Master's thesis
Master's Programme in Data Science

Independent Component Analysis for Binary Data

Vitória Barin Pacela

June 15, 2021

Supervisor(s): Professor Aapo Hyvärinen and Dr. Antti Hyttinen

Examiner(s): Professor Aapo Hyvärinen
Dr. Antti Hyttinen

University of Helsinki
Faculty of Science

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Koulutusohjelma — Utbildningsprogram — Degree programme
Faculty of Science	Master's Programme in Data Science

Tekijä — Författare — Author

Vitória Barin Pacela

Työn nimi — Arbetets titel — Title

Independent Component Analysis for Binary Data

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidantal — Number of pages
Master's thesis	June 15, 2021	58

Tiivistelmä — Referat — Abstract

Independent Component Analysis (ICA) aims to separate the observed signals into their underlying independent components responsible for generating the observations. Most research in ICA has focused on continuous signals, while the methodology for binary and discrete signals is less developed. Yet, binary observations are equally present in various fields and applications, such as causal discovery, signal processing, and bioinformatics. In the last decade, Boolean OR and XOR mixtures have been shown to be identifiable by ICA, but such models suffer from limited expressivity, calling for new methods to solve the problem.

In this thesis, "Independent Component Analysis for Binary Data", we estimate the mixing matrix of ICA from binary observations and an additionally observed auxiliary variable by employing a linear model inspired by the Identifiable Variational Autoencoder (iVAE), which exploits the non-stationarity of the data. The model is optimized with a gradient-based algorithm that uses second-order optimization with limited memory, resulting in a training time in the order of seconds for the particular study cases.

We investigate which conditions can lead to the reconstruction of the mixing matrix, concluding that the method is able to identify the mixing matrix when the number of observed variables is greater than the number of sources. In such cases, the linear binary iVAE can reconstruct the mixing matrix up to order and scale indeterminacies, which are considered in the evaluation with the Mean Cosine Similarity Score. Furthermore, the model can reconstruct the mixing matrix even under a limited sample size. Therefore, this work demonstrates the potential for applications in real-world data and also offers a possibility to study and formalize identifiability in future work.

In summary, the most important contributions of this thesis are the empirical study of the conditions that enable the mixing matrix reconstruction using the binary iVAE, and the empirical results on the performance and efficiency of the model. The latter was achieved through a new combination of existing methods, including modifications and simplifications of a linear binary iVAE model and the optimization of such a model under limited computational resources.

ACM Computing Classification System (CCS):
Computing methodologies → Machine learning → Learning paradigms → Unsupervised learning
Computing methodologies → Machine learning → Machine learning approaches → Learning latent representations
Computing methodologies → Machine learning → Machine learning approaches → Neural networks

Avainsanat — Nyckelord — Keywords

independent component analysis, neural networks, variational autoencoder, binary data

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

To my family,
Janete, Carlos and Vitor.

Acknowledgements

I am immensely grateful for the endless support I have received from my supervisors, Professor Aapo Hyvärinen and Dr. Antti Hyttinen. Thanks for always encouraging me to produce the best work possible, and for giving me full support in this endeavor. I appreciate the rigor and scientific integrity that I have inherited from you, and I have done my best to fulfill this opportunity to improve. Both of you are great inspirations to me! Thank you for being so generous with your time, encouraging my learning, patiently helping with my knowledge gaps, guiding me in my career, and for all the great ideas without which this thesis would not be possible. Thanks, Aapo, for taking me early on as a master's student, I will always be grateful for this chance. Seeing your passion for ICA inspired me every time we virtually met! Thanks for helping me develop more mathematical thinking as we tried to generalize, and formalize thoughts, ideas and results into theoretical properties. Thanks, Antti, for keeping me on track, for finding bugs in my code, for discussing notation as many times as needed, for being so meticulous, for discussing and sharing workflows and pipelines for machine learning research, for sharing fun stories about science, and for being a friend during the pandemic. Thanks to you, I never got stuck for too long. Seeing your passion for causal discovery and causal inference also inspired me every day! Scratching things on the whiteboard and making small discoveries was also fun. Kiitos paljon! This project was also a great source of motivation and energy that helped to keep me mentally healthy during the pandemic chaos.

I would like to acknowledge funding granted to Professor Aapo Hyvärinen by CIFAR (Fellowship) and the Academy of Finland (project #330482). I am grateful to CSC – IT Center for Science, Finland, for generous computational resources, as well as the HPC environment from the University of Helsinki and the Department of Computer Science. I would like to thank Ilyes Khemakhem for sharing the implementation of the continuous iVAE model that has strongly inspired my implementation of the binary iVAE. The notation section of this thesis is based on the notation of the Deep Learning book [11].

I also would like to thank all the supervisors who have supported me in research. My deepest gratitude to the first people to motivate my interest in machine learning:

Professor Maria Spiropulu (Caltech), Dr. Jean-Roch Vlimant (Caltech), and Dr. Maurizio Pierini (CERN). I harbor great admiration for all of you. At CERN, I found my true purpose in science. I was in awe watching and living the passion of scientists from all over the world for trying to understand the universe through particle physics, and the growing interest of the community in deep learning captivated me. Still today, I keep thinking about how my machine learning research can help solve problems in high-energy physics. Jean-Roch, thanks for your patience in getting me started, for having such a strong interest in core machine learning that it caught me too, for inspiring me to develop good computational systems, and for all the fun. Maurizio, thanks for giving me confidence when I needed it, for careful explanations of how the LHC works, and for still being a mentor. Maria, thanks for giving me this opportunity, for the contagious energy that always motivates me, and for the various traits and skills you taught me just by doing, such as the excitement for interdisciplinarity, the collaborative spirit, and the way you manage a group and motivate people. I also would like to thank the organizers of the Openlab and SURF programs, and many of my friends and fellow summer students – to name a few: Danny, Daniel, Gaia, Jesus, Olmo, Aidana, Meghana, Flavia, Professor Sergio Novaes, Professor Sandra Padua. And thanks Tracy Sheffer for all the sympathetic help on the administrative side!

Thanks, Professor Yoshua Bengio, Dr. Karthik Mukkavilli, and Dr. Sasha Luccioni for hosting me at Mila/Université de Montréal, and thanks to everyone in the Visualizing the Impacts of Climate Change team. The deep learning community at Mila is unique and there I had the opportunity to know different areas—It was when I got excited about the topics in this thesis. To all the friends I have made there, I have learned a lot from you! Chiheb, Evan, Gautier, Hans, Ata, Ayesha, Shivam, Raymond, Vikas, Felipe, Pablo. Thanks, Linda Peinthiere, Cristina Eickhoff de Oliveira, and Julie Mongeau for the warm help on the administration side. Merci beaucoup ! I am also thankful for the inclusive community I found when I went to NeurIPS in 2019, especially at the LatinX in AI and Women in Machine Learning workshops.

Thanks, Professor Kai Nordlund, Dr. Antti Kuronen, and Dr. Andrey Ilinov for being the first to take a chance on me when I was still a freshman. The scientific computing background has greatly helped me ever since! I have fond memories from the Accelerator Lab and everyone there. Further thanks to Toffe for helping me in everything from programming to physics, and Alvaro for the lunches in Portuguese. Thanks, Professor Mikko Voutilainen, Dr. Henning Kirschenmann, Joona, Kimmo, and everyone in the Helsinki Institute of Physics.

Thanks, Professor Harry Atwater and Siying Peng for mentoring me for a summer at Caltech, and thanks to all the organizers of the SURF program – Candace Rypisi, Carol Casey, et al. Caltech has a unique environment for science that ensured me that

research is what I want to do with my life. Moreover, Caltech offers unconditional support for undergrads – freshmen like I was – to do serious research work. It's one of the only places that really believes that we can do it at such a young age, that encourages the scientific process from early on, that teaches us to read, write, embrace our failures, and celebrate the small victories. Having this early start was exceptional to me and that is the reason why I think programs like SURF are so important. I may not be working on materials science or photonic crystals anymore, but I will always carry everything I learned about the research methodology. I am also thankful for having met so many wonderful scientists and friends there. For instance, I will never forget my excitement and inspiration when I met Professor Frances Arnold. Also thanks, Tomás, for always encouraging me in science and for the fun conversations about neuroscience.

I am thankful to the Millennium Youth Camp organizers – particularly Professor Maija Aksela – for bringing me to Finland all the way from Brazil and giving me the opportunity to study at the Faculty of Science at the University of Helsinki. I thank all the lovely friends I have made in Finland, who have given me so much strength as I managed to balance studies, research, and growing up in a different culture. To my perhe brasileira, Seija, Aline, and Rodrigo, thanks for embracing me. To Ioanna for being my lovely older sister here. To my MYC friends, in special the dearest friends for life, Yuval and Neja. To my flatmate Marta. To my friends from the Data Science program, in special Maddie, Andres, and Tlahui. To the Data Science program coordinators Professor Jussi Kangasharju and Dr. Pirjo Moen for creating a welcoming environment and always asking for our feedback. To the inspiring lecturers and professors I have learned so much from and who have greatly supported me, in particular Professors Arto Klami, Michael Mathioudakis, Nikolaj Tatti, and Teemu Roos. To Marcelo Hartmann for teaching so much on statistics, pointing me to excellent material, and kindly reminding me to focus on my learning. To the IT4Science team for providing full support on the cluster and the university laptops: Juha Helin, Pekko Metsä, and the always friendly Pekka Niklander. To Kumpula people for friendly random encounters in corridors. To Professor Pedro Camargo, for the wise discussions on academia, Brazilian problems and Brazilian blessings, and for reminding me to be ambitious and work hard to achieve my dreams.

Finally, I am thankful for the opportunities I have had in Brazil that empowered me to get here. To the Brazilian Mathematics Olympiad of Public Schools (OBMEP) for being the widest and deepest educational and scientific initiative for social inclusion project I have ever seen. No words can describe the importance of this work and how badly it is needed in Brazil. I am also thankful for the scholarship from OBMEP-Instituto TIM that I received during my bachelor's degree. I am deeply grateful for all the love and support from my family, even when that comes with the pain and saudade

caused by the distance. Obrigada, mamãe (Janete), papai (Carlos), e Vitor. Obrigada por sempre terem me apoiado em tudo o que faço e torcido incessantemente. Obrigada pela compreensão de que eu teria que me mudar pra muito longe. Obrigada por sempre terem me incentivado nos meus estudos, e ao mesmo por me darem a liberdade de ser quem eu sou e de fazer minhas próprias escolhas na vida. Obrigada por todo o amor e carinho, por nunca terem deixado nada faltar. Amo vocês! Agradeço também aos demais parentes e amigos por momentos de conforto, alegria, e apoio.

Você não percebeu que você é o
único representante dos seus sonhos
na face da Terra?

Emicida

Contents

Notation

Numbers and Arrays

a	A scalar (integer or real)
$\mathbf{a}$	A column vector
$\mathbf{A}$	A matrix

Sets

$\mathbb{A}$	A set
$\mathbb{R}$	The set of real numbers
$\{0, 1\}$	The set containing 0 and 1
$\{0, 1, \dots, n\}$	The set of all integers between 0 and n
$[a, b]$	The real interval including a and b
$\mathbf{A}$	A matrix

Indexing

a_i	Element i of vector $\mathbf{a}$, with indexing starting at 1
$A_{i,j}$ or $A[i, j]$	Element i, j of matrix $\mathbf{A}$
$\mathbf{A}_{i,:}$ or $\mathbf{A}[i, :]$	Row i of matrix $\mathbf{A}$
$\mathbf{A}_{:,i}$ or $\mathbf{A}[:, i]$	Column i of matrix $\mathbf{A}$

Linear Algebra Operations

$\mathbf{A}^\top$	Transpose of matrix $\mathbf{A}$
$\mathbf{A} \odot \mathbf{B}$	Element-wise (Hadamard) product of $\mathbf{A}$ and $\mathbf{B}$
$\det(\mathbf{A})$	Determinant of $\mathbf{A}$

Calculus

$\frac{dy}{dx}$	Derivative of y with respect to x
$\frac{\partial y}{\partial x}$	Partial derivative of y with respect to x
$\nabla_{\mathbf{x}} y$	Gradient of y with respect to $\mathbf{x}$
$\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$	Jacobian matrix $\mathbf{J} \in \mathbb{R}^{m \times n}$ of $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$
$\nabla^2_{\mathbf{x}} \mathbf{f}(\mathbf{x})$ or $\mathbf{H}(\mathbf{f})(\mathbf{x})$	The Hessian matrix of $\mathbf{f}$ at input point $\mathbf{x}$
$\int \mathbf{f}(\mathbf{x}) d\mathbf{x}$	Definite integral over the entire domain of $\mathbf{x}$

Probability

$P(a)$	A probability distribution over a discrete variable
$p(a)$	A probability distribution over a continuous variable, or over a variable whose type has not been specified
$a \sim P$	Random variable a has distribution P
$\mathbb{E}_{\mathbf{x} \sim P}[f(x)]$ or $\mathbb{E}[f(x)]$	Expectation of $f(x)$ with respect to $P(x)$
$\mathbb{V}(f(x))$	Variance of $f(x)$ under $P(\mathbf{x})$
$D_{KL}(P \Vert Q)$	Kullback-Leibler divergence of P and Q
$\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$	Multivariate Normal distribution over $\mathbf{x}$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$

Functions

$f : \mathbb{A} \to \mathbb{B}$	The function f with domain $\mathbb{A}$ and range $\mathbb{B}$
$f(\mathbf{x}; \boldsymbol{\theta})$ or $f_{\boldsymbol{\theta}}(\mathbf{x})$	A function of $\mathbf{x}$ parameterized by $\boldsymbol{\theta}$. Sometimes $\boldsymbol{\theta}$ may be dropped for simplicity
$\log(x)$	Natural logarithm of x
$\sigma(x)$	Logistic sigmoid, $\frac{1}{1 + \exp(-x)}$
$\Vert\mathbf{x}\Vert_p$	L^p norm of $\mathbf{x}$
$\Vert\mathbf{x}\Vert$	L^2 norm of $\mathbf{x}$

Nomenclature

$\mathbf{A}$	Mixing matrix
$\mathbf{z}$	Sources or latent variables
$\mathbf{x}$	Observed variables
$\mathbf{u}$	Additionally observed (auxiliary) variable
m	Number of segments
d_z	Number of sources
d_x	Number of observed variables

Abbreviations

ICA	Independent Component Analysis
VAE	Variational Autoencoder
iVAE	Identifiable Variational Autoencoder
MCS	Mean Cosine Similarity
MCC	Mean Correlation Coefficient
MLE	Maximum Likelihood Estimation

1. Introduction

Generative models are widely studied in the areas of machine learning, statistics, neural computation, and signal processing. They can capture the mechanism that generated the data, as opposed to discriminative models that only learn to distinguish decision boundaries in the data. Generative models provide an elegant and generalizable probabilistic solution: by inferring parameters of the model that generated the data, any decision problem can be solved by averaging over the posterior distribution of such parameters. Latent-variable models define a particular class of generative models that assumes a lower-dimensional latent space that intuitively represents the main generative factors from the data. Therefore, latent variable models perform dimensionality reduction, such that the latent representation can be seen as a compression containing the maximum amount of information about the data. Such methods also belong to the realm of unsupervised learning, since the models learn solely from the observed data, without needing any labels. Unsupervised learning is very useful for applications where labels are unavailable or expensive to produce. Interest in generative models has recently been increased because of the success of deep learning. Deep neural networks are flexible, high-capacity models that can parameterize probability functions, providing a computationally tractable approximation that has unleashed the potential of deep latent variable models for the most varied applications, such as high-energy physics simulations [37], image resynthesis [42], natural language generation [25], and data imputation [35].

Independent Component Analysis (ICA) is a particular type of latent variable model that restricts the latent representation to independent variables. It can separate the observed signal into independent underlying signals, thus solving the problem of blind source separation. A relevant application of ICA is in electroencephalography, to separate the signals from the brain from the signals caused by other parts of the body, such as the eyes [40]. Another application of ICA is to discover the causal structure from observed data [39, 31].

Despite significant progress in both linear and nonlinear ICA in recent years [17, 21, 24], ICA for binary data remains a challenging and important problem. Binary data is abundant in various fields, such as bioinformatics, social sciences, natural lan-

guage, and electrical engineering. Many methods assume an OR mixture model, which has limited expressivity and weak conditions of independence of the sources [14, 32]. On the other hand, the linear model available in literature [22, 23] is designed for feature construction and does not provide a study on the identifiability of the mixing model. In this thesis, we employ a linear model for ICA for binary data inspired by recent developments on the identifiability of nonlinear ICA using auxiliary variables and employing variational autoencoders (VAEs) [27, 35].

In ICA, the latent variables $\mathbf{z}$ are seen as *sources*, and the observations $\mathbf{x}$ are a mixture of such sources via a mixing model $\mathbf{f}$ such that $\mathbf{x} = \mathbf{f}(\mathbf{z})$. In the general case, $\mathbf{f}$ is a nonlinear transformation that can be well-described by a deep neural network. Identifiability is achievable, for example, in identifiable Variational Autoencoders (iVAEs), which use additionally observed variables and sources conditionally independent given such auxiliary variables [24]. Autoencoders are particularly suitable for learning the joint distribution of observed variables and latent variables. Drawing a parallel with VAEs, the decoder is represented by the mixing model from ICA, and the encoder is represented by the inference model.

We investigate the problem of ICA for binary data by estimating the mixing model. The underlying signal is nonstationary, and each segment follows a Normal distribution. The observed signal is also nonstationary, and the signal inside each segment is stationary. Hence, the auxiliary variable used is the segment ID. Regarding the model, we adapt the Identifiable Variational Autoencoder (iVAE) to consist of only affine and linear transformations. The mixing model's linear transformation is followed by an element-wise sigmoid function to give the probability that each element is 1. Similarly, the objective function is modified so that the observations follow a Bernoulli distribution, and not a Normal distribution. The optimization algorithm used is Limited-memory BFGS, which uses second-order derivatives to estimate the gradient direction more efficiently, while still being memory-efficient. The reconstruction of the mixing model is evaluated with the Mean Cosine Similarity (MCS) score, which considers order scale indeterminacies in the columns of the reconstructed mixing matrix.

The main question we investigate is when is the method able to reconstruct the mixing matrix, given different numbers of sources and observations. We find that the reconstruction of the mixing matrix is more reliable the greater is the number of observed variables than the number of sources, and even beyond, the method can identify the mixing matrix in the under-complete ICA problem. In addition, numerical results on the training time of the model indicate time efficiency. Furthermore, an analysis of the performance of the model for different numbers of observations suggests sample efficiency—the performance is high even on small datasets.

