Numerical Estimation of Information Measures

submitted by

Haoran Ni

for the degree of Master of Science

of the

University of Bath

Department of Mathematical Sciences

September 2018

This is a corrected version created at 25th September, 2019. The correction is noted by change bar at the margin.

Contents

List of Figures and Tables

Acknowledgements

I would like to express my deep gratitude to Dr. Keith Briggs, my research supervisor, for his patient guidance, enthusiastic encouragement and useful critiques of this research work. I would also like to thank Dr. Tiago Peixoto, my second supervisor, for his valuable comments and advice. My grateful thanks are also extended to Dr. Simon Thompson for his help in the feature selection applications, to Dr. Warren Lord, who discussed the bias of G-knn estimator with me through email.

This work was supported by the BT Wireless Research Department.

Summary

Mutual information, as a measure of mutual dependence without specific modelling assumptions, is popular in various scientific disciplines. However, numerically estimating mutual information could be difficult. The most popular estimators are the k-nearest neighbour estimators due to the theoretical and practical performance. We implemented two classic and two new k-NN estimators, which are the 3KL, the KSG, the BI-KSG and the G-knn estimators respectively. After that, we tested them on three different families of distribution. We also discussed the efficiency and accuracy of them under different conditions. Furthermore, we implemented the Gao's estimator and proposed an estimator based on the KL entropy estimator to deal with discrete-continuous data. By modifying the local estimates, we proposed a bias-improved G-knn estimator which not only keeps the advantages of the G-knn estimator but also improves the bias. Inspired by approximate k-NN search algorithms, we proposed an approximate k-NN method which improves the efficiency and accuracy of estimation under appropriate parameters. Finally, we showed the application significance of these estimators by testing them on MIMO communication channel and feature selection simulations.

Chapter 1

Introduction

Mutual information is one of many quantities that measure the mutual dependence between two variables. It can be perceived as the reduction in the uncertainty of one random variable due to the information from the other. As an information-theoretic quantity, mutual information plays an important role in a large number of scientific disciplines. There are some properties which make it popular in applications, such as the sensitivity to dependence compared with linear correlation coefficient, the invariance of one-to-one transformations and the obedience to the data processing inequality [19][17]. At the same time, to estimate it from samples becomes a vital issue. This question has totally different solutions depending on the original distribution from which the data are sampled. For discrete distributions, the minimax rate-optimal estimators are considered and developed by researchers. For continuous distributions, the interplay of geometry and the dimensionality become the key in estimating mutual information, which is one of the main foci of this report.

This report implements these classic and newest k-nearest neighbour (k-NN) estimators in the Python environment, investigates their performance, compares them against each other using different families of distribution, and applies them into MIMO communication channels and feature selection. In Chapter 2, we present the theoretical basis of estimating mutual information by four sections. We demonstrate differential entropy, mutual information and two divergences in the first two sections. Based on the understanding of them, we derive the entropy and mutual information for discrete-continuous case, and show that the mutual information is a weighted Jensen-Shannon divergence of a set of conditional distributions in that case. In the final section, we extend the mutual information of multivariate Gaussian to the complex field since the data of MIMO communication channel are from complex Gaussian distributions. After the introduction of theory, we review the previous studies of method and application in

Chapter 3. In Chapter 4, we formally present the algorithms of the estimators. We start from the Kozachenko and Leonenko (KL) estimator of entropy, and the 3KL estimator. Then we introduce the Kraskov, Stögbauer and Grassberger (KSG) estimator which has two slightly different algorithms with similar performances. Next we illustrate the bias-improved KSG (BI-KSG) estimator which uses ℓ_2-ball instead of ℓ_∞-ball to find the k-NN. After that, we present a new estimator named geometric k-NN (G-knn) to mainly deal with dependent samples. In the second section of this chapter, we firstly introduce Gao's estimator, which deals with samples generated from any kind of mixed distribution. Then we propose our estimator which focuses on discrete-continuous samples. In Chapter 5, the estimators are tested on three different families of distribution. We also evaluate the efficiency and discuss the bias of them after the numerical tests. Based on the understanding of the existing estimators, we propose two new methods in Chapter 6. They are the bias-improved G-knn (BIG-knn) estimator and the approximate k-nn method. Then, we test the both on same examples to demonstrate the improvements of them. In Chapter 7, we apply these estimators on MIMO communication channels and feature selection to test the performance of them on applications. In the last chapter, we conclude the main results of this report and propose the possible directions of further development in estimators and applications.

Chapter 2

Theory

This chapter theoretically introduces relevant measures and properties which are used in the following chapters. The base of the logarithm determines the units in which the information is measured. From now on, we use natural logarithm to measure information in nats.

2.1 Discrete and differential entropy

Given two discrete random variable $X \in \mathbb{R}^{d_X}$ and $Y \in \mathbb{R}^{d_Y}$ with possible values $\{x_1, x_2, \cdots, x_N\}$ and $\{y_1, y_2, \cdots, y_N\}$, we define the discrete entropy as below:

Definition 2.1.1. [2] The discrete entropy is defined as

$$H(X) := -\mathbb{E}[\log(p_x(x))] = -\sum_{x}^{N} p_x(x) \log(p_x(x)), \tag{2.1}$$

where $p_x(x)$ is the probability mass function of X, $\mathbb{E}[\cdot]$ is the expectation operator.

Definition 2.1.2. The joint entropy is given as

$$H(X,Y) := -\mathbb{E}[\log(p_{x,y}(x,y))] = -\sum_{x}^{N}\sum_{y}^{N} p_{x,y}(x,y) \log(p_{x,y}(x,y)), \tag{2.2}$$

where $p_{x,y}(x,y)$ is the joint probability mass function of (X,Y).

Definition 2.1.3. The conditional entropy of Y given X is defined as:

$$\begin{aligned} H(Y|X) &:= \sum_{x}^{N} p_x(x) H(Y|X=x) \\ &= -\sum_{x}^{N} p_x(x) \sum_{y}^{N} p_y(y|x) \log p_y(y|x) \\ &= -\sum_{x}^{N} \sum_{y}^{N} p_{x,y}(x,y) \log \left(\frac{p_{x,y}(x,y)}{p_x(x)} \right). \end{aligned} \tag{2.3}$$

Given two continuous random variable $X \in \mathbb{R}^{d_X}$ and $Y \in \mathbb{R}^{d_Y}$, we can define the differential entropy as below:

Definition 2.1.4. [2] The differential entropy is defined as

$$H(X) := -\mathbb{E}[\log(\mu_x(x))] = -\int_X dx \mu_x(x) \log(\mu_x(x)), \tag{2.4}$$

where μ_x is the probability density function of X.

Definition 2.1.5. The joint differential entropy is defined as

$$H(X,Y) := -\mathbb{E}[\log(\mu(x,y))] = -\int_{X,Y} dxdy \mu(x,y) \log(\mu(x,y)), \tag{2.5}$$

where $\mu(x,y)$ is the joint pdf of (X,Y).

Definition 2.1.6. The conditional differential entropy of Y given X is defined as:

$$\begin{aligned} H(Y|X) &:= -\int_X \int_Y dxdy \mu(x,y) \log(\mu_y(y|x)) \\ &= -\int_X \int_Y dxdy \mu(x,y) \log \left(\frac{\mu(x,y)}{\mu_x(x)} \right). \end{aligned} \tag{2.6}$$

2.2 Mutual information

Definition 2.2.1. [19] The mutual information between X and Y is defined as

$$I(X,Y) := H(X) + H(Y) - H(X,Y), \tag{2.7}$$

where $H(X,Y)$ is the joint entropy of X and Y.

We name the operation from entropy estimator to mutual information estimator using the definition of mutual information as *3H-principle* in this report [12]. Here we provide a simple example to help readers understand mutual information.

