_2 \in \mathbb{R}^+$, the authors

define

$$\bar{I}_\alpha(\Psi) = ||\gamma_1 \Psi||_{\alpha_1} ||\gamma_2 / \Psi||_{\alpha_2}. \tag{4.19}$$

We can now cite the main result of (Bouchard and Lakshminarayanan, 2015)

**Theorem 4.4.2** (Variational Hölder bound)**.** *Let $\gamma_1, \gamma_2$ be two positive measures defined on $\mathcal{Z}$. Then*

$$I^* \leq \bar{I}_\alpha(\Psi) \tag{4.20}$$

*holds for any positive scalar $\alpha_1, \alpha_2$ such that $\frac{1}{\alpha_1} + \frac{1}{\alpha_2} = 1$ and any positive function $\Psi : \mathcal{Z} \to \mathbb{R}^+$.*

Variational Bayes based on the Hölder upper bound first selects a tractable family of distributions $\mathcal{F} = \{\Psi(\tau), \tau \in \mathcal{T}\}$ and obtains estimates $\hat{\alpha}_1, \hat{\tau}$ by solving the following optimization problem:

$$\hat{\alpha}_1, \hat{\tau} = \arg\min_{\mathbb{R} \times \mathcal{T}} \bar{I}_\alpha(\Psi(\tau)) \tag{4.21}$$

Note how optimizing $\text{logit}(\frac{1}{\alpha_1})$ makes the problem unconstrained. In this paper, we pick $\Psi$ to be the zero-centered multivariate Gaussian distribution $\Phi_d(0, \Sigma)$ with the independence assumption, that is $\Sigma = \tau \circ I_d$.

To reiterate over the truncated Gaussian integration example given by the authors, we pick $\gamma_1(z) = \prod_{j=1}^d f_j(z_j)$ for univariate functions $f_j : \mathbb{R} \to \mathbb{R}, \forall j = 1, ..., d$ and choose $\gamma_2(z) = \exp\left(-\frac{1}{2} z^T \Sigma z + \mu^T z\right)$ where $\Sigma$ is a symmetric $d \times d$ matrix and $\mu$ a $d-$dimensional real vector. The problem of interest consists in evaluating

$$I^* = \int_{\mathbb{R}^d} \prod_{j=1}^d f_j(z_j) \exp\left(-\frac{1}{2} z^T \Sigma z + \mu^T z\right) dz \tag{4.22}$$

If we take $f_j(z) = \mathbb{I}[a_j^- \leq z \leq a_j]$, we obtain the truncated Gaussian latent model which specifies $P[X_1 = x_1, ..., X_d = x_d] = P[\{z_1 \in (a_1^-, a_1)\} \cap ... \cap \{z_d \in (a_d^-, a_d)\}]$. The left and right integration bounds $\{a_j^-\}_{j=1}^d$ and $\{a_j\}_{j=1}^d$ are fixed for a multivariate Gaussian distribution and correspond to orthogonal truncation limits. However, in the case when the input data isn't normally distributed, we can first estimate the true marginal CDFs $F_j$ with respective empirical CDFs $\hat{F}_j$ and take $a_j^- = \Phi^{-1}(\hat{F}_j(x_j - 1))$ and $a_j = \Phi^{-1}(\hat{F}_j(x_j))$. Hence, by evaluating the multivariate Gaussian rectangle probability, we are able to recover the discrete probability mass function (Nikoloulopoulos, 2013). We could, of course, use standard numerical methods such as Simpsons' rule to evaluate the integral and then, using this estimate, compute the parameters $\mu$ and $\Sigma$. Instead, we examine an ad hoc coordinate ascent method in which we first minimize the variational Hölder upper bound with respect to $(\alpha_1, \tau)$ and then maximize with respect to $(\mu, \Sigma)$. Furthermore, we use the positive semi-definite decomposition of $\Sigma$ into $u^T u + \exp(a) \circ I_d$ and obtain the following simplified upper bound:

$$\mathcal{L}_H = \frac{1}{\alpha_1} \sum_{i=1}^{n} \log U(\tau_i, \alpha_1) + \frac{1}{\alpha_2} J(\alpha_2(\Sigma - \text{diag}(\tau))) - \frac{n}{2} \log 2\pi,$$

where

$$U(\tau_i, \alpha_1) = \sqrt{\frac{\pi}{2\alpha_1 \tau_i}} \left[ \Phi\left( \sqrt{\frac{\tau_i \alpha_1}{2}} r_i \right) - \Phi\left( \sqrt{\frac{\tau_i \alpha_1}{2}} l_i \right) \right] \text{ for } 1 \leq i \leq n, \tag{4.23}$$

$$J(M) = -\frac{1}{2} \log(\det(M)) \text{ and}$$

$$l_{ij} = \Phi^{-1}(\hat{F}_j(x_{ij} - 1)), r_{ij} = \Phi^{-1}(\hat{F}_j(x_{ij}))$$

We aim to solve the following optimization problem:

$$\hat{a}, \hat{u}, \hat{\tau}, \hat{\alpha}_1 = \max_{(a,u)\in\mathbb{R}^d\times\mathbb{R}^d} \min_{(\tau,\alpha_1)\in\mathbb{R}^d\times\mathbb{R}_+} \mathcal{L}_H \tag{4.24}$$

where $\tau_1$ can be initialized to any value which leads $\text{diag}(\tau) \prec \Sigma$, for example to half the minimum eigenvalue of $\Sigma$.