We conclude that the method can be effective for the problem of ICA for binary data, and would contribute to open new avenues of research in binary ICA. The reconstruction of the mixing model for different mixing matrix dimensions sheds light on identifiability, which we hope to formalize in future work.

This thesis is structured as follows. Chapter 2 provides the necessary background for the thesis, summarizing the main concepts on optimization, and providing a literature review on ICA and VAEs. Chapter 3 introduces the methodology employed for solving the problem of ICA for binary data. It starts by describing the continuous iVAE in detail, being followed by the binary iVAE and the modifications introduced in it. Tools for evaluating the methods are also provided. Chapter 4 presents the empirical evaluation of the method on synthetic data, as well as a discussion with the main questions of study and remarks for future work. Chapter 5 summarizes the main conclusions from the experiments, and the lessons learned from the investigation.

2. Preliminaries

This chapter introduces the background knowledge that supports the main methods employed and developed in this thesis. Section 2.1 introduces the optimization algorithms: gradient descent—which lies the foundations and the basics of gradient-based methods—, stochastic gradient descent, and limited-memory BFGS (L-BFGS). L-BFGS is employed in all the experiments in Chapter 4. The understanding of these methods also supports claims made in the Discussion and in the Conclusion in Chapter 5.

Section 2.2 provides an introduction to Independent Component Analysis (ICA), as well as a literature review for nonlinear ICA and binary ICA. We discuss the main models, principles, and frameworks that motivate the methods hereby employed, and also review the most closely related methods in these modern subfields.

Finally, section 2.3 presents the Variational Autoencoder (VAE). We present it as a combination of an autoencoder with variational inference, therefore also outlining each of these concepts. Both sections 2.2 and 2.3 are important for the understanding of Chapter 3, which presents and develops on Identifiable Variational Autoencoders (iVAEs), a framework that unifies ICA and VAEs to achieve reliable and practical latent-variable representations.

2.1 Optimization

2.1.1 Gradient Descent

Let $\mathbf{w}$ be a vector of parameters and $\mathcal{L}$ a cost function – sometimes interchangeably called objective function or loss function. The Gradient Descent algorithm performs unconstrained optimization by minimizing the function $\mathcal{L}(\mathbf{w})$ iteratively. First, an initial parameter vector is initialized, then the gradient of $\mathcal{L}(\mathbf{w})$ is computed at the initial value of the parameters, and lastly, the parameters $\mathbf{w}$ are updated by moving in the gradient direction with a step size α:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} \tag{2.1}$$

where $\leftarrow$ denotes substitution [19].

The gradient computation and parameter update steps are repeated across multiple iterations until convergence. Typically, until the Euclidean distance between two subsequent parameter vectors is lower than a small tolerance level. The algorithm is very sensitive to the initialization of the parameters and to the step size, and can often converge to a local minimum or a saddle point instead of a global minimum.

In the context of the optimization problems in this thesis, all of the following outcomes should be expected if the step size is small enough:

1. Each iteration decreases the objective function.
2. The objective converges to a value.
3. All parameters converge to some values.

Therefore, we would aim to use an optimal or good step size, as defined following.

Optimal step size: largest step size for which every iteration in every run decreases the objective.

Good step size: decreases the objective at every iteration in every run but is reasonably large so that time for convergence is reasonable.

In the unconstrained case, we stop the gradient descent optimization when the norm of the difference of the estimated parameters in two subsequent iterations is small enough, for example, smaller than 10^{-5}. In the general case, we would stop when the norm of the gradient is small enough, such as below 10^{-5}.

2.1.2 Stochastic Gradient Descent

In machine learning, and particularly in ICA and deep learning, the objective function depends on the observed data, having the form $\mathcal{L}(\mathbf{w}) = \mathbb{E}[g(\mathbf{w}, \mathbf{x})]$, where the expectation is approximated by the sample mean and g is the loss per observation. In gradient descent, the whole dataset is used in a gradient computation, which results in a deterministic behavior, but can also be computationally demanding particularly when the dataset has many observations. Considering $\mathbf{x}$ to have m data points, we can write the gradient as

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \frac{1}{m} \sum_{i=1}^{m} \nabla_{\mathbf{w}} g(\mathbf{x}^{(i)}, \mathbf{w}). \tag{2.2}$$

Stochastic gradient descent exploits the fact that the derivatives with respect to $\mathbf{w}$ can be taken inside the expectation with respect to $\mathbf{x}$ [11, 19]. The gradient is computed using a single observation. That leads to a stochastic behavior since the gradient direction can be very different for different data points, but on average the direction should approximate the gradient direction using the whole dataset.

In practice, to reduce the stochasticity of computing the gradient for every single observation and still allow for computational efficiency, *minibatches* are used instead. That is, a subset of m' observations from the dataset is used to compute the gradient:

$$\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w}) = \frac{1}{m'}\nabla_{\mathbf{w}}\sum_{i=1}^{m'} g(\mathbf{x}^{(i)}, \mathbf{w}). \tag{2.3}$$

2.1.3 Limited-memory BFGS

Second-order methods use the information contained in the second-order derivatives of the cost function, which should give better direction to proceed regarding the optimization landscape, and as a result, reduce the number of iterations. The Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm tackles the problem of approximating the inverse of the Hessian matrix with respect to the parameters, which is the bottleneck of Newton's method [33].

For simplicity in the notation, let us define the column vector $\mathbf{f} = \mathcal{L}(\mathbf{w})$, $\nabla\mathbf{f}_k = \nabla_{\mathbf{w}}\mathbf{f}_k = \dfrac{\partial\mathcal{L}(\mathbf{w})}{\partial\mathbf{w}}$, the difference between the new and current parameters $\mathbf{p} = (\mathbf{w}' - \mathbf{w})$, a Hessian matrix $\mathbf{B} = \dfrac{\partial^2\mathcal{L}(\mathbf{w})}{\partial\mathbf{w}^2}$, and its inverse $\mathbf{H} = \mathbf{B}^{-1}$. Then, the quadratic model of the objective function at the current iterate $\mathbf{w}_k$ is:

$$\mathbf{m}_k(\mathbf{p}) = \mathbf{f}_k + \nabla\mathbf{f}_k^{\intercal}\mathbf{p} + \frac{1}{2}\mathbf{p}^{\intercal}\mathbf{B}_k\mathbf{p}. \tag{2.4}$$

where the minimizer $\mathbf{p}_k$ of this convex quadratic model is $\mathbf{p}_k = -\mathbf{B}_k^{-1}\nabla\mathbf{f}_k$.

BFGS starts at a point $\mathbf{x}_0$ and at an initial Hessian approximation $\mathbf{H}_0$. In every iteration, until the norm of the gradient is smaller than the tolerance, the following steps are repeated:

1. Compute the search direction $\mathbf{p}_k = -\mathbf{H}_k\nabla\mathbf{f}_k$.
2. Update $\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha_k\mathbf{p}_k$, where the step size α_k satisfies the Wolfe conditions from line search.
3. Define $\mathbf{s}_k = \mathbf{w}_{k+1} - \mathbf{w}_k$ and $\mathbf{y}_k = \nabla\mathbf{f}_{k+1} - \nabla\mathbf{f}_k$, to be used in the Hessian estimation.
4. Update the Hessian estimate $\mathbf{H}_{k+1} = (\mathbf{I} - \rho_k\mathbf{s}_k\mathbf{y}_k^{\intercal})\mathbf{H}_k(\mathbf{I} - \rho_k\mathbf{y}_k\mathbf{s}_k^{\intercal}) + \rho_k\mathbf{s}_k\mathbf{s}_k^{\intercal}$, where $\rho_k = \dfrac{1}{\mathbf{y}_k^{\intercal}\mathbf{s}_k}$.

In large-scale unconstrained optimization, when the number of parameters is large, storing and manipulating $\mathbf{H}$ can be demanding. Hence, Limited-memory BFGS (L-BFGS) [29] approximates the inverse of the Hessian from BFGS by storing a number m of the vector pairs $\{\mathbf{s}_i, \mathbf{y}_i\}$ and, as a result, computing the product $\mathbf{H}_k\nabla\mathbf{f}_k$ more

efficiently applying recursion. After the first m iterations, the oldest pair is replaced by the new vector pair from the current iteration; such discard is executed before step 3 from the BFGS steps described previously. The higher is the "memory" m, the more precise is the approximation, and the higher is the time complexity. The optimal choice of m depends on the problem, but in general, it can be stated that the L-BFGS can approximate the Hessian reasonably well, converge fast, and reduce the memory consumption when compared to BFGS.

2.2 Independent Component Analysis

2.2.1 Linear Independent Component Analysis

Example: Blind source separation *[19] The long-standing cocktail party problem refers to a cocktail party where multiple people are talking and background music playing simultaneously. Humans are capable of identifying the particular sound signal that they are more interested in paying attention to.*

This is an instance of a mixture of three sound signals simultaneously recorded by three devices located in different places at the party. Let the three original signals be $s_1(t), s_2(t), s_3(t)$, where t is the time index. Here, the number three is completely arbitrary and could be anything larger than one. Hence, the microphones give us three recorded signals and we denote them by $x_1(t), x_2(t), x_3(t)$. We could express the mixed signals $x_i(t)$ as a linear combination of the original signals $s_i(t)$, that is

$$\begin{aligned} x_1(t) &= a_{11}s_1(t) + a_{12}s_2(t) + a_{13}s_3(t) \\ x_2(t) &= a_{21}s_1(t) + a_{22}s_2(t) + a_{23}s_3(t) \\ x_3(t) &= a_{31}s_1(t) + a_{32}s_2(t) + a_{33}s_3(t), \end{aligned} \tag{2.5}$$

where the a_{ij} with $i, j = 1, 2, 3$ are some parameters of location of the microphones.

Therefore, the solution to the problem is to identify the original signals $s_i(t)$ with $i = 1, 2, 3$ from the known signals $x_i(t)$.

Definition A linear ICA model is composed of n random variables $x_1, ..., x_n$ expressed as linear combinations of n independent random variables $s_1, ..., s_n$ [19]. Hence, we have

$$x_i = a_{i1}s_1 + a_{i2}s_2 + ... + a_{in}s_n, \tag{2.6}$$

for all $i, j = 1, ..., n$, and with real coefficients a_{ij}.

We have now dropped the time index t since it is reasonable to assume that each observation $x_i(t)$ is a sample of the random variable x_i. Similarly, the independent component s_j is also a random variable, instead of a proper time signal or time series.

Usually, it is more convenient to denote the ICA model using vectors and matrices. Let $\mathbf{x}$ be the random vector whose elements are the mixtures $x_1, ..., x_n$, $\mathbf{s}$ be the random vector composed of latent variables $s_1, ..., s_n$, and $\mathbf{A}$ be the mixing matrix whose elements are the real, unknown coefficients a_{ij}. Then, our mixing model can be simply represented as

$$\mathbf{x} = \mathbf{A}\mathbf{s}. \tag{2.7}$$

Assumptions

1. The independent components are statistically independent. That is, the joint probability density function of the sources is factorizable: $p(s_1, \ldots, s_n) = p_1(s_1) \ldots p_n(s_n)$.
2. The independent components cannot be normally distributed.
3. For simplicity, we assume that the mixing matrix $\mathbf{A}$ is square, that is the number of independent components is equal to the number of observed variables. This assumption can be relaxed in some cases, but in general it simplifies the estimation and is a pre-requisite for the mixing matrix to be invertible.

Solution Popular methods to solve linear ICA are, for example, maximization of non-Gaussianity and maximum likelihood estimation [19]. The maximization of non-Gaussianity is motivated by the assumption that the sources cannot have Gaussian distributions. Therefore, higher-order moments such as kurtosis can be used to measure the non-Gaussianity of the estimated sources. More precisely, the function to be maximized is the absolute value of the kurtosis, and the optimization is typically implemented with gradient-based methods. An efficient optimization procedure is a fixed-point iteration algorithm, as developed in FastICA [16]. The main idea is to reach a stable point at convergence, so in every iteration, the gradient is computed to reach such a point. Notably, the gradient does not change the direction of the parameters, and the parameters are constrained to have a norm of 1.

In maximum likelihood estimation, assume that there are d observed variables, n observations of each, and that each independent component has density p_i, for $i = 1, \ldots, d$. Let $\mathbf{B} = \mathbf{A}^{-1}$ be the inverse of the mixing matrix. Then, the log-likelihood is given by

$$\log l(\mathbf{B}) = \sum_{j=1}^{d} \sum_{i=1}^{n} \log p_i(\mathbf{b}_i^\intercal \mathbf{x}^{(j)}) + d \log |\det \mathbf{B}|. \tag{2.8}$$

To optimize the likelihood, gradient methods can be used, such as the Bell-Sejnowski algorithm [2]. The inverse of the mixing matrix parameters can be updated by the

following rule

$$\Delta \mathbf{B} \propto (\mathbf{B}^\intercal)^{-1} + \mathbb{E}[\mathbf{g}(\mathbf{B}\mathbf{x})\mathbf{x}^\intercal], \tag{2.9}$$

where $\mathbf{g}$ is the vector consisting of the negative score functions of the distribution of each source. This algorithm has slow convergence due to the inversion of $\mathbf{B}$. A more efficient approach would be FastICA.

This section has been focused on the traditional linear ICA principle, which is to use a non-Gaussian independent and identically distributed (i.i.d.) model. Nevertheless, multiple other methods relax some of the assumptions. According to Cardoso [4], the "three easy routes to ICA" are non-Gaussianity, non-stationarity, and non-whiteness. non-stationarity-based methods can use Gaussian sources, and non-whiteness-based methods can use Gaussian sources stationarily correlated in time. Although beyond the scope of this thesis, such ideas pave the way to identifiability in nonlinear ICA, as will be described in forthcoming sections.

2.2.2 Identifiability

The core question in ICA is whether the parameters learned via the ICA "factorization" are **unique**, which is studied by applying the definition of identifiability. A parameter is identifiable if two different parameter values lead to two different densities. In other words, if two parameters $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ lead to the same marginal density $p_{\boldsymbol{\theta}}(\mathbf{x}) = p_{\boldsymbol{\theta}'}(\mathbf{x})$, then that implies that the two parameters are equal:

$$\forall(\boldsymbol{\theta}, \boldsymbol{\theta}') : p_{\boldsymbol{\theta}}(\mathbf{x}) = p_{\boldsymbol{\theta}'}(\mathbf{x}) \Rightarrow \boldsymbol{\theta} = \boldsymbol{\theta}'. \tag{2.10}$$

Therefore, for identifiability to hold, the mapping $\boldsymbol{\theta} \mapsto p_{\boldsymbol{\theta}}$ should be one-to-one, and different parameter values should lead to different probability distributions. Inference of non-identifiable models can be difficult, since the observations from distinct distributions $p_{\boldsymbol{\theta}}(\mathbf{x})$ and $p_{\boldsymbol{\theta}'}(\mathbf{x})$ would look the same, and it would be impossible to determine if the true value is $\boldsymbol{\theta}$ or $\boldsymbol{\theta}'$ [5, 24].

Identifiability in linear ICA has been proved by Comon in 1994 [7]. The parameters of the mixing model and the reconstructed sources are identifiable up to scale and order indeterminacies. That is, the parameters can be permuted when compared to the real parameter matrix [19].

2.2.3 Nonlinear Independent Component Analysis

Linear mixing may be too simple to express more complex observed data. Hence, nonlinear mixing is defined in the more ambitious nonlinear ICA problem. Consider a nonlinear function $\mathbf{f}$, m observed variables $\mathbf{x}$, and n sources $\mathbf{s}$. The nonlinear mixing

model is, then:

$$\mathbf{x} = \mathbf{f}(\mathbf{s}). \tag{2.11}$$

The nonlinear ICA problem consists of estimating the mapping $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ such that the components $\mathbf{s}$ are statistically independent [19].

Solutions to this problem always exist and are highly non-unique. For instance, two sources may be mixed but still statistically independent. In contrast, the goal of nonlinear blind source separation is to reconstruct the original sources that generated the observations. Given this important distinction, we focus the rest of this subsection on nonlinear ICA. In nonlinear ICA, it has been shown that if the space of the nonlinear mixing functions $\mathbf{f}$ is not limited, infinite solutions always exist and indeterminacies are nontrivial [20, 9]. A method for constructing parameterized families of nonlinear ICA solutions is also presented.

Dinh *et al.* have proposed a flow-based generative model that learns a nonlinear bijective transformation $\mathbf{h} : \mathbb{R}^n \to \mathbb{R}^n$ that gives independent latent variables [10]. The first identifiability result in nonlinear ICA was obtained by Hyvärinen and Morioka, proposing the principle of Time-Contrastive Learning [17]. It assumes that $\mathbf{x}$ and $\mathbf{s}$ are non-stationary time series, and $\mathbf{f}$ is invertible. The sources are divided into non-overlapping time segments such that their distribution varies across time segments. Then, a deep neural network learns to classify the sources into segments. By learning to discriminate the time segments, the model provably also recovers the nonlinear ICA model up to linear mixing and component-wise nonlinearities. Following, identifiability of temporally dependent stationary sources was also proven in Permutation-Contrastive Learning [18]. Both methods exploit the temporal structure of the data, and that allows for identifiability, whereas non-structured data mixtures are much more difficult to "separate" in the nonlinear regime.

A new form of nonlinear ICA with identifiability results was proposed by Hyvärinen *et al.* by using auxiliary variables, such that the independent components are conditionally mutually independent given the auxiliary variable [21]. Training with auxiliary variables occurs similarly to Permutation-Contrastive Learning, by discriminating between a real and a randomized dataset. The assumptions made are that the mixing model is invertible and that there are as many sources as observed variables; with such assumptions, it is possible to solve nonlinear ICA and estimate the sources with no additional indeterminacies. The identifiability proof applies the definition of conditional exponentiality and covers Time-Contrastive Learning as a special case. Building on the idea of nonlinear ICA with auxiliary variables, the Identifiable Variational Autoencoder is proposed as another solution provably identifiable that uses a maximum likelihood framework, and as an important piece of this thesis, it will be described in further detail in the following chapters.