Example 2.2.2. Assuming two discrete random variables $(X, Y) \in \{0, 1\}^2$, where $\Pr[X = 0] = \Pr[X = 1] = 1/2$. If $X = 1$ then $Y = 1$, otherwise $\Pr[Y = 0] = \Pr[Y = 1] = 1/2$. Thus, the entropies of X and Y given by 2.1.1 are

$$\begin{aligned} H(X) &= -\sum_{i=1}^{2} \Pr_X(x_i) \log\left(\Pr_X(x_i)\right) \\ &= -\left(\frac{1}{2}\log\frac{1}{2} + \frac{1}{2}\log\frac{1}{2}\right) \\ &= \log 2 \end{aligned} \tag{2.8}$$

and

$$\begin{aligned} H(Y) &= -\sum_{i=1}^{2} \Pr_Y(y_i) \log\left(\Pr_Y(y_i)\right) \\ &= -\left(\frac{3}{4}\log\frac{3}{4} + \frac{1}{4}\log\frac{1}{4}\right) \\ &= 2\log 2 - \frac{3}{4}\log 3. \end{aligned} \tag{2.9}$$

The joint entropy of (X, Y) given by 2.1.2 is

$$\begin{aligned} H(X, Y) &= -\sum_{i=1}^{2}\sum_{j=1}^{2} \Pr_{X,Y}(x_i, y_j) \log\left(\Pr_{X,Y}(x_i, y_j)\right) \\ &= -\left(\frac{1}{2}\log\frac{1}{2} + 0 + \frac{1}{4}\log\frac{1}{4} + \frac{1}{4}\log\frac{1}{4}\right) \\ &= \frac{3}{2}\log 2. \end{aligned} \tag{2.10}$$

Combining (2.8), (2.9) and (2.10) using Definition 2.2.1, we obtain the mutual information of X and Y:

$$\begin{aligned} I(X, Y) &= H(X) + H(Y) - H(X, Y) \\ &= \log 2 + 2\log 2 - \frac{3}{4}\log 3 - \frac{3}{2}\log 2 \\ &= \frac{3}{4}\log\frac{4}{3}. \end{aligned} \tag{2.11}$$

■

In addition, if we assume there is a continuous random variable $Z = (X, Y)$ with joint density $\mu(x, y)$, $I(X, Y)$ can also be rewritten into the following integral:

$$I(X; Y) = \int\int dxdy\mu(x, y) \log\frac{\mu(x, y)}{\mu_x(x)\mu_y(y)}, \tag{2.12}$$

where the marginal densities of X and Y satisfy $\mu_x(x) = \int dx\mu(x,y)$ and $\mu_y(y) = \int dy\mu(x,y)$.

We collect some well-known properties about entropy and mutual information for analytically deriving the solutions.

Theorem 2.2.3. *[2, 24] (The properties of entropy and mutual information)*

- $H(Y|X) = 0$, *if and only if the value of* Y *is completely determined by the value of* X.
- $H(Y|X) = H(Y)$, *if and only if* X *and* Y *are independent random variables.*
- $H(Y|X) = H(X,Y) - H(X)$.
- $I(X,Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) = H(X,Y) - H(X|Y) - H(Y|X)$.
- $I(X, X+Y) = H(X+Y) - H(Y)$, *if and only if* X *and* Y *are independent random variables.*

Another important property is the invariance of mutual information under reparametizations demonstrated in Theorem 2.2.4.

Theorem 2.2.4. *[19] Given homeomorphisms* $X' = F(X)$ *and* $Y' = G(Y)$, *the pdf of the joint random variable* (X', Y') *is denoted as* $\mu'(x', y')$. *Thus, we obtain*

$$\begin{aligned} I(X', Y') &= \int\int dx'dy'\mu'(x', y') \log \frac{\mu'(x', y')}{\mu'_x(x')\mu'_y(y')} \\ &= \int\int dxdy\mu(x, y) \log \frac{\mu(x, y)}{\mu_x(x)\mu_y(y)} \\ &= I(X, Y). \end{aligned} \tag{2.13}$$

The proof of Theorem 2.2.4 is given in Appendix A.

2.3 Kullback-Leibler divergence and Jensen-Shannon divergence

Given two probability measures P and Q on a measurable space $(\Omega, \mathcal{F})$, we can define Kullback-Leibler divergence and Jensen-Shannon divergence as below:

Definition 2.3.1. [20, 27] The Kullback-Leibler divergence (KLD) between P and Q is defined as

$$D(P\|Q) := \int_\Omega dP \log \frac{dP}{dQ} \tag{2.14}$$

when P is absolutely continuous with respect to Q, and $+\infty$ otherwise, where $\frac{dP}{dQ}$ is the Radon-Nikodym derivative [29].

Definition 2.3.2. [13] The Jensen-Shannon divergence (JSD) $M(A) \times M(A) \to [0, \infty)$ is a symmetrized and smoothed version of the KLD for a set of probability distributions $M(A)$ where A is a set with some σ-*algebra* of measurable subsets. For $M = \frac{1}{2}(P + Q)$, The Jensen-Shannon divergence between P and Q is defined as

$$JSD(P\|Q) := \frac{1}{2}\left(D\left(P\|M\right) + D\left(Q\|M\right)\right). \tag{2.15}$$

2.4 Discrete-continuous mutual information

Mutual information between mixed types of random variable, as an unusual case, is normally avoided by other researchers. However, it is highly relevant to some specific areas of application, such as communication channel using quadrature amplitude modulation (QAM). In this section, we derive the formula of discrete-continuous mutual information and prove some important properties of it. Firstly, we derive the relationship between discrete and continuous entropies.

Consider the range of a random variable X with density $f(x)$ is divided into n small intervals of length Δ. Defining the quantized random variable $X^{\Delta} = x_i$, if $i\Delta \leqslant X < (i+1)\Delta$, we can obtain that:

Lemma 2.4.1. *[2] If the density $f(x)$ of the random variable X is Riemann integrable, then*

$$\lim_{\Delta \to 0} \left(H(X^{\Delta}) + \log \Delta\right) = H(f) = H(X). \tag{2.16}$$

Next, we use the same idea to obtain discrete-continuous joint entropy:

Lemma 2.4.2. *If the density $f(x)$ of the random variable X is Riemann integrable, then*

$$\lim_{\Delta_1 \to 0} \left(H(X^{\Delta_1}, Y^{\Delta_2}) + \log \Delta_1\right) = H(X, Y^{\Delta_2}). \tag{2.17}$$

The proof is given in Appendix A. From Lemma 2.4.1 and Lemma 2.4.2, we define discrete-continuous mutual information as below:

Definition 2.4.3. Given a discrete random variable $X \in \mathscr{X}$ with the probability mass function $p(x) = \Pr[X = x]$ and a continuous random variable $Y \in \mathbb{R}^{d_Y}$ with the probability density function $\mu_y(y)$, the discrete-continuous mutual information is defined

as

$$\begin{aligned} I(X^{\Delta}, Y) &:= H(X^{\Delta}) + H(Y) - H(X^{\Delta}, Y) \\ &= -\sum_x p(x_i) \log p(x_i) - \int_y \mu_y(y) \log \mu_y(y) dy + \sum_x \int_y \mu(x_i, y) \log \mu(x_i, y) dy \\ &= \sum_x \int_y \mu(x_i, y) \log \frac{\mu(x_i, y)}{p(x_i)\mu_y(y)}, \end{aligned} \tag{2.18}$$

where we denote the mixed density by $\mu(x_i, y)$ which satisfies $p(x_i) = \int_y \mu(x_i, y) dy$ and $\mu_y(y) = \sum_x \mu(x_i, y)$.

We collect some properties of it in Theorem 2.4.4.

Theorem 2.4.4. *(The properties of mixed entropy and mutual information)*

- $H(Y|X^{\Delta}) = H(Y)$, *if and only if* X *and* Y *are independent random variables.*
- $H(Y|X^{\Delta}) = H(X^{\Delta}, Y) - H(X^{\Delta})$.
- $I(X^{\Delta}; Y) = H(Y) - H(Y|X^{\Delta})$.
- $I(X'^{\Delta}, Y') = I(X^{\Delta}, Y)$, *for homeomorphisms* $X'^{\Delta} = F(X^{\Delta})$ *and* $Y' = G(Y)$.