**Algorithm 5:** Hölder Variational Bayes parameter updates.

**Input:** Matrix of observations $\boldsymbol{X} \in \mathbb{R}^{n\times d}$, minibatch size $N$, learning rate $\alpha$;

**Output:** Learned $a, \tau, v, \alpha_1$;

Initialize $a, \tau_1, v, \alpha_1$;

Set $\phi \leftarrow u \cup a$;

Set $\theta \leftarrow \tau_1 \cup \alpha_1$;

Estimate the true marginals using the eCDFs $\{\hat{F}_j\}_{j=1}^d$;

**for** *t in* $1, .., \infty$ *(until convergence)* **do**

$\boldsymbol{X}^M \leftarrow$ Sample $M$ points from $\boldsymbol{X}$;

$g_\theta \leftarrow \nabla_\theta \hat{\mathcal{L}}_H^M(\theta, \phi; \boldsymbol{X}^M)$ (Get a tighter approximation to the upper bound);

$\theta \leftarrow \theta - \alpha g_\theta$ (Update the parameters);

$g_\phi \leftarrow \nabla_\phi - \hat{\mathcal{L}}_H^M(\theta, \phi; \boldsymbol{X}^M)$ (Maximize the data probability);

$\phi \leftarrow \phi - \alpha g_\phi$ (Update the parameters);

**end**

Algorithm 5 starts by approximating $p(\boldsymbol{X})$ by minimizing the Hölder upper bound with respect to the variational parameters $(\tau, \alpha_1)$ and then maximizing the evidence with respect to $(u, a)$. The process continues until the estimated quantities

stabilize or until a maximal number of steps.

As discussed in the experimental section, this approach is extremely sensitive to hyperparameter selection. While it has interesting theoretical properties such as consistency (Bouchard and Lakshminarayanan, 2015), the algorithm is less stable than the full-rank VAE and should be used with great care.

# Chapter 5
# Experiments

To assess the performance of our algorithm, we artificially generate multivariate dependent data as follows:

**Algorithm 6:** Generation of correlated observations

**Input:** Collection of distribution functions $\{F_j\}_{j=1}^d$, Gaussian covariance matrix $\Sigma \in \mathbb{R}^{d\times d}$, number of observations $n$;

**Output:** Matrix of observations $\boldsymbol{X} \in \mathbb{R}^{n\times d}$;

$y \leftarrow \{\}$;

**for** $i$ *in* $1, .., n$ **do**

  Sample $z_i \sim \mathcal{N}(0, \Sigma)$;

  $y \leftarrow y \cup z_i$;

**end**

$\boldsymbol{X} \leftarrow \{\}$;

**for** $j$ *in* $1, .., d$ **do**

  $y_j \leftarrow \Phi_1(u_j)$;

  $\boldsymbol{X} \leftarrow \boldsymbol{X} \cup F_j^{-1}(y_j)$;

**end**

Observe how we are using the inverse of the distribution function, $F_j^{-1}$, which is well-defined for continuous random variables. For discrete random variables, we define the inverse as $F^{-1}(x) = \inf\{y : F(y) \geq x\}$.

The output of the algorithm is a matrix of dependent observations, each of which has the specified marginal distribution $F_j$. We use Algorithm 6 to generate 200 bivariate samples from two Gaussian copulas with parameters 0 and 0.9, respectively. We then use the quantile transform to create Poisson random variables with $\lambda_1 = 1$ and $\lambda_2 = 10$, respectively. All two-dimensional experiments use the same Poisson marginals. We ran the multi-armed bandit (MAB) algorithm with $\varepsilon = 0.1$ to estimate

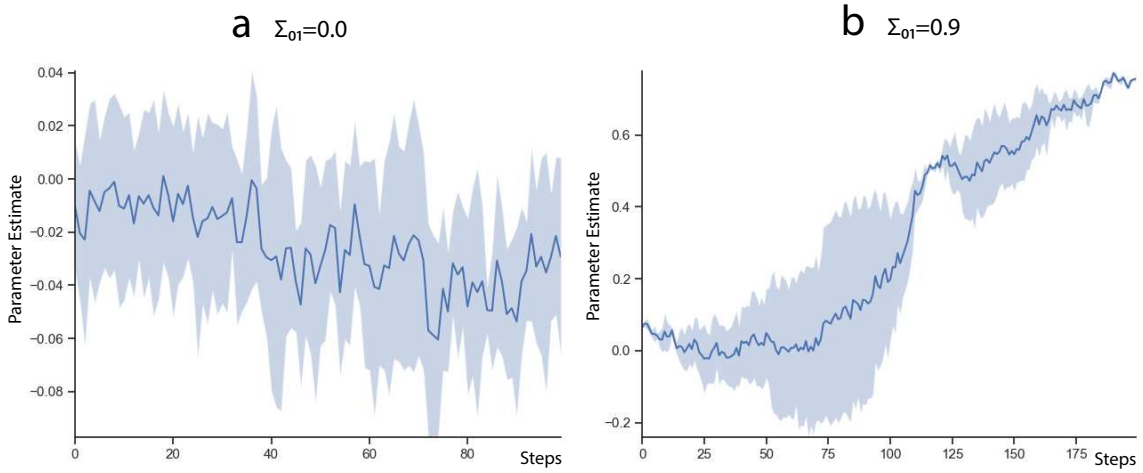



Figure 5–1: Parameters estimated by the MAB algorithm over training iterations for two (a) uncorrelated Poisson variables and (b) positively correlated Poisson variables.

the copula parameters from the simulated data and present the results below.