2.2.4 Binary Independent Component Analysis

Himberg and Hyvarinen [14] approach ICA for binary data by mixing binary signals using OR operations and estimate the binary mixing matrix using cumulant-based ICA algorithms such as FastICA [16]. More precisely, let us define the set of binary numbers as $\mathbb{B} = \{0, 1\}$. The binary ICA model considers binary observed vectors $\mathbf{x} \in \mathbb{B}^m$ and binary sources $\mathbf{s} \in \mathbb{B}^n$, so that the binary ICA model is given by the Boolean expression

$$x_i = \bigvee_{j=1}^{n} a_{ij} \wedge s_j, \quad i = 1, 2, \ldots, m \tag{2.12}$$

where $\wedge$ represents Boolean AND, and $\bigvee$ represents Boolean OR. They describe the Boolean OR mixing to be equivalent to a linear mixing model followed by a unit step function; such alternative is especially suited for sparse data, as evidenced by the results. The method is evaluated by the ratio of correctly retrieved basis vectors of the mixing matrix, such ratio being high particularly in sparse and noiseless signals. Remarkably, this approach suggests a bridge between binary ICA and nonlinear ICA, since the binary OR model is represented as a nonlinearity applied to a linear model.

Nguyen and Zheng have studied binary ICA with OR mixtures by defining a disjunction generative model with a binary mixing matrix and binary sources [32]. Such a model allows for a graphical interpretation by considering the mixing matrix as an adjacency matrix: an edge between source and observation exists if the corresponding element of the mixing matrix is 1. The model presented is identifiable up to permutation. However, one significant limitation is expressiveness: OR mixtures do not allow for all sets of binary variables to be decomposed into binary independent components. The estimated mixing matrix is estimated according to structure and prediction error and the normalized Hamming distance; results suggest that the mixing matrix is recovered well particularly without any added noise.

Kaban and Bingham propose a factorization of binary data as a form of ICA [23]. The sources are continuous and follow a Beta distribution, while the mixing matrix has continuous coefficients. The binary ICA model is linear and the Bernoulli data likelihood is approximated through variational inference. The reconstruction of the sources is evaluated in a denoising task by the Area Under the Curve for reconstructing the 0s from the original data.

The method used in this thesis is a linear model that, while having limited expressivity compared to higher-capacity models such as neural networks, would complement the binary ICA literature that has been heavily focused on OR or XOR-based models. Our approximation resembles most the work from Kaban and Bingham but focuses on the mixing matrix reconstruction.

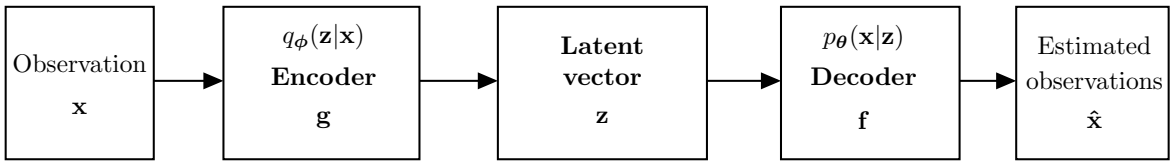

Figure 2.1: Illustration of an autoencoder model.

2.3 Variational Autoencoders

Variational Autoencoders (VAEs) [35, 27] are generative and unsupervised deep latent variable models whose efficiency for large datasets has unleashed applications in many fields that require synthetic data. They have also significantly advanced the field of semi-supervised learning, especially in image classification tasks. This unprecedented success comes from the fact that VAEs are not only generative models, but one of their main functions is to perform representation learning by creating a lower-dimension representation that encodes the maximum amount of information of the data.

VAEs differ from other deep generative models, such as Generative Adversarial Networks (GANs) [12], by providing a principled way to generate samples: they allow for more control over the generation by having a prior on the latent space. We study the probabilistic perspective of how they solve variational inference with the use of deep neural networks.

In this section, we describe the basics of variational autoencoders, presented as a method that employs autoencoders to perform variational inference, which is made practically possible with a reparameterization trick to define a stochastic estimator. We define each of these elements and explain how they are combined into a VAE model. Subsection 2.3.1 introduces autoencoders as deep neural networks that can approximate functions, compress and decompress data. Subsection 2.3.2 introduces variational inference to estimate the posterior. Subsection 2.3.3 introduces how autoencoders are used in variational inference to approximate the likelihood and posterior, enabling sampling. Subsection 2.3.3 introduces the evidence lower bound loss function that is used in VAEs, and also how the optimization is performed with backpropagation.

2.3.1 Autoencoders

Autoencoders, depicted in 2.1, use neural networks as function approximators. We first introduce a probabilistic interpretation of how the model functions. Consider two neural networks, $\mathbf{g}$ and $\mathbf{f}$, with parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ respectively, such parameters being their weights and biases. In the autoencoder framework, they are respectively called *encoder* and *decoder*. The encoder $\mathbf{g}_\phi$ compresses the input $\mathbf{x}$ into a smaller represen-

tation that should contain the most information about the input. This representation, $\mathbf{z} = \mathbf{g}_{\boldsymbol{\phi}}(\mathbf{x})$, is referred to as a *latent vector*. The decoder $\mathbf{f}_{\boldsymbol{\theta}}$ maps the latent vector back into the original input dimensions, forming the *reconstruction* $\mathbf{x}' = \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{g}_{\boldsymbol{\phi}}(\mathbf{x}))$. Thus, autoencoders are particularly suitable for learning the joint distribution of observed and latent variables.

One way to express this compression is the following. When the input is an image, $\mathbf{x} \in \mathbb{R}^m$ can be seen as the vectorized representation of the three-dimensional array (tensor) of the RGB channels of the image. Analogously, the latent representation can also be vectorized into $\mathbf{z} \in \mathbb{R}^p$. In this comparison, $p < m$ since the latent representation should be smaller than the input.

Ideally, the reconstruction should be very similar to the input, and the process is optimized to do so by minimizing the loss function $\mathcal{L}$, which can be simply the mean squared error (MSE) between the input and its reconstruction. Considering n input elements, the loss is given by:

$$\mathcal{L}_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x}) = \frac{1}{n}\|\mathbf{x} - \mathbf{f}_{\boldsymbol{\theta}}(\mathbf{g}_{\boldsymbol{\phi}}(\mathbf{x}))\|_2^2. \tag{2.13}$$

In practice, the encoder is usually modeled as a deep neural network that has sequentially fewer neurons as the layers go deeper until the final layer. The decoder, then, has the reverse architecture than the decoder [15]. When the input data are images, convolutional layers are especially useful, since they account for spatial correlations, being able to identify from low-level features (such as corners and edges) in the first layers to high-level features (such as objects) in the final layers [28].

2.3.2 Variational Inference

A variational autoencoder employs the autoencoder model for variational inference. In order to understand variational inference, we recall Bayes' theorem: given two random variables x and z, the conditional density of z given x is:

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} \tag{2.14}$$

in which $p(z)$ is the prior, $p(x|z)$ is the likelihood, and $p(z|x)$ is the posterior. In the continuous case, the marginal probability density $p(x)$ is given by

$$p(x) = \int p(x|z)p(z)dz. \tag{2.15}$$

Usually, the likelihood and the prior can be computed easily; the problem lies in evaluating the posterior in a computationally tractable manner.

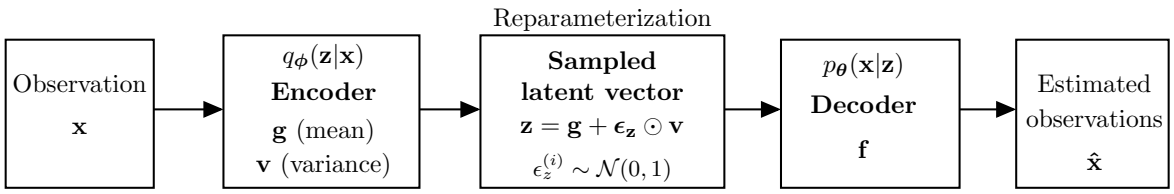

Figure 2.2: Illustration of a VAE that uses a multivariate Gaussian reparametrization.

Let continue using the notation where $\mathbf{x}$ is a vector of observed variables, $\mathbf{z}$ is a vector of latent variables, and $\boldsymbol{\phi}$ is a vector of model parameters. Variational inference proposes to do inference on a known distribution $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$*, and adjust the parameters $\boldsymbol{\phi}$ in order to approximate $q_{\boldsymbol{\phi}}$ to $p(\mathbf{z}|\mathbf{x})$. A divergence metric is, then, used to evaluate how similar the two probability distributions are. The Reverse Kullback-Leibler (KL) divergence is widely used for this purpose

$$\begin{aligned} D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \,||\, p(\mathbf{z}|\mathbf{x})) &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})}\right] \\ &= \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{p(\mathbf{z}|\mathbf{x})} d\mathbf{z}, \end{aligned} \tag{2.16}$$

which can be rewritten as:

$$\begin{aligned} D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \,||\, p(\mathbf{z}|\mathbf{x})) &= \log p(\mathbf{x}) + \Big(\int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{p(\mathbf{z},\mathbf{x})} d\mathbf{z} \Big) \\ &= \log p(\mathbf{x}) + D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \,||\, p(\mathbf{z})) - \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log p(\mathbf{x}|\mathbf{z}). \end{aligned} \tag{2.17}$$

Therefore, variational inference transforms statistical inference (of the form $a|b$) into optimization problems: finding parameters $\boldsymbol{\phi} \in \Phi$ that minimize a loss function $\mathcal{L}$. In latent-variable models, hidden variables are interpreted as priors over to the observed variables, and this connection will be explored in the next section.

2.3.3 Variational Autoencoders

Using large datasets, deep neural network models are suitable solutions for the optimization of large parameter spaces. Thus, being an appropriate candidate to perform variational inference. We recall autoencoders from subsection 2.3.1 and formulate the variational inference problem as an autoencoder that has prior $p_{\boldsymbol{\theta}}(\mathbf{z})$ over the latent variables, likelihood $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$, and posterior $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$, where the densities are parameterized by $\boldsymbol{\theta}$ [41, 25].

Let us consider the problem of how to generate a sample similar to the real data $\mathbf{x}^{(i)}$. First, we draw a sample from the prior density function, $\mathbf{z}^{(i)} \sim p_{\boldsymbol{\theta}^*}(\mathbf{z})$. Then, we

*This notation is equivalent to $q(\mathbf{z}|\mathbf{x};\boldsymbol{\phi})$ and will be used throughout this thesis.

draw a sample from the likelihood density function by using the prior sample from the first step: $\mathbf{x}^{(i)} \sim p_{\boldsymbol{\theta}^*}(\mathbf{x}|\mathbf{z} = \mathbf{z}^{(i)})$. In this example, $\boldsymbol{\theta}^*$ are the estimated parameters, which are optimized by maximizing the marginal likelihood of the data

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{n} p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}). \tag{2.18}$$

As discussed previously, to infer $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$, the following integral would have to be computed:

$$p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{z}) d\mathbf{z}. \tag{2.19}$$

However, such an expression is computationally intractable in practice. Instead, $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ is approximated by $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$, $\boldsymbol{\phi}$ being the parameterization of the function q. In the VAE model, $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is the **probabilistic encoder**, and $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ is the **probabilistic decoder**, as depicted in Figure 2.2. We can also call $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ the **generative model**, and $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ the **inference model**.

As previously described, the Reverse KL divergence is used to "measure the distance" between two probability distributions, so it is suitable to be used in the loss of a VAE. The KL divergence between the approximation and the true density, $D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \,||\, p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}))$, should be minimized with respect to $\boldsymbol{\phi}$. Then, further developing equations 2.16 and 2.17 [41], we have:

$$D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \,||\, p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})) = \log p_{\boldsymbol{\theta}}(\mathbf{x}) + D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \,||\, p_{\boldsymbol{\theta}}(\mathbf{z})) - \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}). \tag{2.20}$$

Or:

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) - D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \,||\, p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})) = \mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) - D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\boldsymbol{x}) \,||\, p_{\boldsymbol{\theta}}(\mathbf{z})). \tag{2.21}$$

Therefore, the VAE loss function $\mathcal{L}_{\text{VAE}}$ is given by:

$$\begin{aligned} \mathcal{L}^{\text{VAE}}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) &= -\log p_{\boldsymbol{\theta}}(\mathbf{x}) + D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \,||\, p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})) \\ &= -\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) + D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \,||\, p_{\boldsymbol{\theta}}(\mathbf{z})), \end{aligned} \tag{2.22}$$

where the term $-\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ is the cross-entropy between $q_{\boldsymbol{\phi}}$ and $p_{\boldsymbol{\theta}}$. Often, $D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \,||\, p_{\boldsymbol{\theta}}(\mathbf{z}))$ is solved for the Gaussian case when $p_{\boldsymbol{\theta}}(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and $q_{\boldsymbol{\phi}}(\mathbf{z}^{(i)}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ [27]:

$$-D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \,||\, p_{\boldsymbol{\theta}}(\mathbf{z})) = \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log\left((\sigma_j)^2\right) - (\mu_j)^2 - (\sigma_j)^2\right). \tag{2.23}$$

Then, we must find optimal parameters $\boldsymbol{\theta}^*, \boldsymbol{\phi}^*$ that minimize the loss:

$$\boldsymbol{\theta}^*, \boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\theta},\boldsymbol{\phi}} \mathcal{L}^{\text{VAE}}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}). \tag{2.24}$$

This loss is known as the *evidence lower bound* (ELBO). By minimizing the loss, the lower bound $-\mathcal{L}^{\text{VAE}}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) \leq \log p_\theta(\mathbf{x})$ is maximized.

VAEs assume a dataset $\mathcal{D} = \left\{\mathbf{x}^{(i)}\right\}_{i=1}^{N}$ of independent and identically distributed samples. The ELBO loss is, then, applied for every data point in $\mathcal{D}$ [25]:

$$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathcal{D}) = \sum_{\mathbf{x}\in\mathcal{D}} \mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}). \tag{2.25}$$

The sharing of parameters across data points can make the model significantly more efficient than methods that would optimize the loss separately for each data point.

Yet, since sampling operations are not differentiable, a reparameterization is necessary and it happens by a change of variable. The solution is to have the random vector $\mathbf{z}$ as a function of another random vector $\boldsymbol{\epsilon}$ at the encoder $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$:

$$\mathbf{z} = \mathbf{g}(\boldsymbol{\epsilon}, \mathbf{x}) \tag{2.26}$$

where $\mathbf{g}$ is a deterministic and **differentiable** function, and $\boldsymbol{\epsilon}$ is independent noise. Then, a Monte Carlo estimator is used to adapt the ELBO loss with the new $\mathbf{z}$. This allows us to approximate the expectation of a function $\mathbf{f}(\mathbf{z})$ with respect to $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ as:

$$\begin{aligned} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})}[\mathbf{f}(\mathbf{z})] &= \int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})\mathbf{f}(\mathbf{z})d\mathbf{z} \\ &\approx \frac{1}{l}\sum_{j=1}^{l} \mathbf{f}(\mathbf{g}(\boldsymbol{\epsilon}^{(j)}, \mathbf{x}^{(i)})) \end{aligned} \tag{2.27}$$

where $\boldsymbol{\epsilon}^{(j)}$ are samples from the independent noise distribution. This expectation is the first term of the loss from equation 2.22 and is where this approximation is applied.

The Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ is typically used to model the transformation $\mathbf{g}$, for its conveniency. For this reason, Gaussian noise is also common in VAEs. On practical terms, first the noise random variable is sampled $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and then $\mathbf{z}$ is computed using such auxiliary random variable: $\mathbf{z} = \boldsymbol{\mu}(\mathbf{x}) + \boldsymbol{\sigma}(\mathbf{x}) \odot \boldsymbol{\epsilon}$, where $\odot$ stands for element-wise multiplication. Importantly, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are transformations learned by the inference model, usually implemented as encoders. The reparameterization trick is also illustrated in Figure 2.2.

Then, the parameters are updated by differentiating $\nabla_{\boldsymbol{\theta},\boldsymbol{\phi}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x})$. Finally, this optimization can be executed by different algorithms, such as stochastic gradient descent or Adam [26], and backpropagation is used to compute the gradients, as in regular

deep learning models. Software libraries like TensorFlow [1] and PyTorch [34] implement automatic differentiation and also provide a modular framework for the design of the models.

Ultimately, VAEs can be seen as autoencoders with a specific regularization term, and that can be seen directly from their loss [25]:

$$\tilde{\mathcal{L}}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x};\boldsymbol{\epsilon}) = \underbrace{\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})}_{\text{Negative reconstruction error}} + \underbrace{\log p_{\boldsymbol{\theta}}(\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}_{\text{Regularization terms}}. \tag{2.28}$$

Disentanglement

Unfortunately, such a model is unidentifiable: although two parameterizations imply the same $p(\mathbf{x})$, they do not imply the same $p(\mathbf{x}, \mathbf{z})$ and, as a result, we cannot discover the prior, decoder or encoder. Identifiability is one interpretation and formalization of the concept of disentanglement, which is supported by Identifiable VAEs (iVAEs) [24, 30].

The definition of disentanglement is widely discussed in the representation learning literature [3]. For example, in image generation, disentangled latent factors would consist of colors, lighting, scale, position, among other factors. Conceptually, according to Ridgeway, a disentangled representation must be [36]:

- **explicit** – the mapping between the representation and the true factors of variation can be implemented simply, such as with a linear classifier;
- **modular** – each representation unit is associated with only one semantic factor of variation;
- **compact** – each semantic factor of variation is represented using only one or a few representation units.

This concept has been formalized by Higgins *et al.* using group and representation theory connected to symmetry transformations.

Disentangled representations result in more interpretable and generalizable VAE models [13], and have also been developed to obtain fair representations [8]. A few different models have been developed to achieve disentanglement properties in VAEs, each one encompassing its own set of definitions. The first VAE model to discuss disentanglement was the Beta VAE [13], and other examples are the Beta Total Correlation VAE [6], and the model used in this thesis, the Identifiable VAE [24].

3. Methods

This chapter introduces the methods used and developed in this project. Section 3.1 starts with a detailed description of identifiable Variational Autoencoders (iVAEs), an existing model that is the basis of our work. Then, the method used in this thesis is introduced: the iVAE for binary data. It is a linear model that adds a few modifications to the continuous iVAE such that it is more suitable for the binary task.

Section 3.2 presents the evaluation scores to be used in the empirical evaluation of the experiments in Chapter 4. The Mean Correlation Coefficient (MCC) is presented as the standard evaluation score of iVAEs, assessing how well it reconstructs the sources. The Mean Cosine Similarity (MCS) is a score that we introduce to evaluate the mixing matrix reconstruction by iVAEs from binary data. Lastly, we discuss why the MCS is an appropriate score to be used in this evaluation by arguing that the reconstruction of the sources (and hence, the use of the MCC) is hampered in the given binarization process.