Note 2.4.5. Notice that the discrete-continuous mutual information equals to a weighted Jensen-Shannon divergence of a set of conditional distributions. Thus, we can rewrite it using Jensen-Shannon divergence as below:

$$\begin{aligned} I(X^{\Delta}, Y) &= H(Y) - H(Y|X^{\Delta}) \\ &= \int_y \mu_y(y) \log \mu_y(y) dy - \sum_x p_x(x_i) \int_y \mu_{x_i}(y) \log \mu_{x_i}(y) dy \\ &= H\left(\sum_x p_x(x_i) X_i\right) - \sum_x p_x(x_i) H(Y|X_i) \\ &=: JSD_{p_x(x)}(X_1, X_2, \cdots, X_n), \end{aligned} \tag{2.19}$$

where $\mu_y(y) = \sum_x p_x(x_i)\mu_{x_i}(y)$ and X_i is the random variable with probability density function $\mu_{x_i}(y)$. ■

Gao, Kannan, Oh and Viswanath [12] provided a more general definition of mutual information based on Radon-Nikodym derivative in 2017, which covers the case of Definition 2.4.3. Each of the two definitions has its advantages. We will use both of them in the following chapters. The definition is shown as follows.

Definition 2.4.6. Consider a probability measure P_{XY} on the space $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are Euclidean spaces. Define $P_X(A) = P_{XY}(A \times \mathcal{Y})$ and $P_Y(B) = P_{XY}(\mathcal{X} \times B)$ for any measurable set $A \subseteq \mathcal{X}$ and $B \subseteq \mathcal{Y}$. The mutual information $I(X,Y)$ of P_{XY} is defined as

$$I(X,Y) := \int_{\mathcal{X}\times\mathcal{Y}} \log \frac{dP_{XY}}{dP_X \times P_Y} dP_{XY}, \tag{2.20}$$

where $\dfrac{dP_{XY}}{dP_X \times P_Y}$ is the Radon-Nikodym derivative.

Notice that the following cases are all included in this definition:

(a) One of X and Y is discrete, and another is continuous;

(b) X or Y has many components where some of them are discrete and others are continuous;

(c) X, Y or (X,Y) is a mixture of discrete and continuous.

2.5 Complex Gaussian channel

In this section, we firstly investigate the properties of complex Gaussian distribution. Then, we derive the formula of entropy for complex Gaussian distribution. Finally, we show the mutual information of the complex Gaussian channel model.

Consider a complex random vector $\boldsymbol{x} \in \mathbb{C}^n$ and a matrix $Q \in \mathbb{C}^{n\times n}$, define

$$\hat{\boldsymbol{x}} = \begin{bmatrix} \Re(\boldsymbol{x}) \\ \Im(\boldsymbol{x}) \end{bmatrix} \quad \text{and} \quad \hat{Q} = \begin{bmatrix} \Re(Q) & -\Im(Q) \\ \Im(Q) & \Re(Q) \end{bmatrix}, \tag{2.21}$$

where $\Re(\boldsymbol{x})$ and $\Im(\boldsymbol{x})$ denote the real and imaginary parts of $\boldsymbol{x}$.

Lemma 2.5.1. *[34] $\boldsymbol{x}$ is said to be Gaussian if $\hat{\boldsymbol{x}}$ is Gaussian. If so, we denote the expectation and covariance of $\hat{\boldsymbol{x}}$ as $\mathbb{E}[\hat{\boldsymbol{x}}] \in \mathbb{R}^{2n}$ and $\mathbb{E}\left[(\hat{\boldsymbol{x}} - \mathbb{E}[\hat{\boldsymbol{x}}])(\hat{\boldsymbol{x}} - \mathbb{E}[\hat{\boldsymbol{x}}])^\dagger\right] \in \mathbb{R}^{2n\times 2n}$.*

Lemma 2.5.2. *[34] A complex Gaussian random vector $\boldsymbol{x}$ is circularly symmetric, if for some Hermitian non-negative definite $Q \in \mathbb{C}^{n\times n}$, we have the covariance of $\hat{\boldsymbol{x}}$ as below:*

$$\mathbb{E}\left[(\hat{\boldsymbol{x}} - \mathbb{E}[\hat{\boldsymbol{x}}])(\hat{\boldsymbol{x}} - \mathbb{E}[\hat{\boldsymbol{x}}])^\dagger\right] = \frac{1}{2}\hat{Q}. \tag{2.22}$$

Lemma 2.5.3. *[34] The covariance of a circularly symmetric complex Gaussian random vector $\boldsymbol{x}$ is $\mathbb{E}\left[(\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}])(\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}])^\dagger\right] = Q$.*

Now we can derive the differential entropy for complex Gaussian distribution.

Lemma 2.5.4. *All eigenvalues of Hermitian non-negative definite $Q \in \mathbb{C}^{n\times n}$ are real and non-negative. Thus,* $\det(Q) = \prod_i^n \lambda_i \geq 0$.

The proof is given in Appendix A.

Theorem 2.5.5. *[34] The differential entropy of a complex Gaussian $\boldsymbol{x}$ with covariance Q is given by*

$$\begin{aligned}\mathcal{H}(\boldsymbol{x}) &= \mathbb{E}_{\gamma_Q}\left[-\log \gamma_Q(\boldsymbol{x})\right] \\ &= \log\det(\pi Q) + (\log e)\mathbb{E}\left[\boldsymbol{x}^\dagger Q^{-1}\boldsymbol{x}\right] \\ &= \log\det(\pi Q) + (\log e)\, tr(I) \\ &= \log\det(\pi e Q).\end{aligned} \tag{2.23}$$

where γ_Q is the pdf of $\boldsymbol{x}$.

For a single user Gaussian channel with multiple transmitting and receiving, we denote the transmitted vector as $\boldsymbol{x} \in \mathbb{C}^t$ and the received vector as $\boldsymbol{y} \in \mathbb{C}^r$. We consider a linear model in which $\boldsymbol{y}$ depends on $\boldsymbol{x}$ via

$$\boldsymbol{y} = H\boldsymbol{x} + \boldsymbol{n}, \tag{2.24}$$

where H is a $r \times t$ complex random matrix and $\boldsymbol{n}$ is complex Gaussian noise with $\mathbb{E}[\boldsymbol{n}] = 0$ and $\mathbb{E}\left[\boldsymbol{n}\boldsymbol{n}^\dagger\right] = I_r$.

Consider a circularly symmetric complex Gaussian $\boldsymbol{x}$ with covariance Q in this model, the covariance of $\boldsymbol{y}$ is $\mathbb{E}\left[\boldsymbol{y}\boldsymbol{y}^\dagger\right] = HQH^\dagger + I_r$. The mutual information between $\boldsymbol{x}$ and $\boldsymbol{y}$ is shown as below.

Lemma 2.5.6. *If Q is a Hermitian non-negative definite matrix, then for random matrix A we have $AQA^\dagger$ is also a Hermitian non-negative definite matrix.*

The proof is given in Appendix A.

Theorem 2.5.7. *[34] The mutual information $\mathcal{I}(\boldsymbol{x}, \boldsymbol{y})$ is given by*

$$\begin{aligned}\mathcal{I}(\boldsymbol{x}, \boldsymbol{y}) &= \mathcal{H}(\boldsymbol{y}) - \mathcal{H}(\boldsymbol{y}|\boldsymbol{x}) \\ &= \mathcal{H}(\boldsymbol{y}) - \mathcal{H}(\boldsymbol{n}) \\ &= \log\det\left(I_r + HQH^\dagger\right).\end{aligned} \tag{2.25}$$

Chapter 3

Literature Review

3.1 Review of methods

There are various methods to do the estimation for continuous distributions. Cumulant expansions give crude approximations of mutual information for distributions close to Gaussian. Entropy maximalization is more robust but is still crude approximations [16]. One kind of the well-known methods is partitioning the supports of the random variables into finite size bins [4, 35], which is called binning estimators. This kind of method does converge to true solutions for proper functions as samplesize tends to infinity and all bin sizes tend to zero, but we will not pursue it due to the limitation of convergence conditions and accuracy. Another kind of promising method is the kernel density estimators (KDE) [25, 32]. KDE performs better than binning estimators and it has both smaller statistical and systematic errors. However, we will not investigate it because of the quick deterioration and high computational costs as mentioned in [19].

