UCLouvain

epl

École polytechnique de Louvain

Mutual Information Neural Estimation

Algorithm Optimization and Application to Wireless Security

Author: **Samuel AHOU**
Supervisors: **Prof. Muriel MÉDARD, Prof. Claude OESTGES, Prof. Luc VANDENDORPE, Dr. Chia-Yi YEH**
Reader: **Anne-Sophie COLLIN**
Academic year 2023-2024
Master [120] in Electrical Engineering

Abstract

Mutual information is a fundamental concept in information theory that measures the dependency between two random variables. It has found application in many fields such as machine learning, signal processing, neuroscience, etc. The estimation of mutual information is however a challenging problem, especially when the probability distributions of the random variables are unknown. In this work, we present a new mutual information estimator called Mutual Information Neural Estimation (MINE) that was introduced in [1]. The latter is based on the use of neural networks and has been shown to outperform other mutual information estimators in many cases. We also present some optimizations we added to an already existing implementation of MINE and how they improve its performances and user-friendliness. Finally, we present an application of MINE to evaluate the security of a new wireless communication scheme as presented in [2]. We also develop a statistical test in order to determine whether the system is secure or not.

Acknowledgements

I would like to thank several people who have helped me throughout the realization of this work.

First of all, I would like to thank my supervisors, Prof. Muriel Médard, Prof. Claude Oestges, Prof. Luc Vandendorpe and Dr. Chia-Yi Yeh for their guidance and support. Their expertise and advice have been invaluable to me.

I would also like to thank Guillaume Thiran and Anne-Sophie Collin for their priceless help, support and feedback throughout the realization of this work. Their kindness and availability have been a great help to me.

Thank you to my mother, father and brother for their unconditional support and love.

I am also incredibly grateful to all of my friends, and to have met Baptiste Sambon and Jérôme Lafontaine this year, their friendship, love and attention meant the world to me. I thank them for being who they are, kind, loving and so full of life. I pray they get nothing but the best in their life.

ChatGPT [3] and *Github Copilot* [4] have been used to assist in the writing of this work.

Contents

List of Acronyms

Adam	Adaptive Moment Estimation
AWGN	Additive White Gaussian Noise
DV	Donsker-Varadhan
GPU	Graphics Processing Unit
KL	Kullback-Leibler
KSG	Kraskov-Stögbauer-Grassberger
LGDE	Local Gaussian Density Estimation
LR	Learning Rate
MI	Mutual Information
MINE	Mutual Information Neural Estimation
MLMI	Maximum Likelihood Mutual Information
MSE	Mean Squared Error
OFDM	Orthogonal Frequency Division Multiplexing
PCC	Pearson Correlation Coefficient
PDF	Probability Density Function
PMF	Probability Mass Function

QAM	Quadrature Amplitude Modulation
ReLU	Rectified Linear Unit
ReMINE	Regularized Mutual Information Neural Estimation
RMSprop	Root Mean Square Propagation
RV	Random Variable
SGD	Stochastic Gradient Descent
SNR	Signal-to-Noise Ratio
SoTA	State-of-the-Art

Introduction

Mutual information (MI) is a fundamental concept in information theory that was first defined by Claude Shannon in his founding paper *"A mathematical theory of communication"* [5]. MI is a measure of dependency between two random variables (RV). It quantifies the amount of information that can be obtained about one RV when observing another. It is a very versatile metric that can be used to measure dependencies in a more general way than other popular metrics such as linear correlation. Indeed, whereas Pearson's correlation will only measure the linear dependency between two RV, the mutual information will also capture non-linear dependencies.

This metric has found applications in many fields such as machine learning, signal processing, neuroscience, etc. It is a very powerful concept that has been used to solve many problems. Recently, [2] introduced a new wireless communication scheme to protect against eavesdropping. They have shown that under certain conditions, the mutual information an eavesdropper can obtain on the original messages sent is null. This implies that the communication scheme is absolutely secure. This information-theoretic approach of security is very interesting and a strong security guarantee. In this work, we wanted to study the effect of the channel on the mutual information leakage to the eavesdropper. Since for general channels, an analytical expression of the mutual information is not always available, we wanted to use numerical estimators to evaluate it.

Unfortunately, the estimation of mutual information is a challenging problem, especially when the probability distributions of the RV are unknown. In this work, we present a new mutual information estimator called Mutual Information Neural Estimation (MINE) that was introduced in [1]. The latter is based on the use

of neural networks and has been shown to outperform other mutual information estimators especially in terms of tractability in high dimensions. We also present some optimizations we added to an already existing implementation of MINE and how they improve its performances and user-friendliness. Finally, we present the results of this estimator to evaluate the security of the wireless communication scheme presented in [2].

Contributions The main contributions of this work are:

- A characterization of MINE in terms of the effects of the hyperparameters on the performance of the estimator.
- Optimizations of an existing implementation of MINE to improve its performance and user-friendliness in the light of the observations made during the characterization.
- An analysis of MINE to evaluate the security of a new wireless communication scheme presented in [2].
- The development of test procedure to determine whether a system is secure based on the estimations of MINE .

Report Structure In this work, Chapter 1 begins by introducing a theoretical background of mutual information and follows by presenting mutual information estimators, including MINE . Chapter 2 starts by giving a reminder of deep learning and neural networks theory. It then presents a characterization of MINE , i.e. the effects of parameters on the training of the neural network. We follow by presenting the optimizations we added to an existing implementation of MINE . Chapter 3 first explains the Absolute Security communication scheme presented in [2] and the modified system model we considered. We then give some theoretical results on the mutual information of this system that will help us evaluate the performance of MINE by comparing the results we obtained on a synthetic dataset. We also compare results on synthetic data against results on a dataset obtained from an experimental setup we created in laboratory. Finally, we give a statistical test procedure for evaluating the security of a communication system based on the estimations of MINE .

At the beginning of each chapter, a brief introduction is given and at the end a conclusion with future works and perspectives is made.

CHAPTER 1

Mutual Information Estimation

In his founding paper [5], Shannon introduced many fundamental concepts and laid the foundations of communication and information theory. Elements such as information sources, transmitters/receivers, channels and channel capacity, unit of information (bits), etc. are first used as building blocks to characterize communication systems. The concept of entropy and mutual information were also first introduced in this paper.

In this chapter, we are first going to present the concept of mutual information and why is it relevant in many fields. We also explain the challenge of mutual information estimation. We then present a few mutual information estimators that have been developed in the litterature. We are going to be especially interested in a new mutual information estimator called Mutual Information Neural Estimation (MINE) that was introduced in [1]. The latter is based on the use of neural networks and has been shown to outperform other MI estimators especially in terms of tractability in high dimensions (i.e. scalability). It is going to be one of the main focuses of this work and will be studied in more details in the following chapters.

1.1 Mutual Information Background

In this section, we are going to introduce the concepts of entropy and mutual information. We are going to give different mathematical definitions, present the relationships between these two concepts and illustrate their meanings intuitively.

Entropy

Entropy can be understood as the average amount of information or "uncertainty" of a random variable (RV).

Definition 1.1: Entropy for Discrete RV

Let $X \in \mathcal{X}$ be a discrete RV with probability mass function (PMF) $p_X(x)$. The entropy of X is then defined as

$$H(X) = -\sum_{x \in \mathcal{X}} p_X(x) \log(p_X(x)). \tag{1.1}$$

Definition 1.2: Entropy for Continuous RV

Let $X \in \mathcal{X}$ be a continuous RV with probability density function (PDF) $f_X(x)$. The entropy of X is then defined as

$$H(X) = -\int_{x \in \mathcal{X}} f_X(x) \log(f_X(x))\, dx. \tag{1.2}$$

The base of the logarithm dictates the units of the entropy. When using the natural logarithm, the entropy is measured in nats, while using the base 2 logarithm, the entropy is measured in bits. Throughout this work, we will use the logarithm in base 2 in order to express the entropy and mutual information in bits which is more conventional especially in communication technologies.

A very simple example to understand the idea behind entropy is the case of a Bernoulli random variable X with parameter p, modeling a random event with binary outcome 1 or 0 with probabilties p and $1-p$ respectively. The entropy of X is then given by (1.3) and is represented in Figure 1.1.

$$H(X) = -p \log(p) - (1-p) \log(1-p) \tag{1.3}$$

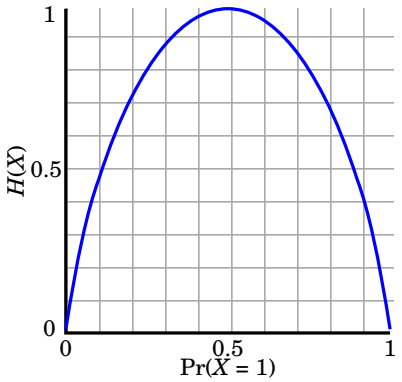


Figure 1.1: Entropy of a Bernoulli RV [6].

We see that the entropy (i.e. the uncertainty of X) is maximal when the probability of the two possible outcomes is equal ($p = 1/2$), and minimal when the probability of one of the outcomes is 1. It makes sense intuitively when we think of the flipping of a coin for example. If the probability of having tails is 0 (the probability of having heads is then 1) the uncertainty of the random event becomes null since the outcome is basically already known. While if the probability of having tails is 1/2, the uncertainty is maximal since we cannot have a priori a preference for one outcome over the other.

Mutual Information

The mutual information between two RVs X and Y is then defined as the reduction of uncertainty of one random variable when observing the other one. Or equivalently, as the average amount of information that can be obtained about X by observing Y, i.e the dependency between random variables.

Definition 1.3: Mutual Information for Discrete RV

Let $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$ be two discrete RVs with joint PMF $p_{XY}(x, y)$ and marginal PMFs $p_X(x)$ and $p_Y(y)$ respectively. The mutual information between X and Y is then defined as

$$I(X;Y) = \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} p_{XY}(x, y) \log \left(\frac{p_{XY}(x, y)}{p_X(x) p_Y(y)} \right). \tag{1.4}$$

Definition 1.4: Mutual Information for Continuous RV

Let $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$ be two continuous RVs with joint PDF $f_{XY}(x, y)$ and marginal PDFs $f_X(x)$ and $f_Y(y)$ respectively. The mutual information between X and Y is then defined as

$$I(X;Y) = \int_{y\in\mathcal{Y}} \int_{x\in\mathcal{X}} f_{XY}(x,y) \log\left(\frac{f_{XY}(x,y)}{f_X(x)f_Y(y)}\right) dx\, dy. \tag{1.5}$$

We can also define it under the lens of the reduction in entropy (uncertainty) of one random variable when observing the other one. This is done by expressing the mutual information as the difference between the entropy of one random variable and the conditional entropy of that random variable given the other one. This relationship is given by Definition 1.5.

Definition 1.5: Mutual Information relation to Entropy

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) \tag{1.6}$$

Looking at the mutual information from this perspective, it is made clearer that the MI is a measure of "how much do we reduce the uncertainty of one RV when observing the other".

Kullback-Leibler Divergence

Another important definition of mutual information can be given using the Kullback-Leibler (KL) divergence. The KL-divergence is a measure of how much one probability distribution diverges from another. It is always positive and is 0 if and only if the two probability distributions are equal.

Definition 1.6: KL-Divergence for Discrete RV

Let $\mathbb{P}$ and $\mathbb{Q}$ be two discrete probability distributions defined on $\mathcal{X}$, with PMFs $p(x)$ and $q(x)$ respectively. The KL-divergence between $\mathbb{P}$ and $\mathbb{Q}$ is then defined as

$$D_{KL}(\mathbb{P}||\mathbb{Q}) = \sum_{x\in\mathcal{X}} p(x) \log\left(\frac{p(x)}{q(x)}\right). \tag{1.7}$$

Definition 1.7: KL-Divergence for Continuous RV

Let $\mathbb{P}$ and $\mathbb{Q}$ be two continuous probability distributions defined on $\mathcal{X}$, with PDFs $p(x)$ and $q(x)$ respectively. The KL-divergence between $\mathbb{P}$ and $\mathbb{Q}$ is then defined as

$$D_{KL}(\mathbb{P}||\mathbb{Q}) = \int_{x\in\mathcal{X}} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx. \tag{1.8}$$

The mutual information can then be expressed as the KL-divergence between the joint distribution $\mathbb{P}_{XY}$ and the product of the marginal distributions $\mathbb{P}_X\mathbb{P}_Y$.

Definition 1.8: Mutual Information as KL-Divergence

Let X and Y be two RVs. Let $\mathbb{P}_{XY}$ be the joint distribution of X and Y, and $\mathbb{P}_X$ and $\mathbb{P}_Y$ be the marginal distributions of X and Y respectively. The mutual information between X and Y can then be expressed as

$$I(X;Y) = D_{KL}(\mathbb{P}_{XY}||\mathbb{P}_X\mathbb{P}_Y). \tag{1.9}$$

Indeed, we know that if X and Y are independent, then $\mathbb{P}_{XY} = \mathbb{P}_X\mathbb{P}_Y$ by definition of independence, and the KL-divergence is then 0.

Mutual information is a powerful concept that has found applications in many fields such as machine learning, signal processing, neuroscience, etc. Its genericity can be used to measure the dependency between RVs in a more general way than other popular metrics such as linear correlation also known as Pearson correlation coefficient (PCC). Indeed, whereas PCC will only measure linear dependencies between RVs, mutual information will also capture non-linear dependencies. Figure 1.2 gives examples of linear correlation values of two RVs X and Y. We see clear examples where X and Y are strongly dependent (non-linearly) but are decorrelated (i.e. $\rho = 0$).

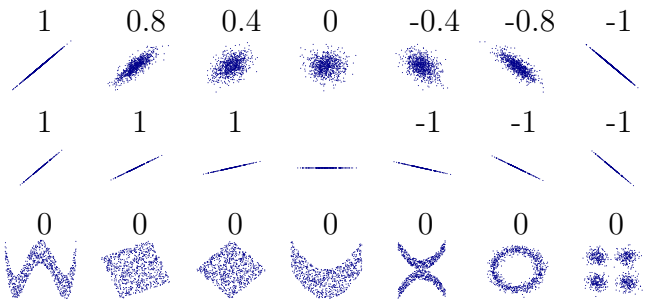


Figure 1.2: Examples of correlation between 2 RVs [7].

The tradeoff for that capacity to capture more general relationships between variables, is the complexity of estimation of the MI. Indeed, it is often the case that the probability distributions of the random variables (especially continuous RVs) are unknown and that direct estimation of the mutual information is intractable or exponentially more demanding in computing ressources. MI estimation is thus a challenging and crucial problem to solve.

1.2 Mutual Information Estimators - MINE and Prior Works

Before the introduction of neural based MI estimators, many other methods have been developed to estimate the mutual information between random variables. Many different approaches have been explored: simple binning estimators, density estimators [8, 9], non-parametric estimators [10, 11, 12, 13], series expansions [14] and many others [15, 16]. We will first present a few of these estimators before introducing MINE.

Prior Works

We are going to present 4 different mutual information estimators that have been developed in the litterature in the last years, in order of complexity and performance: the naive binning estimator, the Kraskov-Stögbauer-Grassberger (KSG) estimator, the Local Gaussian Density Estimation (LGDE) estimator and the Maximum Likelihood Mutual Information (MLMI) estimator.

Naive Binning Estimator

The naive binning estimator is the simplest MI estimator. The idea is to discretize the integral definition of the mutual information (1.5) and to approximate proba-

bility distributions by their empirical distributions. The estimation of the mutual information is then given by

Naive Binning Estimation

$$\hat{I}(X;Y) = \sum_{i,j} f_{xy}(i,j) \log \frac{f_{xy}(i,j)}{f_x(i) f_y(j)}, \tag{1.10}$$

where the i and j indices are the bins of the discretized RVs X and Y. Basically, we evaluate $f_{xy}(i,j)$ by counting the number of samples that fall in the bin (i,j) and dividing by the total number of samples. The same is done for $f_x(i)$ and $f_y(j)$, but by simply looking in the x- and y-axis bins.

Theoretically, the naive binning estimator is consistent, meaning that it converges to the true mutual information when the number of samples $N \to \infty$. It is however known to have a high variance and bias.

Kraskov-Stögbauer-Grassberger Estimator

The Kraskov-Stögbauer-Grassberger (KSG) estimator [10] is a very popular estimator and has been implemented in many libraries as the default MI estimator [17].

It is a non-parametric MI estimator that is based on the concept of nearest neighbors. The idea is to estimate the probability distributions by looking at the K-nearest neighbors of each sample. The estimation of the mutual information is then given by

KSG Estimation

$$\hat{I}(X;Y) = \psi(K) + \psi(N) - \frac{1}{N}\sum_{i=1}^{N} \psi(n_x(i)) - \frac{1}{N}\sum_{i=1}^{N} \psi(n_y(i)), \tag{1.11}$$

where ψ is the digamma function, N is the number of samples, $n_x(i)$ and $n_y(i)$ are the number of samples in the K-nearest neighbors of the i-th sample of X and Y respectively.

The KSG estimator is known to be consistent and to have a lower variance than the naive binning estimator. However, it is also known to have a bias that increases with the dimension of the RVs. On the other hand, it has the advantage to be relatively low in complexity and to be easy to implement.

Local Gaussian Density Estimation

In [9], Gao et al. proposed a density estimation based method to estimate the mutual information. The idea is to estimate PDFs by modeling them locally as Gaussian around each sample, i.e.

$$\hat{f}(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}(\boldsymbol{x}), \boldsymbol{\Sigma}(\boldsymbol{x})), \tag{1.12}$$

where $\boldsymbol{\mu}(\boldsymbol{x})$ and $\boldsymbol{\Sigma}(\boldsymbol{x})$ are the mean and covariance of the Gaussian distribution around the sample $\boldsymbol{x}$. We compute these parameters by solving the following optimization problem:

$$\boldsymbol{\mu}(\boldsymbol{x}), \boldsymbol{\Sigma}(\boldsymbol{x}) = \arg\max_{\boldsymbol{\mu}, \boldsymbol{\Sigma}} L(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}), \tag{1.13}$$

where $L(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the local likelihood function defined as

$$L(\boldsymbol{x}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^{N} \boldsymbol{K_H}(\boldsymbol{x_i} - \boldsymbol{x}) \log \mathcal{N}(\boldsymbol{x}_i; \boldsymbol{\mu}, \boldsymbol{\Sigma}) - \int \boldsymbol{K_H}(\boldsymbol{t} - \boldsymbol{x}) \mathcal{N}(\boldsymbol{t}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) d\boldsymbol{t}, \tag{1.14}$$

where $\boldsymbol{K_H}$ is a product kernel with diagonal bandwidth matrix $\boldsymbol{H}$.

Once the joint and marginal PDFs of X and Y are estimated as in (1.12), the mutual information can be computed as

LGDE

$$\hat{I}(X;Y) = \frac{1}{N} \sum_{i=1}^{N} \log \frac{\hat{f}_{xy}(\boldsymbol{x}_i, \boldsymbol{y}_i)}{\hat{f}_x(\boldsymbol{x}_i) \hat{f}_y(\boldsymbol{y}_i)}. \tag{1.15}$$

The LGDE estimator is known to be consistent and asymptotically unbiased. However, the optimization problem becomes computationally expensive in high dimensions since the number of parameters to estimate grows quadratically with the dimension of the RVs.

Maximum Likelihood Mutual Information

In [8], Suzuki et al. proposed a method called Maximum Likelihood Mutual Information (MLMI) to approximate mutual information using maximum likelihood of a density ratio function. The idea is to directly estimate the density ratio between the joint distribution of X and Y and the product of the marginal distributions of X and Y, $w(\boldsymbol{x}, \boldsymbol{y}) = \frac{f_{xy}(\boldsymbol{x}, \boldsymbol{y})}{f_{\boldsymbol{x}} f_{\boldsymbol{y}}}$.

The MI is then approximated by the expectation of the log-density ratio, i.e

$$\hat{I}(X;Y) = \frac{1}{N} \sum_{i=1}^{N} \log \hat{w}(\boldsymbol{x}_i, \boldsymbol{y}_i), \tag{1.16}$$

where $\hat{w}(\boldsymbol{x}, \boldsymbol{y})$ is the estimated density ratio function they modeled using a linear model:

$$\hat{w}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{\alpha}^T \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}). \tag{1.17}$$

Here, $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_b)$ are parameters learned from samples and $\boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) = (\phi_1(\boldsymbol{x}, \boldsymbol{y}), \cdots, \phi_b(\boldsymbol{x}, \boldsymbol{y}))$ are basis functions such that $\phi_i(\boldsymbol{x}, \boldsymbol{y}) \geq 0 \ \forall i$ and $\forall(\boldsymbol{x}, \boldsymbol{y})$.

The parameters $\boldsymbol{\alpha}$ are then learned by maximizing the log-likelihood:

$$L(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \log(\underbrace{\hat{f}_{xy}(\boldsymbol{x}_i, \boldsymbol{y}_i)}_{\hat{w}(\boldsymbol{x}_i, \boldsymbol{y}_i) f_x(\boldsymbol{x}_i) f_y(\boldsymbol{y}_i)}) = \sum_{i=1}^{N} \log\left(\boldsymbol{\alpha}^T \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})\right) + \underbrace{\sum_{i=1}^{N} \log f_x(\boldsymbol{x}_i) + \sum_{i=1}^{N} \log f_y(\boldsymbol{y}_i)}_{\perp\!\!\!\perp \boldsymbol{\alpha}}.$$

In addition to the non-negativity of $\hat{w}(\boldsymbol{x}, \boldsymbol{y})$, we need to add a normalization constraint such that

$$\begin{aligned} \int f_{xy}(\boldsymbol{x}, \hat{\boldsymbol{y}}) d\boldsymbol{x} d\boldsymbol{y} &= \int \boldsymbol{\alpha}^T \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}) f_x(\boldsymbol{x}) f_y(\boldsymbol{y}) d\boldsymbol{x} d\boldsymbol{y} \\ &\approx \frac{1}{n(n-1)} \sum_{1 \leq i \neq j \leq n} \boldsymbol{\alpha}^T \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{y}_j) = 1. \end{aligned}$$

Where the approximation is done using the U-statistics as described in [18]. The optimization problem is then given by

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{N} \log\left(\boldsymbol{\alpha}^T \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y})\right) \\ &\text{subject to} \quad \frac{1}{n(n-1)} \sum_{1 \leq i \neq j \leq n} \boldsymbol{\alpha}^T \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{y}_j) = 1 \quad \text{and} \quad \boldsymbol{\alpha} \geq 0. \end{aligned}$$

Finally, we have the mutual information estimation:

MLMI Estimation

$$\hat{I}(X; Y) = \frac{1}{N} \sum_{i=1}^{N} \log \hat{w}(\boldsymbol{x}_i, \boldsymbol{y}_i) = \frac{1}{N} \sum_{i=1}^{N} \log\left(\boldsymbol{\alpha}^{*T} \boldsymbol{\phi}(\boldsymbol{x}_i, \boldsymbol{y}_i)\right). \tag{1.18}$$

Mutual Information Neural Estimation

In 2018, Belghazi et al. introduced a new MI estimator called Mutual Information Neural Estimation (MINE) [1]. This estimator is based on a dual representation of the Kullback-Leibler divergence called the Donsker-Varadhan representation

[19], as formulated in Definition 1.9. This representation allows to re-express the KL-divergence as a maximization problem.

Definition 1.9: Donsker-Varadhan Representation

Let $\mathbb{P}$ and $\mathbb{Q}$ be two probability distributions defined on Ω. The KL-divergence between $\mathbb{P}$ and $\mathbb{Q}$ can be expressed as

$$D_{KL}(\mathbb{P}||\mathbb{Q}) = \sup_{T:\Omega\to\mathbb{R}} \mathbb{E}_{X\sim\mathbb{P}}[T(X)] - \log \mathbb{E}_{X\sim\mathbb{Q}}[e^{T(X)}], \tag{1.19}$$

where T lies in the space of functions such that the expectations remain finite.

It is possible to show (Cfr [1]) that the optimal function T^* that maximizes the right-hand side of (1.19) is given by $T^*(X) = \log\left(\frac{d\mathbb{P}}{d\mathbb{Q}}\right) + C$ where C is a constant, i.e. T^* is the log-density ratio between the two distributions.