3.1 Identifiable Variational Autoencoders

The *identifiable VAE* (iVAE) formalizes the idea of disentanglement by defining it as equivalent to identifiability in nonlinear ICA [24]. It integrates a VAE with nonlinear ICA by conditioning the latent prior $\mathbf{z}$ on another observed variable $\mathbf{u}$, such as time index. The main assumption of the method is that the prior on the latent variables is conditionally factorial given the additionally observed auxiliary variable $\mathbf{u}$, such that its probability density function is $p(\mathbf{z}|\mathbf{u}) = \prod_i p_i(z_i|\mathbf{u})$. The iVAE is proved to be identifiable. Previous studies have demonstrated empirically that an arbitrary generative model with a factorized (non-conditional) prior cannot achieve disentanglement with unsupervised learning [30]; this provides intuition for why auxiliary variables used in the iVAE can be effective, by imposing such conditional factorization. As has been described in Chapter 2, the use of auxiliary variables has also been reported to achieve identifiability in nonlinear ICA [21], also based on the conditional mutual independence of the latent variables given the auxiliary variables. In the VAE framework, this is implemented as a probabilistic encoder estimating $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{u})$, and a probabilistic decoder

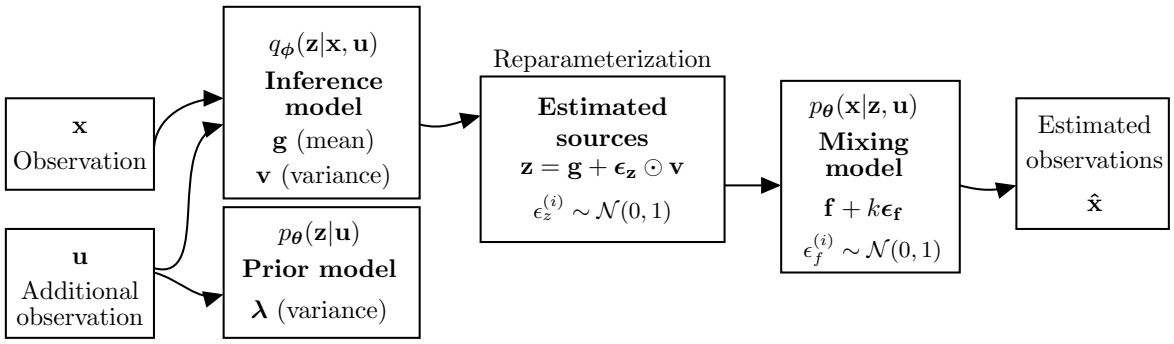

Figure 3.1: Continuous iVAE illustration. In VAE terminology: the inference model is equivalent to the encoder, and the mixing model is equivalent to the decoder. The iVAE uses an additionally observed variable $\mathbf{u}$ to estimate the inference model. Additionally, the iVAE estimates a *prior model* for the sources given such additionally observed variables. The mixing model also specifies a factor k that controls the level of Gaussian noise in the mixture. Variables in bold under the model names denote the transformations learned by the model and are described in detail in the text.

approximating $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x},\mathbf{u})$. The ELBO loss on a data distribution $\mathcal{D}(\mathbf{x}|\mathbf{u})$ is given as:

$$\mathcal{L}^{\text{iVAE}}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) = \mathbb{E}_{q_\mathcal{D}}[\mathbb{E}_{\mathbf{z}\sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})}[\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}|\mathbf{u}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})]]. \tag{3.1}$$

The performance metric utilized to evaluate the iVAE is the mean correlation coefficient (MCC) [24], which is widely used in ICA. It compares the original sources with the corresponding latent variables sampled from the iVAE posterior. The iVAE achieves an MCC score of above 95% with the number of latent variables varying from 2 to 5. In comparison, the VAE and VAE methods that aim to achieve disentanglement, the Beta-VAE [13], and the Beta-TC-VAE [6] have much lower performance as the number of latent variables increases—getting closer or equal to the number of observed variables—for example, with an MCC value below 70% when the number of latent variables is the same as the number of observed variables. Most importantly, identifiability is theoretically guaranteed in the iVAE given the model assumptions. The sources can be **identified** up to permutation and pointwise nonlinearity [24].

The iVAE uses nonstationary data, in which a *segment* contains a set of data points whose distribution is stationary, while the distribution across segments is nonstationary. Therefore, the additionally observed variable $\mathbf{u}$ is simply the segment ID of the observation.

Since the model is based on the VAE, we highlight a few conceptual differences introduced by the iVAE. First of all, the latent variables of the VAE are seen as sources from ICA, and both terms may be used interchangeably throughout this thesis. Second, the VAE's *decoder* $p(\mathbf{x}|\mathbf{z})$ is replaced by $p(\mathbf{x}|\mathbf{z},\mathbf{u})$, now called *mixing model*, since is equivalent to the mixing model of ICA explained in Chapter 2: it mixes the estimated sources to generate estimated observations. Third, the *inference model* of the VAE

$q(\mathbf{z}|\mathbf{x})$ is developed in the iVAE to also consider the additionally observed variable, $q(\mathbf{z}|\mathbf{x}, \mathbf{u})$. Finally, the iVAE introduces an additional model that does not exist in the VAE: the prior model $p(\mathbf{z}|\mathbf{u})$ models the sources given the auxiliary variables. All of these modifications are illustrated in Figure 3.1, and the plate diagrams illustrating the directed graphical model are depicted in Figures 3.2 and 3.3. Under a Bayesian interpretation, $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}, \mathbf{u})$ is the posterior, $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}, \mathbf{u})$ is the likelihood, and $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{u})$ is the prior.

The observed distribution is

$$p(\mathbf{x}|\mathbf{u}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z}|\mathbf{u})d\mathbf{z}. \tag{3.2}$$

This model is parameterized by $\boldsymbol{\theta} = (\mathbf{f}, \mathbf{T}, \boldsymbol{\lambda})$, where $\mathbf{f}$ parameterizes the mixing model, while the remaining parameterize the prior.

Importantly, the prior model factorizes when conditioned on $\mathbf{u}$, such that each source $z_i \in \mathbf{z}$ follows a univariate exponential family distribution given the auxiliary variable $\mathbf{u}$:

$$\begin{aligned} p(\mathbf{z}|\mathbf{u}) &= \prod_i p(z_i|\mathbf{u}) \\ &= \prod_i \frac{Q_i(z_i)}{Z_i(\mathbf{u})} \exp\left[\sum_j T_{i,j}(z_i)\lambda_{i,j}(\mathbf{u})\right]. \end{aligned} \tag{3.3}$$

where Q_i is the base measure, $T_{i,j}$ are the components of the sufficient statistics, Z_i is the normalizing constant, and $\lambda_{i,j}$ the parameters. Here, the parameters $\boldsymbol{\lambda}$ and the normalizing constant Z_i crucially depend on $\mathbf{u}$.

The sources can be **identified** up to permutation and pointwise nonlinearity [24].

Estimation

Let $q_{\mathcal{D}}(\mathbf{x}, \mathbf{u})$ be the empirical data distribution. The model learns by maximizing a lower bound $\mathcal{L}$ of the data log-likelihood

$$\mathbb{E}_{q_{\mathcal{D}}(\mathbf{x},\mathbf{u})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{u})] \geq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}). \tag{3.4}$$

The loss function is:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) := \mathbb{E}_{q_{\mathcal{D}}(\mathbf{x},\mathbf{u})}[\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}|\mathbf{u}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}, \mathbf{u})]] \tag{3.5}$$

where, further,

$$p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}|\mathbf{u}) = p_{\mathbf{f}}(\mathbf{x}|\mathbf{z})p_{\mathbf{T},\boldsymbol{\lambda}}(\mathbf{z}|\mathbf{u}). \tag{3.6}$$

To compute the loss function, the expectation over the data distribution is implemented an average over data samples. In order to deal with expectation over $q_{\phi}(\mathbf{z}|\mathbf{x},\mathbf{u})$, we use the reparametrization trick and *sample* a number of $\mathbf{z}$ vectors from $q_{\phi}(\mathbf{z}|\mathbf{x},\mathbf{u})$.

The mixing model is given by

$$\mathbf{x} = \mathbf{f}(\mathbf{z}) + k\boldsymbol{\epsilon}_{\mathbf{f}} \tag{3.7}$$

where $\boldsymbol{\epsilon}_{\mathbf{f}}$ is an independent Gaussian noise variable, $\mathbf{f}$ is estimated by a multilayer perceptron, and the fixed term k^2 (which is not learned) represents the variance of the Gaussian noise from the data generation process. Note that to satisfy identifiability properties, the mixing model is restricted to transformations with no translation; that is, with no bias vector, only a series of matrix transformations and nonlinearities.

The prior model is multivariate Gaussian with zero mean and variance $\exp(\boldsymbol{\lambda}(\mathbf{u}))$. The inference model $q(\mathbf{z}|\mathbf{x},\mathbf{u})$ uses the reparametrization trick:

$$\mathbf{z} = \mathbf{g}(\mathbf{x},\mathbf{u}) + \boldsymbol{\epsilon}_{\mathbf{z}}(\mathbf{x},\mathbf{u}) \tag{3.8}$$

where

$$\boldsymbol{\epsilon}_{\mathbf{z}}(\mathbf{x},\mathbf{u}) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}(\mathbf{x},\mathbf{u})) \tag{3.9}$$

in which $\boldsymbol{\Sigma}(\mathbf{x},\mathbf{u})$ is a diagonal covariance matrix whose elements depend on $\mathbf{x}$ and $\mathbf{u}$.

3.1.1 Identifiable Variational Autoencoder for Continuous Data

The loss function

Following from equation 3.5, we further develop the loss function so that it can be easily implemented and interpreted:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta},\boldsymbol{\phi}) &= \mathbb{E}_{q_{\mathcal{D}}(\mathbf{x},\mathbf{u})}[\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})}[\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}|\mathbf{u}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})]] \\
&= \mathbb{E}_{q_{\mathcal{D}}(\mathbf{x},\mathbf{u})}[\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})}[\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}|\mathbf{u})] - \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})}[\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})]] \\
&= \mathbb{E}_{q_{\mathcal{D}}(\mathbf{x},\mathbf{u})}[\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})}[\log(p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z},\mathbf{u})p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{u}))] - \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})}[\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})]] \\
&= \mathbb{E}_{q_{\mathcal{D}}(\mathbf{x},\mathbf{u})}[\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z},\mathbf{u})] + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})}[\log p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{u})] \\
&\qquad - \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})}[\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})]]
\end{aligned} \tag{3.10}$$

Thus, the loss can be reduced to three terms, each one relating to the inference model, the prior model, and the mixing model respectively. Note that expectations over $\mathbf{z}$ are calculated using the reparameterization trick: we draw $\mathbf{z}$ from $q_{\boldsymbol{\phi}}$ and compute the average. We can use the same set of sampled $\mathbf{z}$ for each term. This loss is the

evidence lower bound (ELBO) conditioned on the additonal observed variable $\mathbf{u}$. By minimizing the loss, the lower bound $-\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}) \leq \log p_\theta(\mathbf{x}|\mathbf{u})$ is maximized. Next, we further develop each one of the terms.

Prior model The loss term related to the prior model can be written as:

$$\begin{aligned} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})}[\log p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{u})] &= \int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u}) \log p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{u}) d\mathbf{z} \\ &\approx \frac{1}{l} \sum_{i=1}^{l} \log \mathcal{N}(\mathbf{z}^{(i)}; \boldsymbol{\eta}(\mathbf{u}), \boldsymbol{\lambda}(\mathbf{u})) \end{aligned} \tag{3.11}$$

where $\mathcal{N}$ is the probability density function of the multivariate Normal distribution, in which $\boldsymbol{\lambda}(\mathbf{u})$ is a vector containing the variances of the diagonal covariance matrix, and $\boldsymbol{\eta}(\mathbf{u})$ is a vector containing the means. We have applied the reparameterization trick from VAEs to approximate the integral can computed by an average of $p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{u})$ over estimated sources $\mathbf{z}$ drawn from $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})$. In the implementation, typically $l = 1$ is used in all the approximations, since that is also used by the VAE [27].

Mixing model (decoder) The loss term related to the mixing model can be written as:

$$\begin{aligned} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z},\mathbf{u})] &= \int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u}) \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z},\mathbf{u}) d\mathbf{x} \\ &\approx \frac{1}{l} \sum_{i=1}^{l} \log \mathcal{N}(\mathbf{x}; \mathbf{f}(\mathbf{z}^{(i)}), 0.1\mathbf{I}) \end{aligned} \tag{3.12}$$

where $\mathbf{z}^{(i)}$ is drawn from $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})$ in the reparametrization trick, $\mathcal{N}$ is the probability density function of the multivariate Normal distribution with mean vector $\mathbf{f}(\mathbf{z}^{(i)})$ and diagonal covariance matrix $0.1\mathbf{I}$. The mixing model outputs the estimated observations $\hat{\mathbf{x}} = \mathbf{f}(\mathbf{z}^{(i)}) + 0.1\boldsymbol{\epsilon}_{\mathbf{f}}$, where $\boldsymbol{\epsilon}_{\mathbf{f}} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. That is exactly the noisy ICA model, where 0.1 is the standard deviation of the Gaussian noise, as illustrated in Figure 3.1.

Inference model (encoder) The loss term regarding the inference model is:

$$\begin{aligned} \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})}[\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})] &= \int q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u}) \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u}) d\mathbf{z} \\ &\approx \frac{1}{l} \sum_{i=1}^{l} \log \mathcal{N}(\mathbf{z}^{(i)}; \mathbf{g}(\mathbf{x},\mathbf{u}), \mathbf{v}(\mathbf{x},\mathbf{u})) \end{aligned} \tag{3.13}$$

where $\mathbf{v}(\mathbf{x},\mathbf{u})$ is a vector containing the variance values from the diagonal covariance matrix, $\mathbf{g}(\mathbf{x},\mathbf{u})$ is the vector containing the means, and $\mathbf{z}^{(i)}$ is drawn from $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})$ as in the reparameterization trick.

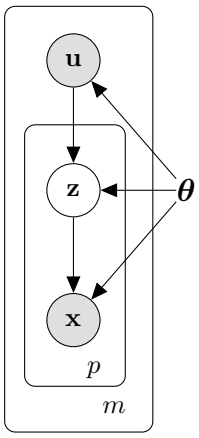

Figure 3.2: Directed graphical model of the iVAE. Solid lines denote the generative model $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z},\mathbf{u})p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{u})$, and shaded nodes denote observed variables as opposed to latent variables. The data contains m segments and p points per segment.

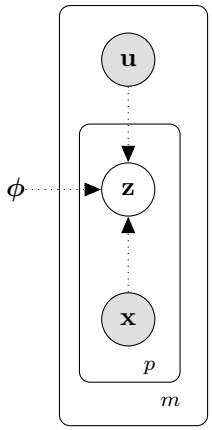

Figure 3.3: Directed graphical model of the iVAE. Dashed lines denote the variational approximation $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})$, and shaded nodes denote observed variables as opposed to latent variables. The data contains m segments and p points per segment.

3.1.2 Identifiable Variational Autoencoder for Binary Data

To further develop iVAEs for binary data—sometimes referred to as binary iVAEs for short—, we notice that we are working with a factorized Bernoulli observational model. The loss terms developed previously in the continuous iVAE model can remain the same for the inference model and the prior model. However, the loss term referring to the **mixing model** should be modified, since the data follows a **multivariate Bernoulli distribution**. We draw $\mathbf{z}^{(i)} \sim q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})$ using the output of the inference model in the reparameterization trick $\mathbf{z}^{(i)} = \mathbf{g}(\mathbf{x},\mathbf{u}) + \mathbf{v}(\mathbf{x},\mathbf{u}) \odot \boldsymbol{\epsilon}^{(i)}$. Thus, the loss term can be

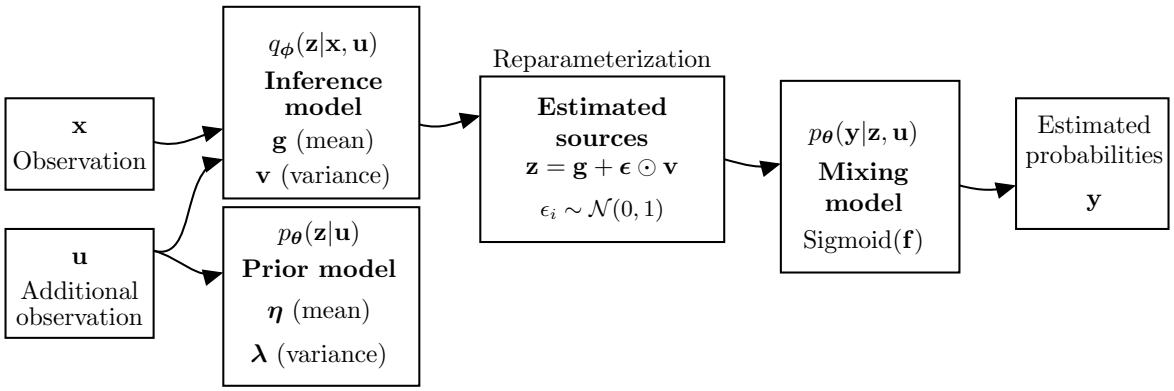

Figure 3.4: Binary iVAE illustration. In VAE terminology: the inference model is equivalent to the encoder, and the mixing model is equivalent to the decoder. The iVAE uses an additionally observed variable $\mathbf{u}$ to estimate the inference model. Additionally, the iVAE estimates a "prior" model for such additionally observed variables. Different from the continuous iVAE, the mixing model does not model the noise explicitly. Also in contrast to the continuous iVAE, the outputs of the model are the estimated probabilities, not the estimated observations. To obtain the probability of each element being 1, a Sigmoid function is applied element-wise to the output of the mixing model. Variables in bold under the model names denote the transformations learned by the model and are described in detail in the text.

given as:

$$\begin{aligned}
\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z},\mathbf{u})] &= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x},\mathbf{u})}[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})] \\
&\approx \frac{1}{l}\sum_{j=1}^{l} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}^{(i)}) \\
&= \frac{1}{l}\sum_{i=1}^{l}\sum_{j=1}^{d_x} \log p_{\boldsymbol{\theta}}(x_j|\mathbf{z}^{(i)}) \\
&= \frac{1}{l}\sum_{i=1}^{l}\sum_{j=1}^{d_x} \left[x_j \log y_j^{(i)} + (1-x_j)\log(1-y_j^{(i)})\right] \\
&= \frac{1}{l}\sum_{i=1}^{l}\sum_{j=1}^{d_x} \log \text{Bernoulli}(x_j; y_j^{(i)})
\end{aligned} \tag{3.14}$$

where y_j is the probability of the observation being 1, $0 \leq y_j \leq 1$, and is modeled by applying an element-wise sigmoid function to the continuous output of the linear mixing model, as will be seen in forthcoming subsections. Notice that $\mathbf{y}^{(i)}$ is a function of the estimated sources $\mathbf{z}^{(i)}$ drawn from the estimated posterior. Hence, the expectation is approximated by computing the log-probability mass function of a Bernoulli distribution given such probability y_j.