Figure 5–1 shows the evolution of the parameter estimates as a function of iterations for both datasets. The algorithm found the following parameter estimates:

$$\hat{\Sigma}_a = \begin{bmatrix} 1 & -0.014 \\ -0.014 & 1 \end{bmatrix} \text{ and } \hat{\Sigma}_b = \begin{bmatrix} 1 & 0.861 \\ 0.861 & 1 \end{bmatrix} \tag{5.1}$$

which are surprisingly accurate considering that we are not making use of the gradient. Now, we assess the performance of the Hölder upper bound optimization

algorithm. Figure 5–2 shows the upper bound estimated by coordinate descent. Ob-

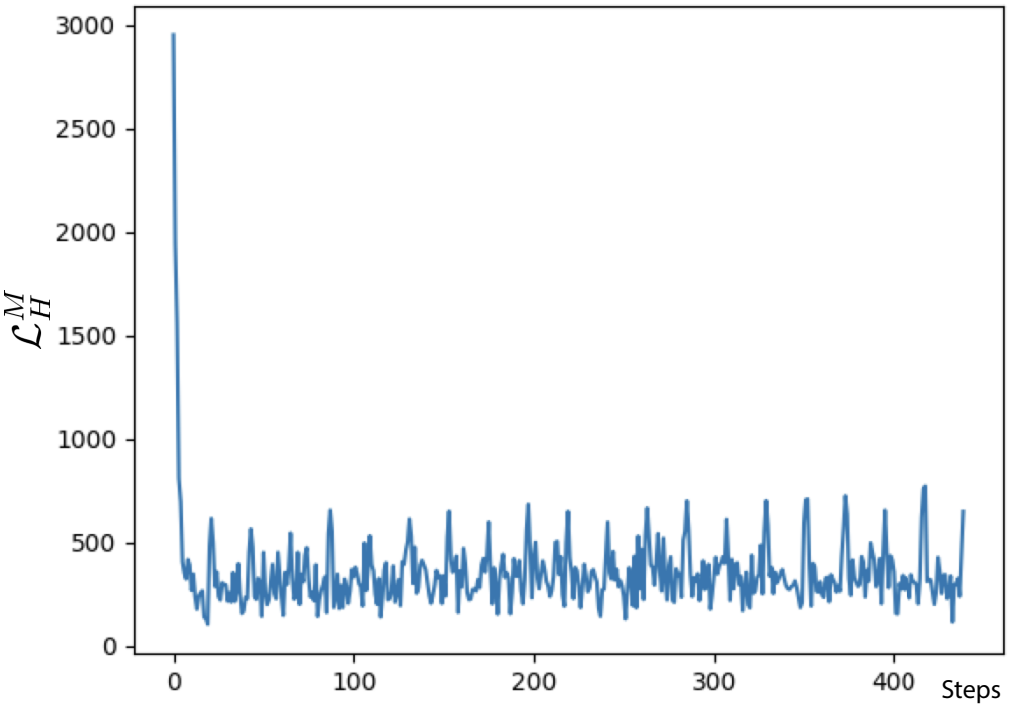



Figure 5–2: The Hölder variational upper bound estimated by Algorithm 5.

serve how phases of minimization and maximization alternate, similarly to Gibbs sampling. The method is quite sensitive to hyperparameter choice. The figure was obtained with samples from a Gaussian copula with parameter 0, run for 20 steps of joint and coordinate-wise optimization. The learning rate was chosen to be $10^{-4}$; $a, u$ and $\alpha_1$ were initialized randomly from a Uniform$(0, 1)$ distribution; $\tau$ was initialized in the convex set of $\Sigma$. After convergence, the algorithm estimated the copula parameter to be 0.227, thus overestimating the true value.

| Method | $R$ $-0.5$ | 0.0 | 0.5 |
|---|---|---|---|
| | $\hat{R}_{01}$ | | |
| Ground truth | $-0.500$ | 0.000 | 0.500 |
| Observed glasso | $-0.468$ | 0.043 | 0.558 |
| Latent glasso | $-0.486$ | **0.000** | 0.415 |
| FR-VAE | **$-0.503$** | 0.044 | **0.495** |
| | $\hat{\Omega}_{01}$ | | |
| Ground truth | 0.667 | 0.000 | $-0.667$ |
| Observed glasso | 0.599 | $-0.016$ | $-0.599$ |
| Latent glasso | 0.486 | **0.000** | $-0.415$ |
| FR-VAE | **0.673** | $-0.044$ | **-0.656** |

Table 5–1: Performance of FR-VAE, observed glasso and latent glasso on the two-dimensional Gaussian model with Poisson marginals.

We next conducted another set of experiments to compare the FR-VAE and glasso algorithms. Applying the graphical lasso algorithm to the discrete observed samples $\boldsymbol{X}$ induces bias into the estimates: we are looking to estimate the parameter of the copula, and not the covariance between the marginals. For this reason, we conduct experiments on two variants of glasso: the naive glasso which estimates the observed ground truth $\Sigma_{ij} = \mathbb{E}[X_i X_j] - \mathbb{E}[X_i]\mathbb{E}[X_j], 1 \leq i, j \leq d$ and the Gaussian glasso which estimates the latent ground truth $\tilde{\Sigma}_{ij} = \mathbb{E}[Z_i Z_j] - \mathbb{E}[Z_i]\mathbb{E}[Z_j]$ where $Z_i = \Phi^{-1}(F_i(X_i)), 1 \leq i \leq d$ and the eCDF is used to estimate the true CDF. In neither case does glasso explicitly take care of the inputs being non-continuous. All observations from the latent continuous model go to the same discrete bin and cannot be recovered by glasso, thus inducing a model bias. We call the glasso which uses the observed covariance matrix the *observed* glasso and the one which uses the

latent covariance matrix the *latent* glasso.

Table 5–1 shows the off-diagonal entries in correlation and precision matrices estimated by observed glasso, latent glasso and by the full-rank VAE. All results were obtained by averaging across 20 trials. Bold entries highlight the algorithm which estimate is closest to ground truth. Note that the observed glasso was always the method to give the furthest estimate from the true copula parameter. This is due not only to the model bias towards discrete data, but also to the marginals providing an additional layer of complexity between the copula and the algorithm.