Another representation of the KL-divergence based on the f-divergence variational representation was presented in [16] and [20]. It is a generalization of the KL-divergence and includes the KL-divergence as a special case. The f-divergence between two probability distributions $\mathbb{P}$ and $\mathbb{Q}$ is defined as

$$D_f(\mathbb{P}||\mathbb{Q}) = \int_\Omega f\left(\frac{d\mathbb{P}}{d\mathbb{Q}}\right) d\mathbb{Q}. \tag{1.20}$$

The KL-divergence is then a special case of the f-divergence where $f(x) = x\log(x)$. The variational representation of an f-divergence is given by (Cfr [16])

$$D_f(\mathbb{P}||\mathbb{Q}) = \sup_{T:\Omega\to\mathbb{R}} \mathbb{E}_{X\sim\mathbb{P}}[T(X)] - \mathbb{E}_{X\sim\mathbb{Q}}[f^*(T(X))], \tag{1.21}$$

where f^* is the convex conjugate of f. For the KL-divergence, we have that $f^*(x) = x\log(x) - x + 1$. The KL-divergence variational representation is then expressed as in Definition 1.10.

Definition 1.10: KL-Divergence Variational Representation

Let $\mathbb{P}$ and $\mathbb{Q}$ be two probability distributions defined on Ω. The KL-divergence between $\mathbb{P}$ and $\mathbb{Q}$ can be expressed as

$$D_{KL}(\mathbb{P}||\mathbb{Q}) = \sup_{T:\Omega\to\mathbb{R}} \mathbb{E}_{X\sim\mathbb{P}}[T(X)] - \mathbb{E}_{X\sim\mathbb{Q}}[e^{T(X)-1}], \tag{1.22}$$

where T lies in the space of functions such that the expectations remain finite.

In practice however, we do not necessarily know the distributions $\mathbb{P}$ and $\mathbb{Q}$ and cannot compute the expectations analytically. We would thus like to find the optimal function by searching in the space of admissible functions. Unfortunately, the complete space of admissible functions T is vast and not searchable realistically. Let us therefore limit the search space to a smaller admissible function space $\mathcal{F}$. Then, the solutions of (1.19) and (1.22) actually become suboptimal and are a lower bound of the KL-divergence. Even though both representations give tight bounds of the KL-divergence for sufficiently large function spaces $\mathcal{F}$, the DV-representation is tighter than the f-divergence variational representation (Cfr [21]), i.e. we always have that

$$\begin{aligned} D_{KL}(\mathbb{P}||\mathbb{Q}) &\geq \sup_{T\in\mathcal{F}} \mathbb{E}_{X\sim\mathbb{P}}[T(X)] - \log \mathbb{E}_{X\sim\mathbb{Q}}[e^{T(X)}] \\ &\geq \sup_{T\in\mathcal{F}} \mathbb{E}_{X\sim\mathbb{P}}[T(X)] - \mathbb{E}_{X\sim\mathbb{Q}}[e^{T(X)-1}]. \end{aligned} \tag{1.23}$$

We are then going to use the DV-representation to define the MINE estimator from here on.

Using (1.9) and (1.23), we can now define a lower bound on the mutual information between two random variables X and Y as follows:

$$I(X;Y) \geq \sup_{T\in\mathcal{F}} \mathbb{E}_{X,Y\sim\mathbb{P}_{XY}}[T(X,Y)] - \log \mathbb{E}_{X,Y\sim\mathbb{P}_X*\mathbb{P}_Y}[e^{T(X,Y)}]. \tag{1.24}$$

The idea behind MINE is to choose $\mathcal{F}$ to be a family of parametric functions $\mathcal{F}_\theta$ parametrized by a neural network with weights θ. The problem then becomes finding the function $T^*_\theta \in \mathcal{F}_\theta$ that maximizes the lower bound of the KL-divergence, or similarly finding the optimal set of parameter $\theta^* \in \Theta$. This will be done by training the neural network to maximize -or similarly to minimize the negative of- the lower bound of the KL-divergence, which will be the loss function of the network. The best estimation we can get given the function space $\mathcal{F}_\theta$ is then given by $I_\theta(X;Y)$, where

$$I(X;Y) \geq I_\theta(X;Y) = \sup_{\theta\in\Theta} \mathbb{E}_{X,Y\sim\mathbb{P}_{XY}}[T_\theta(X,Y)] - \log \mathbb{E}_{X,Y\sim\mathbb{P}_X*\mathbb{P}_Y}[e^{T_\theta(X,Y)}]. \tag{1.25}$$

Unfortunately, we are not able to evaluate the exact mathematical expectations. We can however approximate them by an empirical average over a batch of samples. The mutual information estimation is then given by

$$\hat{I}_\theta(X;Y) = \sup_{\theta\in\Theta} \frac{1}{N}\sum_{i=1}^{N} T_\theta(x_i,y_i) - \log\left(\frac{1}{N}\sum_{i=1}^{N} e^{T_\theta(\bar{x}_i,\bar{y}_i)}\right), \tag{1.26}$$

where (x_i, y_i) are samples drawn from the joint distribution $\mathbb{P}_{XY}$ and $(\bar{x}_i, \bar{y}_i)$ are samples drawn from the product of the marginal distributions $\mathbb{P}_X \mathbb{P}_Y$.

To obtain that estimation, the neural network will be trained by minimizing the loss function $\mathcal{L}(\theta)$, i.e.

$$\begin{aligned}\theta^* &= \arg\min_{\theta \in \Theta} \mathcal{L}(\theta) \\ &= \arg\min_{\theta \in \Theta} -\frac{1}{N}\sum_{i=1}^{N} T_\theta(x_i, y_i) + \log\left(\frac{1}{N}\sum_{i=1}^{N} e^{T_\theta(\bar{x}_i, \bar{y}_i)}\right).\end{aligned} \tag{1.27}$$

As for any neural network, now that we have defined the loss function, we can use backpropagation to train the network (Cfr Chapter 2 for more details). MINE algorithm can then be described as in Algorithm 1.1.

Algorithm 1.1: MINE Algorithm

Input: Samples $(x_i, y_i) \sim \mathbb{P}_{XY}, \quad i = 1, \cdots, N.$

Output: Optimal parameters θ^*.

Parameters: Learning rate λ, Number of epochs N_{epochs}, Number of batches $N_{Batches}$ or Batch size B.

Initialization: Initialize neural network with random weights θ_0.

for *epoch in* $1, 2, ..., N_{epochs}$ **do**

  Shuffling: Shuffle samples (x_i, y_i).

  foreach *batch* $\{x_i, y_i\}_{i=1}^{B}$ **do**

    Marginals: Permutate randomly y_i to artificially create samples $\{\bar{x}_i, \bar{y}_i\}_{i=1}^{B} \sim \mathbb{P}_X \mathbb{P}_Y$.

    Compute Loss:

    $\mathcal{L}(\theta) \leftarrow -\frac{1}{B}\sum_{i=1}^{B} T_\theta(x_i, y_i) + \log\left(\frac{1}{B}\sum_{i=1}^{B} e^{T_\theta(\bar{x}_i, \bar{y}_i)}\right)$.

    Update Weights: $\theta \leftarrow \theta - \lambda \nabla_\theta \mathcal{L}(\theta)$.

  end

end

return θ^*

It is important to understand that here, contrary to most deep learning applications, we are not interested by the output of the network directly, i.e. by the function $T_\theta^*(X)$. We are rather interested in the value of the loss function itself which is an estimation of the mutual information. So that once the network

is trained for a particular set of samples $\{x_i, y_i\}_{i=1}^N$, we can evaluate the mutual information between X and Y by evaluating the loss function on a batch of samples. The estimation of the mutual information is then given by

MINE Estimation

$$\hat{I}_\theta(X;Y) = \frac{1}{N}\sum_{i=1}^{N} T_\theta^*(x_i, y_i) - \log\left(\frac{1}{N}\sum_{i=1}^{N} e^{T_\theta^*(\bar{x}_i, \bar{y}_i)}\right). \tag{1.28}$$

MINE has been shown to outperform other MI estimators especially in tractability (i.e. scalability) in higher dimensions [1]. Its interesting properties have made it a popular choice for MI estimation in many applications. We thus decided to further study this tool and to explore its potential in the context of communication systems.

ReMINE

Although MINE has good estimator properties such as strong consistency and scalability, some issues tend to arise in practice. One of the main issues we encounter with MINE is the instability of the loss. Indeed, the loss function $\mathcal{L}(\theta)$ has been shown to diverge in some cases. Another issue is the drifting of the two terms of the loss function. Both terms tend to decrease indefinitely at the same rate during training, even after the MI estimates converged to a stationnary value. This is due to the fact that the optimal function T^* is defined up to an additive constant. Figure 1.3 illustrates this issue.

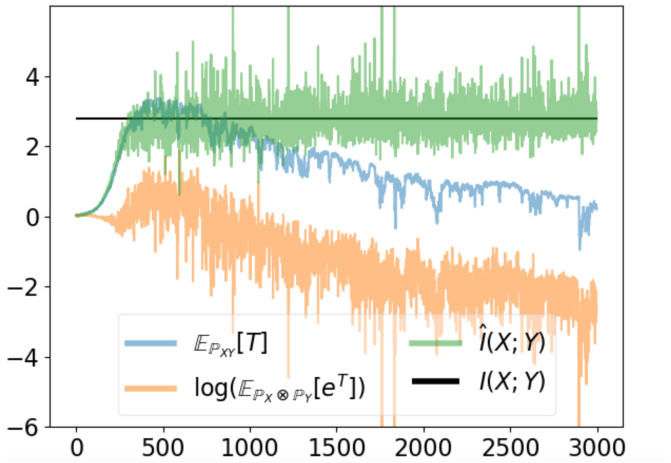


Figure 1.3: Illustration of the drifting issue in MINE. X-axis is the number of epochs and Y-axis is the value of the loss function. The two terms of the loss function tend to decrease indefinitely at the same rate during training. The estimation of the mutual information converges in average to the true MI value [22].

In [22], the authors present and tackle these known issues and propose a modified version of MINE called Regularized Mutual Information Neural Estimation (ReMINE). The idea is to add a regularization term in order to prevent the divergence of the loss and to stabilize the training. The loss function is then given by

$$\mathcal{L}(\theta) = -\frac{1}{N}\sum_{i=1}^{N} T_\theta(x_i, y_i) + \log\left(\frac{1}{N}\sum_{i=1}^{N} e^{T_\theta(\bar{x}_i, \bar{y}_i)}\right) + d\left(\log\left(\frac{1}{N}\sum_{i=1}^{N} e^{T_\theta(\bar{x}_i, \bar{y}_i)}\right), C\right), \tag{1.29}$$

where $d(\cdot,\cdot)$ is a distance measure such as the euclidean distance. This third term penalizes the distance of the second term of the loss function to a constant C. It is important to note that this contant is related to the optimal solution T^*. Indeed, the optimal solution of the KL-divergence variational representation is given by $T^*(X) = \log\left(\frac{d\mathbb{P}}{d\mathbb{Q}}\right) + C$ where C is a constant. The regularization term allows to stabilize the drifting term C and to prevent the divergence of the loss function. Figure 1.4 illustrates the effect of regularization in ReMINE.

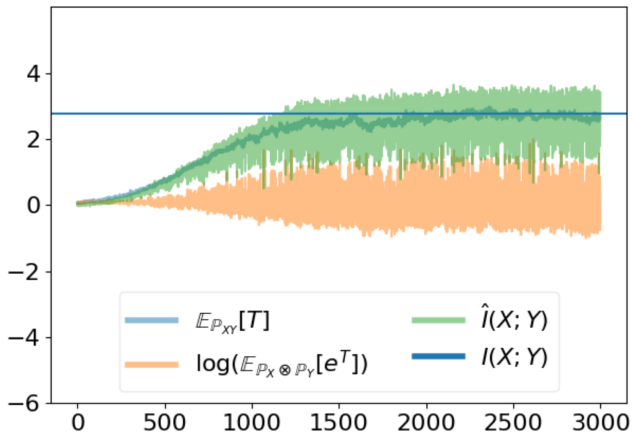


Figure 1.4: Illustration of the regularization term in ReMINE. X-axis is the number of epochs and Y-axis is the value of the loss function. The regularization term penalizes the distance of the second term of the loss function to a constant $C = 0$ here. Indeed we see that the second term is 0 on average. The estimation of the mutual information converges in average to the true MI value [22].

The ReMINE algorithm is similar to the MINE algorithm 1.1, with the addition of the regularization term in the loss function. The training is then done by minimizing the new loss function (1.29) still with respect to the weights θ of the neural network. The estimation of the mutual information remains the same as in (1.28). This version of the algorithm, being more stable and less prone to divergence, will be the one used in the following chapters. For simplicity, we will refer to it as MINE in the following chapters keeping in mind that we are actually using the ReMINE version of the algorithm.

Chapter Conclusion

In this chapter, we have presented the concept of mutual information and its importance in many fields. We have also presented a few mutual information estimators that have been developed in the litterature. We have especially focused on a new mutual information estimator called Mutual Information Neural Estimation (MINE) that was introduced in [1]. The latter is based on the use of neural networks and has been shown to outperform other MI estimators especially in dimensional scalability. We have also presented a modified version of MINE called ReMINE that tackles some of the known issues of MINE. In the following chapters, we are going to study MINE in more details and explore its potential in the context of communication systems.

CHAPTER 2

Algorithm Optimization

MINE is a powerful algorithm for estimating the Mutual Information between two random variables. However, it has some limitations, such as its stability, speed and ease of use for non-expert users, that can make it difficult to use in practice. We thus wanted to improve the algorithm by adding some optimizations that would make it more performant and user-friendly.

In this chapter, we first start with a reminder on deep learning and neural networks theory since it is fundamental to understand the optimizations we have made to the MINE algorithm.

We then discuss the characterization of the network and the impact of the hyperparameters on the performance of the algorithm. We present the results of a series of experiments we conducted to understand the impact of the depth and width of the network, the learning rate, the batch size, etc. on the performance of the algorithm.

Finally, in the light of the previously discussed observations, we present some optimizations we made to the algorithm in order to improve its performance and user friendliness. We will see that these optimizations will help reducing the risk of instabilities and increase the speed and precision of the estimation.

These optimizations were added to an already existing implementation of MINE developed by *William Wu* and *Homa Esfahanizadeh* (*Network coding and Reliable Communication Group, RLE at MIT*) and based on the public repositories of [1] and [22]. This implementation was notably used in [23].

2.1 Deep Learning Background

Deep learning is a subfield of machine learning that focuses on the use of deep neural networks to model complex patterns (Some famous deep learning and machine learning references [24, 25, 26, 27]). Deep learning has been very successful in recent years, in fields such as computer vision, natural language processing, speech recognition, etc. This success is due to the ability of deep neural networks to learn complex patterns in data and to generalize well to unseen data.

Neural Networks

Basically, a neural network represents a parametric function that maps m inputs to n outputs. We can thus simply denote a network by $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) : \mathbb{R}^m \rightarrow \mathbb{R}^n$. They are composed of layers of neurons, connected to each other by weighted edges called synapses. Each neuron is a simple function that takes the weighted sum of its inputs, applies an activation function to it, and passes the result to the next layer. An activation function is a non-linear function that introduces non-linearity in the network, allowing it to learn complex patterns. Common activation functions include the sigmoid, tanh, and ReLU/Leaky ReLU functions and are shown in Figure 2.1.

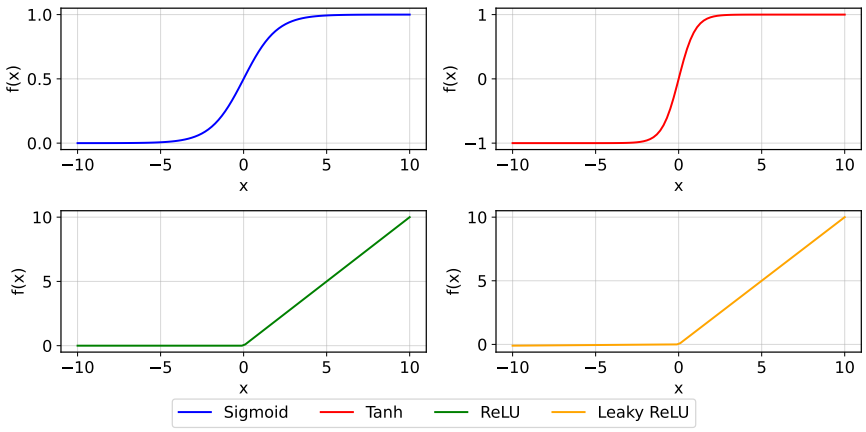


Figure 2.1: Common Activation Functions.

Figure 2.2 shows a general neural network architecture with m inputs, n outputs and 3 hidden layers. The hidden layers are free to be set to any size and does not need to be the same from one layer to another.

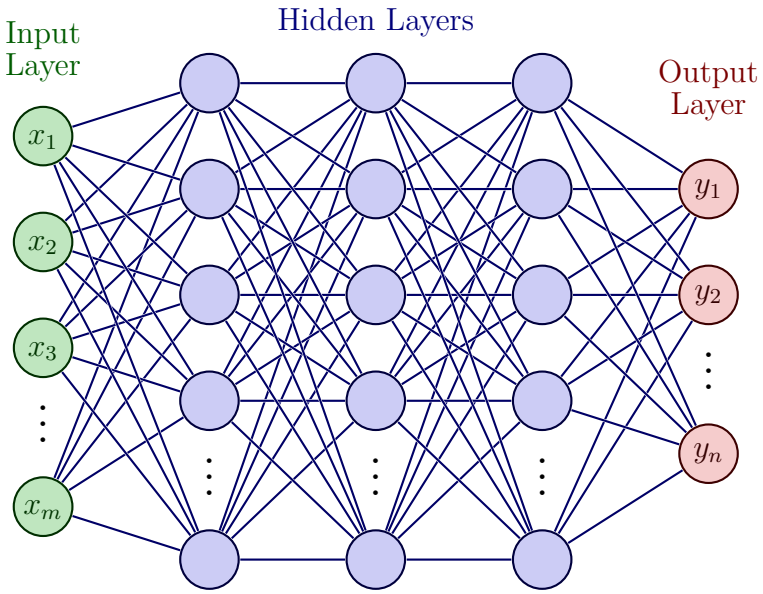


Figure 2.2: A General Neural Network Architecture (Adapted from [28]). Circles represent neurons and lines represent synapses.

Forward Propagation

To evaluate the function $\boldsymbol{f}(\boldsymbol{x})$ for a certain input $\boldsymbol{x}$, we feed the input to the network and propagate it through the layers. This process is called forward propagation. To compute the output of the $(i+1)^{th}$ layer of neurons from the $(i)^{th}$ layer, we apply the following formula:

Forward Propagation

$$\boldsymbol{a}^{(i+1)} = \sigma\left(\mathbf{W}^{(i)}\boldsymbol{a}^{(i)} + \boldsymbol{b}^{(i)}\right), \tag{2.1}$$

where, $\boldsymbol{a}^{(i)}$ is the output of layer (i), $\mathbf{W}^{(i)}$ is the weight matrix of layer (i), $\boldsymbol{b}^{(i)}$ is the bias vector of layer (i) and σ is the activation function. This process is illustrated in more details in Figure 2.3 for any two consecutive layers (i) and $(i+1)$ of size k and l respectively.

Forward propagation is done iteratively starting from the input until we reach the output layer.

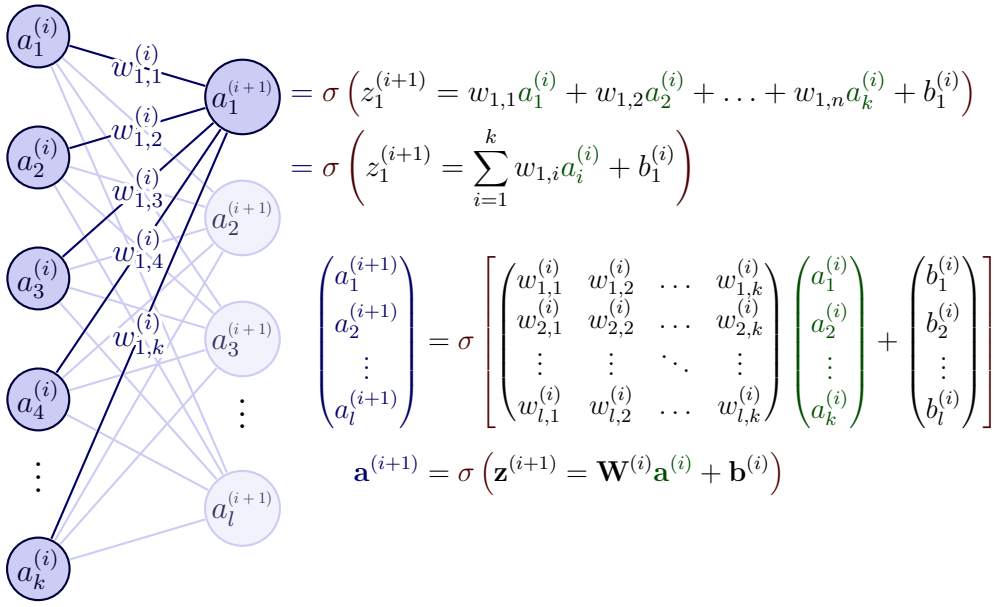


Figure 2.3: Forward Propagation between layer (i) and $(i+1)$ (Adapted from [28]).

Training and Backpropagation

Neural networks need to be trained to learn patterns and relationships within given data samples, enabling them to perform different tasks such as regressions, classifications, etc. The objective is to find an optimal function $\boldsymbol{f}^*(\boldsymbol{x})$, or similarly optimal weights θ^*, to model the task at aim.

In order to do so, we need to define a loss function $\mathcal{L}$ that measures the difference between the predicted output $\boldsymbol{f}(\boldsymbol{x})$ and the true/desired output y. There is an infinite amount of way to define loss functions depending on the task at hand. Common loss functions include the Mean Squared Error (MSE) for regression tasks [29], the Cross-Entropy for classification tasks [30], etc. In our case, we will use the loss function of the MINE algorithm defined in (1.28).

The optimal function $\boldsymbol{f}^*(\boldsymbol{x})$ is found by minimizing the loss function $\mathcal{L}$ with respect to the weights θ of the network, i.e.

$$\theta^* = \arg\min_{\theta} \mathcal{L}(\boldsymbol{f}(\boldsymbol{x}), \boldsymbol{y}). \tag{2.2}$$

The minimum of the loss function is found by using a gradient descent-type of optimization algorithm such as Stochastic Gradient Descent (SGD) [31], Adam [32], RMSprop, etc [33]. These algorithms compute the gradient of the loss function with respect to the weights of the network and update the weights in the opposite direction of the gradient.

The process by which we compute the gradient of the loss with respect to the weights is called backpropagation. The gradient is computed using the chain rule of calculus [34]. This rule states that the derivative of a composition of functions is the product of the derivatives of the functions. We can apply this rule thinking of the each neuron as a function that takes as input the output of another function (the previous layer) and outputs a new function. Refering to Figure 2.3 for the notations, we can compute the gradient of the loss function with respect to the weights of the network as given in (2.3).

Backpropagation

$$\nabla_{w_{j,k}^{(i)}} \mathcal{L} = \nabla_{a_j^{(i+1)}} \mathcal{L} \cdot \nabla_{z_j^{(i+1)}} a_j^{(i+1)} \cdot \nabla_{w_{j,k}^{(i)}} z_j^{(i+1)} \tag{2.3}$$

We can thus start by computing the gradient of the loss function with respect to the output of the network $\nabla_f \mathcal{L}$ and then propagate it back through the layers to compute the gradient of the loss function with respect to the weights of each layer. The backpropagation algorithm is summarized in Algorithm 2.1.

Algorithm 2.1: Backpropagation Algorithm

Input: Loss function $\mathcal{L}$, Training dataset $\{\boldsymbol{x_j}, \boldsymbol{y_j}\}_{j=1}^{N}$.

Output: Optimal weights w^*.

Parameters: Learning rate λ, Number of epochs N_{Epochs}.