Binary model

In the model defined, all the transformations are linear, and the sources are drawn from a Normal distribution given their segment. Compared to the Continuous iVAE, which uses nonlinear transformations in all the models, the binary model is linear and introduces changes to the mixing model and to the prior model. The prior model now estimates not only the log-variances but also the means.

When the observed variables are binary, we use a "Bernoulli MLP" [27] as a decoder in the mixing model, which aims to estimate parameters from a Bernoulli distribution instead of a Normal distribution. The mixing model is modified from the continuous case by applying a sigmoid function element-wise to the output of the mixing model. In addition, in the binary case, we do not have an explicit factor accounting for the noise in the mixture.

Following the derivations from section 3.1.1, we give more details about how the iVAE is modeled. First of all, we notice that for simplicity and numerical stability when modeling the variances in both the inference model and the prior model, the transformations model the log-variances, which can easily be converted to the variances via exponentiation. With this trick, even a linear transformation can suffice for modeling the log-variances, thus making the model simpler.

The **prior model** is composed of a transformation modeling the prior mean, and a transformation modeling the prior log-variance. The prior **mean** is modeled by

$$\begin{aligned} \boldsymbol{\eta} : \mathbb{R}^m &\to \mathbb{R}^{d_z} \\ \mathbf{u} &\mapsto \boldsymbol{\eta}(\mathbf{u}) \end{aligned} \tag{3.15}$$

where $\boldsymbol{\eta}$ is an affine transformation. So the vector of means is given by $\boldsymbol{\eta}(\mathbf{u}) = \mathbf{W}_\eta \mathbf{u} + \mathbf{b}_\eta$, with matrix weights $\mathbf{W}_\eta \in \mathbb{R}^{d_z \times m}$, and a bias vector $\mathbf{b}_\eta \in \mathbb{R}^{d_z}$. The prior **log-variance** is modeled by

$$\begin{aligned} \boldsymbol{\lambda} : \mathbb{R}^m &\to \mathbb{R}^{d_z} \\ \mathbf{u} &\mapsto \boldsymbol{\lambda}(\mathbf{u}) \end{aligned} \tag{3.16}$$

where $\boldsymbol{\lambda}$ is an affine transformation. The vector of log-variances is given by $\boldsymbol{\lambda}(\mathbf{u}) = \mathbf{W}_\lambda \mathbf{u} + \mathbf{b}_\lambda$, in which $\mathbf{W}_\lambda \in \mathbb{R}^{d_z \times m}$ are the weights, and $\mathbf{b}_\lambda \in \mathbb{R}^{d_z}$ are the biases. Notice that $\boldsymbol{\lambda}$ is unrelated to the notation from the exponential family, since we are modeling both the means and variances.

The **mixing model** learns a transformation

$$\begin{aligned} \mathbf{f} : \mathbb{R}^{d_z} &\to \mathbb{R}^{d_x} \\ \mathbf{z} &\mapsto \mathbf{f}(\mathbf{z}) \end{aligned} \tag{3.17}$$

where $\mathbf{f}$ is a linear transformation resulting in the the continuous output $\mathbf{f}(\mathbf{z}) = \mathbf{W}_f\mathbf{z}$, in which $\mathbf{W}_f \in \mathbb{R}^{d_x \times d_z}$ is the matrix of weights. Then, the probability of the estimated observed variables is given by

$$\mathbf{y} = \text{Sigmoid}(\mathbf{W}_f\mathbf{z}). \tag{3.18}$$

It is important to notice that each element of $\mathbf{y}$ is an individual probability of the particular observed variable being 1, $\{y_i = P(x_i = 1)\}_{i=1}^{d_x}$.

The **inference model** has a transformation modeling the mean, and a transformation modeling the log-variance of the data. The data **mean** is modeled by

$$\begin{aligned} \mathbf{g} : \mathbb{R}^{d_x+m} &\to \mathbb{R}^{d_z} \\ (\mathbf{x}, \mathbf{u}) &\mapsto \mathbf{g}(\mathbf{x}, \mathbf{u}) \end{aligned} \tag{3.19}$$

where $\mathbf{g}$ is an affine transformation. We denote the concatenation of the vectors $\mathbf{x}$ and $\mathbf{u}$ as $\mathbf{x}||\mathbf{u}$. The vector of means is given by $\mathbf{g}(\mathbf{x}, \mathbf{u}) = \mathbf{W}_g(\mathbf{x}||\mathbf{u}) + \mathbf{b}_g$, for a matrix $\mathbf{W}_g \in \mathbb{R}^{d_z \times (d_x+m)}$, and a bias vector $\mathbf{b}_g \in \mathbb{R}^{d_z}$. The data **log-variance** is modeled by

$$\begin{aligned} \mathbf{v} : \mathbb{R}^{d_x+m} &\to \mathbb{R}^{d_z} \\ (\mathbf{x}, \mathbf{u}) &\mapsto \mathbf{v}(\mathbf{x}, \mathbf{u}) \end{aligned} \tag{3.20}$$

where $\mathbf{v}$ is an affine transformation. The vector of log-variances is given by $\mathbf{v}(\mathbf{x}, \mathbf{u}) = \mathbf{W}_v(\mathbf{x}||\mathbf{u}) + \mathbf{b}_v$, where $\mathbf{W}_v \in \mathbb{R}^{d_z \times d_x+m}$ are the weights and $\mathbf{b}_v \in \mathbb{R}^{d_z}$ the biases.

3.2 Evaluation

3.2.1 Mean Correlation Coefficient

To evaluate the reconstruction of the sources, which is particularly relevant for the continuous iVAE, the **Mean Correlation Coefficient (MCC)** is implemented as in traditional ICA literature [17, 24] by first computing Pearson's correlation between the true source $\mathbf{z} = (z_1, \ldots, z_n)$ and the estimated source $\hat{\mathbf{z}} = (\hat{z_1}, \ldots, \hat{z_n})$ for a batch of n data points:

$$r(\mathbf{z}, \hat{\mathbf{z}}) = \frac{\sum_{i=1}^{n} (z_i - \bar{z}) \left(\hat{z}_i - \bar{\hat{z}}\right)}{\sqrt{\sum_{i=1}^{n} (z_i - \bar{z})^2} \sqrt{\sum_{i=1}^{n} \left(\hat{z}_i - \bar{\hat{z}}\right)^2}}, \tag{3.21}$$

where $\bar{\hat{z}} = \frac{1}{n}\sum_{i=1}^{n} \hat{z}_i$ is the sample mean of the estimated sources and $\bar{z}$ is the sample mean of the true sources. Hence, the MCC is defined as:

$$\text{MCC}(\mathbf{z}, \hat{\mathbf{z}}) = \max_p \frac{1}{d_z} \sum_{i=1}^{d_z} \mid r(\mathbf{z}_i, \hat{\mathbf{z}}_{p[i]}) \mid, \tag{3.22}$$

where $\hat{\mathbf{z}}_{p[i]}$ denotes a permutation on the source $\hat{\mathbf{z}}$ in use.

For a practical implementation, we refer to a combinatorial optimization solution to this problem. The range of Pearson's correlation coefficient is [-1, 1], so we use the absolute coefficient value. This will define the cost function $\mathbf{C} \in \mathbb{R}^{d_z \times d_z}$ from the matrix of correlation coefficients $\mathbf{R} \in \mathbb{R}^{d_z \times d_z}$:

$$\mathbf{C}(\mathbf{z}, \hat{\mathbf{z}}) = -|\mathbf{R}(\mathbf{z}, \hat{\mathbf{z}})|, \tag{3.23}$$

where the negative sign is used in order to transform the minimization problem into a maximization one. A linear sum assignment problem is solved using the auction algorithm in order to determine the components that maximize such score.

3.2.2 Mean Cosine Similarity

In the binary case, it is more relevant to evaluate the estimated **mixing matrix** than the sources, since the binarization process adds much more noise than simply adding Gaussian noise to the observations – as will be shown in the next subsection. For this purpose, a similar procedure to the MCC is applied between the estimated mixing matrix and the true mixing matrix, with a few considerations.

When there are only two components, the mixing matrix $\mathbf{A} \in \mathbb{R}^{2\times 2}$ can be written considering its column vectors $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2]$. Each vector contains only two elements, so the correlation coefficient cannot be used, since $r(\mathbf{v}_1, \mathbf{v}_2) = 1 \quad \forall\ \mathbf{v}_1, \mathbf{v}_2 \in \mathbb{R}^2$. In addition, even if $d > 2$, the MCC is undesired because by subtracting the means of each vector, the correlation between "shifted" vectors is the same as if they were not shifted: $r(\mathbf{v}_1 + \mathbf{d}, \mathbf{v}_2) = r(\mathbf{v}_1, \mathbf{v}_2)$ for any $\mathbf{d} \in \mathbb{R}^2$.

Therefore, we employ the **Mean Cosine Similarity (MCS)** instead of the MCC. The MCS uses the cosine similarity – instead of the correlation coefficient – to determine whether the vectors of the true and estimated matrices are aligned.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{a}_1 \cdot \mathbf{a}_2}{\|\mathbf{a}_1\|\|\mathbf{a}_2\|} = \frac{\sum_{i=1}^{n} a_1^{(i)} a_2^{(i)}}{\sqrt{\sum_{i=1}^{n} a_1^{(i)2}}\sqrt{\sum_{i=1}^{n} a_2^{(i)2}}}. \tag{3.24}$$

Let us denote the i^{th} column of a matrix $\mathbf{A} \in \mathbb{R}^{d\times d}$ as $A[, i]$. In the MCS calculation, we aim to compare each column of $\mathbf{A}$ with each column of the estimated matrix $\hat{\mathbf{A}}$, thus getting a pair-wise cosine similarity. For simplicity, we consider a column permutation p of matrix $\hat{\mathbf{A}}$ as $\hat{\mathbf{A}}[, p[i]]$. We compute the mean cosine similarity across all the columns for each permutation, and take the maximum of all the means, hence defining the MCS:

$$\text{MCS}(\mathbf{A}, \hat{\mathbf{A}}) = \max_p \left(\frac{1}{d} \sum_{i=1}^{d} |\ \cos(\mathbf{A}_{[,i]}, \hat{\mathbf{A}}_{[,p[i]]})\ | \right). \tag{3.25}$$

In the general case of a matrix $\mathbf{A} \in \mathbb{R}^{d_x \times d_z}$,

$$\text{MCS}(\mathbf{A}, \hat{\mathbf{A}}) = \max_p \left(\frac{1}{d_z} \sum_{i=1}^{d_z} | \cos(\mathbf{A}_{[,i]}, \hat{\mathbf{A}}_{[,p[i]]}) | \right). \tag{3.26}$$

Similar to the MCC, we use combinatorial optimization for a scalable implementation. For each pair of matrices, we consider all possible combinations of **columns** – as will be explained at the end of this subsection – from each matrix, and record all such pairwise cosine similarities in a matrix $\mathbf{E} \in \mathbb{R}^{d_z \times d_z}$. Since the cosine similarity range is [-1,1], we take the absolute cosine similarity. Then, the cost function $\mathbf{C} \in \mathbb{R}^{d_z \times d_z}$ considers the negative similarities for the linear assignment problem.

$$\mathbf{C}_{i,j}(\mathbf{a}_i, \mathbf{a}_j) = -|\mathbf{E}_{i,j}(\mathbf{a}_i, \mathbf{a}_j)| \quad \forall\ i, j \in D_z \tag{3.27}$$

The rest proceeds in the same way as the MCC of the sources.

Following is an example of why we take the **columns** (and not the rows) of $\mathbf{A}$ as a guarantee of scale and order indeterminacies. Consider the two-dimensional case; $\mathbf{x_1}, \mathbf{x_2}, \mathbf{z_1}, \mathbf{z_2}$ are row vectors. Only in this example, a, b, c, d are redefined to new constants! Let

$$\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{pmatrix} = \begin{pmatrix} a\mathbf{z}_1 + b\mathbf{z}_2 \\ c\mathbf{z}_1 + d\mathbf{z}_2 \end{pmatrix} \tag{3.28}$$

When the sources are scaled by constants e and f, we need to have the same scaling in the columns of $\mathbf{A}$.

$$\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} a\mathbf{z}_1 + b\mathbf{z}_2 \\ c\mathbf{z}_1 + d\mathbf{z}_2 \end{pmatrix} = \begin{pmatrix} ea & fb \\ ec & fd \end{pmatrix} \begin{pmatrix} \mathbf{z}_1/e \\ \mathbf{z}_2/f \end{pmatrix} \tag{3.29}$$

Similarly, when the order of the sources is switched, the order of the columns of $\mathbf{A}$ needs to be switched as well.

$$\begin{pmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{pmatrix} = \begin{pmatrix} b\mathbf{z}_2 + a\mathbf{z}_1 \\ d\mathbf{z}_2 + c\mathbf{z}_1 \end{pmatrix} = \begin{pmatrix} b & a \\ d & c \end{pmatrix} \begin{pmatrix} \mathbf{z}_2 \\ \mathbf{z}_1 \end{pmatrix} \tag{3.30}$$

Remark: If two vectors $\mathbf{a}_1$ and $\mathbf{a}_2$ are orthogonal, $\langle \mathbf{a}_1, \mathbf{a}_2 \rangle = 0$. As a result, if two matrices $\mathbf{A}$ and $\hat{\mathbf{A}}$ are orthogonal (that is, their columns are orthonormal), then $\text{MCS}(\mathbf{A}, \hat{\mathbf{A}}) = 0$. This is important because it accounts for the orthogonal indeterminacy in the mixing matrix to be described in future sections.

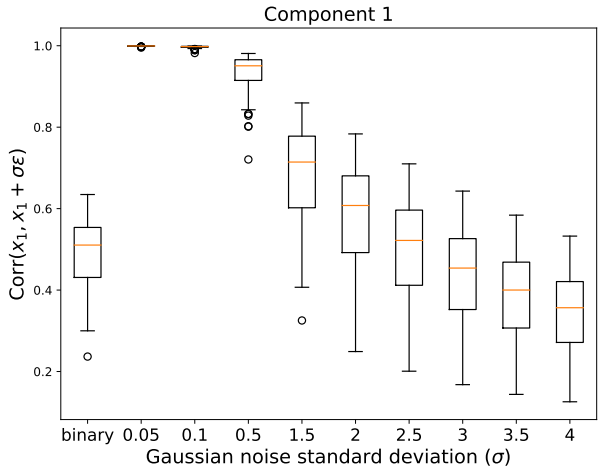

Figure 3.5: Quantification of the noise introduced in the binarization process compared to continuous Gaussian noise. The correlation between the continuous observation and the binary observation is as strong as the correlation of the continuous observation with Gaussian noise of variance 9.

3.2.3 Noise quantification

We aim to study how much noise the binarization process introduces to the observations and to compare it with the continuous case, which has an explicit Gaussian noise model. For such, we check the correlation coefficient of the observations before and after adding noise: $\text{Cor}(\mathbf{x}_i, \mathbf{x}_i + k\epsilon)$. In the continuous case, the noise comes from a Gaussian distribution from which different standard deviation values are evaluated. By default, the standard deviation is $k = 0.1$. In the binary case, the noise is produced by "binarizing" the continuous observations when sampling from a Bernoulli distribution.

Figure 3.5 illustrates the study of the correlation applied to a dataset that has two observed variables and two sources, with results shown for one of the components. We observe that the correlation coefficient in the binary case corresponds to Gaussian noise generated with a standard deviation of 3. This is a significant difference compared to the standard deviation of 0.1 typically used in the continuous case, which enlightens how much more difficult the binary problem is. Therefore, we conclude that the estimation of the sources and evaluation using the MCC is not viable for the binary model.

4. Experiments

This chapter presents the empirical evaluation of the linear binary iVAE model introduced in Chapter 3. Section 4.1 describes the procedure to generate the synthetic data utilized in the experiments. Section 4.2 briefly outlines the computing environment where the experiments were performed, as well as specific details about the implementation of certain methods and optimizers. Section 4.3 presents the main questions investigated in this thesis and the results regarding the performance of the method. Finally, Section 4.4 discusses in detail the results, premises, interpretations, and future work.

4.1 Data generation

In the data generation, first generate continuous data and then binarize it. We denote the set $\mathbb{D}_z$ to contain all the source component indexes $\mathbb{D}_z = \{1, 2, \ldots, d_z\}$ and the set $\mathbb{D}_x$ to contain all the observation component indexes $\mathbb{D}_x = \{1, 2, \ldots, d_x\}$. The data comprises sources $\mathbf{z} = \{z_i\}_{i=1}^{d_z}$, observed variables $\mathbf{x} = \{x_i\}_{i=1}^{d_x}$, and an additional observed variable $u = 1, 2, \ldots, m$. We define the set containing all the possible values of u as $\mathbb{U} = \{1, 2, \ldots, m\}$. The original sources $\mathbf{s} = \{s_i\}_{i=1}^{d_z}$ are drawn from a Normal distribution.

$$s_i \sim \mathcal{N}(0, 1) \quad \forall\, i \in \mathbb{D}_z \tag{4.1}$$

A new vector $\mathbf{s}$ is generated for each new sample, n times, originating the matrix $\mathbf{S} \in \mathbb{R}^{d_z \times n}$.

Consider j to be a single segment ID of u. Each segment, of each source, has its mean drawn from a Uniform distribution

$$\mu_i^{(j)} \sim \mathcal{U}(-b, b) \quad \forall\, i \in \mathbb{D}_z,\ j \in \mathbb{U}, \tag{4.2}$$

where b is typically 0.5, as will be justified. In the continuous case, it is not necessary that the segments have different means. Experiments with zero-mean segments $\boldsymbol{\mu}_i = \mathbf{0}$ are also performed.

Similarly, the segments' standard deviation is drawn from an uniform distribution

$$\sigma_i^{(j)} \sim \mathcal{U}(0.5, 3) \quad \forall\ i \in \mathbb{D}_z,\ j \in \mathbb{U}. \tag{4.3}$$

Thus, $\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i \in \mathbb{R}^m \quad \forall\ i \in \mathbb{D}_z$. Hence, the sources are "modulated" by the mean and variance for each segment:

$$z_i^{(k,j)} = \sigma_i^{(j)} \cdot s_i^{(k,j)} + \mu_i^{(j)} \quad \forall\ i \in \mathbb{D}_z,\ j \in \mathbb{U},\ k = \{1, \ldots, p\}. \tag{4.4}$$

From the equation above, we can notice that the variance of the sources are given as $\mathbb{V}(\mathbf{z}_i) = \boldsymbol{\sigma}_i^2 \mathbb{V}(\mathbf{s}_i) \quad \forall\ i \in \mathbb{D}_z$.