Finally, we estimated the copula parameters using FR-VAE for equally spaced $R_{01}$ from $-1$ to 1. Figure 5–3 shows the true value of the copula used to generate the synthetic dataset versus the difference between the true and estimated values of the parameter found by the variational auto-encoder.

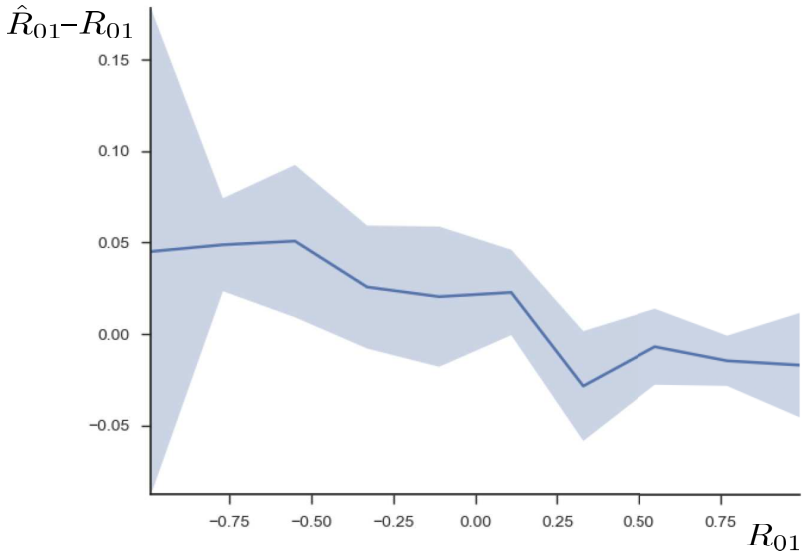



Figure 5–3: True value of the copula parameter versus the values estimated by the FR-VAE algorithm.

The mean and the standard deviation shown in the plot were computed from 20 trials on the same dataset (that is, with Poisson($\lambda = 1$) and Poisson($\lambda = 10$) marginals). Note that the 95% confidence interval is narrow for values of $R_{01} \geq -0.75$; estimates of the parameter at the boundary of the space (when $R_{01} < -0.75$) have higher variance. This is mostly due to numerical stability issues, in which case even though $\Sigma$ is at least positive semi-definite, transforming it into a correlation matrix based on small sample sizes causes an avalanche effect and a small perturbation in the entries of $\hat{\Sigma}$ causes large perturbations in $\hat{R}$ and $\hat{\Omega}$.

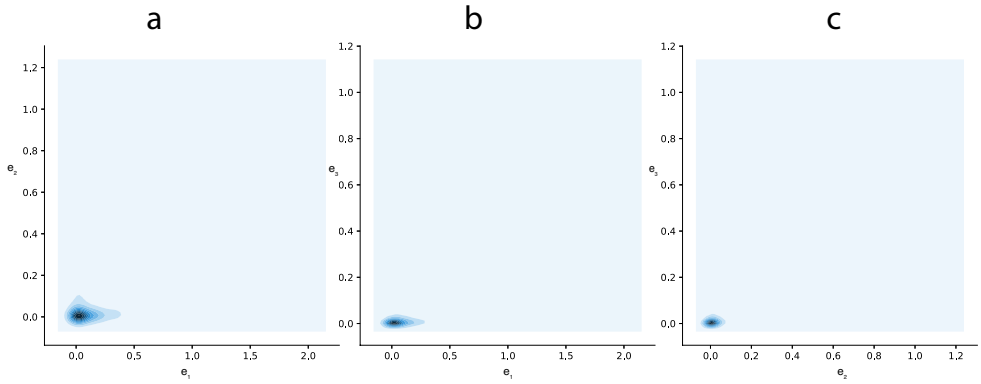



Figure 5–4: Pairwise contour plots of the squared error between the true value of the covariance parameters and the estimates returned by the FR-VAE algorithm.

We conducted an additional series of experiments in which the dataset was sampled from a three-dimensional Gaussian copula model with Poisson($\lambda = 10$),Poisson($\lambda =$

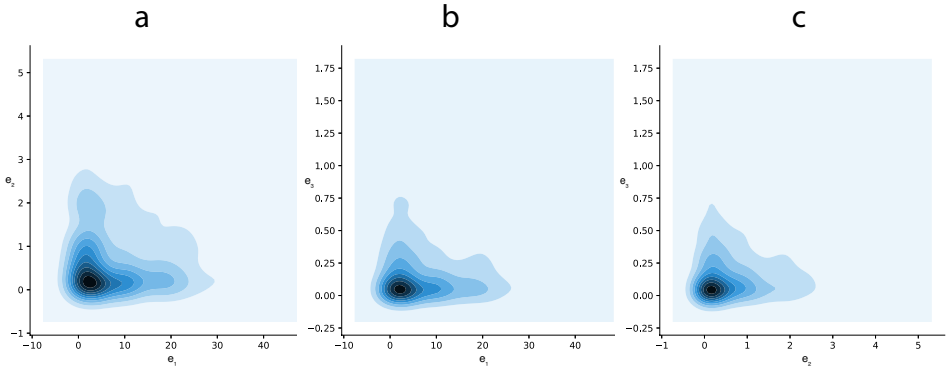



Figure 5–5: Pairwise contour plots of the squared error between the true value of the covariance parameters and the estimates returned by the observed glasso algorithm.