Among these estimators, the k-nearest neighbour estimators are singled out due to the theoretical and practical performance. The most classic k-NN estimator of entropy was proposed with a bias correction term by Kozachenko and Leonenko [18] named KL estimator in 1987, which also led to a fairly good estimator by the definition of mutual information denoted as 3KL estimator. In 2004, Kraskov, Stögbauer and Grassberger [19] proposed a different k-NN estimator of mutual information named KSG estimator and illustrated the improvements in variant settings. Due to superior performance, it became one of the most popular estimators. In 2017, Gao, Oh and Viswanath [11] investigated the basic theoretical properties, such as consistency and convergence, of KL and KSG estimators. They believe that the better performance of KSG over 3KL is due to the *correlation boosting effect*. Based on the understanding of this effect, they introduced a new method denoted as bias-improved KSG (BI-KSG) estimator, which

has further improvements in vanishing bias. Lord, Sun and Bollt (2018) [23] noticed that most k-NN methods use regular local volume elements which allows the estimators to be asymptotic consistent. However, it is not descriptive of the geometry of the probability measures and results in bias. They introduced the geometric k-NN estimators (G-knn) of entropy and mutual information based on elliptical local volume elements. This method outperforms the KSG for small sample size and highly dependent samples, but it is not corrected for asymptotic bias.

In some applications, the samples are generated from discrete-continuous mixed distributions. There are only few studies about estimating mutual information in this case. Ross [28] presented a rough definition of mutual information in 2014. Given the understanding of it, he proposed a k-NN estimator and compared it with binning method. It generally performs well on the given examples. However, we will not introduce this estimator due to the bad performance on higher dimension samples. Gao, Kannan, Oh and Viswanath [12] gave a more general definition of mutual information which includes any mixed case and proposed an estimator (we denote it as Gao's estimator) which is inspired by the KSG estimator.

3.2 Review of applications

Mutual information has been applied in a variety of disciplines. As demonstrated by Hyvärinen, Karhunen and Oja [16], mutual information occupies an important position in independent component analysis (ICA). In ICA problems, mutual information is not only used for testing the actual independence of the components, but also for testing the uniqueness and robustness [19]. Another possible application area mentioned in [28] is the gene detection. For example, we can estimate the mutual information between base sequence and the gene expression level on patient survival time to provide ideas of the mechanism of some genetic diseases. In big data area, there are a large number of recent research about link prediction and topic discovery algorithms of different kinds of social network based on mutual information, such as [3] and [30]. Furthermore, the mutual information can also be used to measure the similarity of two datasets to give an evaluation of algorithm performance. Mutual information are also applied in financial market studies. Guo, Zhang and Tian [14] believe that mutual information is more effective than Pearson correlation coefficient for characterising the nonlinear relationship between stocks. Wang and Hui [36] investigated the distribution of financial contagion using mutual information.

Apart from the above applications, feature selection and multi-input and multi-output (MIMO) communication channels are two popular application fields involving

mutual information. There are numerous feature selection algorithms using mutual information as their measures [26][7][22]. However, as discussed by Kwak and Choi [21], estimating mutual information could be difficult, and the performance of these algorithms depends on the accuracy of the estimates. For MIMO communication channels, mutual information is the ideally suitable measure for the capacity of channels. This idea was originally proposed by Shannon [31] in 1948 and widely used by other researchers [15][33]. In 1999, Telatar [34] extended the mutual information to complex distributions which introduces more possibility in relevant applications. In this report, we will discuss the possible applications in MIMO communication channels and feature selection using our estimators, and test the performance of estimator on given examples.

Chapter 4

Methods

4.1 Continuous estimators

Differential entropy and mutual information estimation is the main focus in this section. We present four continuous estimators, which are the KL estimator, the KSG estimator, the BI-KSG estimator and the G-knn estimator. The main idea and performance of them are partly similar, but each has its own merits. We start from the KL estimator.

KL estimator

Consider N i.i.d. samples $x_1, x_2, \cdots, x_N$ generated from a random variable $X \in \mathbb{R}^d$ which has the probability density function $f_X(x)$. The KL estimator estimates entropy by the following formula [18]:

$$\begin{aligned} H(X) &= -\mathbb{E}[\log f_X(X)] \\ &\approx -\frac{1}{N}\sum_{i}^{N} \log \hat{f}_X(x_i), \end{aligned} \tag{4.1}$$

where $\hat{f}_X(x_i)$ is the estimate of $f_X(x)$ locally at $x = x_i$. To estimate the local pdfs, we firstly find the distance $\epsilon_i^{k,p}$ from x_i to its k-th nearest neighbour in ℓ_p $(p \geqslant 1)$ space. Then, we compute the volume of the unit ℓ_p-ball in d dimension [11]:

$$V^{p,d} = 2^d \frac{\Gamma\left(1+\frac{1}{p}\right)^d}{\Gamma\left(1+\frac{d}{p}\right)}. \tag{4.2}$$

The local estimation $\hat{f}_X(x_i)$ is given by

$$\begin{aligned}\hat{f}_X(x_i) &\simeq \frac{k/N}{\mathrm{Vol}_i} \\ &= \frac{k/N}{V^{p,d}(\epsilon_i^{k,p})^d},\end{aligned} \tag{4.3}$$

where Vol_i denote the volume of the neighbourhood of x_i. Combining Equation (4.1) and Equation (4.3), we obtain the KL estimator of entropy [11]:

$$\hat{H}_{\mathrm{KL}}(X) = \log N + \log\left(V^{p,d}\right) + \frac{d}{N}\sum_i^N \log\left(\epsilon_i^{k,p}\right) - \psi(k), \tag{4.4}$$

where $\psi(x)$ is the digamma function defined as $\psi(x) = \Gamma^{-1}(x)d\Gamma(x)/dx$. Note that $\psi(x) = \log x - 1/2x + o(1/x)$ has an important correction term $-1/2x + o(1/x)$ for debiasing the estimator [18]. Using 3H-principle, we can simply obtain an estimation of mutual information by applying the KL estimator three times:

$$\hat{I}_{3\mathrm{KL}}(X,Y) = \hat{H}_{\mathrm{KL}}(X) + \hat{H}_{\mathrm{KL}}(Y) - \hat{H}_{\mathrm{KL}}(X,Y), \tag{4.5}$$

which is denoted as 3KL estimator.

KSG estimator

Based on the KL estimator, the KSG estimator with two slightly different algorithms is introduced in [19]. The main difference between the KSG estimator and the 3KL estimator is the value of k that they choose to estimate the marginal entropies. In the 3KL estimator, the entropies are estimated separately using fixed k. However, the authors of [19] believe that the bias of $\hat{H}(X)$, $\hat{H}(Y)$ and $\hat{H}(X,Y)$ will cancel out by using the number of points in the neighbourhood of each sample in X and Y space respectively, which is *sample dependent*, where the radius of the neighbourhood depends on the k-NN distances of the sample in the joint space $Z = (X,Y)$. Now we introduce the two algorithms of KSG estimator.

Consider N i.i.d. samples $z_1 = (x_1, y_1), \cdots, z_N = (x_N, y_N)$ generated from $Z = (X,Y)$, where X and Y are two random variables. We denote the distance from z_i to its k-th nearest neighbour by $\epsilon(i)$, and the distance between the same points in the subspaces by $\epsilon_x(i)$ and $\epsilon_y(i)$. Since the KSG estimator use the maximum norm distance, we have $\epsilon(i) = \max\{\epsilon_x(i), \epsilon_y(i)\}$.