The mixing matrix is $\mathbf{A} \in \mathbb{R}^{d_x \times d_z}$. Each of its elements is drawn from a Uniform distribution $\mathcal{U}(-1, 1)$. The mixing matrices should be well-conditioned for the ICA problem to be solvable. We recall that the condition number a matrix is the ratio of the magnitude of its largest and smallest eigenvalue, $\text{cond}(\mathbf{A}) = \dfrac{|\lambda_{\max}|}{|\lambda_{\min}|}$, where $\lambda_{\max}$ and $\lambda_{\min}$ are the maximal and minimal (by absolute value) eigenvalues of $\mathbf{A}$. For example, we establish the threshold for the condition number of $\mathbf{A}$ to be 2 when $\mathbf{A} \in \mathbb{R}^{2\times 2}$. A new mixing matrix should be drawn until $\text{cond}(\mathbf{A}) \leq 2$. As the dimension of A grows, the condition number threshold grows as well. Therefore, in the general case, we estimate a threshold on condition number such that the matrices generated with condition numbers smaller or equal to it result in well-conditioned matrices. We sample 1000 mixing matrices, order them by their condition number, and set the condition number threshold to the maximum value in the 20^{th} percentile.

Finally, the observed variables are generated as usual in Independent Component Analysis by multiplying the sources by the mixing matrix:

$$x_i = \sum_{j \in \mathbb{D}_z} a_{ij} s_j \quad \forall\ i \in \mathbb{D}_x. \tag{4.5}$$

In matrix notation, we consider all the n data pints in the dataset $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^n$, $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$, so

$$\mathbf{X} = \mathbf{A}\mathbf{Z} \tag{4.6}$$

where $\mathbf{Z} \in \mathbb{R}^{d_z \times n}$, and $\mathbf{X} \in \mathbb{R}^{d_x \times n}$.

Gaussian noise of variance $k^2 = 0.01$ is added to the observations:

$$\epsilon_i \sim \mathcal{N}(0, 1) \quad \forall\ i \in \mathbb{D}_x \tag{4.7}$$

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + k\boldsymbol{\epsilon}_i \quad \forall\ i \in \mathbb{D}_x. \tag{4.8}$$

Lastly, we one-hot encode u to treat it as a categorical variable. We denote the set of one-hot encoded vectors as $\mathbb{B} = \{\mathbf{e}_i\}_{i=1}^m$, where $\mathbf{e}_i$ is a vector of the standard basis of

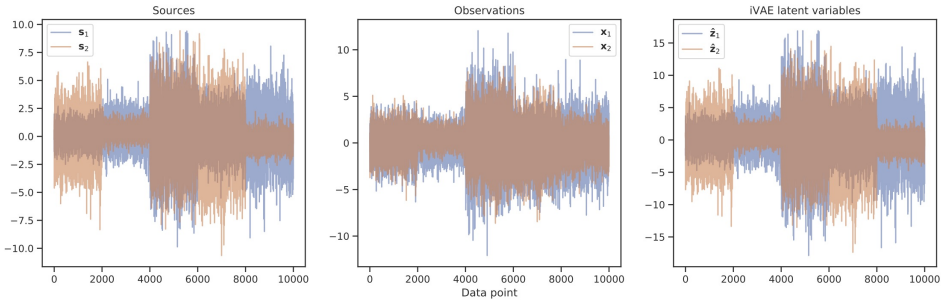

Figure 4.1: Illustration of the continuous data signals for 2 observed variables and 2 sources. Sources, observations, and reconstructed sources are depicted. The reconstructed sources visually resemble the true sources.

the m-dimensional Euclidean space $\mathbb{R}^m$. The matrix of additional observed variables is $\mathbf{U} \in \mathbb{B}^n$.

As opposed to when the observations are continuous, with **binary** data it is necessary to change the segment means, besides their variance, in order to generate less homogeneous joint distributions $p(\mathbf{x}_1, \mathbf{x}_2)$. Thus, the mean of each segment is drawn from a Uniform distribution:

$$\mu_i^{(j)} \sim \mathcal{U}(-b, b) \quad \forall\, i \in \mathbb{D}_z,\ j \in \mathbb{U}. \tag{4.9}$$

Suitable values for b are in the interval $[0.5, 3]$, as will be shown in Figure 4.3.

In addition, the following changes are introduced to the observations. First, the continuous outputs of the mixing model x_i^c are binarized by first transforming the mixing model output into a probability. The probability of each observed variable being 1, $p(x_i = 1)$, is defined by applying the sigmoid function element-wise to the observations:

$$y_i = P(x_i = 1) = \frac{1}{1 + e^{-x_i^c}} \quad \forall\, i \in \mathbb{D}_x. \tag{4.10}$$

Then, binary observations are drawn from a Bernoulli distribution where the probability of success is y_i. Such procedure is applied element-wise:

$$x_i \sim \text{Bernoulli}(y_i) \quad \forall\, i \in \mathbb{D}_x. \tag{4.11}$$

Figures 4.1 and 4.2 illustrate an example of the sources and observations in the continuous case, as well as the iVAE reconstruction of the sources. The plots refer to the case when both the number of observed variables and of sources is 2.

Evaluating the non-stationarity We develop a score to measure the non-stationarity of the data. For each segment u, we compute the joint distribution of the

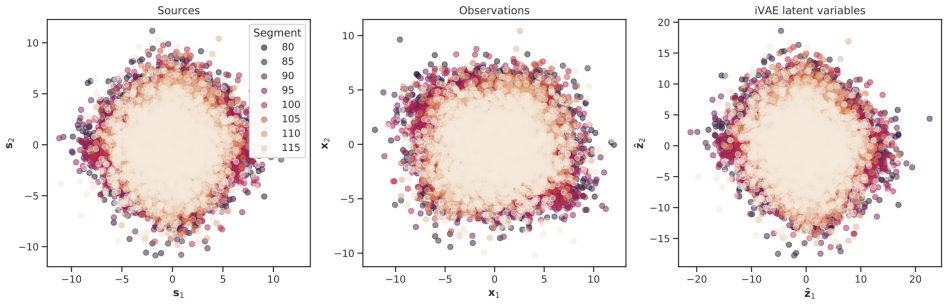

Figure 4.2: Illustration of the continuous data signals for 2 observed variables and 2 sources. Sources, observations, and reconstructed sources are depicted. The reconstructed sources visually resemble the true sources.

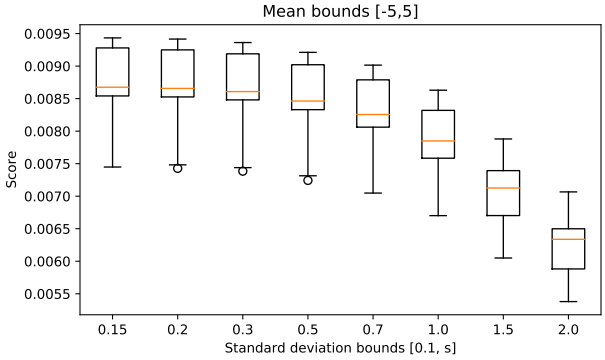

Figure 4.3: Non-stationarity score as a function of the standard deviation bounds, while the means are drawn from Uniform(-5, 5).

observed variables given the segment ID $P(x_1, x_2, x_3, x_4, x_5|u)$ for $(x_1, x_2, x_3, x_4, x_5) \in \{0, 1\}^5$. Where the d-ary Cartesian power of the binary set $\{0, 1\}^5$ is defined in the implementation as a list containing $2^5 = 32$ elements, and each element having 5 entries: $\{(0, 0, 0, 0, 0), (0, 0, 0, 0, 1), \ldots, (1, 1, 1, 1, 1)\}$.

Since there are 40 segments, the joint distribution computation described above will result in 40 vectors, each one with 2^5 entries. Then, for each assignment of the d-ary Cartesian power $(x_1, x_2, x_3, x_4, x_5)$, we compute the variance over u. That is, we form a 40-element vector that gives probabilities (0,0,0,0,0) for each value of u. Then we take the variance of this vector. We repeat this procedure for the other elements of the 5-ary Cartesian power, and since we get 2^5 different variance values, we take the average of them. This is the non-stationarity score.

In the general case, the score f is given as:

$$f = \frac{1}{||C||} \sum_{i=1}^{d} \mathbb{V}(P(x_1, \ldots, x_d|u) \quad \forall\ (x_1, \ldots, x_d) \in C \tag{4.12}$$

where d is the number of observed variables, C is the d-ary Cartesian power $C = \{0, 1\}^d$, and $||C||$ is the length of C (the total number of elements in it).

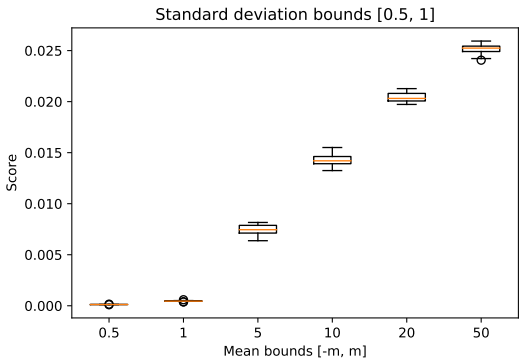

Figure 4.4: Non-stationarity score as a function of the mean bounds, where the standard deviation is drawn from Uniform(0.5, 1.0).

Figures 4.3 and 4.4 illustrate how this non-stationarity score can auxiliate in selecting the bounds for the mean and variance of the distributions. We notice that the bound value from which to draw the means cannot be too big, since that can result in many of the sampled values being close to the bound. Since such high values are then binarized, they can result in joint probabilities $p(x_1 = 0, x_2 = 0; u)$ and $p(x_1 = 1, x_2 = 1; u)$ being very high—in a model with two observed variables as an example.

4.2 Experimental setup

We implemented the model in PyTorch [34] and conducted the experiments at the Puhti AI Cluster, using Intel Xeon Processors and Nvidia Volta V100 GPUs running at 2.1 GHz. Most experiments were performed using 10 CPU cores and 8 GB of main memory. Experiments to study how the method scales to high dimensions use 1 GPU core coupled with 4 CPU cores and 8 GB of memory.

We used the L-BFGS optimizer implemented in PyTorch, which is inspired by the *minFunc* function [38]. The functions are evaluated on the full dataset, as opposed to mini-batches. The update history size is 200, a line search is performed with strong Wolfe conditions, the termination tolerance on first order optimality is 10^{-5}, and the termination tolerance on function value/parameter changes is 10^{-9}.

We ran simulations for 100 different datasets for a given number of observed variables and sources. For each dataset, we ran 3 different learning seeds for the optimization. Then, we selected the learning seed that yielded the lowest final loss and computed the MCS of the mixing matrix for those, as illustrated in Figure 4.5. The learning seed sets both the initial point of the optimization algorithm and the sampling seed for estimating the sources with the inference model. Setting the seeds is needed in

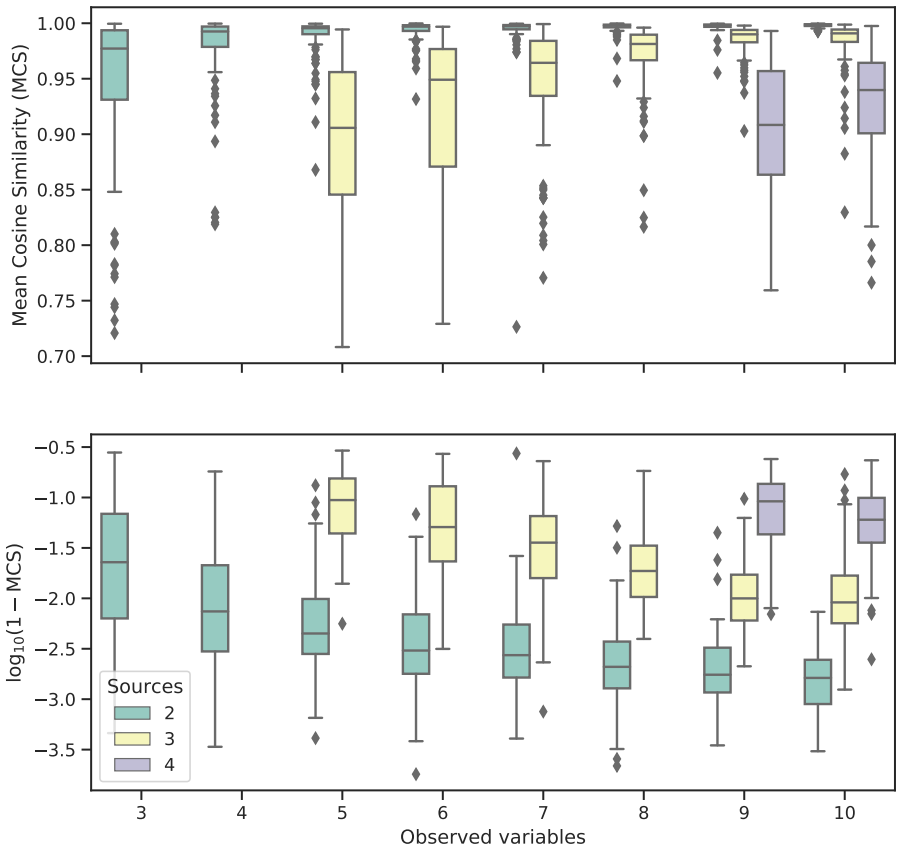

Figure 4.5: Mean Cosine Similarity (MCS) for different combinations of observed variables and sources – under-complete ICA model. On the left, the performance is depicted (the higher, the better): an average MCS of 0.99 is achieved for 2 sources and 6 observed variables or more, and for 3 sources and 10 observed variables. On the right, the log of the estimation error is given (the lower, the better) for more detailed visualization.

order for the results to be reproducible. The optimization results depend on the initial point, so picking the best learning seed is an attempt towards global optimization and avoids convergence to a local minimum.

4.3 Performance

In the following experiments, the questions we study are:

1. Which combinations of sources and observed variables can lead to a high MCS?
2. How time-efficient is the binary iVAE in solving the previous question?

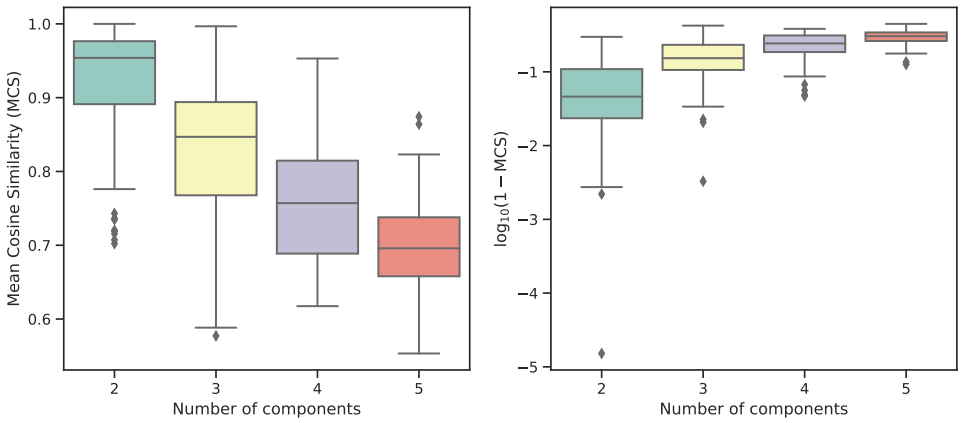

Figure 4.6: Mean Cosine Similarity (MCS) for different numbers of components, where the number of sources is equal to the number of observed variables – complete ICA model. On the left, the performance is depicted (the higher, the better): an average MCS of 0.97 is achieved for 2 components, and the performance degrades as the number of components increases. n the right, the log of the estimation error is given (the lower, the better) for more detailed visualization.

3. How does the number of data samples affect the performance?
4. How does the total number of segments influence the performance of the binary iVAE?

4.3.1 Question 1: Dimensions

For question 1, we evaluate the MCS for different dimensions of observed variables and sources. The datasets are generated with 40 segments and 2000 points per segment.

Figure 4.5 shows results on the under-complete ICA case; that is, when there are more observed variables than sources. For 10 observed variables and 2 sources, it is clear that the MCS achieved is 0.99 for all the cases. On the other hand, when there are 6 observed variables and 2 sources, the average MCS is also 0.99, but there are a few outliers with lower MCS. Such a behavior could be explained by the difficulty of the optimization task, as will be discussed in the next sections. Overall, the plot suggests that for 2 sources, the average MCS is 0.99 when the number of observed variables is between 6 and 10, and the number of outliers with lower MCS decreases as the number of sources increases, which may indicate that the problem becomes more approachable (either because of optimization, statistics, or because the model is appropriate) as the number of observed variables increases. For 3 sources, the average MCS is 0.99 for both 9 and 10 observed variables, but both examples contain a few outliers with lower MCS. When the number of sources is 4, the average MCS is below 0.95 for both 9 and 10 observed variables. These results suggest that the higher is the ratio of observed

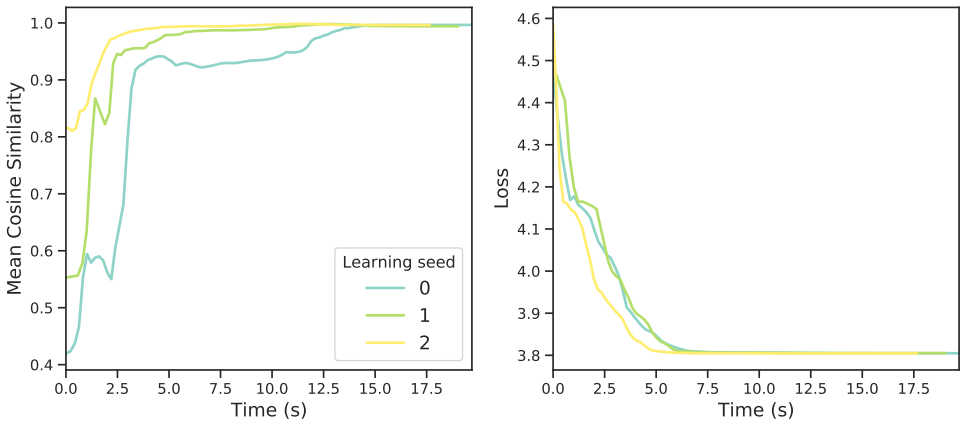

Figure 4.7: Mean Cosine Similarity and loss for different learning seeds for a certain dataset with 6 observed variables and 2 sources. In this particular dataset, all the learning seeds have achieved an MCS score of 0.99.

variables per source, the higher is the MCS. The intuition behind such reasoning is that if the problem is seen as a system of linear equations, more observed variables would relate to more equations in the system for estimating the statistics of interest.