5) and Poisson($\lambda = 1$) marginal distribution functions. The Gaussian copula parameter in the three-dimensional case has the following form:

$$R = \begin{bmatrix} 1 & R_{10} & R_{20} \\ R_{10} & 1 & R_{21} \\ R_{20} & R_{21} & 1 \end{bmatrix} \tag{5.2}$$

where the parameters of interest are $\theta = \{R_{10}, R_{20}, R_{21}\}$. We construct the element-wise squared error which is defined for the element $\theta_i$ as $e_i = (\theta_i - \hat{\theta}_i)^2$. We then conducted a grid search on 1000 values of $\theta$ where $\theta_i \in (-1, 1), 1 \leq i \leq 3$ for 10 equally spaced values along each dimension. For instance, the vector $(0, 0, 0)$ corresponds to the independence copula and therefore is equivalent to generating a dataset with independent Poisson marginals.

Figures 5–4, 5–5 and 5–6 show the pairwise contour plots of the squared error between the true value of the covariance parameters and the estimate given by the FR-VAE, observed glasso and latent glasso algorithms, respectively. The points on

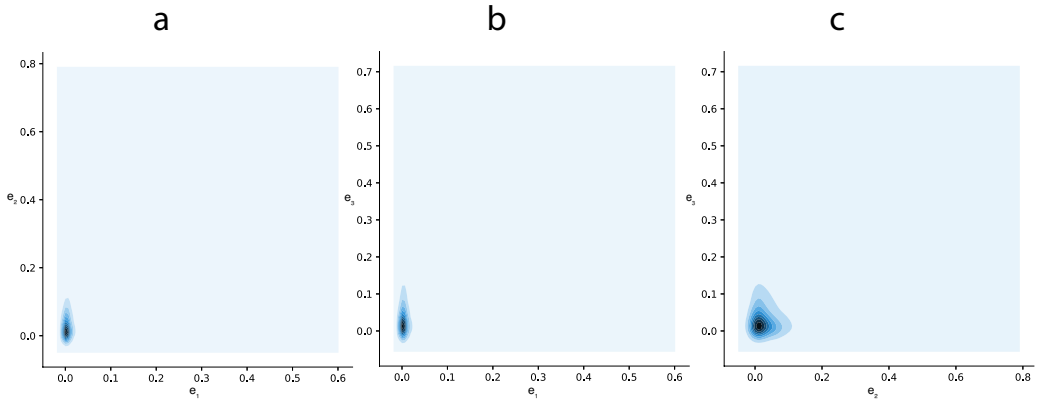



Figure 5–6: Pairwise contour plots of the squared error between the true value of the covariance parameters and the estimates returned by the latent glasso algorithm.

the plot were obtained by joining $(e_1, e_2, e_3)$ into a $3-$dimensional vector.

We can see that while the errors of the FR-VAE and latent glasso predictions are concentrated around the origin and going to a maximum value of 2, the errors of the observed glasso algorithm range up to 40. Such high variation in the observed glasso predictions might be due to stability issues on the boundary of the parameter space. To compare the numerical stability of a linear system of the form $Ax = b$ for a $A \in \mathcal{M}_{n\times d}$ and real vector $x, b$, the condition number $\kappa(A)$ is used. If $e$ is the error in $b$, then

$$\begin{aligned} \kappa(A) &= \max_{e,b\neq 0} \frac{\frac{||A^{-1}e||}{||A^{-1}b||}}{\frac{||e||}{||b||}} \\ &= ||A^{-1}|| \cdot ||A||, \end{aligned} \tag{5.3}$$

that is the ratio of the norm of the error in the solution $A^{-1}b$ to the error in $b$. Using properties of norms, $\kappa(A) = ||A^{-1}|| \cdot ||A|| \geq ||A^{-1}A|| = 1$. As a rule of thumb, when $\kappa(A) = 10^k$, then we may lose up to $k$ digits of accuracy due to the instability of

the system. In fact, since solving the system of linear equations $Ax = b$ requires a matrix inversion, the stability of the solution (i.e. the magnitude of its norm) is directly connected to the stability of the matrix inversion algorithm.

We compare the condition numbers $\kappa(\Sigma)$ for the same 1000 values used in the grid-

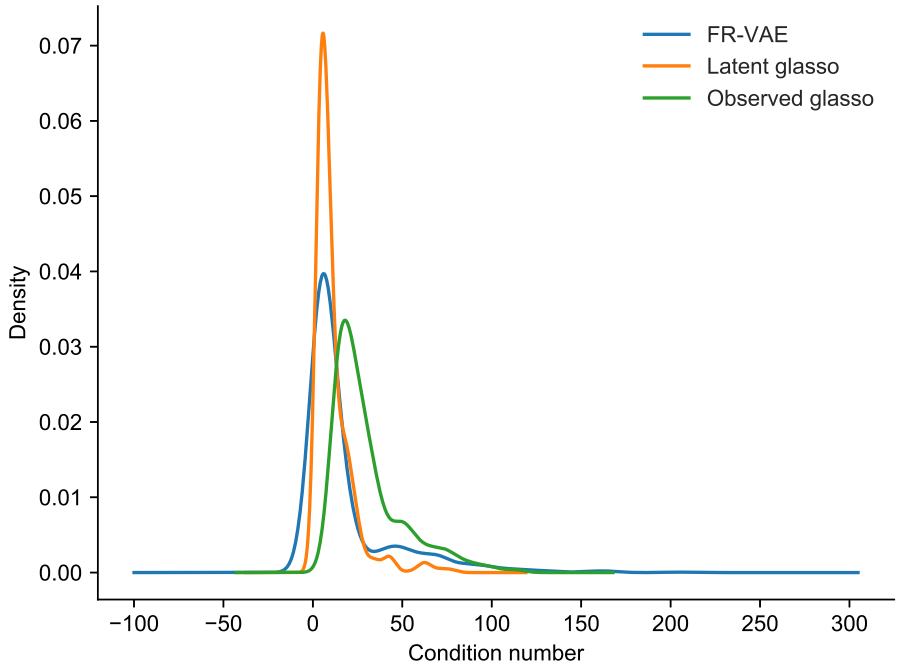



Figure 5–7: Distribution of the condition numbers of $\Sigma$ for various true copula parameters $\theta \in (-1, 1)^3$ as estimated by the FR-VAE and glasso methods.