Initialization: Initialize randomly the weights $w \leftarrow w_0$.

for *i in range* N_{Epochs} **do**
- **Shuffling:** Shuffle the dataset $\{\boldsymbol{x_j}, \boldsymbol{y_j}\}_{j=1}^{N}$.
- **foreach** *Sample* $(\boldsymbol{x_j}, \boldsymbol{y_j})$ **do**
  - **Forward Propagation:** Compute output $\boldsymbol{f}(\boldsymbol{x_j})$.
  - **Compute Loss:** $\mathcal{L} \leftarrow \mathcal{L}(\boldsymbol{f}(\boldsymbol{x_j}), \boldsymbol{y_j})$.
  - **Back Propagation:** Compute gradient of loss with respect to output $\nabla_{\boldsymbol{f}}\mathcal{L}$.
  - **foreach** *layer l in* N_{layers}, $N_{layers} - 1$, *...*, *1* **do**
    - **Compute Gradient:** Compute gradient of loss with respect to weights $\nabla_{w^{(l)}}\mathcal{L}$ using (2.3).
    - **Update Weights:** $w_{i+1}^{(l)} = w_i^{(l)} - \lambda \nabla_{w^{(l)}}\mathcal{L}$.
  - **end**
- **end**

end

In Algorithm 2.1, we see that we update the weights of the network after each sample. This method is called Stochastic Gradient Descent (SGD). However, in practice, we often use $N_{Batches}$ mini-batches of $B = N/N_{Batches}$ samples to compute the gradient of the loss function. This is called Mini-Batch Gradient Descent. If $N_{Batches} = N$, we get the SGD back and when $N_{Batches} = 1$, we get what is called a Batch Gradient Descent. The gradient is then computed by averaging the gradients of the loss function with respect to the weights of the network over the mini-batch. The weights are then updated using the average gradient (see Alorithm 2.2). Mini-Batch Gradient Descent has the advantage of offering a smoother convergence of the optimization algorithm. But even more importantly, it allows to efficiently use the parallelism of modern hardware such as GPUs to speed up the computation of the gradient.

This feature however was not included in the original implementation of MINE,

we thus wanted to add it to improve the performance of the algorithm. This addition, as well as the fact that we used *PyTorch* [35] instead of *PyTorch Lightning* [36], increased tremendously the speed of the algorithm, allowing us to converge in the matter of a few minutes instead of a few hours (on the dataset considered for our application).

In practice, we often use more complex gradient descent algorithms such as Adagrad, Adadelta, RMSprop, Adam, etc. These algorithms use adaptive learning rates that change during the optimization process to speed up the convergence. They also include momentum terms that help to escape local minima and saddle points. Since optimization algorithms are not the main focus of this work, we will not go into the details of these algorithms, for more informations, please refer to [32, 33, 37].

Algorithm 2.2: Batch-Wise Backpropagation Algorithm

Input: Loss function $\mathcal{L}$, Training dataset $\{\boldsymbol{x_j}, \boldsymbol{y_j}\}_{j=1}^{N}$.

Output: Optimal weights w^*.

Parameters: Learning rate λ, Number of epochs N_{Epochs}, Number of batches $N_{Batches}$ or Batch Size B.

Initialization: Initialize randomly the weights $w \leftarrow w_0$.

for *i in range* N_{Epochs} **do**
- **Shuffling:** Shuffle the dataset $\{\boldsymbol{x_j}, \boldsymbol{y_j}\}_{j=1}^{N}$.
- **Splitting:** Split the dataset into mini-batches of size B.
- **foreach** *batch* $\{x_i, y_i\}_{i=1}^{B}$ **do**
  - **foreach** *Sample* $(\boldsymbol{x_j}, \boldsymbol{y_j})$ *in Batch* **do**
    - **Forward Propagation:** Compute output $\boldsymbol{f}(\boldsymbol{x_j})$.
    - **Compute Loss:** $\mathcal{L}(\boldsymbol{f}(\boldsymbol{x_j}), \boldsymbol{y_j})$.
    - **Back Propagation:** Compute gradient of loss with respect to the output $\nabla_{\boldsymbol{f}}\mathcal{L}$.
    - **foreach** *layer l in* N_{layers}, $N_{layers} - 1$, ..., *1* **do**
      - **Compute Gradient:** Compute gradient of loss with respect to weights $\nabla_{w^{(l)}}\mathcal{L}$ using (2.3).
      - **Average Gradients:** Aggregate gradients of loss with respect to the weights $\nabla_{w^{(l)}}\mathcal{L} \leftarrow \nabla_{w^{(l)}}\mathcal{L} + \nabla_{w^{(l)}}\mathcal{L}^{(j)}/B$.
    - **end**
  - **end**
  - **Update Weights:** $w_{i+1}^{(l)} = w_i^{(l)} - \lambda\nabla_{w^{(l)}}\mathcal{L}$.
- **end**

end

2.2 Characterization of the Network

The network used in the MINE algorithm (Algorithm 1.1) is a simple feedforward neural network. It is composed of an input layer, a number of hidden layers and an output layer. The size of the input layer must equal the sum of the dimensions of the input data X and Y. The output layer has a single neuron since we are trying to model $T : (X, Y) \to \mathbb{R}$, which is real valued. The hidden layers can have any size and any number of neurons. The activation function used in the hidden layers is the Leaky ReLU function which has the advantage of being less prone to the vanishing gradient problem [38]. Figure 2.4 represents such a network.

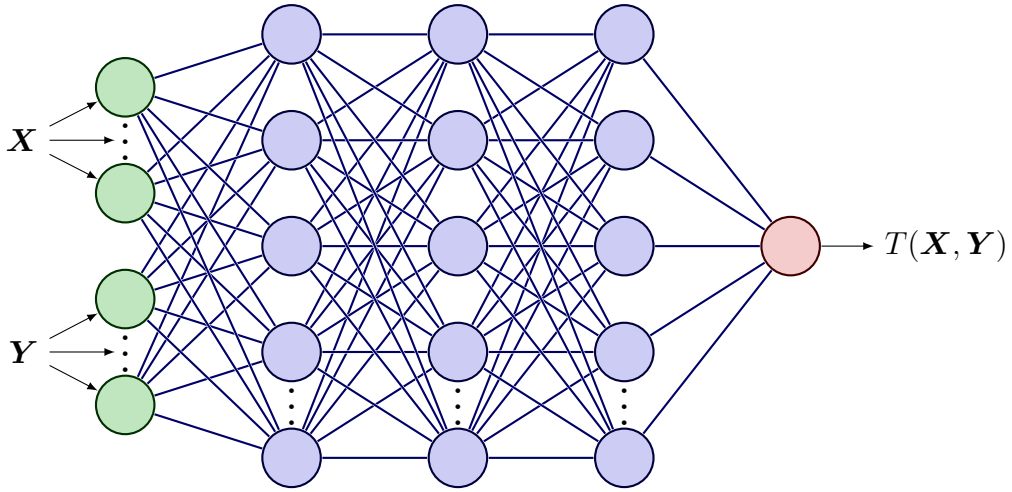


Figure 2.4: Neural Network Architecture of MINE.

As mentioned in the previous section, neural networks require a lot of hyperparameters to be set prior training. Before trying to optimize the algorithm, we first need to understand and characterize the impact of these hyperparameters on the performance of the algorithm, we conducted a series of experiments to that end. We tested the impact of the number of hidden layers (i.e. the depth of the network), the size of the hidden layers (i.e. the width of the network), the learning rate, the batch size, the number of epochs etc.

To study the network, we will use a dataset commonly used in the MI estimation litterature [1, 22, 39]: the correlated Gaussian dataset. This dataset is composed of 2 correlated standard ($\mathbb{E}[X_i] = 0$ and $\mathbb{V}ar[X_i] = 1$, $\forall i$) Gaussian random vectors $\boldsymbol{X}$ and $\boldsymbol{Y} \in \mathbb{R}^d$ with component-wise correlation $\text{Corr}(X_i, Y_j) = \rho \cdot \delta_{ij}$. Same entry components thus follow a bivariate standard Gaussian distribution with correlation ρ:

$$\begin{bmatrix} X_i \\ Y_i \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \right), \tag{2.4}$$

and different entries components are independent. The MI can be computed analytically in this case which is useful to compare our estimations against the true MI. The MI between X and Y is given by (2.5).

$$I(X;Y) = -\frac{d}{2}\log(1-\rho^2) \tag{2.5}$$

Although the Gaussian dataset is not the most complex dataset and might not fully reflect the behaviour of more intricate data, it is a good starting point to understand the impact of the network architecture on the performance of the algorithm.

By default, we will use the parameters as described in Table 2.1 in the following experiments. For each experiment we will vary one parameter and keep the others fixed to the default values.

Parameter	**Value**
Depth	3
Width (Same for every layer)	128
Batch Size	2048
Number of Samples	10^6
Learning Rate	0.001
Number of Epochs	100

Table 2.1: Default Parameters for the Experiments.

Effect of Network Architecture

In this section, we are going to be mainly interested in the effects of the depth and the width of the neural network on the performance of MINE .

- Depth The depth of a neural network is a crucial parameter that can greatly impact its performance. The number of hidden layers determines the complexity of the function that the network can model. A network with more hidden layers can model more complex functions but is also more prone to overfitting and instabilities as we are going to show. We will vary the number of hidden layers from 1 to 6. The results are shown in Figure 2.5.

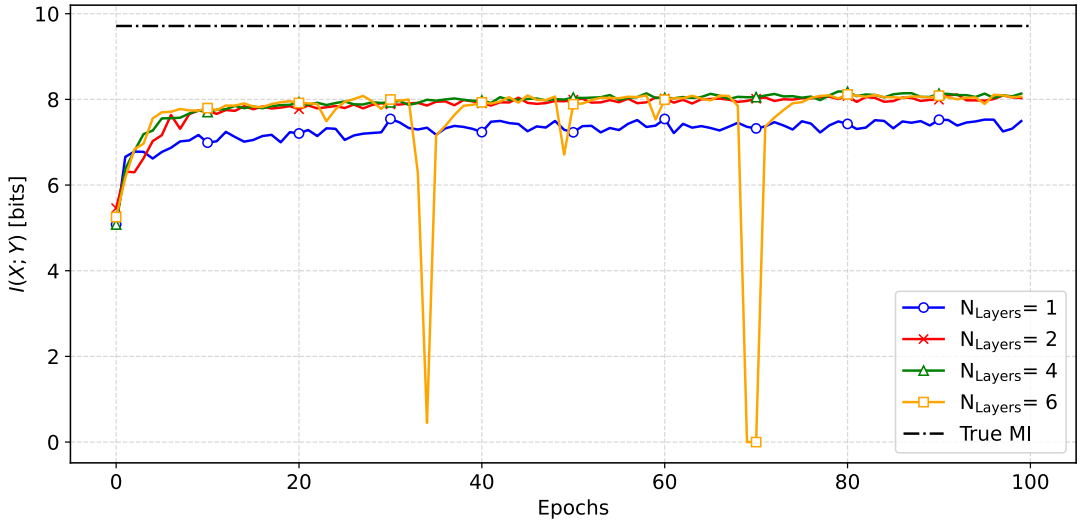


Figure 2.5: Effect of network depth. Gaussian Dataset with $d = 20$ and $\rho = 0.7$.

As is shown in Figure 2.5, we observe that having too few hidden layers (e.g. $N_{Layers} = 1$), i.e. a not very expressive network, leads to poorer estimations than with more hidden layers (e.g. $N_{Layers} = 2$ or 4), up to a certain point (e.g. $N_{Layers} = 6$). After a certain depth, the network starts to become unstable and dips to very low values of MI, although it tends to come back to a certain level in a few epochs.

- Width The width of the hidden layers is another important parameter to consider. It also determines the complexity of the function that the network can model. Again, a network with too many neurons will be prone to overfitting/instabilities, although, we usually prefer a wide network against a deep network due to vanishing gradients issues. Here, we are going vary the number of neurons in each layer from 64 to 2048. The results are shown in Figure 2.6.

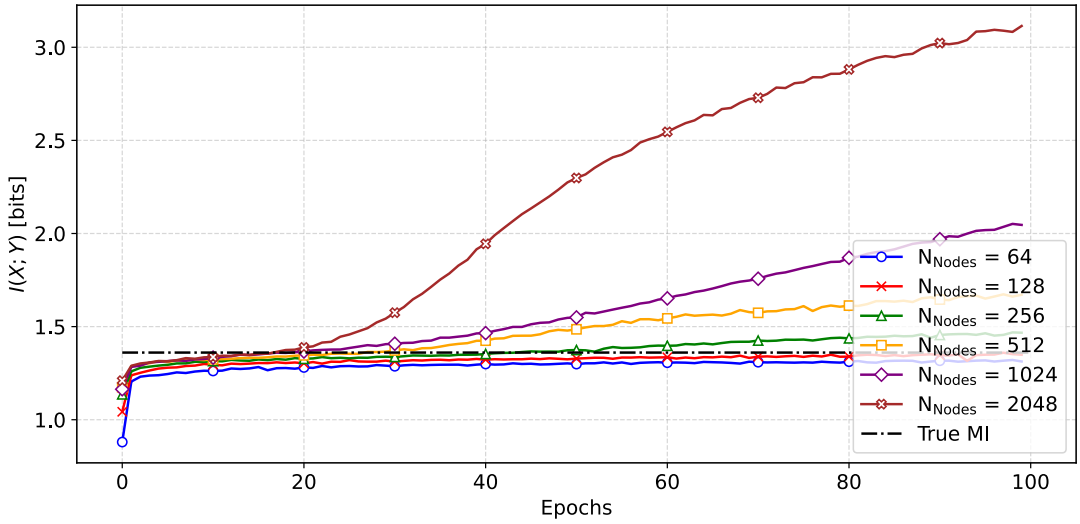


Figure 2.6: Effect of network width. Gaussian Dataset with $d = 20$ and $\rho = 0.3$.

In Figure 2.6, we observe a different behavior than when we increased the complexity by increasing the width instead of the depth of the network. We now do not observe instabilities anymore but rather smooth convergences. On the contrary, we see that for very expressive networks (e.g. $N_{Nodes} \geq 256$), the network tends to converge very well to the true MI value but then starts to overfit the data and diverge. The network being too expressive, starts to model noise and finds patterns in the data that do not exist, leading to an overestimation of the MI.

Effect of Batch Size

Generally speaking, in most deep learning applications, a small batch size leads to noisy gradients but a faster convergence. While larger batch sizes lead to smoother convergence but will also slow down the process. We will vary the batch size from 32 to 32768. Usually, we choose the batch size to be a power of 2 for hardware optimization reasons. The results are shown in Figure 2.7.

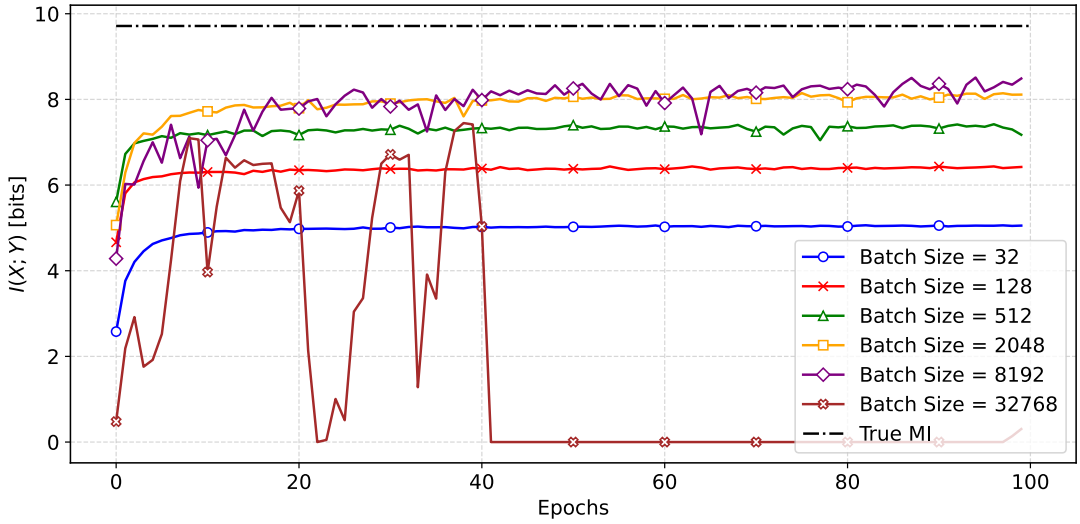


Figure 2.7: Effect of Batch Size. Gaussian Dataset with $d = 20$ and $\rho = 0.7$.

We see that the batch size has an impact on both the variance and the accuracy of estimation. Contrary to most deep learning applications, we see here that a larger batch size leads to a more accurate convergence but with a higher variance, up to a certain batch size where it starts to break down ($BatchSize = 32768$ in Figure 2.7). On the other hand a smaller batch size leads to a smoother convergence (i.e. less variance) but a decreased accuracy.

We can understand this behaviour by looking at the formula of the loss of MINE:

$$\mathcal{L}(\theta) = -\frac{1}{N_{Batches}} \sum_{i=1}^{N_{Batches}} T_\theta(X_i, Y_i) + \log\left(\frac{1}{N_{Batches}} \sum_{i=1}^{N_{Batches}} e^{T_\theta(X_i, Y_{\sigma(i)})}\right), \quad (2.6)$$

where $N_{Batches}$ is the number of batches. We see that when we increase the batch size B, or similarly decrease the number of batches $N_{Batches}$, we actually compute the empirical averages in (2.6) using less samples which naturally increases the variance of the average estimation. Indeed, it is well known that the variance of the average of N i.i.d samples is $\frac{\sigma^2}{N}$ where σ^2 is the variance of the samples.

It is interesting to note that we observe a variance-bias tradeoff in the choice of the batch size. Indeed, as explained above, we see that a smaller batch size will lead to a smoother (small variance) convergence but a lower estimation accuracy (high bias). On the other hand, a larger batch size will lead to a better estimation accuracy but a higher variance in the estimation.

Effect of Number of Samples

The number of samples in the dataset can also impact the performance of the network. We will vary the number of samples in the dataset from 10^3 to 10^7. The results are shown in Figure 2.8.

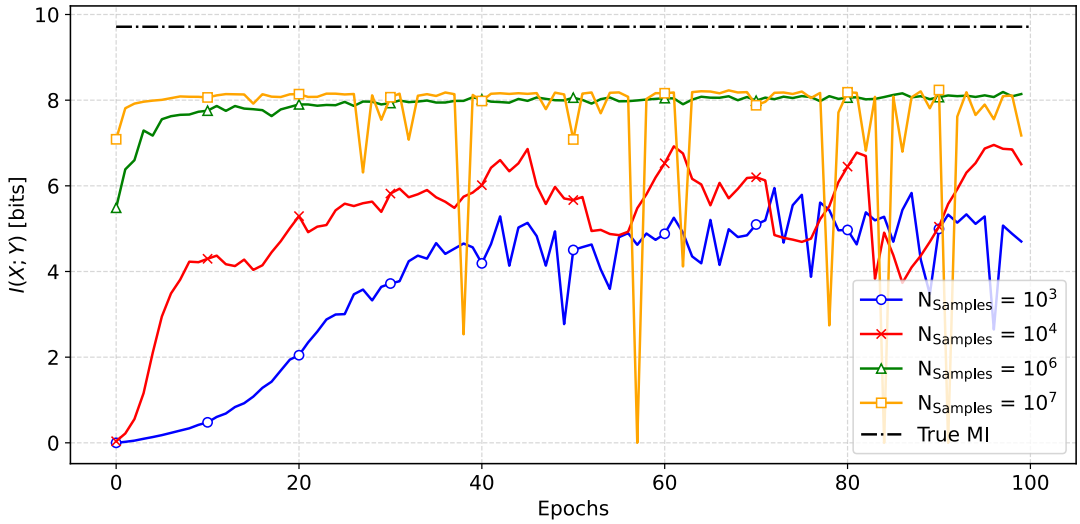


Figure 2.8: Effect of Number of Samples. Gaussian Dataset with $d = 20$ and $\rho = 0.7$.

In Figure 2.8, we see that the number of samples has a significant impact on the convergence. We observe that for a small number of samples, the network does not seem to converge to a good estimation and is very unstable. On the other hand, we see that for 10^6 samples, the convergence is very smooth and the estimation is much closer to the true value. However, we note that once we increase the number of samples to 10^7, the network starts to behave as in the experiment where we increased the depth of the network too much. At first, the convergence is smooth and faster than for 10^6 samples but at some point we observe these instabilities in the convergence.

It is not clear to us why this phenomenon occurs in this particular case as we would expect the network to have a smoother and faster convergence as we increase the number of samples available. We know however how to mitigate this issue. Indeed, both the *Early Stopping* and the *Learning Rate Scheduler* optimizations presented in the following sections will help solve this issue.

Effect of Learning Rate

The learning rate is of paramount importance for a fast, smooth and accurate estimation. It is likely the most impactful hyperparameter in terms of convergence speed. We will vary the learning rate from 10^{-5} to 10^{-1}. The results are shown in Figure 2.9.

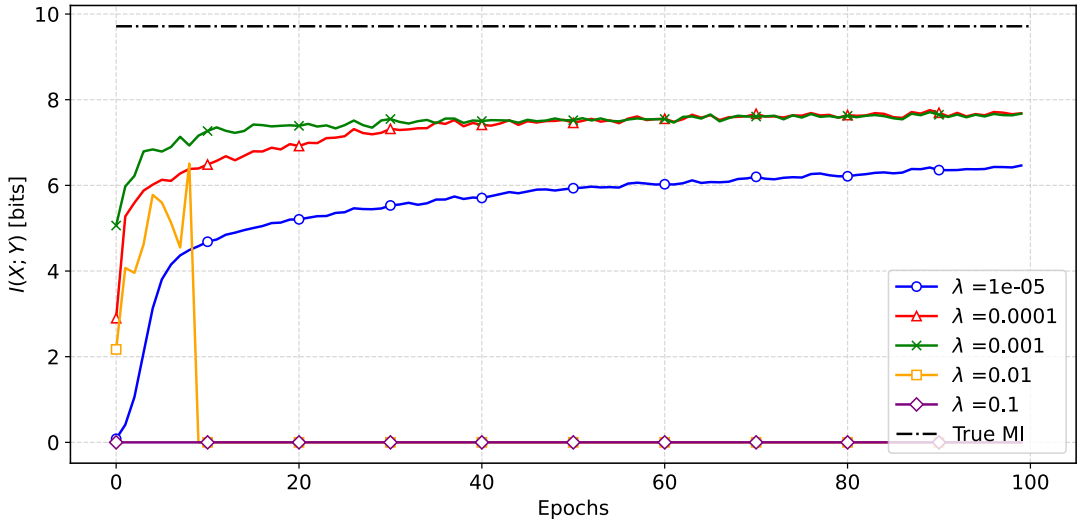


Figure 2.9: Effect of Learning Rate. Gaussian Dataset with $d = 20$ and $\rho = 0.7$.

We see that the learning rate has an impact both on the stability and on the speed of the estimation. A learning rate that is too high (e.g. $\lambda = 0.01$ and 0.1) will lead to an unstable convergence and a learning rate that is too small ($\lambda = 10^{-5}$) to a slow convergence. Here we see that a learning rate of $\lambda = 10^{-3}$ leads to the fastest and most stable convergence. It is interesting to note that the optimal learning rate, in the sense of accuracy, will (for all cases tested) be the one leading to the fastest convergence, i.e. the highest estimates during the first epochs. This fact will be used as part of the *Automatic Learning Rate Tuner* optimization presented in the following section.

2.3 Optimizations

In this section, we present a series of optimizations that we implemented to improve the performance of the MINE algorithm. We will present the *Early Stopping*, the *Automatic Learning Rate Tuner* and the *Learning Rate Scheduler*. The goal of these optimizations is to improve the convergence of the algorithm, to save time

and resources as well as to offer a better user-friendliness for non deep learning expert users.

Early Stopping

Early stopping is a mechanism often used in machine learning to stop the training under certain constraints. The idea is basic yet very time-saving. It consists in monitoring a metric of interest until it reaches a plateau or starts to degrade. When this happens, we stop the training. This is very useful to avoid overfitting and to save time. We implemented this method in our algorithm by monitoring the MINE estimation. We stop the training when we reach a plateau. The definition of plateau for early stopping depends on some parameters such as the patience and the threshold. The patience is the number of epochs we wait before stopping the training when the metrics does not improve. The threshold is the minimum improvement we expect from the metrics within a patience period, it is expressed as a percentage of the metrics. The results are shown in Figure 2.10.