Figure 4.6 shows results on the complete ICA model, when there are as many sources as observed variables – that is, when the mixing matrix is square. It is noticeable that an average MCS of approximately 0.97 is achieved when there are 2 components. The MCS decreases as the number of components increases. Therefore, these results suggest that the method and the model may not be suitable for solving the complete ICA case as such, and also corroborates the hypothesis that the performance is low when the ratio between observed variables and sources is as low as 1.

Lastly, we analyze the optimization procedure by visualizing the learning seeds for particular datasets. For 6 observed variables and 2 sources, Figure 4.5 shows that the method is successful in reconstructing the mixing matrix for most datasets up to the said indeterminacies; one particular dataset is illustrated in 4.7, where all of the learning seeds converge to the same likelihood value and to an MCS value of 0.99. However, Figure 4.5 also displays a few outliers for the same mixing matrix dimensions; this case is illustrated in Figure 4.8, where none of the learning seeds has reached an MCS score above 0.9. This problem may arise due to the learning seed setting both the initial point of the optimizer and also the seed for sampling the estimated sources of the inference model in the reparameterization trick. More examples of this behavior can be found in Figure 4.9, where all the learning seeds achieve an MCS of 0.99 in a dataset with 10 observed variables and 2 sources, and in Figure 4.10, where none of the learning seeds have reached an MCS score of 0.86 in a dataset with 10 observed variables and 4 sources.

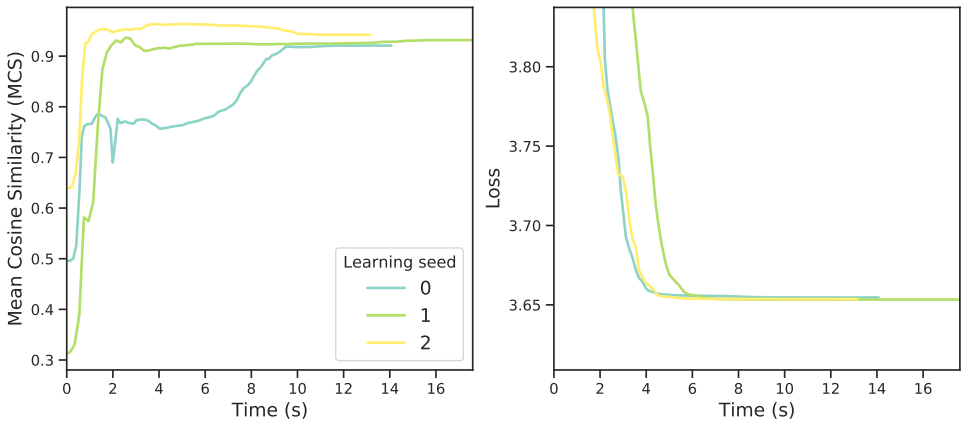

Figure 4.8: Mean Cosine Similarity and loss for different learning seeds for a certain dataset with 6 observed variables and 2 sources. In this particular dataset, all of the learning seeds have achieved an MCS score of 0.9, thus not having succeeded in reconstructing the mixing matrix.

Figure 4.11 offers a comparison of the standard deviation of the final loss and MCS values across the three different learning seeds, for all the combinations of sources and observed variables, and each combination containing 100 datasets. We can observe that the optimization indeed becomes more difficult as the number of dimensions (in both sources and observed variables) increases, as can be seen from the greater variation in the loss on the upper plot. This behavior is plausible since more parameters typically imply more local minima. On the lower plot, it is visible that a higher number of sources results in a higher standard deviation of the MCS, which is reasonable since the estimation error is usually higher when there are more parameters to estimate. The standard deviation of the MCS slightly decreases as the number of observed variables increases, which can be explained by the increase in the statistics available in the data. It would be interesting to verify this in higher dimensions since the effect is not steep in the current plot. Furthermore, we can compare these results with the ones from Figure 4.5, noticing that the cases with low MCS are also the ones where the learning seeds had greater MCS variation.

4.3.2 Question 2: Running time performance

Relevant to how long the method takes to reconstruct the mixing matrix is whether the problem is empirically identifiable (achieves MCS of 0.99) or not. We investigate this question for 6 observed variables and 2 sources. In Figure 4.12, we visualize the MCS and loss of 10 random datasets out of the 100 datasets from question 1. The average running time is of 16.11 ± 0.04 seconds.

We must also check the running time when there are more observed variables and

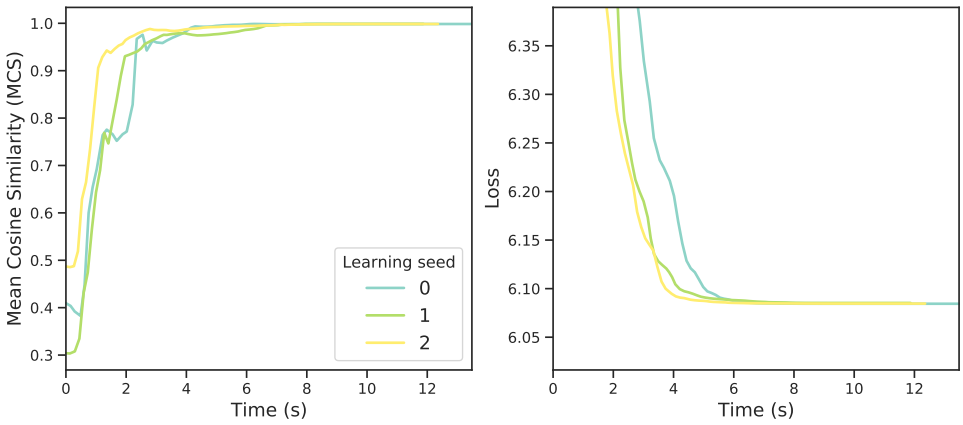

Figure 4.9: Mean Cosine Similarity and loss for different learning seeds for a certain dataset with 10 observed variables and 2 sources. In this particular dataset, all of the learning seeds have achieved a MCS score of 0.99.

more sources, such as with 10 observed variables and 3 sources, as in Figure 4.13. The average running time is of 27.96 ± 0.05 seconds, which is higher than the previous case as expected. It is also visible that, once again, the loss converges to a different value for each dataset.

We analyze the results of a mixing matrix that had not reached a high MCS in question 1, with 10 observed variables and 4 sources. We visualize the first 10 datasets out of 100 for better comprehension. The average running time of the datasets in the figure is of 50.41 ± 1.16 seconds. Interestingly, there is a visible overlap in the loss values of data seeds 0 and 3. In addition, the dataset with the lowest MCS is run for the longest time, 80 seconds.

Finally, Figure 4.15 compares the total running time for different numbers of sources and observed variables, where it is clear that the number of sources is most significant to the increase in the running time.

4.3.3 Question 3: Sample size

In question 3, we study how the number of data samples affects the performance of the binary iVAE. We study the case with 6 observed variables and 2 sources since they have achieved an MCS of 0.99 in previous experiments. We fix the number of segments to 40 and vary the number of points per segment.

As seen in Figure 4.16, the MCS is higher when there are more points per segment. Simultaneously, it is visible that even with 50 points per segment, the average MCS is very close to 0.99. Noticeably, both the variance and the number of outliers with lower MCS is higher with inverse proportion to the number of points per segment.

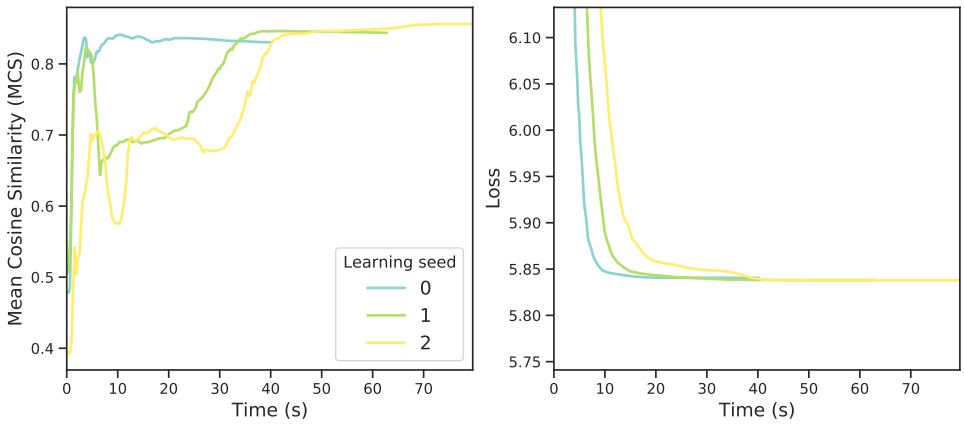

Figure 4.10: Mean Cosine Similarity and loss for different learning seeds for a certain dataset with 10 observed variables and 4 sources. In this particular dataset, all the learning seeds have converged to a MCS score below 0.86.

We can also examine if the estimators are asymptotically consistent. If an estimator is consistent, as the sample size reaches infinity, the estimator will be close to the true parameter with high probability [5]. We cannot claim that any of the estimators is *strictly* consistent because in this case study the number of segments is fixed. The MCS is an indirect evaluation of how close the estimators are to the true parameters up to order and scale indeterminacies, so at least we can note that some of the estimators are more consistent than others. From the plot, we can assert that (up to the scale and order indeterminacies) the estimators are reasonably consistent for 6 observed variables and 2 sources, since a higher number of points per segment presents higher mean, lower variance, and fewer outliers with regards to the MCS. The plot on the right of Figure 4.16 shows that, for 4000 points per segment, the average error is slightly higher than for 2000 points per segment; that can be explained by the challenging optimization task as argued previously, and the fact that the optimization landscape may become less smooth when there are more data points.

4.3.4 Question 4: Number of segments

In question 4, we evaluate how the number of segments affects the performance of the binary iVAE. We divide this study into two parts: when the dataset size is fixed (the number of points per segment decreases as the number of segments increases), and when it increases as the number of segments increases (the number of points per segment is fixed). The first part can be related loosely to the consistency of the estimators, while the second part can be related to the effect of nonstationarity in the estimators.

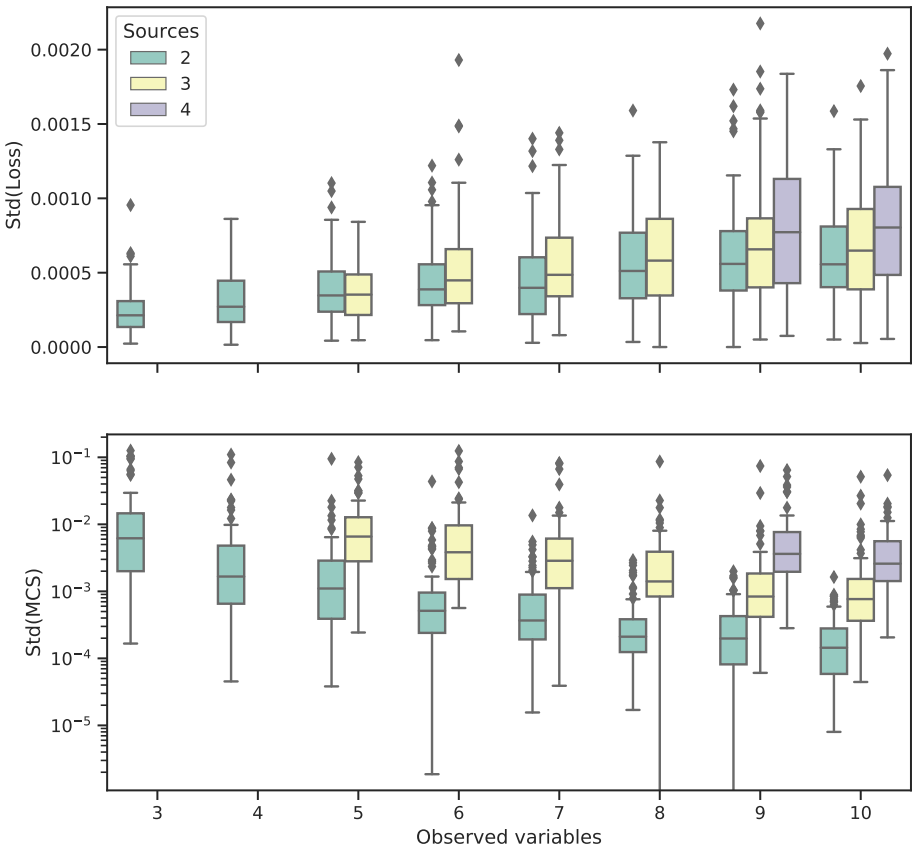

Figure 4.11: Standard deviation of the final loss and Mean Cosine Similarity (MCS) values across different learning seeds. Each bin contains 100 datasets. It is visible that the greater is the number of sources, the higher is the standard deviation on both the loss and the MCS.

Unfixed dataset size

Here, we study how the number of samples affects the performance of the method when the number of points per segment is fixed to 2000. The data consists of 6 observed variables and 2 sources. In Figure 4.17, it is visible that a higher number of segments results in a higher MCS. Notably, when there is only one point per segment, the MCS has a lower mean and significantly higher variance. In this example, the total number of data points also increases as the number of segments increases. The estimator does not seem to be consistent since there is no significant reduction in the error when the number of segments is greater than 20.

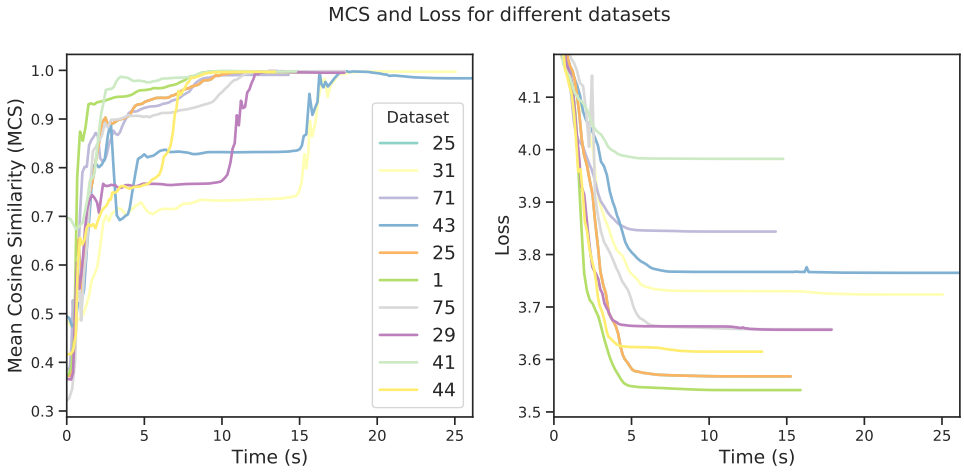

Figure 4.12: Mean Cosine Similarity and Loss for different datasets when the mixing matrix has dimensions 6×2. The average running time is of approximately 16 seconds.

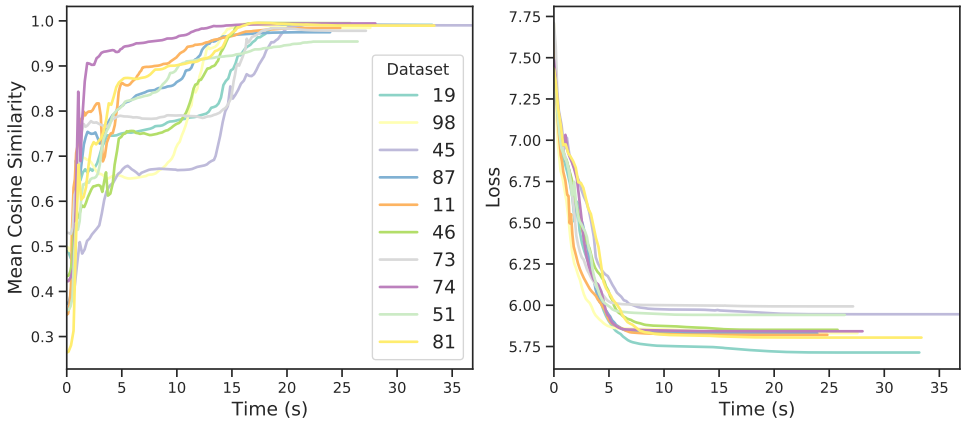

Figure 4.13: Mean Cosine Similarity and Loss for different datasets when the mixing matrix has dimensions 10×3. The average running time is of approximately 28 seconds.

Fixed dataset size

We study the effect of the number of segments on the performance when the dataset size is fixed to 80000 samples. As in previous study questions, the number of observed variables is 6 and the number of sources is 2. In fixing the total number of samples, the number of points per segment decreases at the inverse proportion that the number of segments increases.

In Figure 4.18, it can be observed that the MCS is higher when the number of segments is greater than one. The estimation error decreases as the number of segments increases up to 20 segments, after which the reduction in the error becomes less significant. These results reinforce the importance of non-stationarity in the data for our model, which exploits it through the use of an additionally observed variable.

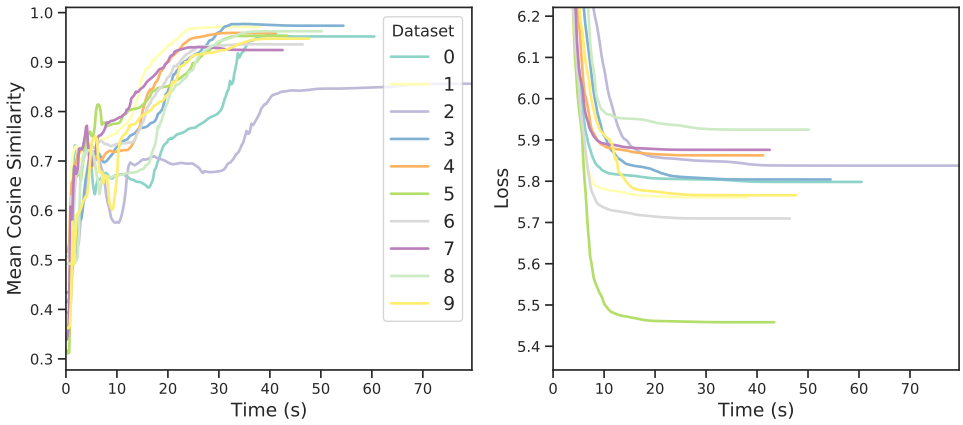

Figure 4.14: Mean Cosine Similarity and Loss for different datasets when the mixing matrix has dimensions 10×4. The average running time is of approximately 50 seconds.