In the first algorithm, we count, for each x_i in the subspace, the number of points $n_x(i)$ whose distance from x_i is strictly less than $\epsilon(i)$, and similarly for each y_i to obtain

$n_y(i)$. Then the estimate of mutual information is given by

$$\hat{I}_{\text{KSG}^1}(X,Y) = \psi(k) - \frac{1}{N}\sum_{i}^{N}\left(\psi\left(n_x\left(i\right)+1\right)+\psi\left(n_y\left(i\right)+1\right)\right)+\psi(N). \tag{4.6}$$

In the second algorithm, we count the number of points $n_x(i)$ and $n_y(i)$ by using the conditions $\|x_i - x_j\|_\infty \leqslant \epsilon_x(i)$ and $\|y_i - y_j\|_\infty \leqslant \epsilon_y(i)$. Thus, we obtain

$$\hat{I}_{\text{KSG}^2}(X,Y) = \psi(k) - 1 - \frac{1}{N}\sum_{i}^{N}\left(\psi\left(n_x\left(i\right)\right)+\psi\left(n_y\left(i\right)\right)\right)+\psi(N). \tag{4.7}$$

BI-KSG estimator

In [11], Gao, Oh and Viswanath gave an explanation about the superior performance of KSG estimator, which is the correlation boosting effect. They found out that the bias of estimators will be reduced if the bias of the joint entropy estimate is positively correlated to the bias of marginal entropy estimates. Based on the understanding of this effect, they built a new estimator named BI-KSG estimator. In this estimator, they use ℓ_2-norm to measure k-NN distances instead of ℓ_∞-norm. They also replace $\psi(n_x(i)+1)$ and $\psi(n_y(i)+1)$ by $\log\left(n_x(i)\right)$ and $\log\left(n_y(i)\right)$ separately. Denoting $\epsilon(i)$ as the ℓ_2-distance from z_i to its k-th nearest neighbour, $n_x(i)$ and $n_y(i)$ are the number of points satisfying the conditions $\|x_i - x_j\|_2 \leqslant \epsilon(i)$ and $\|y_i - y_j\|_2 \leqslant \epsilon(i)$. Thus, we have

$$\hat{I}_{\text{BI-KSG}}(X,Y) = \psi(k)+\log N+\log\left(\frac{V_{d_x}V_{d_y}}{V_{d_x+d_y}}\right)-\frac{1}{N}\sum_{i}^{N}\left(\log\left(n_x\left(i\right)\right)+\log\left(n_y\left(i\right)\right)\right), \tag{4.8}$$

where $V_d = \dfrac{\pi^{d/2}}{\Gamma(d/2+1)}$ is the volume of d-dimension unit ℓ_2-ball.

G-knn estimator

As shown in [23], the KSG estimator does not perform well when the two random variables are highly dependent or the sample size is small. To fill this gap, a svd-based geometric k-NN estimator was defined by Lord, Sun and Bollt [23] in 2018. The estimator is inspired by Gao, Steeg, and Galstyan (GSG) estimator [10], which uses the principal component analysis of local data to fit a hyper-rectangle. We start from the entropy estimator.

Recall the formula (4.1) and the local pdf estimate formula (4.3). In G-knn estimator, we use elliptical local volume elements, which is estimated by singular-value decomposition (svd) of local data. For each sample x_i, denote $\boldsymbol{x}_i = \left(x_i^0, x_i^1, x_i^2, \cdots, x_i^k\right)$

where $x_i^0 = x_i$ to be the k-neighbourhood of sample points. In order to indicate directions of maximal stretching by using svd, firstly we center the data to the centroid $z_i = \frac{1}{k+1}\sum_{j=0}^{k} x_i^j$. Denote $y_i^j = x_i^j - z_i$ for $j \in \{0, 1, 2, \cdots, k\}$, we obtain a $(k+1) \times d$ matrix

$$Y_i = \begin{pmatrix} y_i^0 \\ y_i^1 \\ \vdots \\ y_i^k \end{pmatrix}. \tag{4.9}$$

Using singular-value decomposition on Y_i, we obtain $Y_i = U_i \Sigma_i V_i^T$, where the diagonal entries of Σ_i are the singular values of Y_i, and the columns of U_i and V_i are the left and right singular vectors. Define $\epsilon(x_i, k)$ to be the ℓ_2-distance from x_i to the k-nearest data point, we let $\boldsymbol{r}_i = \epsilon(x_i, k)\left[\frac{\sigma_i^1}{\sigma_i^1}v_i^1, \frac{\sigma_i^2}{\sigma_i^1}v_i^2, \cdots, \frac{\sigma_i^d}{\sigma_i^1}v_i^d\right]$ to be the axes, and x_i to be the centroid of the ellipsoid to fit the local data, where σ_i^l and v_i^l $(l = 1, \cdots, d)$ are the singular values and vectors from Σ_i and V_i. Denote $n_x(i)$ to be the number of points inside the local ellipsoid, we can derive the local density estimate $\hat{f}_X(x_i)$ directly by (4.3) to obtain

$$\hat{f}_X(x_i) = \frac{n_x(i)/N}{V_i} \tag{4.10}$$

where V_i is the volume of the local ellipsoid computed by the formula

$$V_i = \frac{\pi^{d/2}}{\Gamma(1 + d/2)}\epsilon(x_i, k)^d \prod_l^d \frac{\sigma_i^l}{\sigma_i^1}. \tag{4.11}$$

Substituting into Equation (4.1) yields the entropy estimator:

$$\begin{aligned} \hat{H}_{\text{G-knn}}(X) =& -\frac{1}{N}\sum_i^N \log\frac{n_x(i)/N}{V_i} \\ =& \log N + \log\left(\frac{\pi^{d/2}}{\Gamma(1 + d/2)}\right) - \frac{1}{N}\sum_i^N \log\left(n_x\left(i\right)\right) \\ &+ \frac{d}{N}\sum_i^N \log\left(\epsilon\left(x_i, k\right)\right) + \frac{1}{N}\sum_i^N\sum_l^d \log\left(\frac{\sigma_i^l}{\sigma_i^1}\right). \end{aligned} \tag{4.12}$$

We can then derive the estimate of mutual information by 3H-principle:

$$\hat{I}_{\text{G-knn}}(X, Y) = \hat{H}_{\text{G-knn}}(X) + \hat{H}_{\text{G-knn}}(Y) - \hat{H}_{\text{G-knn}}(X, Y). \tag{4.13}$$

4.2 Discrete-continuous estimators

In this section, we introduce two estimators which estimate mutual information from discrete-continuous samples. They are the Gao's estimator and the Multi-KL estimator. Both the two estimators are developed from continuous estimators.

Gao's estimator

Inspired by the KSG estimator, Gao, Kannan, Oh and Viswanath [12] presented an estimator (we denote it as Gao's estimator) which estimates the mutual information satisfying Definition 2.4.6. For $z_i = (x_i, y_i)$, denote $\epsilon(i)$ as the k-smallest distance among $d_{i,j} = \max\{\|X_j - X_i\|, \|Y_j - Y_i\|\}$ for $j \neq i$, and k_i as the number of samples satisfying $d_{i,j} \leqslant \epsilon(i)$. Since $\epsilon(i) = 0$ can only be satisfied when both X and Y are discrete, which is not the focus of this report, we let $k_i = k$ all the time. Thus, the estimator is given as follows.

$$\hat{I}_{\text{Gao's}}(X, Y) = \log N + \psi(k) - \frac{1}{N}\sum_{i}^{N}\left(\log\left(n_x(i) + 1\right) + \log\left(n_y(i) + 1\right)\right), \tag{4.14}$$

where $n_x(i)$ and $n_y(i)$ are the number of samples with $\|X_j - X_i\| \leqslant \epsilon(i)$ and $\|Y_j - Y_i\| \leqslant \epsilon(i)$.

Multi-KL estimator

As mentioned in Note 2.4.5, discrete-continuous mutual information can be treated as a weighted Jensen-Shannon divergence of conditional distributions. Inspired by Ross [28], we propose a the Multi-KL estimator which follows the formula (2.19). The idea of this estimator is to estimate each entropy separately by the KL estimator and add them together by the estimated weights. The estimator is given as

$$\begin{aligned}\hat{I}_{\text{Multi-KL}}(X^{\Delta}, Y) = \frac{d}{N}\left(\sum_{i}^{N}\log\left(\epsilon_y(i)\right) - \sum_{j}^{x}\sum_{i}^{N_{x_j}}\log\left(\epsilon_{y_j}(i)\right)\right) \\ - \sum_{j}^{x}\frac{N_{x_j}}{N}\log N_{x_j} + \log\left(N\right),\end{aligned} \tag{4.15}$$

where $\epsilon_y(i)$ is the distance between y_i and the k^{th} nearest neighbour in the full data set of Y, $\epsilon_{y_j}(i)$ is the distance between y_i and the k^{th} nearest neighbour in the subset of Y whose value of the discrete variable equals x_j, and N_{x_j} is the number samples for each possible value of X^{Δ}.