search above and use them to construct a kernel density estimate plot (see Figure 5–7). The plot shows that the FR-VAE and latent glasso algorithms have condition numbers of $\Sigma$ concentrated at values close to 1 as opposed to larger values produced by observed glasso. The matrices $\Sigma$ estimated by FR-VAE have, on average, a condition number of 18.09 while observed glasso outputs covariance matrices with a

condition number of 30.66 and latent glasso yields an average condition number of 11.22. Precision matrices obtained through the variational auto-encoder are therefore less susceptible to have blown-out components than the ones estimated through observed glasso. On the other hand, latent glasso seems to be producing more stable solutions overall but is susceptible to model bias due to the discrete nature of the observations.

# Chapter 6
# Discussion

In the previous chapter, we have assessed the performance of three proposed methods of graphical model reconstruction. The first algorithm made use of multi-armed bandits to estimate the value of Gaussian copula parameters, the second method leveraged the Hölder variational upper bound in order to estimate the truncated Gaussian integral which corresponds to the probability mass function over the input dataset. The last algorithm used a variational auto-encoder to learn the parameters of the latent copula function. The third methodology worked the best among all suggested algorithms. The third approach had the most accurate estimate of the copula parameters, because we have used the correct prior over the copula parameter. The prior over $z$ was specified by the $\mathcal{N}(0, \hat{\Sigma})$ distribution, where $\hat{\Sigma}$ was estimated through the unbiased estimator $\hat{\Sigma} = \frac{1}{n-1}\sum_{i=1}^{n}(x_{i\cdot} - \bar{x})^T(x_{i\cdot} - \bar{x})$ for $x_{i\cdot} = \frac{1}{d}\sum_{j=1}^{d} x_{ij}$.

## 6.1 Multi-Armed Bandits as a genetic algorithm

Taking a closer look at the MAB algorithm, we can see that it is similar to a genetic algorithm. Both approaches optimize some objective function: the multi-armed bandit maximizes the expected returns and genetic algorithms maximize the fitness of an individual in the hopes of increasing the average fitness of the population. Neither method has access to the gradients since they have to work with any

kind of objective function, including non-continuous and non-differentiable functions. Both algorithms rely on stochastic (i.e. suboptimal) decision making to explore the parameter space and achieve a global optimum. The MAB algorithm which we used in the experiments had the simplest $\varepsilon-$greedy exploration strategy, in which a random arm is pulled with probability $\varepsilon$. On the other hand, genetic methods make use of two stochastic mechanisms: mutations and crossovers. A mutation is a random perturbation in the state of the individual. For example, if we want to find the optimal undirected graph connectivity structure, the state would be the adjacency matrix and random mutations would remove or add and edge with some (typically small) probability. Crossovers occur between two potential solutions and mimic the reproduction mechanism: random parts of the state of one individual are replaced by the complementary parts of the other individual, hence taking the "average" of both states.

Unlike genetic algorithms, bandits explicitly interact with a given environment and the reward (fitness) function is a property of the system as opposed to being chosen by the researcher. When designing a genetic algorithm for a specific task, the fitness score is typically chosen to reflect a notion of proximity to the true solution (a heuristic is used). The exploration strategy is directly dependent on the fitness function, and more fit individuals will have higher crossover rates, driving the population's average fitness up.

The notion of state in a multi-armed bandit is rather blurred: the sequence in which arms are pulled can be seen as a transition into the same state but through different

actions and therefore different rewards. Genetic algorithms are closer to the classical reinforcement learning problem formulation, in the sense that they have feasible configurations which induce a notion of state.

## 6.2 Applying glasso to a latent variable model

Originally, graphical lasso was designed to provide an $L_1$ penalized precision matrix estimated from data. As mentioned earlier, the algorithm assumes that the data is jointly distributed as a $d-$variate Gaussian distribution with unknown mean $\mu$ and precision matrix $\Omega$. Glasso relies on maximizing the penalized normal likelihood (3.3). However, if the dataset $\boldsymbol{X}$ is discrete, the *observed* glasso will wrongly assume a multivariate Gaussian model for what is in fact a Gaussian copula model with discrete marginals. The estimated precision and covariance matrices will therefore be biased.


Figure 6–1: Latent Gaussian model $Z$ with discrete observations $X$. In order to recover the probability mass function of $X$, we marginalize out the hidden variable $Z$.

Recall the latent copula model, in which an unobserved continuous function $C$ generates discrete observations $X_j, 1 \leq j \leq d$. If $C$ is the Gaussian copula, we obtain the latent Gaussian copula model. Discrete observations are generated through truncations of the multivariate Gaussian at points corresponding to the values of the

discrete pmf.

For continuous distributions such as the exponential, mapping the original observations $X_j$ onto the multivariate Gaussian and back is possible through $Z_j = \Phi^{-1}(F_{X_j}(X_j))$ and $X_j = F_{X_j}^{-1}(\Phi(Z_j))$ for $1 \leq j \leq d$, respectively. However, once the data has been discretized through some discrete CDF, it is impossible to recover the original continuous formulation since multiple $z$ values lead to the same discrete bin.