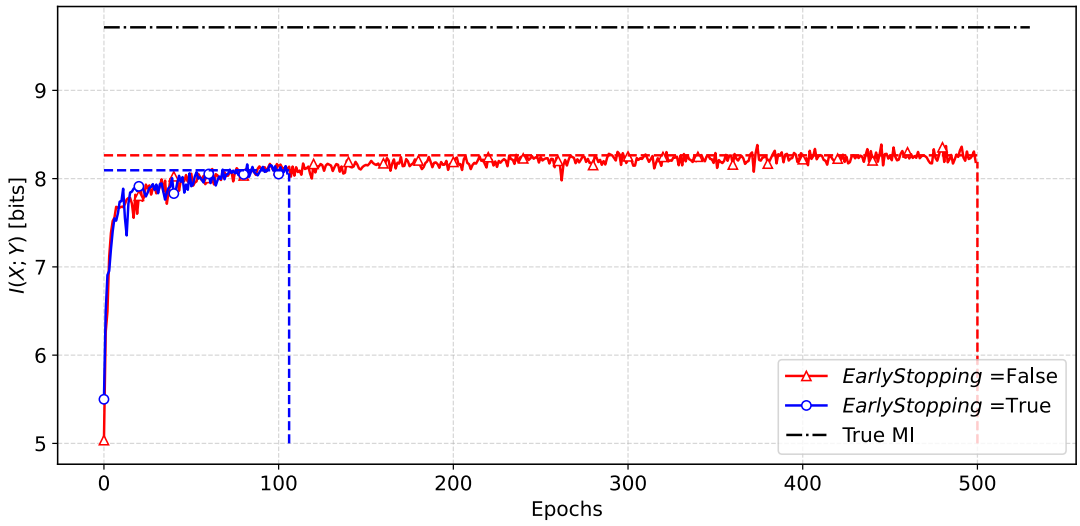


Figure 2.10: Effect of Early Stopping with a patience of 20 and a threshold of 1%. Gaussian Dataset with $d = 20$ and $\rho = 0.7$.

We see that we gain a lot of epochs by using early stopping. Here we used 500 epochs for the training to let a long enough time for the network to converge. Indeed, for each application, the number of epochs necessary might not be the same and is a priori unknown. We must thus keep the default number of epochs high enough hoping that the network will converge within that time span. However, the default number of epochs might be too high for some applications and we might be

wasting a lot of time and ressources by training the network for too long. Which is why we implemented the early stopping mechanism. We see indeed that the accuracy gain we have from 100 to 500 epochs is not very significant. With the parameters used here (patience=20 and threshold=0.01), we stop the training a little bit after 100 epochs. This allows us to save a lot of time and ressources by not continuing the training for the remaining 400 epochs at the expense of small loss in accuracy. This is a trade-off that we have to make depending on the application and that can be tuned using the aforementioned parameters at the conveniance of the user.

Automatic Learning Rate Tuner

As explained in the previous section, the learning rate is a crucial hyperparameter that can greatly impact the performance of the network. We have seen that the optimal learning rate in the sense of accuracy is the one leading to the fastest convergence. We can then use this information to automatically tune the learning rate before the final training.

The procedure is rather simple yet very effective and time-saving. We simply start to train the network for a few epochs (e.g. 5 or 10) for a range of learning rates (e.g. 5 LR from 10^{-5} to 10^{-1} logarithmically separated). We choose the learning rate that leads to the greatest estimation after the first epochs. If needed, we can choose to refine the learning rate by training the network for a few more epochs with a smaller range of learning rates around the first optimal LR, this is controlled by the *Depth* parameter in Algorithm 2.3. Once the optimal learning rate is found, we can then start the final training. The results are shown in Figure 2.11.

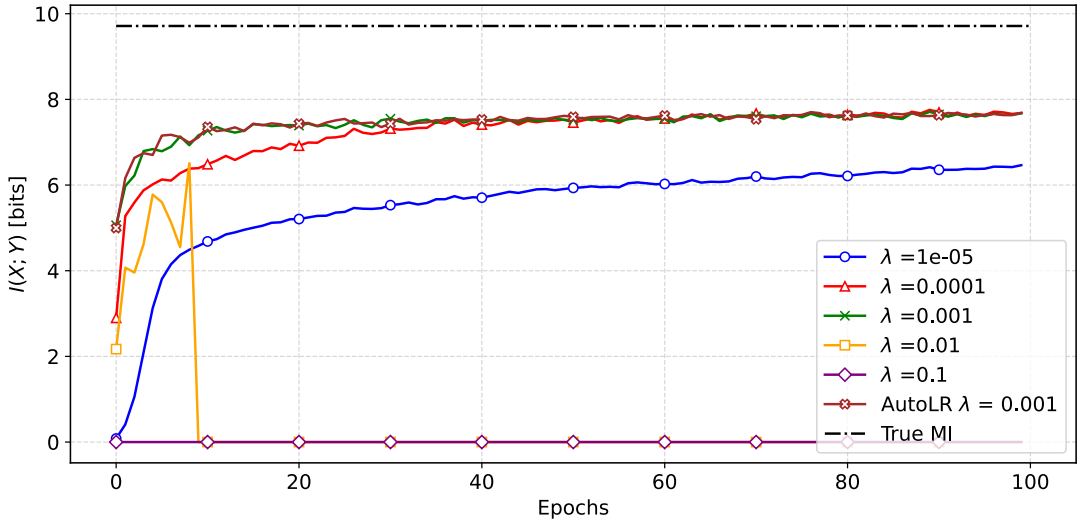


Figure 2.11: Automatic LR Tuner finds the optimal LR, in this case $\lambda = 10^{-3}$. Gaussian Dataset with $d = 20$ and $\rho = 0.7$.

This method is very useful to avoid the trial and error process of finding the optimal learning rate. Indeed, the optimal LR is dependant on many factors such as the dataset, the network architecture, the optimization algorithm, etc. It is thus very hard to find the optimal learning rate without trying many values by hand and see after convergence which value is the best. This method is very useful to save time and resources. It is also more user-friendly as it creates a black box estimator for the non-experienced user in which he only need to plug in the data to obtain an estimation of the MI. The algorithm is summarized in Algorithm 2.3.

Algorithm 2.3: Automatic LR Tuner

Input: Loss function $\mathcal{L}$, Training dataset $\{\boldsymbol{x_j}, \boldsymbol{y_j}\}_{j=1}^{N}$.

Output: Optimal learning rate λ^*.

Parameters: Learning rate range $(\lambda_{min}, \lambda_{max})$, Number of test LR N_λ, Number of epochs for pretraining N_{Pre}, Depth of pretraining D.

Initialization: $\lambda^* \leftarrow 0$, $\hat{I}^* \leftarrow 0$.

for *depth in range* D **do**

  for λ *in range logspace*$(\lambda_{min}, \lambda_{max}, N_\lambda)$ **do**

    Train Network: Train the network for N_{Pre} epochs with learning rate λ.

    Save Last MI Estimation: $\hat{I}_\lambda \leftarrow \hat{I}[N_{Pre}]$.

    if $\hat{I}_\lambda > \hat{I}^*$ **then**

      $\lambda^* \leftarrow \lambda$, $\hat{I}^* \leftarrow \hat{I}_\lambda$

    end

  end

  Refine Learning Rate: Refine the learning rate around λ^*

  $\lambda_{min} \leftarrow \lambda^*/10$, $\lambda_{max} \leftarrow 10\lambda^*$.

end

Learning Rate Scheduler

An issue that might occur with the previous method is that the optimal LR might give good results at the beginning of the training but will be too high for the rest of the training and cause the estimation to diverge. To prevent this issue, we can use a learning rate scheduler. A learning rate scheduler is a method that allows to modify the learning rate during the training. The learning rate is usually decreased during the training although some schedulers such as the Cyclic LR Scheduler can increase the learning rate during the training.

We tested different LR Schedulers such as the Step, Cyclic and Reduce On Plateau LR Scheduler [40]. The results are shown in Figure 2.12.

The Step LR Scheduler is a simple scheduler that decreases the learning rate by a factor γ every *step* epochs. The Reduce On Plateau LR Scheduler is a scheduler

that decreases the learning rate when a monitored metric, i.e. the MI estimation in our case, does not improve for a certain number of epochs, similar to the early stopping mechanism. Finally, the Cyclic LR Scheduler is a scheduler that cycles the learning rate between a minimum and a maximum value, here we considered triangular cycles.

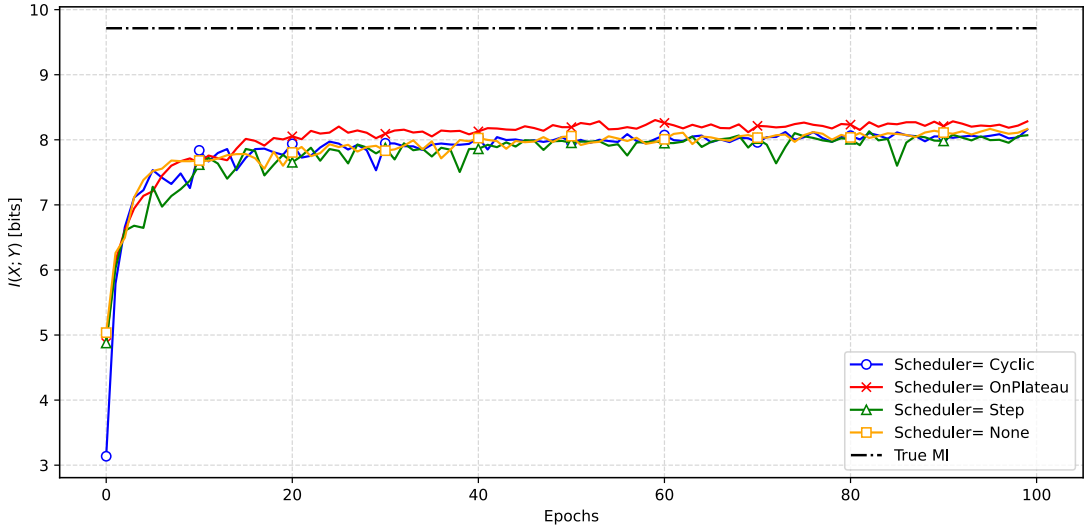


Figure 2.12: Learning Rate Schedulers results. Gaussian Dataset with $d = 20$ and $\rho = 0.7$.

We found that, generally speaking, the Reduce On Plateau Scheduler gave the best results both in terms in stability and accuracy. Indeed, we see a small increase in accuracy for the Reduce On Plateau Scheduler, although this dataset might not be the most representative of this benefit. The Step Scheduler was also very good and can lead to the same results as the Reduce On Plateau but required a lot of tuning to find the optimal step and gamma. The Cyclic Scheduler gave relatively good result in this experiment but this is not always the case as it strongly depends on the parameters of the scheduler as well as the application considered.

As claimed in the previous section, we see on Figure 2.13 that a LR Scheduler can help to mitigate the instabilities we might encounter with some setups. Here we see that the LR Scheduler allows us to completely stabilize the convergence of the network for 10^7 samples that used to be very unstable. This does however happens at the expense of a small loss of convergence speed at the beginning of the training, even though we quickly converge back to the same level of MI estimation.

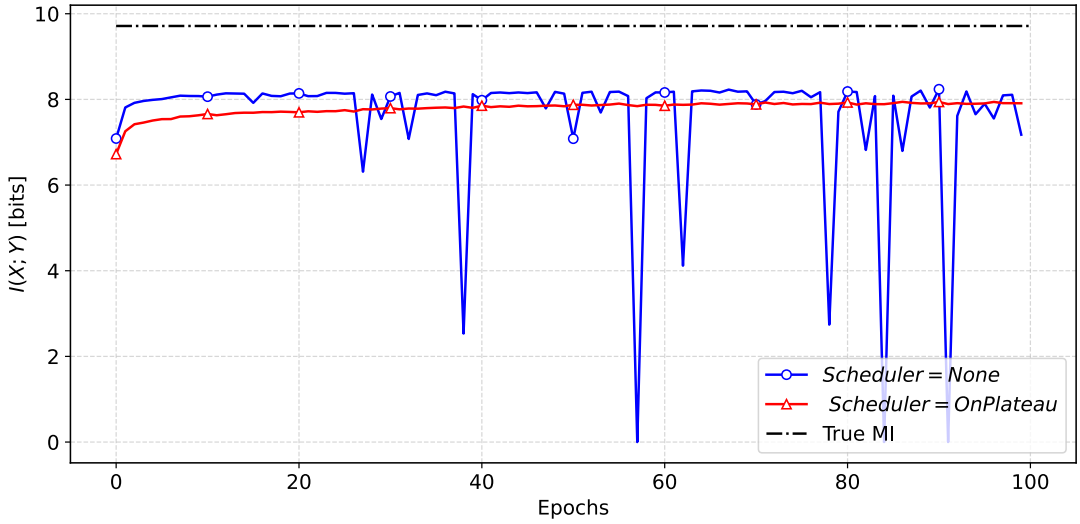


Figure 2.13: Learning Rate Schedulers with $N_{Samples} = 10^7$, we see that the scheduler stabilizes the estimation. Gaussian Dataset with $d = 20$ and $\rho = 0.7$.

Chapter Conclusion

In this chapter, we wanted to record experiments to understand the behavior of MINE under different setups in order to enlighten us in regard to the directions we should take to optimize the algorithm.

We showed through different experiments the effects of various parameters on the performance of the algorithm. Table 2.2 summarizes these effects.

Parameter	**Too Low**	**Too High**
Depth	Cannot Model Data Complexity	Unstable Estimation
Width	Cannot Model Data Complexity	Overfitting → Overestimation
Batch Size	Smoother/Less Accurate	Noisier/More Accurate
Number of Samples	Noisier/Less Accurate	Faster/Unstable
Learning Rate	More Stable/Slower	Unstable

Table 2.2: Summary of the effects of hyperparameters on the performance.

In the light of these experiments, we implemented a series of optimizations to improve the performance of the algorithm. We presented the *Early Stopping*, the *Automatic Learning Rate Tuner* and the *Learning Rate Scheduler*. We showed that these optimizations can greatly improve the performance of the algorithm in terms of time, stability and accuracy. Table 2.3 summarizes the effects of these optimizations.

Optimization	**Additions**
Early Stopping	Faster Estimation/ Prevent Overfitting and Late Stage Instabilities
Auto LR Tuner	Faster/More Stable Estimation/ User Friendliness
LR Scheduler	Stabilizes Convergence/ (may be) More Accurate

Table 2.3: Summary of the optimizations implemented in the MINE algorithm.

We see that the optimizations we added allow us to alleviate some of the issues we encountered when we characterized the network, especially by reducing convergence time and by increasing convergence stability.

Furthermore, the *Automatic Learning Rate Tuner* optimization allows us to avoid the trial and error process of finding the optimal learning rate. Which is also very useful for non deep learning expert users.

Future Works and Perspectives

We wanted to present here other possible optimizations/features that could possibly be added to the algorithm although we did not have the time to fully implement or explore them in depth in this work. We still believe that they could be very useful to improve the performance of the algorithm and leave them as suggestions for future work.

First, Transfer Learning could be a very interesting paradigm to explore in the context of MI estimation. This could be useful in an application where we do not have access to a lot of data but where we could have pretrained a network on a dataset sharing similarities with the data we want to estimate the MI from. An example using the dataset of Chapter 3 is given in Appendix A.1.

Another improvement we did not have the chance to fully implement was inspired from [41]. In this paper, the authors propose a method to tighten variational

representations of divergences and to speed up their estimation process. This means that, theoretically, we could implement an improved DV bound loss that would speed up the estimation of MINE and hopefully make it more accurate. For more informations and a theoretical implementation of this new loss in MINE, please refer to A.2.

CHAPTER 3

Application to Security in Wireless Communications

One of the main concerns in wireless communication is security against eavesdropping. New approaches, such as the one discussed in [2], were developed in high frequency wireless links in order to provide absolute security. As defined in the original paper, absolute security means that it is possible to set a communication scheme such that an eavesdropper (Eve) does not obtain any information about messages sent by a transmitter (Alice) to a receiver (Bob). More precisely, we suppose that Eve has a signal intensity detection threshold under which she does not detect the signal at all. The region of space where Eve does not detect at least one channel is called the blind region. Then, if Eve is located in the blind region, the communication scheme guarantees that Eve will have 0 mutual information with the messages sent by Alice, implying that the system is secure. This information-theoretic approach gives a strong security guarantee on the eavesdropper ability to decode hidden messages.

However, we wanted to consider another threat model where Eve would not have to be necessarily located in the blind region. Indeed, in practice, it might happen that Eve moves during transmission and ends up in a region where she can detect every channel. In this case, we wanted to estimate the amount of information she could potentially extract from the messages, i.e. how much information are we leaking to the eavesdropper. To do so, we used the estimator developed in Chapter 2 to compute the mutual information between the messages and the eavesdropper's observations.

In this chapter, we will first present the communication scheme used in [2]

and the new threat model we want to consider. We will then derive some general theoretical results regarding the mutual information leakage to an eavesdropper and give the analytical solution of the MI leakage in our threat model. We then use a synthetic dataset to estimate the MI using MINE and confront the results to the theory. We will also compare the previous results to the estimation of MINE on an experimental dataset we generated in a laboratory to simulate the communication scheme. Finally, we will give a procedure to evaluate the security of a system using MINE.

3.1 System and Threat Model

The absolute security approach in [2] jointly designs the physical layer radiation pattern and secure coding to achieve information-theoretic security. In the physical layer, the radiation patterns, and more precisely the radiation minima, are engineered to vary with frequency, to achieve the goal that Eve fails to detect at least one frequency channel in a significant spatial area. For a given frequency channel i, define the channel's blind region Ω_i as the region of space (Cfr Figure 3.1) where the signal intensity is below the eavesdropper Eve's detection threshold $\delta > 0$, i.e. the intensity threshold under which the eavesdropper cannot detect a transmitted signal. The spatial region where Eve fails to detect at least one frequency channel is defined as the blind region and is given by $\Omega = \bigcup_i \Omega_i$.

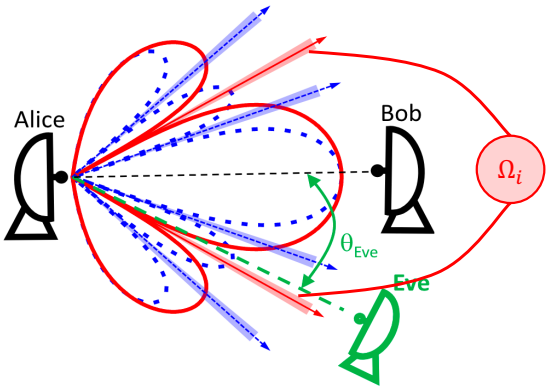


Figure 3.1: Blind Region Ω_i for frequency channel i (Adapted from [2]).

The secure encoding scheme for Alice sending N messages $\{M_i\}_{i=1}^N$ to Bob over K channels is as follows. Consider N messages $\{M_i\}_{i=1}^N \in \mathbb{F}_q$, assumed to be uniformly distributed on $\mathbb{F}_q$, that we want to send wirelessly to Bob with $N \leq K$, where $\mathbb{F}_q$ is the q-Galois Field. If $N < K$, we add $K - N$ uniformly random

messages $\{T_i\}_{i=1}^{K-N}$ to send together with the other N messages as described in [2]. Defining the secure communication efficiency as $\eta = \frac{N}{K}$, the number of meaningful messages sent over the total number of messages sent. We see that it is more advantageous for Alice and Bob to have $N = K$ to maximize her communication efficiency. For the sake of simplicity, we will consider that $N = K$ in the rest of the chapter.

The encoded messages are then obtained as

$$\boldsymbol{X} = \boldsymbol{E}\boldsymbol{M}, \tag{3.1}$$

where $\boldsymbol{X} = [X_1, X_2, \cdots, X_N] \in \mathbb{F}_q^N$ is the vector containing the encoded messages, $\boldsymbol{M} = [M_1, M_2, \cdots, M_N] \in \mathbb{F}_q^N$ is the vector of original messages and $\boldsymbol{E} \in \mathbb{F}_q^{N \times N}$ is an encoding matrix with $\text{rank}(\boldsymbol{E}) = N$ and such that the encoded symbols will also be uniformly distributed on $\mathbb{F}_q$. In [42]-[43], the authors give systematic methods to create such matrix in order to guarantee the security requirements.

The encoded messages will then be mapped onto a constellation $\mathcal{C}$ to obtain $\{Y_i\}_{i=1}^{N} \in \mathcal{C} \subset \mathbb{C}$, the N constellation points that will be sent through the channels. When received and demapped correctly, the received encoded messages $\tilde{\boldsymbol{X}}$ can be used to retrieve the original messages $\boldsymbol{M}$ as follows:

$$\tilde{\boldsymbol{M}} = \boldsymbol{E}^{-1}\tilde{\boldsymbol{X}}. \tag{3.2}$$

Obviously, if $\tilde{\boldsymbol{X}} \neq \boldsymbol{X}$, we will not obtain the correct original messages so $\tilde{\boldsymbol{M}} \neq \boldsymbol{M}$. Figure 3.2 summarizes the communication scheme.

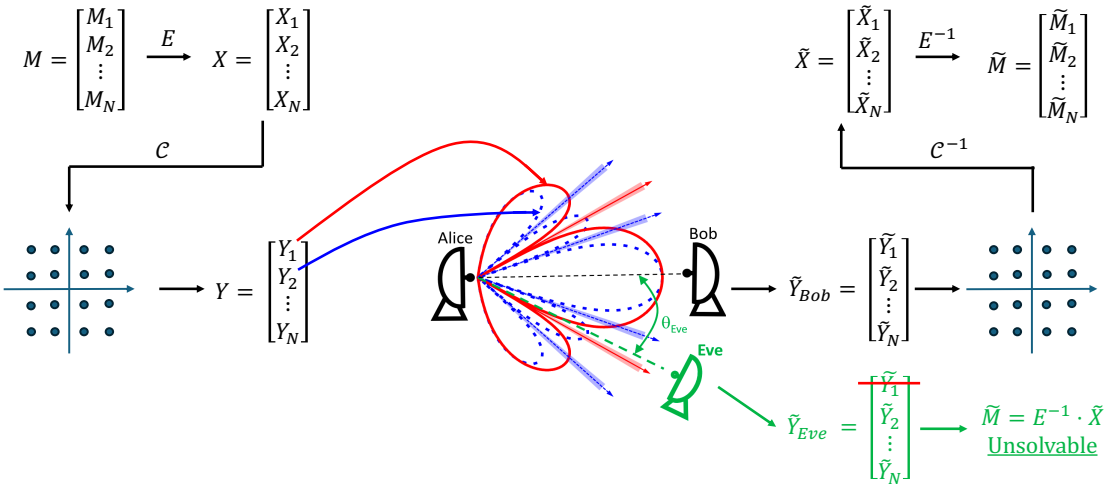


Figure 3.2: Absolute Security Scheme. Original messages $\boldsymbol{M}$ are encoded into $\boldsymbol{X}$, then mapped to $\boldsymbol{Y}$ before transmition The received symbols $\tilde{\boldsymbol{Y}}$ are demapped to $\tilde{\boldsymbol{X}}$ and decoded to retrieve the original messages $\tilde{\boldsymbol{M}} = \boldsymbol{M}$ if $\tilde{\boldsymbol{X}} = \boldsymbol{X}$.

But since we assumed that Eve is in the blind region, she is lacking at least one equation ($\tilde{\boldsymbol{Y}}_{\mathbf{1}}$ in Figure 3.2) so the system $\tilde{\boldsymbol{M}} = \boldsymbol{E}^{-1} \cdot \tilde{\boldsymbol{X}}$ cannot be solved and the original messages cannot be retrieved. Indeed, it has been proven in [2] that the mutual information between any original message M_i and any set of $N-1$ encoded messages is null, i.e. $I(M_i; \{X_j\}_{j\in\mathcal{R}}) = 0$ for any (N-1)-index set $\mathcal{R}$. This is summarized and proven in Proposition 3.1. This security claim is called individual security, meaning that the eavesdropper cannot decode any message individually. The joint security, i.e. $I(\{M_i\}_{i=1}^N; \{X_j\}_{j\in\mathcal{R}}) = 0$, is in general not respected. However, the information that Eve can obtain in that scenario is trivial, it actually only allows her to know linear combinations of the $\{M_i\}_{i=1}^N$.