Chapter 5

Results

In this chapter, we test the different estimators of mutual information and compare them against each other. There are many environments for us to implement these estimators such as C, Java and MATLAB. We choose Python in this report not only because of the widespread use in relevant industry but also the rich set of numerical computation libraries. In the implementation, we use k-d tree algorithm [1] in SciPy library to query the k-nearest neighbour, which improves the efficiency of algorithm compared with direct implementation. In the first section, we choose some classic and new families of distribution to test the performance of them. Next we evaluate the efficiency and accuracy of them for variant k and samplesize. In the last section, we discuss the bias of these estimators.

5.1 Numerical tests

This section compares different estimates of mutual information on simulated examples. The examples are divided into three families of distribution and the analytical solutions of mutual information are given. The first two families are designed so that the dependence of the samples is controlled by a single scalar parameter α. The third family is two special distributions which the true solutions were derived by Darbellay and Vajda [5] in 2000. We choose these special distributions not only because they are untested by other researchers but also because they have some kind of tail behaviours which may introduce bias into estimators.

Gaussian distributions

The first family of distribution is bivariate Gaussian and 4-dimensional multivariate Gaussian. The model is

$$(X, Y) \sim \mathcal{N}(0, \Sigma), \tag{5.1}$$

where Σ is the covariance of the joint variable (X, Y). The exact solution is

$$I(X, Y) = \frac{1}{2} \log \left(\frac{|\Sigma_X||\Sigma_Y|}{|\Sigma|} \right), \tag{5.2}$$

where Σ_X and Σ_Y are the covariance matrices of X and Y. The proof is given in Appendix A.

The covariance of the bivariate Gaussian is

$$\Sigma_1 = \begin{bmatrix} 1 & \alpha \\ \alpha & 1 \end{bmatrix}. \tag{5.3}$$

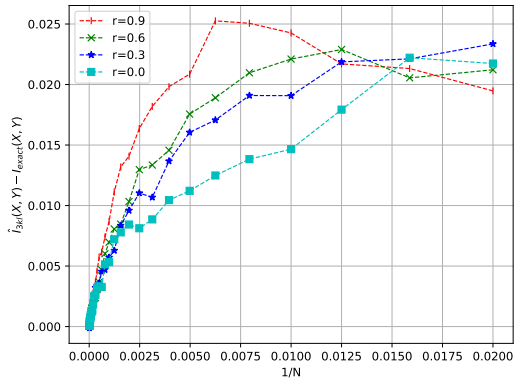
We estimate $\hat{I}(X, Y) - I_{\text{exact}}(X, Y)$ for $r = 0.0, 0.3, 0.6$, and 0.9, plot against $1/N$ for each estimator except G-knn, since it is not asymptotic unbiased. The results are shown in Figure 5-1. We also plot the RMS errors of the estimates for $r = 0.9$ against N and $1/k$. The results are shown in Figure 5-2.

For $N \to \infty$, the systematic errors of the four estimators converge to 0 as expected. It can be seen clearly that the errors of the KSG estimator are smaller than the 3KL for variant N. The two algorithms of the KSG perform similarly well for large samplesize, but the first algorithm of the KSG outperforms the second one for small samplesize. For the BI-KSG estimator, the convergence rate is higher than the 3KL, but for small samplesize it does not look competitive. For medium samplesize, the BI-KSG performs better than other estimators. It also generally performs well for variant r.

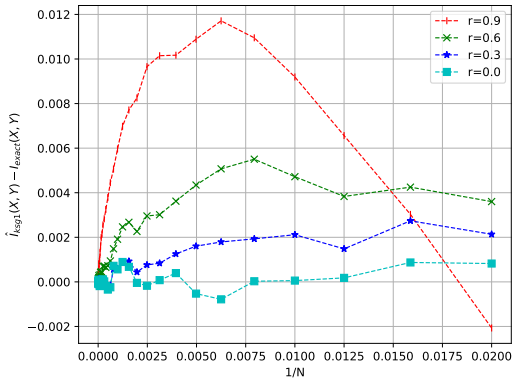
The next example is 4-dimensional multivariate Gaussian. The covariance of 4-dimensional multivariate Gaussian in the simulation is

$$\Sigma_2 = \begin{bmatrix} 3 & -1 & -1 & -2 \\ -1 & 2+\alpha & -3 & 4 \\ -1 & -3 & 11 & -4 \\ -2 & 4 & -4 & 9 \end{bmatrix}. \tag{5.4}$$

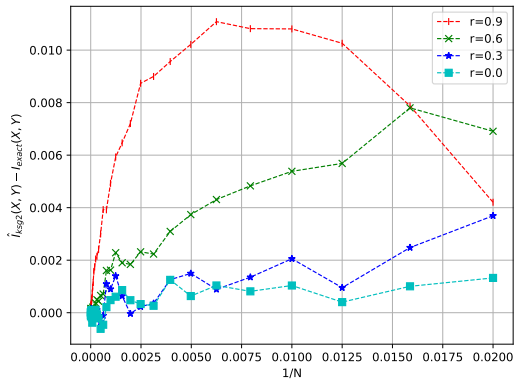
We estimate $\hat{I}(X, Y)$ for variant α and N in Figure 5-3. In this simulation, we include G-knn estimator in the plotting. For each realization, we generate one sample of size N of the joint random variable and test it on all the estimators. For the G-knn estimator, we choose $k = 20$ to be small enough for estimating the local density and large enough



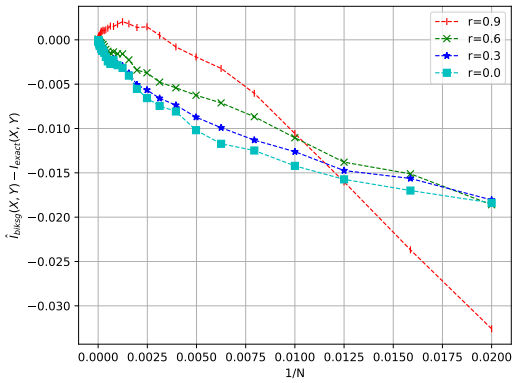
(a) Errors of 3KL estimator



(b) Errors of KSG^1 estimator



(c) Errors of KSG^2 estimator

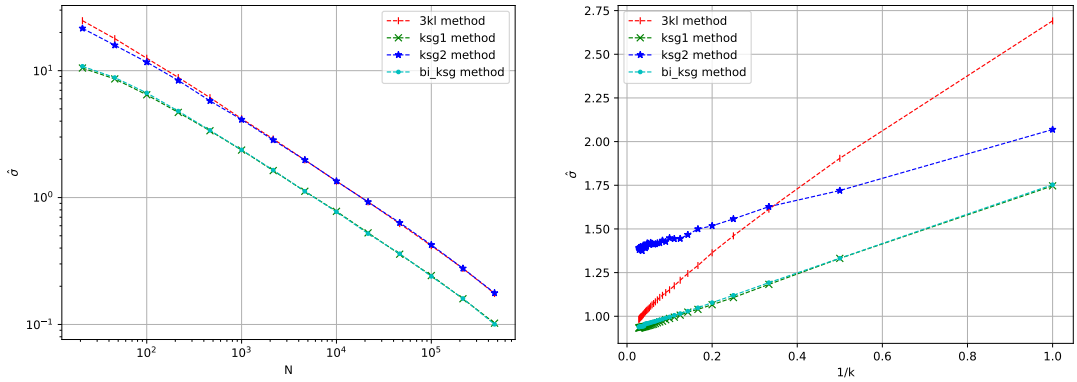


(d) Errors of BI-KSG estimator

Figure 5-1: Errors of estimate, where k is fixed at 1.