4.4 Discussion

In this section, we further discuss the results of the questions presented in the study of the performance of our method. In question 1, we have learned that the number of observations being higher than the number of sources leads to a higher MCS score. We have determined which dimensions of the mixing matrix have resulted in an MCS of 0.99, but have noticed the presence of outliers due to the challenging optimization task. Experiments with more learning seeds might achieve global optimization and result in fewer outliers. In addition, using a separate sampling seed for the reparameterization trick than the initialization seed could contribute to better results. Another possible reason for the outliers is simply that some datasets may have been generated by ill-conditioned matrices (the condition number threshold estimated was not low enough). Decreasing the percentile of accepted condition numbers would address this problem.

We hope to formalize the findings from question 1 by proving which cases we expect to be identifiable. Finally, we also notice that the MCS achieved is never exactly 1.0 because of the variational approximation inherent to the method and because of the binarization procedure introducing noise to the observations.

In addressing question 2, we have found that the binary iVAE can converge to a high MCS very fast when the number of sources and observed variables is adequate: in the mixing matrix dimensions studied, the runs have taken from 15 to 50 seconds on average. The exact running times have a degree of imprecision since they are affected, for example, by the number of users in the computing node; yet, we consider them reliable and would expect to obtain even lower running times if we had full control of the computing infrastructure. Furthermore, a comparison of the total running time for

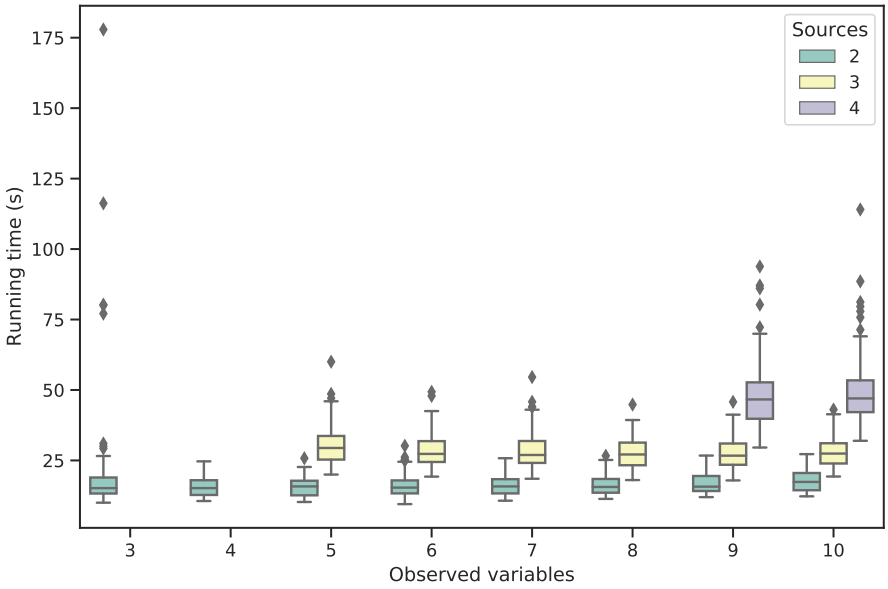

Figure 4.15: Running time for different numbers of sources and observed variables. More sources correlate with higher running times.

each combination of observed variables and sources shows that the total running time is correlated with the number of sources. The running time is expected to be longer when there are more parameters to estimate, and the number of sources determines both the number of columns of the estimated mixing matrix and the number of estimated sources.

The results on the low running time are strongly related to the optimization procedure. The L-BFGS optimizer using the whole dataset to compute the gradient is particularly appropriate in this case. Other optimization algorithms that could have been used are Gradient Descent and Stochastic Gradient Descent with the Adam optimizer. However, Gradient Descent is very sensitive to the step size and an effective step-size adaptation method can be difficult to design. On the other hand, SGD-based methods like Adam offer a solution for learning rate adaptation, but the stochasticity of the method is also present in the results, which are often less interpretable or congruous. L-BFGS offers a good alternative to such issues, offering fast convergence to the optimal parameters. The only drawback is that it can be memory-intensive and, as a result, may not necessarily scale well to high dimensions of data.

In the study of question 3, we find that 1000 points per segment are enough data for our method to estimate a 6×2 mixing matrix given 40 segments. In future work, we will repeat this experiment for different mixing matrices and different numbers of segments. We would expect more consistency in cases with a higher ratio of observed

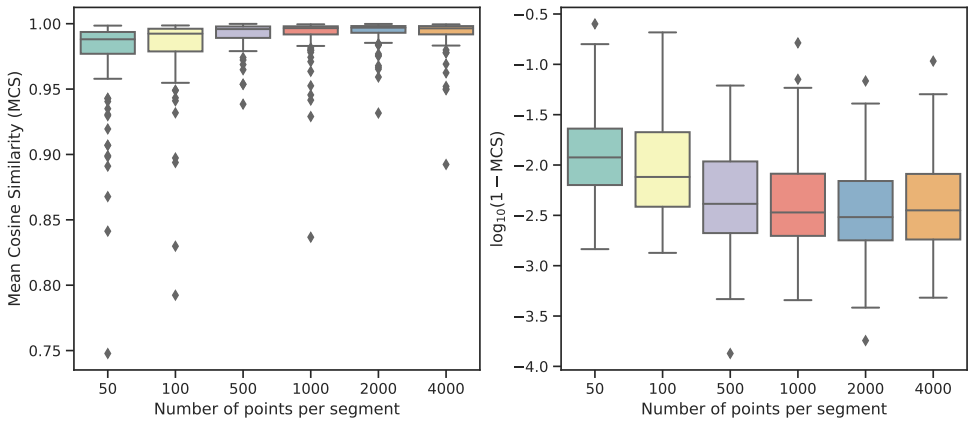

Figure 4.16: On the left, Mean Cosine Similarity (the higher, the better) as a function of the number of points per segment for 6 observed variables, 2 sources, and 40 segments. The performance improves as the number of samples increases, but even 100 points per segment lead to an average MCS close to 0.99. On the right, the log of the estimation error is given (the lower, the better).

variables per source. Furthermore, as explained previously, by utilizing more learning seeds, we hope to improve the optimization and also achieve more consistent results. Similarly, in question 4, we find that 20 segments are enough for our method to estimate a 6×2 mixing matrix with 2000 points per segment. Repetitions of this experiment for different mixing matrices and different numbers of points per segment would offer more insight into the question.

By combining the findings from both question 3 and question 4, we conclude that the estimators for a 6×2 mixing matrix are not consistent. There are a few possible plausible explanations for such a result. First, our method utilizes a variational approximation, therefore we cannot necessarily expect to obtain consistent estimators. Second, the particular mixing matrix studied may not define an identifiable ICA problem, as suggested by the few outliers with MCS below 0.99 from our method; we would hope that a mixing matrix that results in an identifiable problem would offer more reliable results and consistent estimators. Third, the number of datasets used in these experiments may not be enough to illustrate consistency; repeating the experiment with more datasets could yield a consistent plot where the error is visibly lower as the number of segments increases. The importance of having enough datasets under study is due to the variational approximation having a stochastic nature.

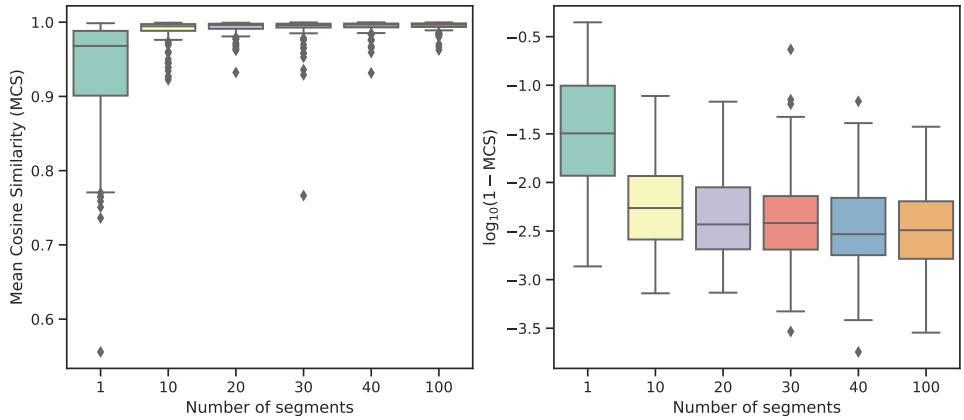

Figure 4.17: On the left, Mean Cosine Similarity (the higher, the better) as a function of the number of segments. A higher number of segments correlate with higher average performance, but this performance improvement is less significant for more than 20 segments. On the right, the log of the estimation error is given (the lower, the better) for more detailed visualization.

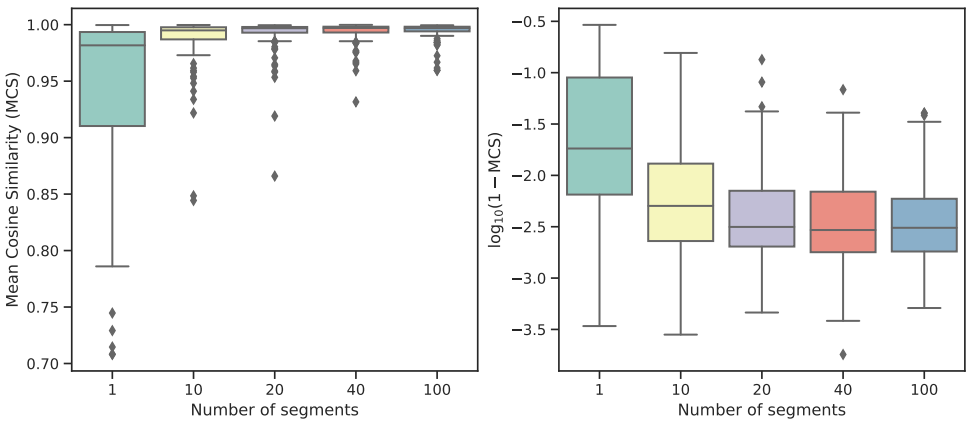

Figure 4.18: Mean Cosine Similarity and estimation error as a function of the number of segments when the total dataset size is fixed to 80000 data points. The performance improvement stabilizes after 20 segments.

5. Conclusions

We have employed a linear model based on the Identifiable Variational Autoencoder, intending to perform ICA on binary data. We are most interested in estimating the mixing matrix that generated the observations. Since the mixing model is linear, we use the Mean Cosine Similarity to evaluate the reconstruction of the mixing matrix, which accounts for order and scale indeterminacies. We obtain the likelihood from the Evidence Lower Bound and perform maximum likelihood estimation. The optimization algorithm employed is L-BFGS, which has demonstrated empirical efficiency in solving the problem under less than a minute in the specified computing environment.

The iVAE approach uses additionally observed variables such that the sources are independent given such auxiliary variables, which can be realistic in many real-world datasets. Furthermore, the various unsupervised learning goals that the continuous iVAE solves also apply to our binary iVAE model. We can approximate the data distribution, generate new samples, and learn useful features. We can also learn the latent variables that generated the data when the ratio of the number of observed variables per source is high enough, but not in the general case due to the high level of noise introduced by the binarization process.

The most important question of this thesis is which combinations of observed variables and sources lead to an MCS of 0.99. This question aims to an interpretation of empirical identifiability that could support the development of an identifiability proof. We have determined the dimensions of the mixing matrix in which the desired performance is achieved, and discovered that the model can identify the mixing matrix when the ratio of observed variables per source is high enough. In such cases, the model can converge very fast. We have also found that the estimators are not generally consistent, which can be justified by the employment of a variational approximation and by the difficulty of the optimization task.

Previous solutions of binary ICA in literature are very limited and identifiability remains largely unexplored, and our linear model has demonstrated effectiveness. A nonlinear model could allow for even more expressivity, but our current method already offers application potential, since many existing binary ICA models are based on OR mixtures. We plan to provide a direct comparison of the performance of the binary

iVAE with other methods in future work, as well as an application to a real-world dataset.

Finally, this method has the potential to be useful in many application areas, such as causal discovery. ICA can correspond to a structural equation model. For example, in the case of a pair of observations, the cause-effect relationship between them can be represented as a transformation of their independent disturbances. From an ICA perspective, the sources can be seen as disturbances, and thus can be used to determine the causal structure. Another interesting outcome of this work is that it has opened the possibility to formalize identifiability in binary ICA and to develop novel methods with maximum likelihood estimation using auxiliary variables, which will be explored in prospective work. Ultimately, we are also interested in extending the model beyond the binary case such that it can model categorical variables in general.

Bibliography

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems. *White paper*, 2015.

[2] A. J. Bell and T. J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Comput.*, 7(6):1129–1159, Nov. 1995.

[3] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[4] J.-F. Cardoso. The three easy routes to independent component analysis, contrasts and geometry. In *Proc. ICA 2001*, pages 1–6, 2001.

[5] G. Casella and R. L. Berger. *Statistical Inference*. Wadsworth and Brooks/Cole, Pacific Grove, CA, 1990.

[6] T. Q. Chen, X. Li, R. Grosse, and D. Duvenaud. Isolating sources of disentanglement in variational autoencoders. *International Conference on Learning Representations, ICLR 2018, Workshop Track Proceedings*, (NeurIPS), 2018.

[7] P. Comon. Independent component analysis, a new concept? *Signal Process.*, 36(3):287–314, Apr. 1994.

[8] E. Creager, D. Madras, J.-H. Jacobsen, M. Weis, K. Swersky, T. Pitassi, and R. Zemel. Flexibly fair representation learning by disentanglement. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1436–1445, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[9] G. Darmois. Analyse générale des liaisons stochastiques: etude particulière de l'analyse factorielle linéaire. *Revue de l'Institut International de Statistique / Review of the International Statistical Institute*, 21(1/2):2–8, 1953.

[10] L. Dinh, D. Krueger, and Y. Bengio. NICE: non-linear independent components estimation. In Y. Bengio and Y. LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, 2015.

[11] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[12] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. In *Conference on Neural Information Processing Systems*, 2014.

[13] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. Beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework. *International Conference on Learning Representations, ICLR*, 2016.

[14] J. Himberg and A. Hyvärinen. Independent component analysis for binary data: An experimental study. In *Proceedings 3rd International Conference on Independent Component Analysis and Blind Signal Separation, ICA2001*, pages 552–556, 2001.

[15] G. E. Hinton. Reducing the dimensionality of data with neural networks. *Science*, 2006.

[16] A. Hyvärinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Trans. Neural Networks*, 10(3):626–634, 1999.

[17] A. Hyvärinen and H. Morioka. Unsupervised Feature Extraction by Time-Contrastive Learning and Nonlinear ICA. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[18] A. Hyvärinen and H. Morioka. Nonlinear ICA of Temporally Dependent Stationary Sources. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 460–469, Fort Lauderdale, FL, USA, 20–22 Apr 2017. PMLR.

[19] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13:411–430, 2000.

[20] A. Hyvärinen and P. Pajunen. Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks*, 12(3):429–439, 1999.

[21] A. Hyvärinen, H. Sasaki, and R. E. Turner. Nonlinear ICA Using Auxiliary Variables and Generalized Contrastive Learning. In K. Chaudhuri and M. Sugiyama, editors, *AISTATS*, volume 89 of *Proceedings of Machine Learning Research*, pages 859–868. PMLR, 2019.

[22] A. Kabán and E. Bingham. ICA-based Binary Feature Construction. In *International Conference, ICA*, 2006.

[23] A. Kabán and E. Bingham. ICA-Based Binary Feature Construction. In *Proceedings of the 6th International Conference on Independent Component Analysis and Blind Signal Separation*, ICA'06, pages 140–148, Berlin, Heidelberg, 2006. Springer-Verlag.

[24] I. Khemakhem, D. P. Kingma, R. P. Monti, and A. Hyvärinen. Variational Autoencoders and Nonlinear ICA: A Unifying Framework. In *International Conference on Artificial Intelligence and Statistics, AISTATS*, 2020.

[25] D. P. Kingma. *Variational Inference & Deep Learning: A New Synthesis*. PhD thesis, 2017.

[26] D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations, ICLR*, 2015.

[27] D. P. Kingma and M. Welling. Auto-encoding Variational Bayes. In *International Conference on Learning Representations, ICLR*, 2014.

[28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, 1998.

[29] D. C. Liu and J. Nocedal. On the Limited Memory BFGS Method for Large Scale Optimization. *Math. Program.*, 45(1-3):503–528, Aug. 1989.

[30] F. Locatello, S. Bauer, M. Lucic, G. Raetsch, S. Gelly, B. Schölkopf, and O. Bachem. Challenging Common Assumptions in the Unsupervised Learning of Disentangled Representations. In *International Conference on Machine Learning, ICML*, 2019.

[31] R. P. Monti, K. Zhang, and A. Hyvärinen. Causal discovery with general non-linear relationships using non-linear ica. In R. P. Adams and V. Gogate, editors, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume

115 of *Proceedings of Machine Learning Research*, pages 186–195. PMLR, 22–25 Jul 2020.

[32] H. Nguyen and R. Zheng. Binary Independent Component Analysis With or Mixtures. *IEEE Transactions on Signal Processing*, 59(7):3168–3181, 2011.

[33] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, NY, USA, second edition, 2006.

[34] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems, NeurIPS*, 2019.

[35] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *International Conference on Machine Learning, ICML*, 2014.

[36] K. Ridgeway and M. C. Mozer. Learning Deep Disentangled Embeddings With the F-Statistic Loss. *Conference on Neural Information Processing Systems, NeurIPS*, 2018.

[37] D. Salamani, T. Golling, S. Gadatsch, G. Stewart, D. Rousseau, and A. Ghosh. Deep generative models for fast shower simulation in ATLAS. 2018.

[38] M. Schmidt. MinFunc: Unconstrained Differentiable Multivariate Optimization in MatLab, 2005.

[39] S. Shimizu, P. O. Hoyer, A. Hyvärinen, and A. Kerminen. A Linear Non-Gaussian Acyclic Model for Causal Discovery. *Journal of Machine Learning Research*, 7, 2006.

[40] R. Vigario, J. Sarela, V. Jousmiki, M. Hamalainen, and E. Oja. Independent Component Approach to the Analysis of EEG and MEG Recordings. *IEEE Transactions on Biomedical Engineering*, 47(5):589–593, 2000.

[41] L. Weng. From Autoencoder to Beta-VAE, 2018.

[42] T. White. Sampling generative networks: Notes on a few effective techniques. *ArXiv*, 1609.04468, 2016.