Proposition 3.1

The mutual information between one original message M_i and any set of $N-1$ encoded messages $\{X_j\}_{j\in\mathcal{R}}$ is zero i.e. $I(M_i; \{X_j\}_{j\in\mathcal{R}}) = 0$, with $\mathcal{R}$ a set of indices s.t. $|\mathcal{R}| = N-1$.

Proof

Let $\mathcal{R}$ be a set of indices s.t. $|\mathcal{R}| = N - 1$. We have

$$\begin{aligned} I(M_i; \{X_j\}_{j\in\mathcal{R}}) &= H(\{X_j\}_{j\in\mathcal{R}}) - H(\{X_j\}_{j\in\mathcal{R}}|M_i) \\ &= H(\{X_j\}_{j\in\mathcal{R}}) - H(\{\boldsymbol{E}^{-1}_{j,:} \cdot \boldsymbol{M}\}_{j\in\mathcal{R}}|M_i) \\ &= H(\{X_j\}_{j\in\mathcal{R}}) - (N-1)\log_2(|\mathbb{F}_q|) \\ &\leq \sum_{j\in\mathcal{R}} H(X_j) - (N-1)\log_2(q) \\ &= (N-1)\underbrace{H(X_j)}_{\log_2(q)} - (N-1)\log_2(q) \\ &= 0, \end{aligned}$$

where the equality $H(X_j) = \log_2(q)$ comes from the fact that the encoded messages $\{X_j\}_{j=1}^N$, just as the original messages $\{M_i\}_{i=1}^N$, are uniformly distributed in $\mathbb{F}_q$. The notation $\boldsymbol{E}^{-1}_{j,:}$ is used to denote the j^{th} line of the matrix $\boldsymbol{E}^{-1}$. □

One unique feature that is worth noting regarding this encoding scheme is the fact that by increasing the number of frequency channels K, in addition to increasing the data rate to Bob, we also broaden the blind region Ω and thus improve the security at the same time.

Absolute Security over AWGN Channels

The previous explanations considered the case where the eavesdropper received every symbol either perfectly or not at all, i.e. when Eve is in the blind region. However, this model does not allow us to understand what happens in the case where the signals' intensity received by Eve are not below her detection threshold. More precisely, we need to take into account scenarios where the eavesdropper might receive all the constellation points $\{Y_i\}_{i=1}^N$ that were initially sent. These received symbols should be degraded in some way by the channel via fading, noise, etc. and we want to understand how much information the eavesdropper can extract from these received symbols.

In this work, we will consider the following threat model: Eve is able to receive all the encoded messages $\{Y_i\}_{i=1}^N$ but degraded by noise. We will suppose that the channels are Additive White Gaussian Noise (AWGN) channels. We also consider that we are facing a known-plaintext attack [44], meaning that Eve knows the encoding matrix $\boldsymbol{E}$ as well as the constellation $\mathcal{C}$ and, for a given period of time,

knows the messages sent by Alice. This will allow her to have access to a given number of pair samples $(M_i; \{Z_j\}_{j=1}^N)$ in order to estimate herself the information leakage and judge if the eavesdropping is worthwhile in her current position.

Let $\{Z_i\}_{i=1}^N \in \mathbb{C}$ be the N points received by Eve. We suppose that each one of the channels follows the AWGN channel model, i.e.

$$Z_i = \alpha_i Y_i + N_i,$$

where $\alpha_i \in \mathbb{C}$, with $|\alpha_i| < 1$, is an attenuation factor accounting for the transmitter gain, receiver gain, path loss, etc. and $N_i \sim \mathcal{CN}(0, \sigma_i^2)$ is an additive white Gaussian noise of variance σ_i^2. Both parameters are specific to each channel. The effective Signal-to-Noise-Ratio (SNR) for each channel is $SNR_i = |\alpha_i|^2/\sigma_i^2$. In practice, we will thus use the following equivalent channel model:

$$Z_i = Y_i + \tilde{N}_i,$$

where now $\tilde{N}_i \sim \mathcal{CN}(0, \tilde{\sigma}_i^2 = \sigma_i^2/|\alpha_i|^2)$ is the effective noise for the i-th channel. For simplicity, we express as our model $Z_i = Y_i + N_i$ with $N_i \sim \mathcal{CN}(0, \sigma_i^2)$ in the rest of the section with the understanding of the complete model described hereabove.

Our objective is now to evaluate $I(M_i; \{Z_j\}_{j=1}^N)$, the mutual information between one original message M_i and all of the eavesdropper's noisy observations Z_j in that more general scenario.

3.2 Theoretical Analysis

In this section, we derive some theoretical results about convergence and bounds for the mutual information $I(M_i; \{Z_j\}_{j=1}^N)$ in order to establish a ground truth to confront the results of MINE against. This will give us confidence in the reliability of our estimator, hopefully even for scenarios with no analytical solutions. The results on the MI upper bound derived hereunder are especially useful for us since it allows us to bound a numerically intractable case by a tractable one with an analytical solution. This upper bound will actually represent the worst case scenario where the eavesdropper receives every channel perfectly except one, which is a good approximation of the general case where the eavesdropper receives every channel with low noise on $N - 1$, the last channel being much more noisy.

General Properties of Information Leakage

We first give a general expression of the mutual information leakage to an eavesdropper in the communication scheme described above. We then derive an upper

bound on the information leakage. These results are general and do not depend on the specific channel model we are considering.

Proposition 3.2: General Expression of $I(M_i; \{Z_j\}_{j=1}^N)$

The general expression of the information leakage to an eavesdropper of one original message M_i considering she observes degraded signals $\{Z_j\}_{j=1}^N$ through an arbitrary channel is given by

$$I(M_i; \{Z_j\}_{j=1}^N) = h(Z_k) - h(Z_k|\{Z_j\}_{j\neq k}, M_i). \tag{3.3}$$

Proof

From Definition 1.5, we can write in general

$$I\left(M_i; \{Z_j\}_{j=1}^N\right) = h\left(\{Z_j\}_{j=1}^N\right) - h\left(\{Z_j\}_{j=1}^N|M_i\right). \tag{3.4}$$

Let's compute each of the terms separately. For the first term, we have

$$\begin{aligned} h\left(\{Z_j\}_{j=1}^N\right) &= h\left(Z_1, ..., Z_N\right) \\ &= \sum_{n=1}^N h\left(Z_n|\{Z_j\}_{j<n}\right) \quad \text{(by the chain rule)} \\ &= \sum_{n=1}^N h(Z_n). \end{aligned} \tag{3.5}$$

The last equality is justified by the independence of Z_i and $\{Z_j\}_{j=1}^{i-1}$. Indeed, even observing $N-1$ non-noisy encoded messages (i.e., $\{Y_j\}_{j=1}^{N-1}$) does not give the observer any information about the last encoded message (i.e., Y_N) which can take any value in $\mathbb{F}_q$ uniformly randomly.

Next, the second term is given by

$$\begin{aligned} h\left(\{Z_j\}_{j=1}^N\middle|M_i\right) &= \sum_{n=1}^N h\left(Z_n\middle|\{Z_j\}_{j<n}, M_i\right) \quad \text{(by the chain rule)} \\ &= h\left(Z_N\middle|\{Z_j\}_{j<N}, M_i\right) + \sum_{n=1}^{N-1} h\left(Z_n\middle|\{Z_j\}_{j<n}, M_i\right) \\ &= h\left(Z_N\middle|\{Z_j\}_{j<N}, M_i\right) + \sum_{n=1}^{N-1} h(Z_n). \end{aligned} \tag{3.6}$$

The simplification from $h\left(Z_n\middle|\{Z_j\}_{j<n}, M_i\right)$ to $h(Z_k)$ is similar to the argument in (3.5). When observing at most $N-1$ equations with N unknown

related variables (messages), one does not obtain any information regarding the last equation. In this case, observing $\{Z_1, \cdots, Z_{n-2}\}$ and M_i is equivalent to observing $n-1$ equations and therefore does not change the entropy of Z_n. This equation also holds in general for Z_n instead of Z_N, with the sum and conditioning being on $\{Z_j\}_{j\neq k}$.

Thus, using (3.5) and (3.6) in (3.4), we have as a general expression for the information leakage to Eve:

$$I(M_i; \{Z_j\}_{j=1}^N) = h(Z_k) - h(Z_k|\{Z_j\}_{j\neq k}, M_i). \quad \square$$

Proposition 3.3: Upper Bound on Information Leakage

The mutual information between one original message M_i and all of the eavesdropper's noisy observations Z_j is upper bounded by the mutual information between M_i and one noisy observation Z_k as well as the other encoded messages $\{X_j\}_{j\neq k}$, i.e.

$$I(M_i; \{Z_j\}_{j=1}^N) \leq I(M_i; Z_k, \{X_j\}_{j\neq k}). \tag{3.7}$$

Proof

To prove this bound, we can make use of the *Data Processing Inequality*:

$$I(\boldsymbol{X}; \boldsymbol{Z}) \leq I(\boldsymbol{X}; \boldsymbol{Y}) \quad \text{if} \quad \boldsymbol{X} \to \boldsymbol{Y} \to \boldsymbol{Z},$$

where $\boldsymbol{X}$, $\boldsymbol{Y}$ and $\boldsymbol{Z}$ can be in general random vectors. Here, we will have the correspondance $M_i \to \{X_j\}_{j=1}^N \to \{Z_j\}_{j=1}^N$. The bound is thus directly given and we indeed have

$$I(M_i; \{Z_j\}_{j=1}^N) \leq I(M_i; Z_k, \{X_j\}_{j\neq k}). \quad \square$$

Properties of the AWGN Channel

We will derive properties specific to the AWGN channel model, which will give us a ground truth to compare the results from MINE against in the following sections. We first study the convergence of the mutual information when the SNR of one channel tends to 0, and when the smallest SNR tends to $+\infty$. We then develop Proposition 3.2 using the AWGN channel model to obtain an analytical expression of the mutual information leakage for this particular scenario. Finally,

we particularize the expression of the mutual information leakage to the case where the eavesdropper receives every channel perfectly except one which will be affected by noise.

Convergence Properties

Proposition 3.4: Convergence of $I(M_i; \{Z_j\}_{j=1}^N)$ when $\text{SNR}_k \to 0$

The mutual information between one original message M_i and all of the eavesdropper's noisy observations Z_j converges to 0 when the SNR of one noisy channel k tends to 0, i.e.

$$\lim_{\text{SNR}_k \to 0} I(M_i; \{Z_j\}_{j=1}^N) = 0. \tag{3.8}$$

This reassures us in the fact that when, in the limit, one channel becomes completely noisy or attenuated, the mutual information between the original message and the eavesdropper's observations will be null as in the absolute security model.

Proof

Since we know we can upper bound $I(M_i; \{Z_j\}_{j=1}^N) \leq I(M_i; Z_k, \{X_j\}_{j\neq k})$, let's prove that the upper bound converges to 0 when $\text{SNR}_k \to 0$.
We can write

$$\begin{aligned} I(M_i; Z_k, \{X_j\}_{j\neq k}) &= h(Z_k) - h(Z_k|\{X_j\}_{j\neq k}, M_i) \\ &= h(Z_k) - h(N_k). \end{aligned}$$

Let's express each of these terms separately. First express the individual probability density functions,

$$\begin{aligned} f_{N_k}(n) &= \frac{1}{\pi\sigma_k^2} e^{-\frac{|n|^2}{\sigma_k^2}}, \qquad \text{and} \\ f_{Z_k}(z) &= f_{Y_k} \circledast f_{N_k}(z) \\ &= [\sum_{c_i \in \mathcal{C}} \frac{1}{|\mathcal{C}|} \delta(y - c_i)] \circledast \frac{1}{\pi\sigma_k^2} e^{-\frac{|n|^2}{\sigma_k^2}} \\ &= \frac{1}{|\mathcal{C}|\pi\sigma_k^2} \sum_{c_i \in \mathcal{C}} e^{-\frac{|z-c_i|^2}{\sigma_k^2}}. \end{aligned}$$

We can now express $h(Z_k)$ as

$$h(Z_k) = -\int_{-\infty}^{+\infty} \frac{1}{|\mathcal{C}|\pi\sigma_k^2} \sum_{c_i \in \mathcal{C}} e^{-\frac{|z-c_i|^2}{\sigma_k^2}} \log\left(\frac{1}{|\mathcal{C}|\pi\sigma_k^2} \sum_{c_i \in \mathcal{C}} e^{-\frac{|z-c_i|^2}{\sigma_k^2}}\right) dz.$$

Now, we see that when $\mathrm{SNR}_k \to 0 \Leftrightarrow \sigma_k^2 \to +\infty$, f_{Z_k} approaches f_{N_k}:

$$\begin{aligned}
\lim_{\sigma_k^2 \to +\infty} f_{Z_k}(z) &= \lim_{\sigma_k^2 \to +\infty} \frac{1}{|\mathcal{C}|\pi\sigma_k^2} \sum_{c_i \in \mathcal{C}} e^{-\frac{|z-c_i|^2}{\sigma_k^2}} \\
&= \frac{1}{|\mathcal{C}|\pi\sigma_k^2} \sum_{c_i \in \mathcal{C}} e^{-\frac{|z|^2}{\sigma_k^2}} = \frac{1}{\pi\sigma_k^2} e^{-\frac{|z|^2}{\sigma_k^2}} \\
&= f_{N_k}(z).
\end{aligned}$$

Which also means that $h(Z_k) \to h(N_k)$ when $\mathrm{SNR}_k \to 0$. We thus have for the mutual information:

$$\lim_{\mathrm{SNR}_k \to 0} I(M_i; Z_k, \{X_j\}_{j \neq k}) = \lim_{\mathrm{SNR}_k \to 0} \underbrace{h(Z_k)}_{\to h(N_k)} - \underbrace{h(Z_k|\{X_j\}_{j \neq k})}_{h(N_k)} = 0. \quad \square$$

Proposition 3.5: Convergence of $I(M_i; Z_k, \{X_j\}_{j \neq k})$ when $\mathrm{SNR}_k \to +\infty$

The mutual information between one original message M_i and all of the eavesdropper's noisy observations Z_j converges to $h(X_k) = \log(q)$ when the SNR of the less noisy channel k tends to $+\infty$, i.e.

$$\lim_{\mathrm{SNR}_k \to +\infty} I(M_i; \{Z_j\}_{j=1}^N) = \log(q). \tag{3.9}$$

This means that when the eavesdropper receives the channel perfectly, the mutual information between the original message and the eavesdropper's observations is equal to the entropy of the constellation.

Proof

When $SNR_k \to +\infty$ where $SNR_k = \min_i SNR_i$, we have that $Z_i \to Y_i$, $\forall i$, and consequently $I(M_i; \{Z_j\}_{j=1}^N) \to I(M_i; \{X_j\}_{j=1}^N)$.
Then, we have that

$$I(M_i; \{X_j\}_{j=1}^N) = H(M_i) - H(M_i|\{X_j\}_{j=1}^N) = \log(q) - 0 = \log(q). \quad \square$$

Proposition 3.6: Rate of Convergence of $I(M_i; \{Z_j\}_{j=1}^N)$ to 0

The mutual information between one original message M_i and the noisy observation $\{Z_j\}_{j=1}^N$ converges to 0 at a rate of at least $1/\sigma_k^2$ when $SNR_k \to 0$, i.e.

$$I(M_i; \{Z_j\}_{j=1}^N) \underset{1/\sigma_k^2}{\to} 0. \tag{3.10}$$

Meaning that SNRs $I(M_i; \{Z_j\}_{j=1}^N) \propto 1/\sigma_k^2$ or that $\log_{10}\left(I(M_i; \{Z_j\}_{j=1}^N)\right) \propto SNR_k$ for low SNRs.

Proof

First, let us again upper bound $I(M_i; \{Z_j\}_{j=1}^N) \leq I(M_i; Z_k, \{X_j\}_{j\neq k})$ and prove that the upper bound converges to 0 at a rate of $1/\sigma_k^2$ when $\text{SNR}_k \to 0$.

Define $\rho(\sigma_k) = \frac{f_{N_k}(n)}{f_{Z_k}(z)} = \frac{e^{-|z|^2/\sigma_k^2}}{\frac{1}{|\mathcal{C}|}\sum_{c_i \in \mathcal{C}} e^{-|z-c_i|^2/\sigma_k^2}}$.

We see that, since the c_i are constants, $e^{-|z-c_i|^2/\sigma_k^2} \to e^{-|z|^2/\sigma_k^2}$ at a rate of $1/\sigma_k^2$. This also implies that the denominator tends to the numerator at a rate of $1/\sigma_k^2$. Thus, $\rho(\sigma_k) \underset{\mathcal{O}(1/\sigma_k^2)}{\to} 1$.

We can rewrite the mutual information to make $\rho(\sigma_k)$ appear:

$$\begin{aligned} &I(M_i; Z_k, \{X_j\}_{j\neq k}|\sigma_k) = h(Z_k) - h(N_k) \\ &= -\int_{-\infty}^{+\infty} f_{Z_k} \log(f_{Z_k}) + \int_{-\infty}^{+\infty} f_{N_k} \log(f_{N_k}) \\ &= \int_{-\infty}^{+\infty} f_{Z_k}(z)[\underbrace{\rho(\sigma_k)\log(\rho(\sigma_k) f_{Z_k}(z))}_{\underset{\mathcal{O}(1/\sigma_k^2)}{\to} \log(f_{Z_k})} - \log(f_{Z_k}(z))]\, dz. \end{aligned}$$

Thus, we see that the integrand converges to 0 at a rate of $1/\sigma_k^2$, implying that

$$I(M_i; Z_k, \{X_j\}_{j\neq k}|\sigma_k) \underset{\mathcal{O}(1/\sigma_k^2)}{\to} 0. \quad \square$$

Expression of Information Leakage on AWGN Channel

Based on Proposition 3.2, we see that we are going to need to compute 2 differential entropies, $h(Z_k)$ and $h(Z_k|\{Z_j\}_{j\neq k}, M_i)$ to obtain $I(M_i; \{Z_j\}_{j=1}^N)$. To do so, we will need to obtain an expression for the probability density function of Z_k (already given in the proof of Proposition 3.4) and the conditional probability density function of Z_k given the other observations and the original message M_i.

Proposition 3.7: Entropy of Z_k

The entropy of the noisy observation Z_k is given by

$$h(Z_k) = -\int_{-\infty}^{+\infty} \frac{1}{|\mathcal{C}|\pi\sigma_k^2} \sum_{c_i \in \mathcal{C}} e^{-\frac{|z-c_i|^2}{\sigma_k^2}} \log\left(\frac{1}{|\mathcal{C}|\pi\sigma_k^2} \sum_{c_i \in \mathcal{C}} e^{-\frac{|z-c_i|^2}{\sigma_k^2}}\right) dz. \quad (3.11)$$

Proof: Entropy of Z_k

We have already computed the probability density function of Z_k in the proof of Proposition 3.4. We can now use it to compute the differential entropy of Z_k as follows:

$$\begin{aligned} h(Z_k) &= -\int_{-\infty}^{+\infty} f_{Z_k}(z)\log(f_{Z_k}(z))\,dz \\ &= -\int_{-\infty}^{+\infty} \frac{1}{|\mathcal{C}|\pi\sigma_k^2}\sum_{c_i\in\mathcal{C}} e^{-\frac{|z-c_i|^2}{\sigma_k^2}} \log\left(\frac{1}{|\mathcal{C}|\pi\sigma_k^2}\sum_{c_i\in\mathcal{C}} e^{-\frac{|z-c_i|^2}{\sigma_k^2}}\right) dz. \end{aligned}$$

Proposition 3.8: Conditional Entropy of Z_k given $\{Z_j\}_{j\neq k}$ and M_i

The conditional entropy of the noisy observation Z_k given the other observations and the original message M_i is given by

$$\begin{aligned} h(Z_k|\{Z_j\}_{j\neq k}, M_i) = -\int_{-\infty}^{+\infty} &\left(\frac{1}{\pi\sigma_k^2}\sum_{c_i\in\mathcal{C}} f_{Y_k}(y_k = c_i|\{Z_j\}_{j\neq k}) e^{-\frac{|z_k-c_i|^2}{\sigma_k^2}}\right) \\ &\left(\prod_{m\neq k}\frac{1}{|\mathcal{C}|\pi\sigma_m^2}\sum_{c_i\in\mathcal{C}} e^{-\frac{|z_m-c_i|^2}{\sigma_m^2}}\right) \\ &\log\left(\frac{1}{\pi\sigma_k^2}\sum_{c_i\in\mathcal{C}} f_{Y_k}(y_k = c_i|\{Z_j\}_{j\neq k}) e^{-\frac{|z-c_i|^2}{\sigma_k^2}}\right) d\boldsymbol{z}. \end{aligned} \quad (3.12)$$

Proof: Conditional Entropy of Z_k given $\{Z_j\}_{j\neq k}$ and M_i

We can express the conditional probability density function of Z_k given the other observations and the original message M_i as follows:

$$f_{Z_k}(z_k|\{Z_j\}_{j\neq k}, M_i) = f_{Y_k}(y_k|\{Z_j\}_{j\neq k}, M_i) \circledast f_{N_k}(n). \quad (3.13)$$

We thus need to compute the conditional probability density function of Y_k given the other observations and the original message M_i.
Remembering that there is a one-to-one mapping between the constellation points Y_i and the encoded messages X_i, themselves related to the original messages M_i through the encoding matrix $\boldsymbol{E}$, we can say

$$f_{Y_k}(y_k|\{Z_j\}_{j\neq k}, M_i) = f_{X_k}(x_k|\{Z_j\}_{j\neq k}, M_i). \quad (3.14)$$

Now, let us use the relationship between encoded symbols and original messages:

$$\boldsymbol{X} = \boldsymbol{E}\boldsymbol{M} \Rightarrow M_i = \boldsymbol{E}_{i,:}^{-1}\boldsymbol{X} \Leftrightarrow \boldsymbol{E}_{i,k}^{-1}X_k = M_i - \sum_{j\neq k} \boldsymbol{E}_{i,j}^{-1}X_j.$$

Knowing this relationship, we see that the probability of observing X_k is related to the probability of observing $\{X_j\}_{j\neq k}$.
Consequently, the probability of having $X_k = x_k$ conditioned on knowing $\{Z_j\}_{j\neq k} = \{z_j\}_{j\neq k}$ and $M_i = m_i$ is the same as the probability $\sum_{j\neq k} \boldsymbol{E}_{i,j}^{-1}X_j = \sum_{j\neq k} \boldsymbol{E}_{i,j}^{-1}x_j$ with the same conditioning.
Let us first define the set $\mathcal{I} = \left\{\{x_j\}_{j\neq k} : \sum_{j\neq k} \boldsymbol{E}_{i,j}^{-1}x_j = m_i - \boldsymbol{E}_{i,j}^{-1}x_j\right\}$ then, we have for $f_{X_k}(x_k|\{Z_j\}_{j\neq k}, M_i)$:

$$\begin{aligned}
&f_{X_k}(x_k|\{Z_j\}_{j\neq k}, M_i) = f_{X_k}(m_i - \boldsymbol{E}_{i,:}^{-1}x_k = \sum_{j\neq k} \boldsymbol{E}_{i,j}^{-1}x_j|\{Z_j\}_{j\neq k}, M_i)\\
&= f_{X_k}(\underbrace{\{\{x_j\}_{j\neq k} : \sum_{j\neq k} \boldsymbol{E}_{i,j}^{-1}x_j = m_i - \boldsymbol{E}_{i,j}^{-1}x_j\}}_{=\mathcal{I}}|\{Z_j\}_{j\neq k}, M_i)\\
&= \sum_{\{x_j\}_{j\neq k}\in\mathcal{I}} f_{X_k}(\{x_j\}_{j\neq k}|\{Z_j\}_{j\neq k}, M_i)\\
&= \sum_{\mathcal{I}} \prod_{\substack{m=1\\ m\neq k}}^{N} f_{X_k}(x_m|\{X_j\}_{j\neq k, j<m}, \{Z_j\}_{j\neq k}, M_i).
\end{aligned}$$