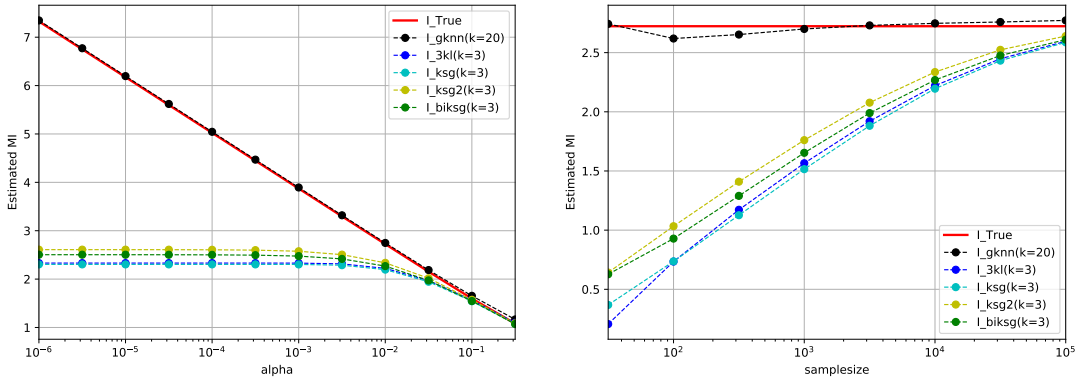
to have a good local ellipsoid element from the svd. For the KSG estimator, k is commonly chosen from 2 to 6 [23]. In the following experiments, we choose k to be 3 for the estimators excluding G-knn. In addition, the choice of k should depend on the dimension of the given joint samples since the dimension of the local geometry equals to the dimension of space.

It can be observed from the results that the G-knn outperforms other estimators in the most cases of this example. Although the G-knn estimator is not corrected for bias, it still performs better than other estimators for small samplesize and high dependence due to the sample-dependent local volume elements.



(a) $\hat{\sigma}$ against N (b) $\hat{\sigma}$ against $1/k$

Figure 5-2: RMS errors of estimators for $r = 0.9$, where k is fixed at 1 in Fig (a) and N is fixed at 10^4 in Fig (b).



(a) Estimated MI against α (b) Estimated MI against samplesize N

Figure 5-3: Estimated MI of different estimators, where N is fixed at 10^4 in Fig (a) and α is fixed at 0.01 in Fig (b).

Uniform distribution with Gaussian noise

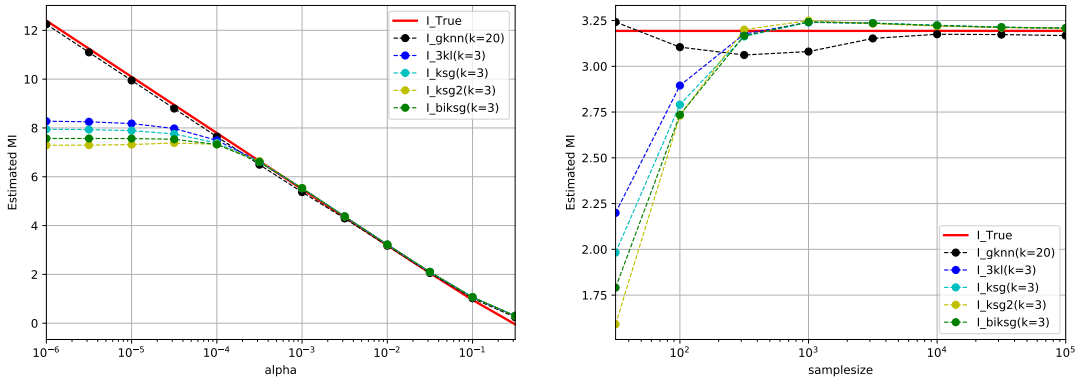
The second family of distribution is built by adding random Gaussian noise with different standard deviation α on uniform distribution. The model is given by

$$\begin{aligned} Y &= X + \alpha U \\ X &\sim \text{Unif}\,(0,1), \\ U &\sim \mathcal{N}(0,1), \end{aligned} \tag{5.5}$$

where X and U are independent. Then the exact value of the mutual information is

$$I(X,Y) = -\int_{-\alpha}^{\alpha} \left(\Phi\left(1-\frac{y}{\alpha}\right) - \Phi\left(-\frac{y}{\alpha}\right)\right) \log\left(\Phi\left(1-\frac{y}{\alpha}\right) - \Phi\left(-\frac{y}{\alpha}\right)\right) dy \\ - \log\alpha - \frac{1}{2}\log(2\pi e), \tag{5.6}$$

where $\Phi(x)$ is the cdf of the standard normal distribution. Here we numerically calculate the exact solution in the simulations. We plot the estimated MI using the same settings in the multivariate Gaussian case. The results are shown in Figure 5-4.



(a) Estimated MI against α (b) Estimated MI against samplesize N

Figure 5-4: Estimated MI of different estimators, where N is fixed at 10^4 in Fig (a) and α is fixed at 0.01 in Fig (b).

In this example, the G-knn estimator only performs better when α is no larger than 10^{-4} in the left figure and samplesize is no larger than 10^2 in the right figure. It still has the better performance when the measure is thinly supported as we mentioned in the last example, but the range of the parameters which gives better performance is smaller due to the local stretching mode of samples between the two models.

Two new distributions

In previous studies, mutual information estimators are always tested with common distributions such as uniform and Gaussian. However, the estimators are designed to deal with samples from unknown distributions and they indeed perform differently depending on the distributions. In this section, we choose two new distributions with special kind of tail behaviours to test the estimators. The first example is bivariate

Pareto distribution of type I. The joint pdf in this case is given by

$$f(x_1, x_2) = \frac{\alpha(\alpha+1)}{\theta_1\theta_2}\left(\frac{x_1}{\theta_1} + \frac{x_2}{\theta_2} - 1\right)^{-(\alpha+2)}. \tag{5.6}$$

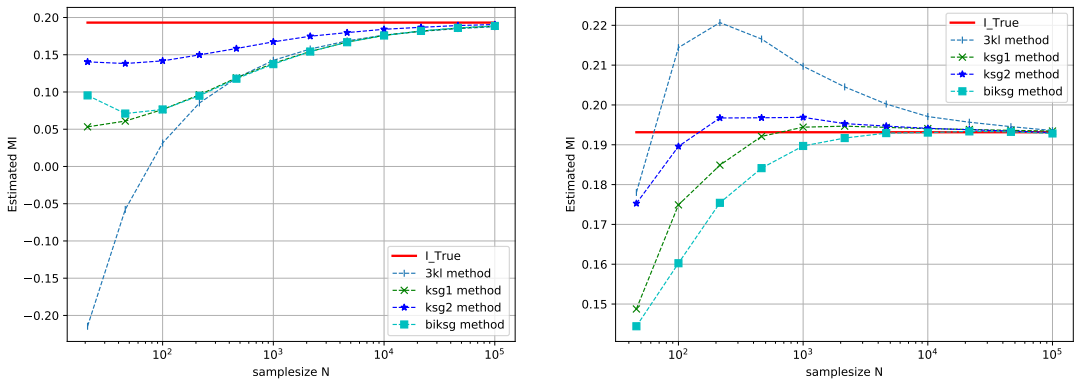
The second example is bivariate logistic distribution, the joint pdf is given by

$$f(x_1, x_2) = \frac{2}{\theta_1\theta_2}\exp\left(-\frac{x_1}{\theta_1} - \frac{x_2}{\theta_2}\right)\left(1 + \exp\left(-\frac{x_1}{\theta_1}\right) + \exp\left(-\frac{x_2}{\theta_2}\right)\right)^{-3}. \tag{5.7}$$

For these two distributions, the true solution derived by Darbellay and Vajda [5] is

$$I(X, Y) = \log\left(\frac{\alpha+1}{\alpha}\right) - \frac{1}{\alpha+1}. \tag{5.8}$$

We plot the estimated errors of mutual information against samplesize N, where we set $\alpha = 1$, $\theta_1 = 2$ and $\theta_2 = 5$. The results are shown in Figure 5-5. The four estimators generally perform well for large samplesize, but the errors could be unacceptable compared with the true mutual information for small samplesize. In Figure 5-5a, the BI-KSG and the KSG_1 have similar performance for $N \geqslant 10^2$, which is fairly good. However, the KSG_2 outperforms other estimators in all cases. In Figure 5-5b, all the estimators give better results than those in the first figure. The KSG_2 still outperforms the other estimators for small N, but the BI-KSG and the KSG_1 perform better for medium N.