One major advantage of the FR-VAE method over glasso is that the latent gaussian integral representation of the discrete pmf allows us to simultaneously learn the mapping $X \mapsto Z$ with an encoding network and the mapping $Z \mapsto X$ with a decoding network. Because the dimension of $Z$ is the same as the dimension of $X$ by construction, there is a one-to-one correspondence between observed and latent variables.

### 6.3 Performance analysis of the variational Hölder upper bound optimization

The variational Hölder upper bound estimates multivariate Gaussian integrals by fitting the variational family to a sensible prior over the parameters. The choice of the prior dictates the form of the Kullback-Leibler divergence: certain tractable priors do not yield a closed-form KL divergence when combined with variational densities. However, in our case, we pick both the prior and the variational density to be multivariate Gaussian. We picked a $\mathcal{N}(0, \Sigma)$ prior over the latent variables $Z_1, ..., Z_d$, where $\Sigma$ was the unbiased maximum likelihood estimate. As mentioned in previous sections, $\Sigma$ estimated in such a way needs not to be full-rank nor invertible. The vector decomposition of $\Sigma$ ensures that the variational parameters always produce an invertible matrix. Figure 5–2 shows the phases of minimization of the

upper bound to get a tighter approximation to the multivariate Gaussian integral and of maximization of the data probability through prior estimation. We can see the algorithm as alternating between estimating the variational parameters and estimating the prior from data. The variational Hölder upper bound uses alternating gradient descent to project the initial covariance estimate onto the positive-definite cone, making the matrix full rank.

The method turned out to be much more sensitive to hyperparameter choice than the FR-VAE algorithm. For instance, choosing a large learning rate would cause numerical stability issues during training, sometimes causing the upper bound to jump over the true estimate. Initializing the parameter $\tau$ to be in the convex set of $\Sigma$ in order to guarantee proper convergence yields sensible results in the case when $\Sigma$ is fixed prior to training. However, if we dynamically update $\Sigma$, then we should always clip $\tau$ to the nearest positive-definite matrix.

### 6.4 Concluding remarks

We proposed three novel methods of estimating the dependence structure of a Gaussian copula model with discrete marginals. The first approach was based on multi-armed bandits, the second maximized the variational Hölder upper bound with respect to prior parameters and the last algorithm assumed a latent Gaussian copula structure, using a variational auto-encoder to simultaneously learn the variational density (i.e. copula) parameters together with the marginal mass functions.

All three methods learn a vector decomposition of the covariance matrix which is shown to always be positive-definite. The precision matrix (i.e. the inverse of the covariance matrix) encodes the dependence graph between the random variables of

interest and can be easily obtained through the Sherman-Morrison inversion formula. We conducted a series of experiments on artificially generated data obtained from both $2-$dimensional and $3-$dimensional Gaussian copula models with Poisson marginals with parameters $\lambda = 1, 5, 10$. We assessed the performance of all three algorithms by comparing their respective mean squared error to the current state-of-the-art in the field algorithm known as graphical lasso.

The multi-armed bandit and variational Hölder methods suffer from stability issues and failed to correctly estimate the copula correlation parameter. On the other hand, FR-VAE was compared to graphical lasso estimates obtained directly from the observed dataset and from the dataset mapped onto a multivariate Gaussian. Both FR-VAE and latent glasso yielded a similarly low mean squared error and condition numbers, which highlighted the numeric stability of their estimates of $\Sigma$.

A potential extension of the present thesis would be to adapt the FR-VAE algorithm to highly dimensional data. For instance, the MNIST dataset which contains handwritten digits from 0 to 9 of size 28 by 28 pixels would have a $784 \times 784$ covariance matrix which even for modern day algorithms is a large search space. The challenge would be to ensure numerical stability of the gradient update step to keep the covariance matrix invertible.

## Chapter 7
## Definitions

- **Big-O**: Also denoted as $O()$, $Big-O$ notation is a shorthand for asymptotic dominance of one function over another. Let $f$ and $g$ be defined on the same real interval. Then, $f(x) \in O(g(x))$ if and only if $\exists M \in \mathbb{R}^+, x_0 \in \mathbb{R} : |f(x)| \leq M|g(x)|,\ \forall x \geq x_0$. For instance, if $\sqrt{x} \in O(x)$, then $\sqrt{x}$ is eventually dominated by $x$ for all $x \geq 1$. In computer science, this notation is used to denote the worst-case complexity of an algorithm or approximation errors.
- **Random variable**: A measurable function $X : \Omega \to R_X$ is called a *random variable* if it maps elements $\omega \in \Omega$ in the sample space onto a measurable space called the support of $X$, that is $R_X \subseteq \mathbb{R}$. The probability that an event $E \subseteq \Omega$ occurs is equivalent to the expression $P[X \in E] = P[w \in \Omega : X(\omega) \in E]$, hence the requirement for $R_x$ to be measurable.
- **Support of a random variable**: The *support* of a random variable $X$ is the set $R_X = \{x \in \mathbb{R} : \forall \omega \in \Omega, P[X(\omega) = x] > 0\}$. Alternatively, $R_X \subseteq \mathbb{R}$ is the smallest closed set such that $P[X \in R_X] = 1$.

References

Abaker, I., Hashem, T., Yaqoob, I., Anuar, B., Mokhtar, S., Gani, A., and Khan, S. U. (2015). The rise of "big data" on cloud computing: Review and open research issues. *Information Systems*, 47:98–115.

Bellman, R. E. (1952). The Theory of Dynamic Programming. *Proceedings of the National Academy of Sciences*, 38(8):716–719.

Blei, D. M., Kucukelbir, A., and Mcauliffe, J. D. (2017). Variational Inference : A Review for Statisticians. *Journal of the American Statistical Association*, 112(518):859–877.