Let us now compute $f_{X_k}(x_m|\{X_j\}_{j\neq k,j<m}, \{Z_j\}_{j\neq k}, M_i)$:

$$\begin{aligned}
&f_{X_k}(x_m|\{X_j\}_{j\neq k,j<m}, \{Z_j\}_{j\neq k}, M_i)\\
&= \frac{f_{Z_k}(z_m|\{X_j\}_{j\neq k,j<m}, X_m, M_i)\cdot f_{X_k}(x_m)}{f_{Z_k}(z_m)} = \frac{f_{Z_k}(z_m|X_m)\cdot f_{X_k}(x_m)}{f_{Z_k}(z_m)}\\
&= \frac{\frac{1}{\pi\sigma_m^2}e^{-\frac{|z_m-c_m|^2}{\sigma_m^2}}\cdot\frac{1}{|\mathcal{C}|}}{\frac{1}{|\mathcal{C}|\pi\sigma_m^2}\sum_{c_i\in\mathcal{C}} e^{-\frac{|z_m-c_i|^2}{\sigma_m^2}}} = \frac{e^{-\frac{|z_m-c_i|^2}{\sigma_m^2}}}{\sum_{c_i\in\mathcal{C}} e^{-\frac{|z_m-c_i|^2}{\sigma_m^2}}} \triangleq \gamma_m(z_m),
\end{aligned}$$

where $\gamma_m(z_m)$ is in fact the soft-max function of the noisy observation Z_m given the constellation points c_m and the noise variance σ_m^2. We can now express $f_{X_k}(x_k|\{Z_j\}_{j\neq k}, M_i)$ as follows:

$$f_{X_k}(x_k|\{Z_j\}_{j\neq k}, M_i) = \sum_{\mathcal{I}} \prod_{m=1}^{N} \gamma_k(z_m). \tag{3.15}$$

Using (3.14) and (3.15) in (3.13), we can now obtain an expression for $f_{Z_k}(z|\{Z_j\}_{j\neq k}, M_i)$:

$$\begin{aligned} f_{Z_k}(z|\{Z_j\}_{j\neq k}, M_i) &= f_{Y_k}(y|\{Z_j\}_{j\neq k}, M_i) \circledast f_{N_k}(n) \\ &= \left(\sum_{c_i \in \mathcal{C}} f_{Y_k}(y_k = c_i|\{Z_j\}_{j\neq k}) \cdot \delta(y_k - c_i) \right) \circledast f_{N_k}(n) \\ &= \sum_{c_i \in \mathcal{C}} f_{Y_k}(y_k = c_i|\{Z_j\}_{j\neq k}) \frac{1}{\pi\sigma_k^2} e^{-\frac{|z-c_i|^2}{\sigma_k^2}}. \end{aligned}$$

It is interesting to notice that the conditional probability density function of Z_k given the other observations and the original message M_i is a mixture of Gaussian distributions centered around the constellation points c_i with updated weights given by the probability distribution $f_{Y_k}(y_k = c_i|\{Z_j\}_{j\neq k})$. Finally, we simply need to compute $f(\{Z_j\}_{j=1}^N, M_i)$ to obtain the conditional entropy of Z_k given the other observations and the original message M_i:

$$\begin{aligned} &f(\{Z_j\}_{j=1}^N, M_i) \\ &= f_{Z_k}(z_k|\{Z_j\}_{j\neq k}, M_i) \cdot \prod_{m\neq k} \underbrace{f_{Z_m}(z_m|\{Z_j\}_{j\neq k, j<m}, M_i)}_{=f_{Z_m}(z_m)} \cdot f_M(M_i) \\ &= f_{Z_k}(z_k|\{Z_j\}_{j\neq k}, M_i) \cdot \prod_{m\neq k} f_{Z_m}(z_m) \cdot f_M(M_i). \end{aligned}$$

We can now express the conditional entropy of Z_k given the other observations and the original message M_i as follows:

$$h(Z_k|\{Z_j\}_{j\neq k}, M_i) = -\int_{-\infty}^{+\infty} f(\{Z_j\}_{j=1}^N, M_i)\log(f_{Z_k}(z_k|\{Z_j\}_{j\neq k}, M_i))\,d\boldsymbol{z}$$

$$= -\int_{-\infty}^{+\infty} f_{Z_k}(z_k|\{Z_j\}_{j\neq k}, M_i)\cdot\prod_{m\neq k} f_{Z_m}(z_m)\cdot f_M(M_i)$$
$$\log(f_{Z_k}(z|\{Z_j\}_{j\neq k}, M_i))\,d\boldsymbol{z}$$

$$= -\int_{-\infty}^{+\infty}\left(\frac{1}{\pi\sigma_k^2}\sum_{c_i\in\mathcal{C}} f_{Y_k}(y_k = c_i|\{Z_j\}_{j\neq k})e^{-\frac{|z_k-c_i|^2}{\sigma_k^2}}\right)\left(\prod_{m\neq k}\frac{1}{|\mathcal{C}|\pi\sigma_m^2}\sum_{c_i\in\mathcal{C}} e^{-\frac{|z_m-c_i|^2}{\sigma_m^2}}\right)$$
$$\log(\frac{1}{\pi\sigma_k^2}\sum_{c_i\in\mathcal{C}} f_{Y_k}(y_k = c_i|\{Z_j\}_{j\neq k})e^{-\frac{|z-c_i|^2}{\sigma_k^2}})\,d\boldsymbol{z},$$

where the integral is over the N-dimensional space of the noisy observations $\{Z_j\}_{j=1}^N$, i.e. $d\boldsymbol{z} = dz_1\ldots dz_N$.

We can now compute the mutual information between one original message M_i and all of the eavesdropper's noisy observations Z_j using (3.11) and (3.12) in (3.3).

Proposition 3.9: Mutual Information Leakage - AWGN Channel

The mutual information between one original message M_i and all of the eavesdropper's noisy observations Z_j is given by

$$I(M_i;\{Z_j\}_{j=1}^N) = h(Z_k) - h(Z_k|\{Z_j\}_{j\neq k}, M_i)$$
$$= -\int_{-\infty}^{+\infty}\frac{1}{|\mathcal{C}|\pi\sigma_k^2}\sum_{c_i\in\mathcal{C}} e^{-\frac{|z-c_i|^2}{\sigma_k^2}}\log\left(\frac{1}{|\mathcal{C}|\pi\sigma_k^2}\sum_{c_i\in\mathcal{C}} e^{-\frac{|z-c_i|^2}{\sigma_k^2}}\right)dz$$
$$-\int_{-\infty}^{+\infty}\left(\frac{1}{\pi\sigma_k^2}\sum_{c_i\in\mathcal{C}} f_{Y_k}(y_k = c_i|\{Z_j\}_{j\neq k})e^{-\frac{|z_k-c_i|^2}{\sigma_k^2}}\right)\left(\prod_{m\neq k}\frac{1}{|\mathcal{C}|\pi\sigma_m^2}\sum_{c_i\in\mathcal{C}} e^{-\frac{|z_m-c_i|^2}{\sigma_m^2}}\right)$$
$$\log(\frac{1}{\pi\sigma_k^2}\sum_{c_i\in\mathcal{C}} f_{Y_k}(y_k = c_i|\{Z_j\}_{j\neq k})e^{-\frac{|z-c_i|^2}{\sigma_k^2}})\,d\boldsymbol{z}. \tag{3.16}$$

It is however important to note that the computation of the mutual information is intractable in practice due to the high dimensionality of the integral. Indeed, while it should be feasible to compute numerically this integral for 2 or 3 channels (4

and 6 integral dimensions respectively), it quickly starts to become unmanageable as the number of noisy channels grows. Which is why we want to make use of MINE to estimate the mutual information between the original messages and the eavesdropper's observations.

Particular Case: One Noisy Channel

Let us study the particular case where only one channel is noisy. We can use the results obtained in the proof of Proposition 3.4 to obtain an expression for the mutual information between one original message M_i and one noisy observation Z_k as well as the other encoded messages $\{X_j\}_{j\neq k}$. Indeed, we have that

$$\begin{aligned} &I(M_i; Z_k, \{X_j\}_{j\neq k}) = h(Z_k) - h(Z_k|\{X_j\}_{j\neq k}) = h(Z_k) - h(N_k) \\ &= -\int_{-\infty}^{+\infty} \frac{1}{|\mathcal{C}|\pi\sigma_k^2} \sum_{c_i\in\mathcal{C}} e^{-\frac{|z-c_i|^2}{\sigma_k^2}} \log\left(\frac{1}{|\mathcal{C}|\pi\sigma_k^2} \sum_{c_i\in\mathcal{C}} e^{-\frac{|z-c_i|^2}{\sigma_k^2}}\right) dz - \log(\pi e\sigma_k^2). \end{aligned} \tag{3.17}$$

We could also have used Proposition 3.9 to obtain the same expression. However in this particularly simple case we already have an expression and do not need to make use of the general formula.

We can now compute the above expression relatively easily using numerical integration. This will give us a ground truth to compare the results from MINE against.

As a study case for this work, we considered the scenario presented in [2]. We have a system with $N = 3$ messages chosen in the Galois field $\mathbb{F}_{11}$ sent on $K = 3$ frequency channels. The encoding matrix $\boldsymbol{E}$ is given by

$$\boldsymbol{E} = \begin{bmatrix} 3 & 8 & 1 \\ 3 & 4 & 4 \\ 6 & 10 & 6 \end{bmatrix}. \tag{3.18}$$

The constellation is chosen to be the 11-QAM represented on Figure 3.3. This choice is completely arbitrary and does not affect the general results as all 11-points constellations behave the same in the low SNR regime, which is our regime of interest.

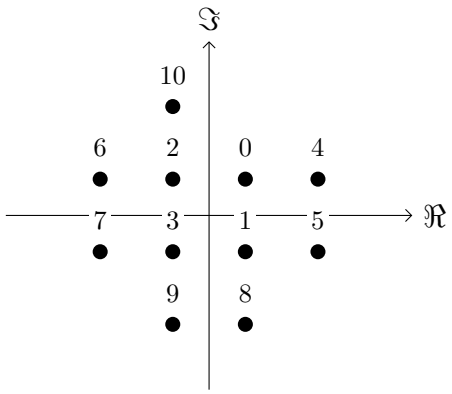
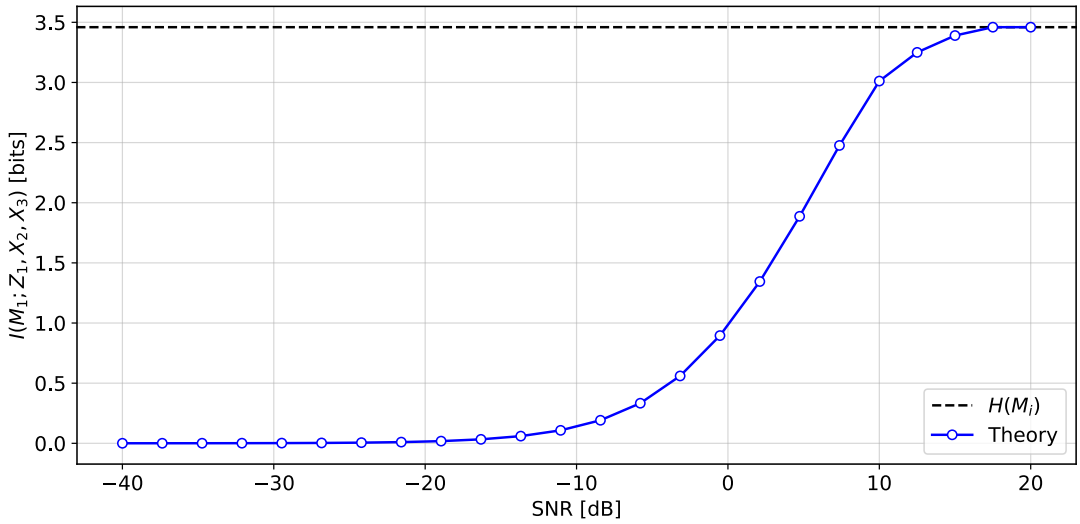
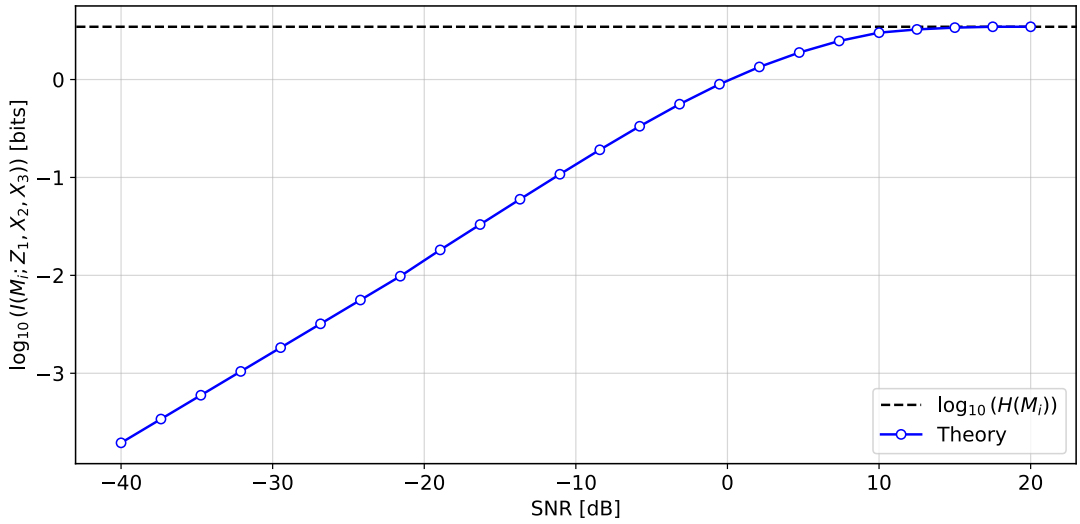


Figure 3.3: 11-QAM.

The mutual information computed numerically based on the theoretical expressions derived hereabove is presented on Figure 3.4. We note that the mutual information converges to 0 when the SNR of the noisy channel tends to 0, and that it converges to $h(M_i) = \log_2(11)$ as predicted by the theoretical results. On Figure 3.4b, we can also observe the linear rate of convergence of the mutual information to 0 when the SNR of the noisy channel tends to 0.



(a) Natural Scale



(b) Logarithmic Scale

Figure 3.4: Theoretical MI - SNR_1, using $\mathcal{C}$ = 11-QAM, $\mathrm{SNR}_2 = \mathrm{SNR}_3 = +\infty$ dB i.e. Channels 2 and 3 received perfectly.

3.3 Security Evaluation with MINE

Now that we have obtained a few theoretical statements about the mutual information between the original messages and the eavesdropper's observations, we can

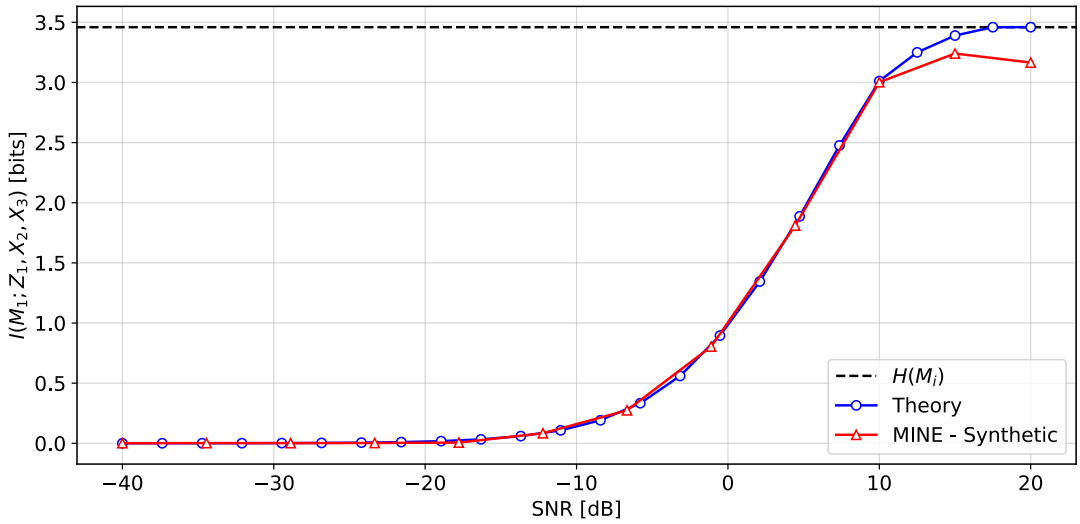
establish a ground truth to compare the results from MINE against. We will use the results from the previous section to evaluate the performance of MINE in the context of absolute security.

MINE on Synthetic Data

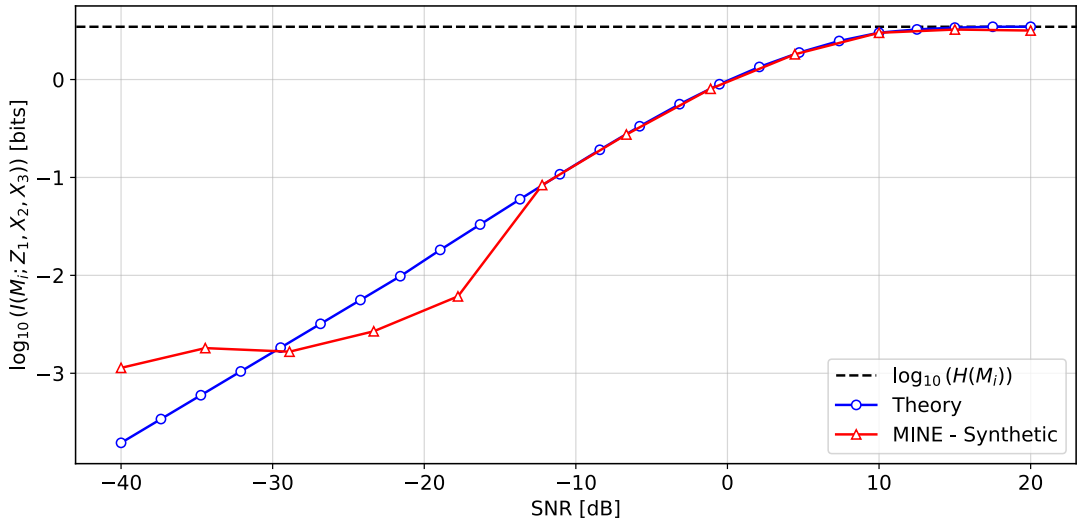
We will now use MINE to estimate the mutual information between the original messages and the eavesdropper's observations. We will compare the results from MINE against the theoretical values computed in the previous section with the same particular scenario.

To that end, we created a synthetic dataset of eavesdropper's observation samples by generating valid triplets of messages (M_1, M_2, M_3) mapping them onto the constellation and adding noise on the first channel. We made the noise variance vary to create samples with different SNRs.

The results are shown in Figure 3.5.



(a) Natural Scale



(b) Logarithmic Scale

Figure 3.5: Comparison between theory and MINE on synthetic data. MINE Parameters: $N_{Layers} = 3$, $N_{Nodes} = 128$, $N_{Samples} = 10^6$, $N_{Epochs} = 100$.

We see that the estimations of MINE are consistent with the theoretical values, at least for a given SNR range (approximately $[-12$ dB; 10 dB$]$). The results are however less accurate for lower SNRs. Indeed, we observe that MINE reaches a plateau at some point and is unable to converge to 0. This was expected as for low SNRs, the mutual information is closer and closer to 0 and the neural network is

unable to capture the small variations in the mutual information. This limitation is inherent to the method and is not a problem with the implementation. What is interesting to note is that the plateau coincides with the estimations of MINE when the input are truly independent, i.e. when one input is pure noise, or equivalently that $\text{SNR}_1 = -\infty$ dB. Figure 3.6 illustrates this phenomenon. We also note that fo very high values of SNR (e.g. $\text{SNR}_1 = 20$ dB), MINE underestimate more significantly the mutual information. This is not really an issue since we are not interested in this regime as the mutual information is close to the theoretical value of $h(M_i) = \log_2(11)$. Indeed, this would be more a problem of discrete RV entropy estimation which is rather simple to compute if we know the distribution of the RV (uniform distribution in our case).

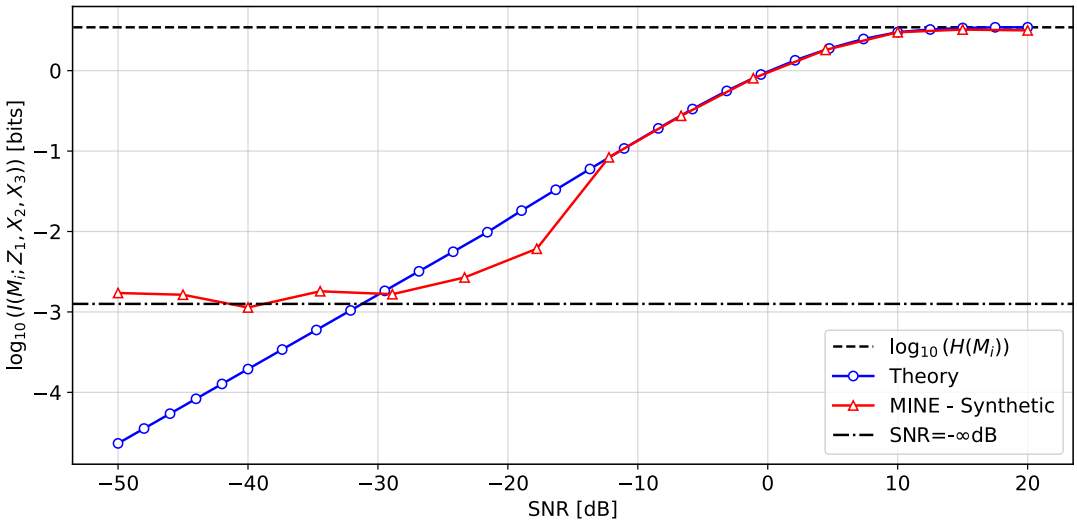


Figure 3.6: MINE converging to a plateau corresponding to the estimation of MINE when pure noise is given as input, i.e. $\text{SNR}_1 = -\infty$ dB. MINE Parameters: $N_{Layers} = 3$, $N_{Nodes} = 128$, $N_{Samples} = 10^6$, $N_{Epochs} = 100$.

However, these inaccuracies can be mitigated by increasing the capacities (e.g. Size of the network) or ressources (e.g. Number of samples) of the neural network. This tradeoff between accuracy and computational cost must be weighted depending on the application and is left to the user's discretion. Figure 3.7 illustrates this tradeoff with respect to the complexity of the network. For ease of visualization, we will now only consider the logarithmic scale for the rest of the study as it is more informative in the low SNR regime.

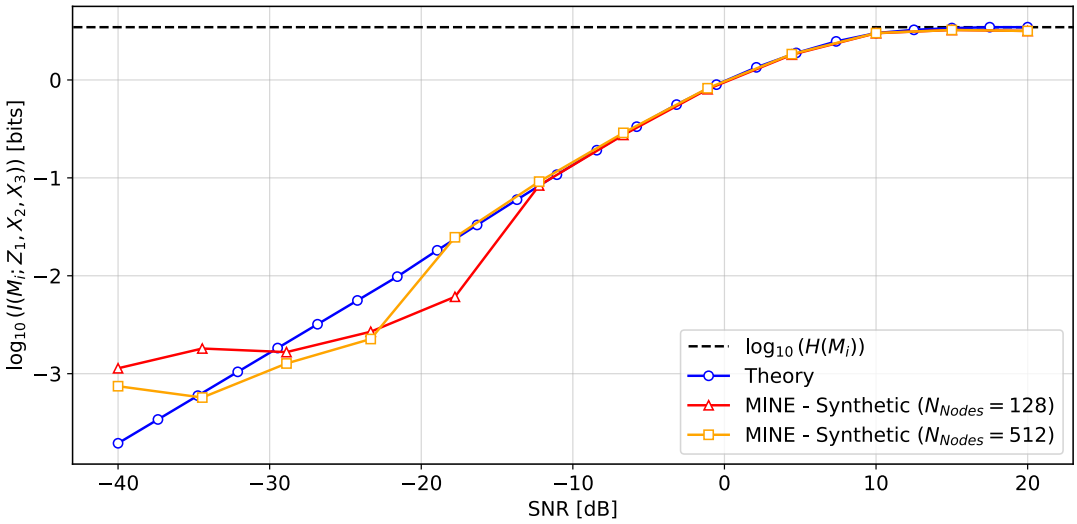


Figure 3.7: Tradeoff between accuracy and computational cost in terms of the complexity of the network. MINE Parameters: N_{Layers}, $N_{Samples} = 10^6$, $N_{Epochs} = 100$.