Bouchard, G. and Lakshminarayanan, B. (2015). Approximate Inference with the Variational Holder Bound. *arXiv preprint*.

Casella, G. and Berger, R. L. (2002). *Statistical inference*. Thomson Learning.

Chambers, John M and Cleveland, William S and Kleiner, Beat and Tukey, P. A. (2017). *Graphical methods for data analysis*. Chapman and Hall/CRC.

Coppersmith, D. and Winograd, S. (1987). Matrix multiplication via arithmetic progressions. In *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, pages 1–6. Academic Press.

Dunn, P. K. and Smyth, G. K. (1996). Randomized Quantile Residuals. *Journal of Computational and Graphical Statistics*, 5(3):236–244.

Elidan, G. (2013). Copulas in Machine Learning. *Copulae in mathematical and quantitative finance*, pages 39–60.

Ery, J. (2016). Semiparametric Inference for Copulas with Discrete Margins. Master's thesis, EPFL.

Franzke, B. and Kosko, B. (2015). Using Noise to Speed up Markov Chain Monte Carlo Estimation. *Procedia - Procedia Computer Science*, 53:113–120.

Friedman, J., Hastie, T., and Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441.

Gelman, A. (2013). *Bayesian data analysis*. Chapman and Hall/CRC.

Genest, C., Ghoudi, K., and Rivest, L.-P. (1998). Understanding Relationships Using Copulas. *North American Actuarial Journal*, 2(3):143–149.

Genest, C. and Nešlehová, J. (2007). A primer on copulas for count data. *ASTIN Bulletin*, 37:475–515.

Gerschgorin, S. (1931). Uber die Abgrenzung der Eigenwerte einer Matrix. *Bulletin de l'académie des sciences de l'URSS*, (6):749–754.

Graves, A., Bellemare, M. G., Menick, J., Munos, R., and Kavukcuoglu, K. (2017). Automated Curriculum Learning for Neural Networks. In *International Conference on Machine Learning*.

Hall, M. A. (1999). *Correlation-based Feature Selection for Machine Learning*. PhD thesis, The University of Waikato.

Hernández-Lobato, J. M., Edu, J. H., Li, Y., Rowland, M., Hernández-Lobato, D., Es, D. H., Bui, T. D., Turner, R. E., and Uk, R. A. (2016). Black-Box $\alpha$-Divergence Minimization. In *International Conference on Machine Learning*.

Hinton, G. E., Srivastava, N., and Swersky, K. (2012). Lecture 6a- overview of mini-batch gradient descent. *COURSERA: Neural Networks for Machine Learning*, page 31.

Højsgaard, S., Edwards, D., and Lauritzen, S. (2012). *Graphical Models with R*. Springer US, Boston, MA.

Joe, H. (1997). *Multivariate models and multivariate dependence concepts*. Chapman and Hall/CRC.

Kingma, D. P. and Ba, J. L. (2015). Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*.

Kingma, D. P. and Welling, M. (2013). Auto-Encoding Variational Bayes. *International Conference on Learning Representations*.

Kullback, S. and Leibler, R. A. (1951). On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86.

Larjo, A. and Lähdesmäki, H. (2015). Using multi-step proposal distribution for improved MCMC convergence in Bayesian network structure learning. *EURASIP journal on bioinformatics & systems biology*, 2015(1):6.

Lauritzen, S. L. (1996). *Graphical models*. Clarendon Press.

Li, X., Mikusiński, P., Sherwood, H., and Taylor, M. D. (1997). On Approximation of Copulas. In *Distributions with given Marginals and Moment Problems*, pages 107–116. Springer Netherlands, Dordrecht.

Li, Y. and Turner, R. E. (2016). Rényi Divergence Variational Inference. In *Advances in Neural Information Processing Systems*.

Liu, H., Lafferty, J., Wasserman, L., and Wainwright, M. J. (2009). The Nonparanormal: Semiparametric Estimation of High Dimensional Undirected Graphs. *Journal of Machine Learning Research*, 10:2295–2328.

Mccallum, A. and Nigam, K. (1998). A Comparison of Event Models for Naive Bayes Text Classification. In *Association for the Advancement of Artificial Intelligence*.

Nelsen, R. B. (2007). *An introduction to copulas*. Springer Science & Business Media.

Nešlehová, J. (2007). On rank correlation measures for non-continuous random variables. *Journal of Multivariate Analysis*, 98(3):544–567.

Nikoloulopoulos, A. K. (2013). A survey on multivariate copula-based models for multivariate discrete response data. *Copulae in Mathematical and Quantitative Finance*, pages 231–249.

Nowozin, S., Cseke, B., and Tomioka, R. (2016). f -GAN: Training Generative Neural Samplers using Variational Divergence Minimization. In *Advances in Neural Information Processing Systems*.

Popovic, G. C., Hui, F. K., and Warton, D. I. (2018). A general algorithm for covariance modeling of discrete data. *Journal of Multivariate Analysis*, 165:86–100.

Rummery, G. A. and Niranjan, M. (1994). On-Line Q-Learning Using Connectionist Systems. Technical report.

Sutton, R. S. (1988). Learning to Predict by the Methods of Temporal Differences. *Machine Learning*, 3(1):9–44.

Trivedi, P. K. and Zimmer, D. M. (2006). Copula Modeling: An Introduction for Practitioners. *Foundations and Trends in Econometrics*, 1(1):1–111.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3-4):279–292.

Wei Koh, P. and Liang, P. (2017). Understanding Black-box Predictions via Influence Functions. *arXiv preprint*.

Williams, R. J. (1992). Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3):229–256.

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67(2):301–320.