In Figure 3.7, we compare the estimations of MINE for two different architectures. We see that when we increase the expressivity of the network by increasing the width of the layers from 128 to 512, the estimations of MINE become more accurate. Indeed, we see that the estimations fit with the theoretical values for a larger range of SNRs and that the plateau is reached at lower SNRs and has a smaller value. This is consistent with the fact that a more expressive network is able to capture more complex relationships in the data and thus is able to estimate the mutual information more accurately. This gain in accuracy comes at the cost of computational resources as the network is more complex, it is thus left at the user's discretion to choose the architecture that fits the best its application requirements.

MINE on Experimental Data

To go further in the analysis of our scenario, we wanted to compare our current results against estimations on real world data. We thus created an experimental setup in a laboratory to emulate the communication scheme described in [2].

In order to simulate the 3 frequency channels, we decided to use an OFDM scheme in which the 3 messages are sent on 3 different subcarriers. The system we used has 64 subcarriers available, 3 of which are used to send the messages at position 1, 30 and 62, the other subcarriers are put to 0. The DC subcarrier is at position 32. The carrier frequency is set to 130GHz and the bandwidth is 20GHz,

we thus occupy the frequency range $[120, 140]$ GHz. The subcarrier spacing is thus $\Delta f = 312.5$ MHz. We have a symbol rate of 10 GHz and a sampling frequency of 90 GHz.

To create the dataset, we considered every possible combination of messages $(M_1, M_2, M_3) \in \mathbb{F}_{11}^3$ to obtain the symbols $(Y_1, Y_2, Y_3) \in \mathcal{C}^3$ to send on the subcarriers. Then, to simulate the degradation of the channel due to the antenna gain in the eavesdropper's direction, we artificially added attenuation on the third subcarrier. The attenuation is controlled by an attenuation factor α that we made vary to create samples with different SNRs. Noise is naturally present in the communication system and is added to the received symbols. It does not seem to be white over the considered bandwidth, rather it seems to be frequency dependant. Figure 3.8 shows the power of the received symbols over the subcarriers.

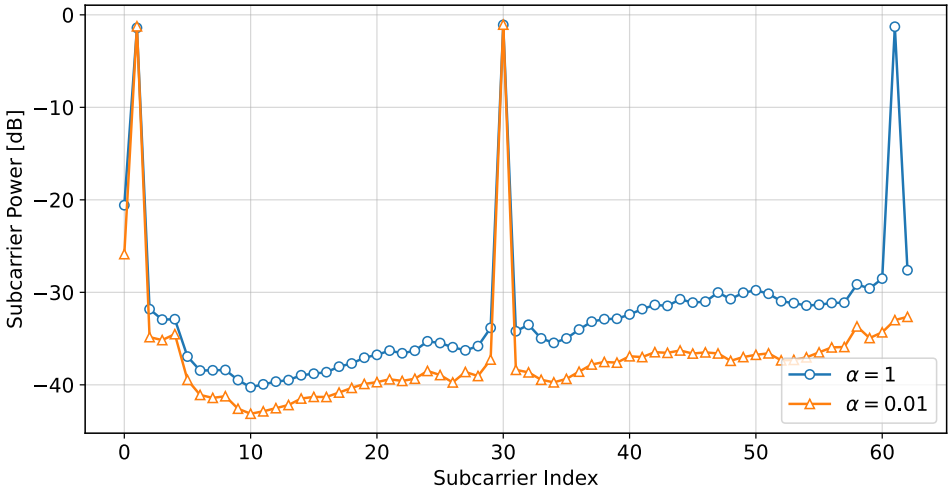


Figure 3.8: Power of the Received Symbols over the Subcarriers.

We see however that the noise can be approximated to be locally flat around the subcarriers of interest. We thus used the average noise estimated on the neighbor subcarriers of the ones of interest to compute the SNRs.

We then used MINE to estimate the mutual information between the original messages and the eavesdropper's observations, i.e. here the receiver antenna. Figure 3.9 summarizes the experimental setup communication scheme.

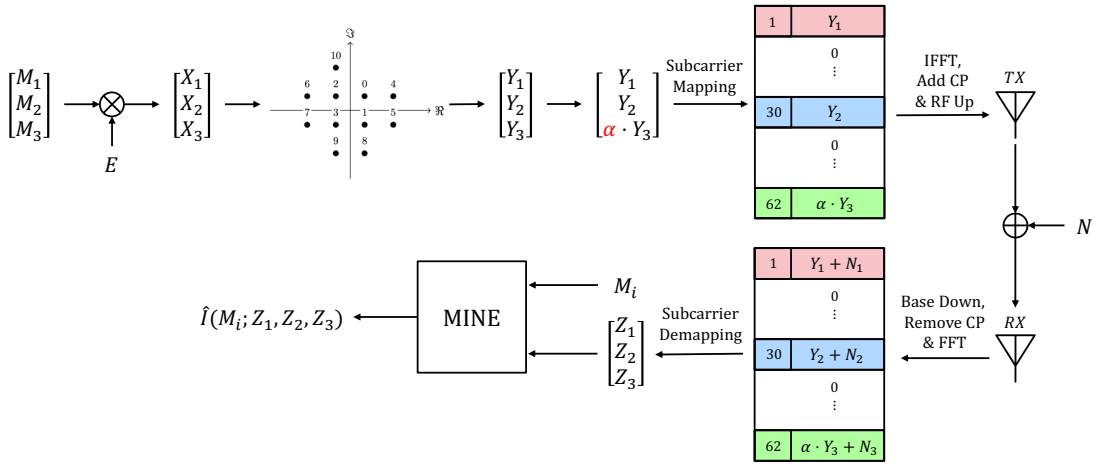


Figure 3.9: Experimental Setup for the Communication System. IFFT: Inverse Fast Fourier Transform, FFT: Fast Fourier Transform, CP: Cyclic Prefix, RF Up: RF Upconversion, Base Down: Baseband Downconversion.

In Figure 3.10, we see that the experimental results are consistent both with the theoretical values and the estimations on synthetic data, for about the same range of SNR as the synthetic data (approximately $[-15 \text{ dB}; 20 \text{ dB}]$). We also observe the plateau phenomenon at low SNRs. It is however interesting to note that the plateau has a higher value than the one obtained with synthetic data. This can be explained because, contrary to the synthetic data, the experimental data also present some non-zero level of noise on the supposedly perfect channels. This noise will be captured by MINE and will thus worsen its estimation capacity. The difference in the plateau values is thus due to the fact that the experimental data is more complex than the synthetic data and that MINE is unable to capture the small variations in the mutual information due to the noise.

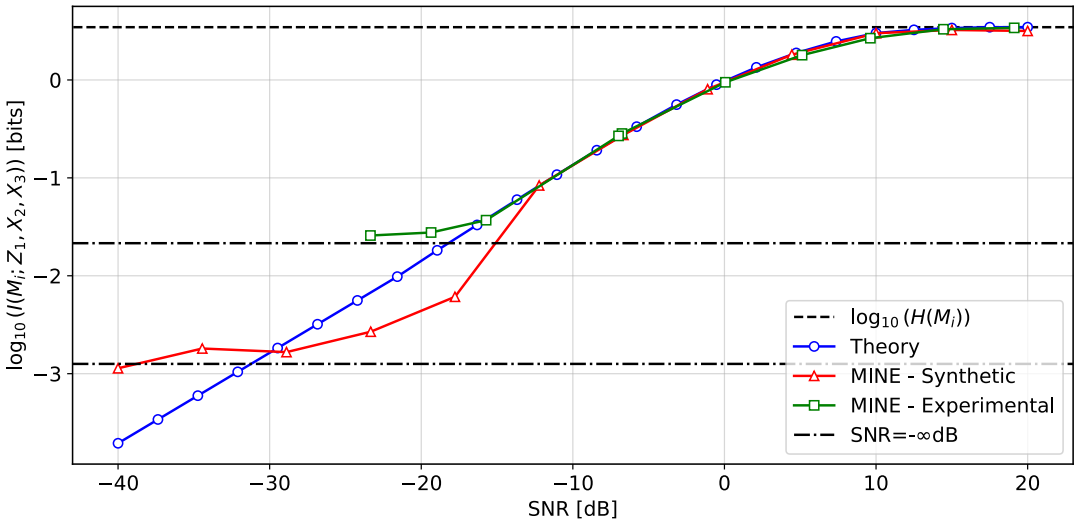


Figure 3.10: Comparison between Theory, MINE on Synthetic Data and MINE on Experimental Data. MINE Parameters: $N_{Layers} = 3$, $N_{Nodes} = 128$, $N_{Samples} = 10^6$, $N_{Epochs} = 100$.

Security Evaluation Procedure

Now that we studied the performance of MINE in the context of absolute security, we propose a scheme to evaluate the security of the communication system. This procedure give us a practical way to assert if the communication system is secure or if information could be leaked to an eavesdropper in some locations.

The strategy we consider is based on the limitation of MINE to estimate the mutual information for low SNRs. The argument is founded on the indinstinguishability between very low SNR scenarios and the case where the eavesdropper receives no information at all. Indeed, we saw that at low SNR the estimations of MINE converge to a plateau that actually coincides with the mutual information MINE would estimate if the true mutual information was 0. It is important to understand that this argument is different from the absolute security argument presented in [2], yet provides a practical way to evaluate the security of the communication system based on the optimized algorithm we developed in Chapter 2.

The approach we suggest is to make use of statistical hypothesis tests in order to assert if the mutual information estimated by MINE is significantly different from 0. More precisely, in statistical terms, we have samples coming from 2 populations: estimations of MINE in a low SNR scenario and estimations when the eavesdropper truly receives no information. We can then use a statistical test to verify if the

samples come from the same population or not. If we cannot claim that the samples come from different populations, we can conclude that the eavesdropper will not be able to distinguish between the two scenarios and thus that the communication system is "empirically" secure.

By "empirically", we mean that using the state-of-the-art tools we developed in this work, it is not possible to distinguish between two situations where one is absolutely secure and where one is not. Again, the absolute security claim is different and is stronger in the sense that if the needed constraints are fulfilled the mutual information is guaranteed to be null. On the other hand, our claims are more practically/numerically founded and are based on the current limitations of the SoTA MI estimators. It should be however possible to reconcile both claims, at least theoretically. Indeed, if we manage to build an estimator capable of estimating reliably down to the SNR value corresponding to limits required in [2], then we would be able to tell for sure if the system is absolutely secure or not.

The statistical test we want to use is the Welch's t-test [45]. This test is used to compare the means of two independent samples when the variances are not assumed to be equal, which is an assumption made in the renowned Student's t-test [46]. The Welch's t-test is thus more general than the Student's t-test and is actually more adequate in our case since we do not know if the variances of the two populations are equal and we do not necessarily want to make this assumption. Welch's t-test is described in Algorithm 3.1.

Algorithm 3.1: Welch's t-test

Given two set of samples $X_1 = \{x_1^{(1)}, x_2^{(1)}, \cdots, x_{n_1}^{(1)}\}$ and $X_2 = \{x_1^{(2)}, x_2^{(2)}, \cdots, x_{n_2}^{(2)}\}$, we want to test the null hypothesis that the means of the two populations are equal against the alternative hypothesis that the means are different, i.e.

$$\begin{cases} H_0 : \mu_1 = \mu_2 \\ H_1 : \mu_1 \neq \mu_2 \end{cases}. \tag{3.19}$$

Assumptions:

- The samples are independent.
- The samples are normally distributed.

Testing Procedure:

1. Compute the means $\bar{x}_1$ and $\bar{x}_2$ of the two populations.
2. Compute the corrected sample variances s_1^2 and s_2^2 of the two samples. Where $s_i^2 = \frac{\sum_j (x_j - \bar{x}_i)^2}{n_i - 1}$.
3. Compute the degrees of freedom ν as follows:

$$\nu = \frac{(s_1^2/n_1 + s_2^2/n_2)^2}{\frac{(s_1^2/n_1)^2}{n_1 - 1} + \frac{(s_2^2/n_2)^2}{n_2 - 1}}. \tag{3.20}$$

4. Compute the t-statistic as follows:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{s_1^2/n_1 + s_2^2/n_2}}. \tag{3.21}$$

5. Compute the p-value using the Student t-distribution with ν degrees of freedom. Where $p = P(T > |t|)$ (two-sided test) or $p = P(T > t)$ (one-sided test) and $T \sim t_\nu$.
6. Reject the null hypothesis if the p-value is less than the significance level α.

As a reminder, the p-value [47] is the probability of observing a test statistic at least as extreme as the one computed from the sample data, assuming that

H_0 is true. The significance level α is the value that we set to determine if the p-value is low enough to reject the null hypothesis. It is also the probability of rejecting the null hypothesis when it is true, which is called a Type I error. The lower the p-value is, the more confident we are in rejecting the null hypothesis. Usual values for the significance level are $\alpha = 0.05$ or $\alpha = 0.01$. In our application however, we give the user the choice to set freely the significance level as this value is a proxy of the empirical security. Indeed, the surer we are when rejecting the null hypothesis, i.e. that the eavesdropper cannot distinguish from 0 MI, the more secure our assumption of security is.

In order to make sure that the Welch's t-test is well suited for our application, we need to verify that the samples are indeed normally distributed. We can do so by using the Shapiro-Wilk test [48]. This test is used to test if samples come from a normally distributed population. The Shapiro-Wilk test is described in Algorithm 3.2. It is important to keep in mind that the Shapiro-Wilk test is put here as a verification tool against extreme scenarios but that the Welch's t-test is still robust to small violations of the normality assumption.

Algorithm 3.2: Shapiro-Wilk Test

Given a set of samples $X = \{x_1, x_2, \cdots, x_n\}$, we want to test the null hypothesis that the samples come from a normally distributed population against the alternative hypothesis that the samples do not, i.e.

$$\begin{cases} H_0 : X \sim \mathcal{N} \\ H_1 : X \nsim \mathcal{N} \end{cases} . \tag{3.22}$$

Assumptions:

- The samples are independent.

Testing Procedure:

1. Compute the sample mean $\bar{x}$.
2. Compute the test statistic W as follows:

$$W = \frac{\left(\sum_{i=1}^{n} a_i x_{(i)}\right)^2}{\sum_{i=1}^{n}(x_i - \bar{x})^2}, \tag{3.23}$$

   where:

   - $x_{(i)}$ is the i^{th} ordered statistics, i.e. i^{th} smallest sample value.

   - a_i are the coefficients of the Shapiro-Wilk test and are given by $[a_1, \cdots, a_n] = \frac{m^T V^{-1}}{(m^T V^{-2} m)^{1/2}}$.

   - m is the vector of the expected values of the order statistics of a sample of size n from a normal distribution.

   - V is the covariance matrix of the order statistics of a sample of size n from a normal distribution.

3. Compute the p-value using the Shapiro-Wilk distribution. Where $p = P(W > w)$ and $W \sim$ Shapiro-Wilk.
4. Reject the null hypothesis if the p-value is less than the significance level α.

We can now combine the two former tests in one procedure to evaluate the security of the communication system. The procedure is described in Algorithm 3.3.

Algorithm 3.3: Security Evaluation Procedure

Input: Set of estimation samples of MINE when given independent inputs $I_0 = \{I_0^{(1)}, \cdots, I_0^{(N_{Est})}\}$, Set of samples of eavesdropper's observations $Z = \{\boldsymbol{z}^{(1)}, \cdots, \boldsymbol{z}^{(N_{Samples})}\}$.

Output: Security of the communication system.

Parameters : Number of MINE estimations N_{Est}, Significance level α.

Initialization: $I \leftarrow zeros(N_{Est})$

for *iteration in range*(N_{Est}) **do**
| $I[iteration] \leftarrow \text{MINE}(Z)$
end

1. Test the normality of the samples using the Shapiro-Wilk test 3.2.
2. If the samples are normally distributed, compute the Welch's t-test 3.1 to compare the means of the two populations I and I_0.
3. Reject the null hypothesis if the p-value is less than the significance level α.
4. If the null hypothesis is rejected, i.e. $p < \alpha$, conclude that the communication system is not secure.

In Figure 3.11, are presented the histograms of the estimations of MINE for SNRs=$\{-\infty$ dB, -40 dB, -20 dB$\}$. We clearly see that the distribution of estimation at -40 dB is very close to the distribution when the inputs are independent (i.e. SNR=$-\infty$ dB). On the other hand, the distribution at -20 dB is significantly different from the other two. However, we note that the normality assumption might not be respected especially at low SNRs. This is due to the fact that the mutual information is close to 0 and the distribution of the estimations is not necessarily normal. It seems to be more skewed and should remain always positive in theory, which is not the case for normal distributions. Table 3.1 summarizes the results of the tests for the SNRs considered. We see indeed that the p-values obtained for the Shapiro-Wilk test are relatively low especially for SNR=-40 dB, implying that the samples are likely non-normal. The Welch's t-test however gives us a clear answer on the security of the communication system, we see that the system is secure for SNR=-40 dB and definitely not secure for SNR=-20 dB.

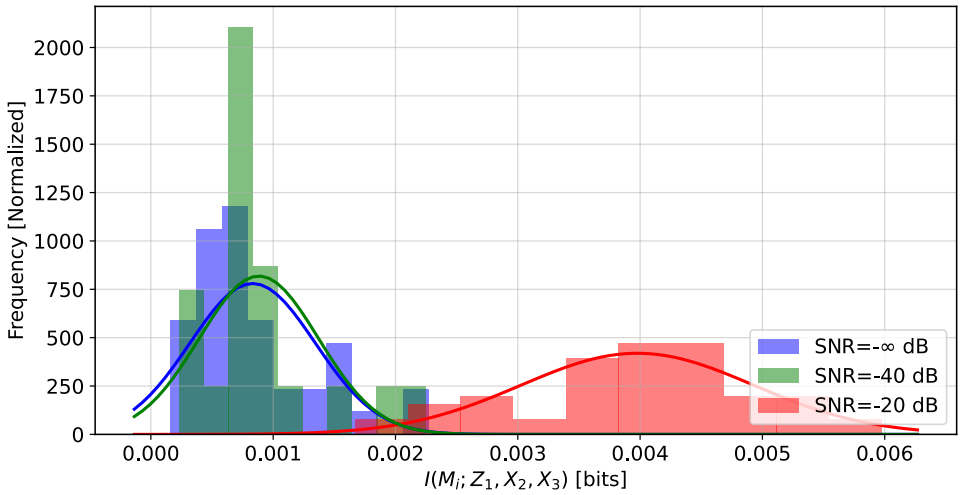


Figure 3.11: Histogram of the estimations in natural scale for SNR=$\{-\infty$ dB, -40 dB, -20 dB$\}$. Are also plotted the normal distributions fitted to the samples.

	Shapiro-Wilk p-value	**Welch's t p-value**	**Secure**
SNR=-∞ dB	0.003	-	-
SNR=-40 dB	$7.39 \cdot 10^{-5}$	0.65	Yes
SNR=-20 dB	0.003	$3.65 \cdot 10^{-37}$	No

Table 3.1: Security Evaluation Results, Natural Scale, $N_{Est} = 40$. MINE Parameters: $N_{Layers} = 3$, $N_{Nodes} = 128$, $N_{Samples} = 10^6$, $N_{Epochs} = 500$.

A possible solution to the normality issue is to consider the logarithm of the mutual information. The log-transform [49] is a common technique used to reduce the skewness of a distribution and to make it more symmetric so that the Welch's t-test can be applied more reliably. In Figure 3.12, we see that the distribution of the log-transformed estimations is more symmetric and that the normality assumption is closer to be respected. Table 3.2 summarizes the results of the tests for the log-transformed estimations. We see that the normality assumption is better respected looking at the p-values for the Shapiro-Wilk test, and that the Welch's t-test gives us the same results as for the natural scale.

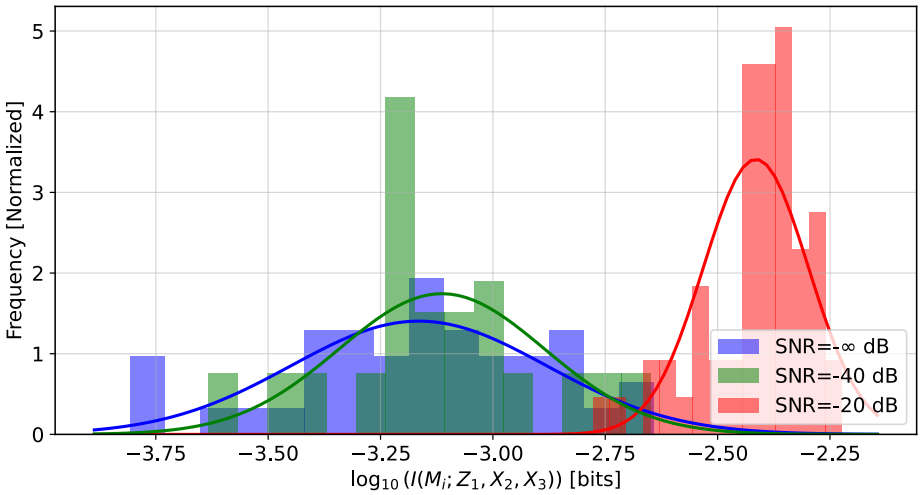


Figure 3.12: Histogram of the estimations in logarithmic scale for SNR=$\{-\infty$ dB, -40 dB, -20 dB$\}$. Are also plotted the normal distributions fitted to the samples.

	Shapiro-Wilk p-value	**Welch's t p-value**	**Secure**
SNR=-∞ dB	0.416	-	-
SNR=-40 dB	0.1	0.38	Yes
SNR=-20 dB	0.0016	$1.91 \cdot 10^{-20}$	No

Table 3.2: Security Evaluation Results, Log Scale, $N_{Est} = 40$. MINE Parameters: $N_{Layers} = 3$, $N_{Nodes} = 128$, $N_{Samples} = 10^6$, $N_{Epochs} = 500$.

Even though in this example, both scales give the same results in terms of security evaluation, it might not be the case for all scenarios, especially in limit cases where the differentiation between the two populations is very small. The log-transform is thus a useful tool to make the Welch's t-test more reliable. Other more complex transformations such as the Box-Cox transform (or Power transform) [50] could also be used to make the distribution more normal.

Chapter Conclusion

In this chapter, we studied the performance of MINE in the context of absolute security. We derived theoretical expressions for the mutual information between the original messages and the eavesdropper's observations and compared the results from MINE against these theoretical values. We showed that its estimations are consistent with the theoretical values and that the method is able to capture the behavior of the mutual information in the context of absolute security within a given range of SNR. We also saw that at low SNR, the estimations tended toward a plateau that corresponded to the value MINE would estimate for completely independent inputs.

Based on that observation, we proposed a strategy to evaluate the security of the communication system. Assuming that estimations in the same setup would be normally distributed, we showed that the Welch's t-test could be a suitable statistical test to evaluate the security of the communication system. We illustrated the procedure on a simple scenario to show that the procedure was able to differentiate between a secure and not secure case. We also showed that the log-transform could be used to make the distribution more normal and thus make the Welch's t-test more reliable.

Future Works and Perspectives

In the future, we would like to extend the study to more complex scenarios. Indeed, the current study was limited to a simple scenario with 3 channels. In practical systems however, the number of channels can be much higher and we need to study the scalability of MINE to such scenarios. Especially as the number of noisy channels grows. Even though, in theory, we know we can upper bound the information leakage using only the noisiest channel, if we are limited to real samples a non-negligible amount of noise will be present on the other channels and we will need to take this into account.

It could also be interesting to study more complex channel models such as Rayleigh or Rician fading channels. These models are more realistic than the AWGN channel and would allow us to study the performance of MINE in more realistic scenarios.

Conclusion

In this work, we focused on the estimation of mutual information and especially on a new estimator called MINE based on neural networks. This estimator was then used to evaluate the mutual information between the original messages and the eavesdropper's observations in the context of absolute security in wireless communications.

In Chapter 1 we first introduced the concept of mutual information and its importance in various fields. We then presented some estimators including MINE and the theory behind it.

In Chapter 2, we characterized the performance of MINE when varying some parameters such as the network's depth and width, batch size or learning rate. We noted that these parameters had various impacts on the performance of the estimator, in terms of accuracy, variance, stability and convergence speed. We then added some optimizations to the original MINE algorithm to make it more robust and efficient based on our observations during the characterization. The *Early Stopping* was implemented to avoid overfitting and to stop the training when the estimator starts to converge in order to save time and ressources. The *Automatic LR Tuner* enables us to find the optimal learning rate for the training, i.e. the rate leading to the fastest and most accurate estimation. It also allows the non-experienced user to use the estimator as a black box without needing to fine tune the learning rate manually. The *Learning Rate Scheduler* was implemented to stabilize the training and avoid the estimator to diverge as we observed during the characterization.

Finally, in Chapter 3 we developed a theoretical framework to evaluate the mutual information between the original messages and the eavesdropper's observations

in the context of absolute security. We then used MINE to estimate the mutual information based on synthetic data we generated and compared the results to the theoretical expressions we derived. We also compared these results against experimental data obtained in a laboratory setup. We observed that the estimator was able to capture the general trend of the mutual information and that the results were consistent with theory for a range of SNR. For low SNRs however, we observed that MINE reached a plateau and was not able to estimate beyond a certain value that corresponded with the estimation when given truly independant data as inputs. We used the former limitation of this State-of-The-Art estimator to define empirical security using an indistinguishability argument. To formalize this security claim, we defined a statistical test procedure based on Welch's t-test in order to verify if the eavesdropper's estimations can be distinguished from the null mutual information estimation of MINE.

Bibliography

[1] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R. Devon Hjelm. MINE: Mutual Information Neural Estimation, August 2021. arXiv:1801.04062 [cs, stat].

[2] Alejandro Cohen, Rafael G. L. D'Oliveira, Chia-Yi Yeh, Hichem Guerboukha, Rabi Shrestha, Zhaoji Fang, Edward Knightly, Muriel Médard, and Daniel M. Mittleman. Absolute Security in High-Frequency Wireless Links, August 2022. arXiv:2208.05907 [cs, math].

[3] ChatGPT: Openai's conversational AI model. `https://openai.com/chatgpt`.

[4] Github copilot. `https://copilot.github.com`.

[5] C E Shannon. A Mathematical Theory of Communication. 1948.

[6] Wikipedia contributors. Entropy (information theory) — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Entropy_(information_theory)&oldid=1226148324`, 2024. [Online; accessed 29-May-2024].

[7] Wikipedia contributors. Pearson correlation coefficient — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Pearson_correlation_coefficient&oldid=1225314034`, 2024. [Online; accessed 29-May-2024].

[8] Taiji Suzuki, Masashi Sugiyama, Jun Sese, and Takafumi Kanamori. Approximating Mutual Information by Maximum Likelihood Density Ratio Estimation. In *Proceedings of the Workshop on New Challenges for Feature Selection in*

Data Mining and Knowledge Discovery at ECML/PKDD 2008, pages 5–20. PMLR, September 2008. ISSN: 1938-7228.

[9] Shuyang Gao, Greg Ver Steeg, and Aram Galstyan. Estimating Mutual Information by Local Gaussian Approximation, February 2016. arXiv:1508.00536 [physics].

[10] Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical Review E*, 69(6):066138–066138, June 2004. ARXIV_ID: cond-mat/0305641 MAG ID: 2092939357 S2ID: 5729e9bc42a8c006fcdeaf6ca3434fabf49e7e7c.

[11] Thomas Benjamin Berrett and Caius College. Modern k-Nearest Neighbour Methods in Entropy Estimation, Independence Testing and Classification.

[12] Damiano Lombardi, Damiano Lombardi, and Sanjay Pant. Nonparametric k-nearest-neighbor entropy estimator. *Physical Review E*, 93(1):013310–013310, January 2016. ARXIV_ID: 1506.06501 MAG ID: 2206302789 S2ID: 0954ee594ae594c4bad59808ec4d596ab119ae3b.

[13] Alexander Marx and Jonas Fischer. Estimating Mutual Information via Geodesic kNN. In *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)*, Proceedings, pages 415–423. Society for Industrial and Applied Mathematics, January 2022.

[14] M.M. Van Hulle. Multivariate Edgeworth-Based Entropy Estimation. In *2005 IEEE Workshop on Machine Learning for Signal Processing*, pages 311–316, Mystic, CT, USA, 2005. IEEE.

[15] Sudipto Mukherjee, Himanshu Asnani, and Sreeram Kannan. CCMI : Classifier based Conditional Mutual Information Estimation, June 2019. arXiv:1906.01824 [cs, math, stat].

[16] XuanLong Nguyen, Martin J. Wainwright, and Michael I. Jordan. Estimating divergence functionals and the likelihood ratio by convex risk minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861, November 2010. arXiv:0809.0853 [cs, math, stat].

[17] Scikit-learn: Mutual Information Estimator. `https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_regression.html`.

[18] R. J. Serfling. *Approximation Theorems of Mathematical Statistics*. Wiley, New York, 1980.

[19] M. D. Donsker and S. R. S. Varadhan. Asymptotic evaluation of certain markov process expectations for large time. iv, 1983.

[20] Sebastian Nowozin, Botond Cseke, and Ryota Tomioka. f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization, June 2016. arXiv:1606.00709 [cs, stat].

[21] Avraham Ruderman, Mark D Reid, Darío García-García, and James Petterson. Tighter Variational Representations of f -Divergences via Restriction to Probability Measures.

[22] Kwanghee Choi and Siyeong Lee. Regularized Mutual Information Neural Estimation. 2021.

[23] Homa Esfahanizadeh, William Wu, Manya Ghobadi, Regina Barzilay, and Muriel Médard. InfoShape: Task-Based Neural Data Shaping via Mutual Information. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, Rhodes Island, Greece, June 2023. IEEE.

[24] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[25] François Chollet. *Deep Learning with Python*. Manning Publications, 2018.

[26] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[27] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[28] Neural networks — Tikz.net. `https://tikz.net/neural_networks`. [Online; accessed 2024-05-19].

[29] Wikipedia contributors. Mean squared error — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Mean_squared_error&oldid=1226060918`, 2024. [Online; accessed 29-May-2024].

[30] Wikipedia contributors. Cross-entropy — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Cross-entropy&oldid=1223539435`, 2024. [Online; accessed 29-May-2024].

[31] Wikipedia contributors. Stochastic gradient descent — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Stochastic_gradient_descent&oldid=1222784950`, 2024. [Online; accessed 29-May-2024].

[32] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

[33] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2017.

[34] Wikipedia contributors. Chain rule — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Chain_rule&oldid=1222545557`, 2024. [Online; accessed 29-May-2024].

[35] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[36] William Falcon and The PyTorch Lightning team. Pytorch lightning. `https://github.com/PyTorchLightning/pytorch-lightning`.

[37] Wikipedia contributors. Stochastic gradient descent — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Stochastic_gradient_descent#RMSProp`, 2024. [Online; accessed 29-May-2024].

[38] Wikipedia contributors. Vanishing gradient problem — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Vanishing_gradient_problem&oldid=1222680571`, 2024. [Online; accessed 29-May-2024].

[39] Ben Poole, Sherjil Ozair, Aaron van den Oord, Alexander A. Alemi, and George Tucker. On Variational Bounds of Mutual Information, May 2019. arXiv:1905.06922 [cs, stat].

[40] PyTorch Learning Rate Schedulers — PyTorch 1.10.0 documentation. `https://pytorch.org/docs/stable/optim.html#module-torch.optim.lr_scheduler`.

[41] Jeremiah Birrell, Markos A. Katsoulakis, and Yannis Pantazis. Optimizing Variational Representations of Divergences and Accelerating their Statistical Estimation, March 2022. arXiv:2006.08781 [cs, math, stat].

[42] Danilo Silva and Frank R. Kschischang. Universal weakly secure network coding. In *2009 IEEE Information Theory Workshop on Networking and Information Theory*, pages 281–285, 2009.

[43] Danilo Silva and Frank R. Kschischang. Universal secure network coding via rank-metric codes. *IEEE Transactions on Information Theory*, 57(2):1124–1135, 2011.

[44] Wikipedia contributors. Known-plaintext attack — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Known-plaintext_attack&oldid=1194222939`, 2024. [Online; accessed 29-May-2024].

[45] Wikipedia contributors. Welch's t-test — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Welch%27s_t-test&oldid=1225182216`, 2024. [Online; accessed 29-May-2024].

[46] Wikipedia contributors. Student's t-test — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Student%27s_t-test&oldid=1224247091`, 2024. [Online; accessed 29-May-2024].

[47] Wikipedia contributors. P-value — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=P-value&oldid=1223374181`, 2024. [Online; accessed 29-May-2024].

[48] Wikipedia contributors. Shapiro–wilk test — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Shapiro%E2%80%93Wilk_test&oldid=1218522924`, 2024. [Online; accessed 29-May-2024].

[49] Robert M West. Best practice in statistics: The use of log transformation. *Annals of Clinical Biochemistry*, 59(3):162–165, 2022.

[50] Wikipedia contributors. Power transform — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/w/index.php?title=Power_transform&oldid=1217530637`, 2024. [Online; accessed 29-May-2024].

Appendix A

A.1 Transfer Learning

Transfer learning is a technique in machine learning that consists in using a model trained on a task to solve another task. This is very useful when we do not have enough data to train a model from scratch. We can use a pre-trained model and fine-tune it on our data.

We thought that this technique could be very useful in the context of MINE. Indeed, we could use a pre-trained model on a similar dataset and fine-tune it on our data. This kind of approach could make sense especially in applications based on real data where we do not have enough data to train a model from scratch and for which we have a prior idea of the structure of the data. In a communication system application, for example, we could have an a priori knowledge of the SNR regime of the data and use a pre-trained model on a similar SNR regime to then fine-tune it on the real data measured afterwards.

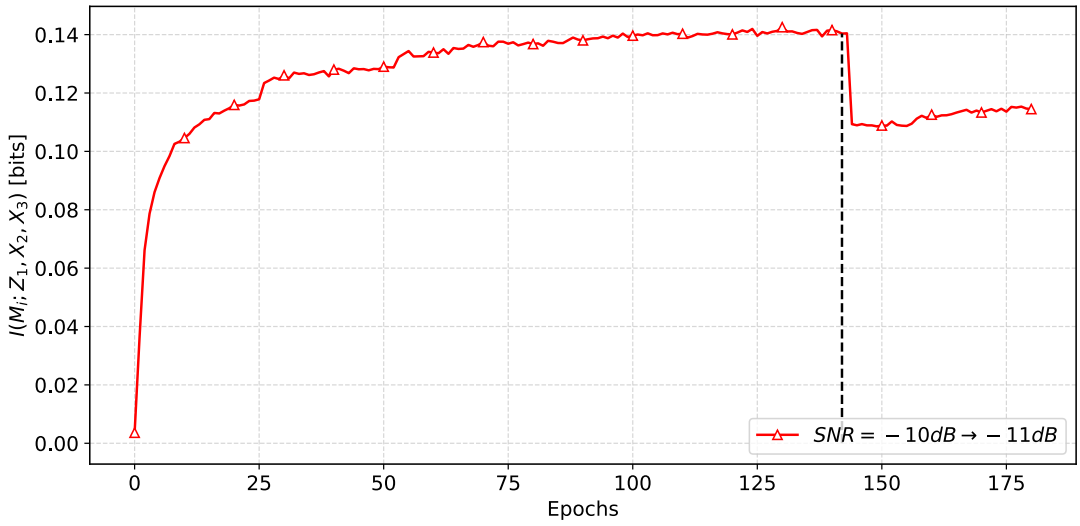


Figure A.1: Transfer Learning Experiment. We train a network on a dataset with a SNR of -10dB until convergence. We then use this pre-trained network to fine-tune it on a dataset with a SNR of -11dB. We see that the fine-tuning process allows us to converge to another value of MI.

Figure A.1 shows the results of a simple experiment we conducted to give a proof of concept of the potential of Transfer Learning in the context of MINE. We used the dataset from Chapter 3 (please refer to this chapter for further details on the dataset) and trained a network on a dataset with a SNR of -10dB until convergence. We then used this pre-trained network to fine-tune it on a dataset with a SNR of -11dB. We see that the fine-tuning process allows us to converge to another value of MI.

Further experiments and optimizations could be conducted to fully explore the potential of Transfer Learning in the context of MINE. Especially to test if it truly allows to converge faster and to a better value of MI than training from scratch.

A.2 Improved DV Bound

In [41], the authors present an extensive work on variational representations of divergences (such as the Donsker-Varadhan representation) and a systematic method to tighten them by using a family of auxiliary functions. They also show that these tighter representation can lead to faster and more accurate estimation of divergences.

The main theorem presented in the paper goes as follows:

Proposition A.1

Let $f \in \{f : (a,b) \to \mathbb{R} \text{ with } f(1) = 0\}$ be a convex function, and suppose that the f-divergence $D_f(\mathbb{P}||\mathbb{Q}) = \mathbb{E}_P(f(d\mathbb{P}/d\mathbb{Q})) < \infty$ for some probability measures $\mathbb{P}$ and $\mathbb{Q}$ on (a,b). We can also use the variational representation of the f-divergence $D_f(\mathbb{P}||\mathbb{Q}) = \sup_{\phi \in \mathcal{M}_b} \mathbb{E}_{\mathbb{P}}[\phi] - \mathbb{E}_{\mathbb{Q}}[f^*(\phi)]$, where $\mathcal{M}_b$ denotes the set of all bounded measurable functions and f^* is the Legendre transform of f. The supremum is reached for $\phi^* = f'(d\mathbb{P}/d\mathbb{Q})$.

Let also $\Phi \subset L^1(Q)$ be a family of test functions such that $\phi^* \in \Phi$, where $\phi^*(x) = f'(d\mathbb{P}/d\mathbb{Q}(x))$.

Consider any family of transformations $\mathcal{T} \subset \{T(\phi)|T : \Phi \to L^1(\mathbb{P})\}$, that includes the identity transformation. Then,

$$D_f(\mathbb{P}||\mathbb{Q}) = \sup_{\phi \in \Phi} H_{\mathcal{T},f}[\phi] \tag{A.1}$$

$$\text{With, } H_{\mathcal{T},f}[\phi] = \sup_{T \in \mathcal{T}} \mathbb{E}_{\mathbb{P}}[T(\phi)] - \mathbb{E}_{\mathbb{Q}}[f^*(T(\phi))] \tag{A.2}$$

The supremum is reached for $\phi^* = f'(d\mathbb{P}/d\mathbb{Q})$. And, the function $H_{\mathcal{T},f}$ is tighter than the variational representation of the f-divergence, i.e.

$$\mathbb{E}_{\mathbb{P}}[\phi] - \mathbb{E}_{\mathbb{Q}}[f^*(\phi)] \geq H_{\mathcal{T},f}[\phi] \geq D_f(\mathbb{P}||\mathbb{Q}), \quad \forall \phi \in \Phi \tag{A.3}$$

The families of transformations $\mathcal{T}$ and test functions Φ can be chosen in many ways. The authors present a few examples in the paper: Shifting, Scaling, Affine and Power transformations.

The Donsker-Varadhan representation is a special case of this theorem with $\mathcal{T}$ chosen to be the set of shifting transformations. In [21], it has been shown that the DV representation is indeed tighter than the f-divergence variational representation of the KL-divergence (with $f(x) = x\log(x)$).

Using this theorem, the authors derived a tighter representation of the KL-divergence than the DV representation, they call this representation *Improved DV representation*. They derived this result by extending the family of transformations to the affine transformations instead of the shifting transformations (by introducing a scaling factor η on ϕ). Starting from the DV representation, and adding the scaling factor η, we get the following representation of the KL-divergence:

$$D_{KL}(\mathbb{P}||\mathbb{Q}) = \sup_{\phi \in \Phi} H_{\mathcal{T},f}[\phi] = \sup_{\phi \in \Phi} \sup_{\eta \in \mathbb{R}} \eta \mathbb{E}_{\mathbb{P}}[T(\phi)] - \log(\mathbb{E}_{\mathbb{Q}}[e^{\eta\phi}]) \tag{A.4}$$

Which is necessarily tighter than the DV representation since $\mathcal{T}_{Shift} \subset \mathcal{T}_{Affine}$.

The supremum over η in Eq. A.4 cannot be computed analytically in general. We can however obtain an analytical approximation that is still tighter than the DV representation. The approximation is obtained as such: Define $G_\phi(\eta) = \eta\mathbb{E}_\mathbb{P}[T(\phi)] - \log(\mathbb{E}_\mathbb{Q}[e^{\eta\phi}])$ and compute the Taylor series expansion around $\eta = 1$ (i.e. the DV representation). We get:

$$\begin{aligned} G_\phi(1+\Delta\eta) =& \mathbb{E}_\mathbb{P}[\phi] - \log(\mathbb{E}_\mathbb{Q}[e^{\eta\phi}]) + (\mathbb{E}_\mathbb{P}[\phi] - \mathbb{E}_{\mathbb{Q}_\phi}[\phi])\Delta\eta \\ & - \frac{Var_{\mathbb{Q}_\phi}[\phi]}{2}\Delta\eta^2 + \mathcal{O}(\Delta\eta^3) \end{aligned} \tag{A.5}$$

And since we want to maximize $G_\phi(\eta)$, we maximize A.5 with respect to $\Delta\eta$. We get:

$$\Delta\eta^*(\phi) = \frac{\mathbb{E}_\mathbb{P}[\phi] - \mathbb{E}_{\mathbb{Q}_\phi}[\phi]}{Var_{\mathbb{Q}_\phi}[\phi]} \tag{A.6}$$

Where $d\mathbb{Q}_\phi = \frac{e^\phi}{\mathbb{E}_P[e^\phi]}d\mathbb{Q}$ is the tilted measure. We can then use this approximation to compute the Improved DV representation of the KL-divergence from A.4:

$$D_{KL}(\mathbb{P}||\mathbb{Q}) = \sup_{\phi\in\Phi}(1+\Delta\eta^*(\phi))\mathbb{E}_\mathbb{P}[\phi] - \log(\mathbb{E}_\mathbb{Q}[e^{(1+\Delta\eta^*(\phi))\phi}]) \tag{A.7}$$

This representation is only guaranteed to be tighter when $\Delta\eta^*$ is sufficiently small.

Let us now present the algorithmic implementation of A.7 in the context of MINE.

The main challenge lays in the computation of $\Delta\eta^*$. Especially in computing the expectation and variance of ϕ under the tilted measure $\mathbb{Q}_\phi$, since we only have samples from $\mathbb{P}$ and $\mathbb{Q}$. We can however use the samples to compute the empirical expectation and variance of ϕ under $\mathbb{Q}_\phi$ in the following way:

$$\begin{aligned} \mathbb{E}_{\mathbb{Q}_\phi}[\phi] &= \int \phi d\mathbb{Q}_\phi = \int \phi\frac{e^\phi}{\mathbb{E}_\mathbb{Q}[e^\phi]}d\mathbb{Q} = \mathbb{E}_\mathbb{Q}[\phi\frac{e^\phi}{\mathbb{E}_\mathbb{Q}[e^\phi]}] \\ &\approx \frac{1}{N}\sum_{i=1}^N \phi_i^{(\mathbb{Q})}\frac{e^{\phi_i^{(\mathbb{Q})}}}{\frac{1}{N}\sum_{j=1}^N e^{\phi_j^{(\mathbb{Q})}}} = \frac{1}{\sum_{j=1}^N e^{\phi_j^{(\mathbb{Q})}}}\sum_{i=1}^N \phi_i^{(\mathbb{Q})}e^{\phi_i^{(\mathbb{Q})}} \end{aligned} \tag{A.8}$$

In the same way, we can compute the empirical variance of ϕ under $\mathbb{Q}_\phi$:

$$
\begin{aligned}
Var_{\mathbb{Q}_\phi}[\phi] &= \mathbb{E}_{\mathbb{Q}_\phi}[\phi^2] - \mathbb{E}_{\mathbb{Q}_\phi}[\phi]^2 \\
&\approx \frac{1}{\sum_{j=1}^{N} e^{\phi_j^{(\mathbb{Q})}}} \sum_{i=1}^{N} \phi_i^{2(\mathbb{Q})} e^{\phi_i^{(\mathbb{Q})}} - \left(\frac{1}{\sum_{j=1}^{N} e^{\phi_j^{(\mathbb{Q})}}} \sum_{i=1}^{N} \phi_i^{(\mathbb{Q})} e^{\phi_i^{(\mathbb{Q})}} \right)^2
\end{aligned} \tag{A.9}
$$

We can now obtain an empirical approximation for $\Delta\eta^*$:

$$
\Delta\eta^*(\phi) = \frac{\frac{1}{N}\sum_{i=1}^{N} \phi_i^{(\mathbb{P})} - \frac{1}{\sum_{j=1}^{N} e^{\phi_j^{(\mathbb{Q})}}} \sum_{i=1}^{N} \phi_i^{(\mathbb{Q})} e^{\phi_i^{(\mathbb{Q})}}}{\frac{1}{\sum_{j=1}^{N} e^{\phi_j^{(\mathbb{Q})}}} \sum_{i=1}^{N} \phi_i^{(\mathbb{Q})} e^{\phi_i^{(\mathbb{Q})}} - \left(\frac{1}{\sum_{j=1}^{N} e^{\phi_j^{(\mathbb{Q})}}} \sum_{i=1}^{N} \phi_i^{(\mathbb{Q})} e^{\phi_i^{(\mathbb{Q})}} \right)^2} \tag{A.10}
$$

$$
= \frac{(\sum_{j=1}^{N} e^{\phi_j^{(\mathbb{Q})}})^2 \sum_{i=1}^{N} \phi_i^{(\mathbb{Q})} - (\sum_{j=1}^{N} e^{\phi_j^{(\mathbb{Q})}}) \sum_{i=1}^{N} \phi_i^{(\mathbb{Q})} e^{\phi_i^{(\mathbb{Q})}}}{(\sum_{j=1}^{N} e^{\phi_j^{(\mathbb{Q})}}) \sum_{i=1}^{N} (\phi_i^{(P\mathbb{Q})})^2 e^{\phi_i^{(\mathbb{Q})}} - \left(\sum_{i=1}^{N} \phi_i^{(\mathbb{Q})} e^{\phi_i^{(\mathbb{Q})}}\right)^2} \tag{A.11}
$$

We can then use this empirical approximation to compute the Improved DV representation of the KL-divergence in MINE. The updated algorithm is presented in Algorithm A.1.

Algorithm A.1: Improved DV MINE Algorithm

Input: Samples $(x_i, y_i) \sim \mathbb{P}_{XY}$

Output: Optimal parameters θ^*

Parameters: Learning rate λ, Number of epochs N_{epochs}, Number of batches $N_{batches}$, Batch size B

Initialize the neural network with random weights θ

for *epoch in* $1, 2, ..., N_{epochs}$ **do**

  for *batch in* $1, 2, ..., N_{batches}$ **do**

    Permutation: Randomly permutate the y_i to artificially create samples $\{\bar{x}_i, \bar{y}_i\}_{i=1}^{B}$

    Compute $\Delta\eta^*$:

      Compute (1) $\leftarrow \sum_{i=1}^{B} e^{T(\bar{x}_i, \bar{y}_i)}$

      Compute (2) $\leftarrow \frac{1}{B} \sum_{i=1}^{N} T(x_i, y_i)$

      Compute (3) $\leftarrow \sum_{i=1}^{B} T(\bar{x}_i, \bar{y}_i) e^{T(\bar{x}_i, \bar{y}_i)}$

      Compute (4) $\leftarrow \sum_{i=1}^{B} T^2(\bar{x}_i, \bar{y}_i) e^{T(\bar{x}_i, \bar{y}_i)}$

      Compute $\Delta\eta^* \leftarrow \frac{(1)^2 * (2) - (1) * (3)}{(1) * (4) - (3)^2}$

    Compute Loss:

    $\mathcal{L}(\theta) \leftarrow -(1 + \Delta\eta^*) * (2) + \log\left(\frac{1}{B} \sum_{i=1}^{B} e^{(1+\Delta\eta^*) T_\theta(\bar{x}_i, \bar{y}_i)}\right)$

    Update Weights: $\theta \leftarrow \theta - \lambda \nabla_\theta \mathcal{L}(\theta)$

  end

end

return θ^*

From the claims made in [41], we would expect this algorithm to converge faster and to a better value of MI than the standard MINE algorithm. Although we did not have the time to fully implement and test this algorithm, we believe that it could be a very interesting direction to take to improve the performance of MINE.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**