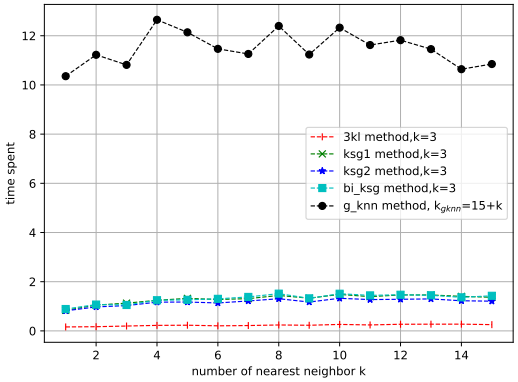
(a) Estimated MI for bivariate Pareto

(b) Estimated MI for bivariate logistic

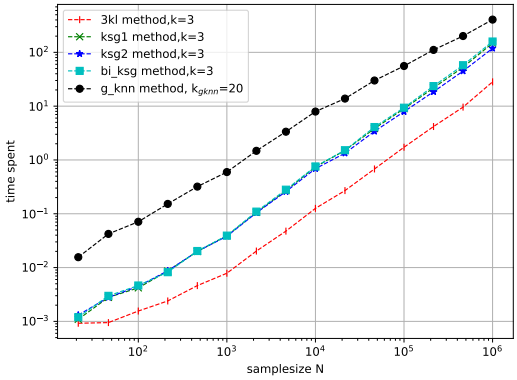
Figure 5-5: Estimated MI of different estimators for Pareto and logistic distributions, where k is fixed at 3.

5.2 Algorithm evaluation

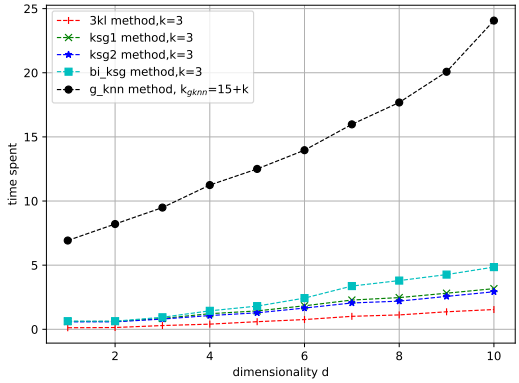
In this section, we evaluate the efficiency of the estimators for variant k, samplesize N and dimensionality d. In Figure 5-6a and Figure 5-6b, we plot the average time spent of one realization against k and samplesize N using the bivariate Gaussian model with $r = 0.3$. In Figure 5-6c, we estimate the average time spent against the dimensionality d using multivariate Gaussian model with random covariance matrices.



(a) Time spent against k



(b) Time spent against samplesize N



(c) Time spent against dimensionality d

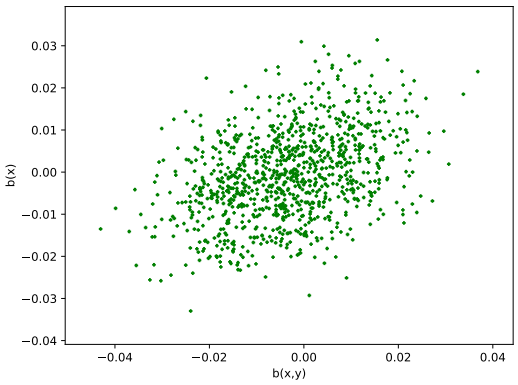
Figure 5-6: Average time spent of one realization of the estimators for different k, N and d, where N is fixed at 5×10^4 in Fig (a) and 10^4 in Fig (c).

Notice that all the estimators spend most of the time in finding the k-nearest neighbour point and querying the distance for each local estimate. The G-knn also spends time on the svd of each k-nearest neighbour point set, which generally runs slowly than other estimators. For increasing values of k, the time spent of the G-knn estimator roughly keeps invariant, while the time spent of other estimators grow slowly. In the Figure 5-6b, the time spent are approximately linear correlated with samplesize,

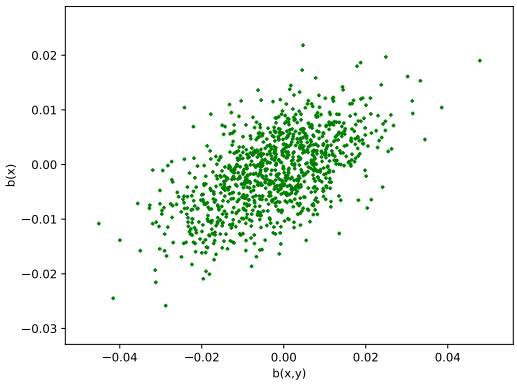
which gives the possibility to efficiently deal with large-size samples. In Figure 5-6c, the G-knn estimator seems quadratic to d due to the extra time spent of svd operation, while the other estimators are still efficient when the dimensionality increases.

5.3 Bias discussion

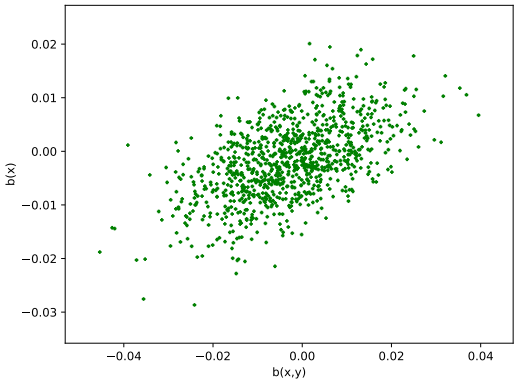
In this section, we focus on the bias of the estimators. To explain the superior performance of the KSG^1 estimator, Gao, Oh and Viswanath [11] proposed the correlation boosting effect in 2016. They believe the bias of the estimators is reduced if the bias of joint entropy is positively correlated with the biases of marginal entropies. Here we simulate 1000 i.i.d. samples from the bivariate Gaussian model with $r = 0.3$ and map the scatter-plot of the biases in Figure 5-7 to verify this hypothesis.



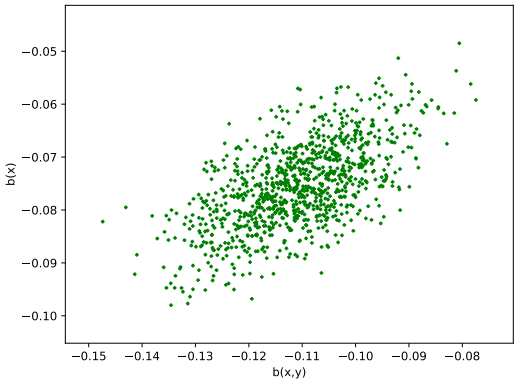
(a) Entropy bias of 3KL estimator



(b) Entropy bias of KSG^1 estimator



(c) Entropy bias of BI-KSG estimator



(d) Entropy bias of G-knn estimator

Figure 5-7: Bias of $\hat{H}(X,Y)$ against bias of $\hat{H}(X)$ of estimators.

As shown in Figure 5-7, the correlations of bias for the KSG^1 and the BI-KSG

estimators are visibly more significant than 3KL estimator, which allows the biases to cancel out when estimating mutual information. To prove the observations above, we tabulate the Pearson correlation coefficients of the biases.

	$(X,Y) \sim \mathcal{N}(0, \Sigma_1)$			
N	1000	2000	5000	10000
3KL	0.3872	0.3711	0.3725	0.3615
KSG^1	0.5805	0.5419	0.5586	0.5556
BI-KSG	0.5889	0.5678	0.5794	0.5672
G-knn	0.6257	0.6248	0.6072	0.6188

Table 5.8: Pearson correlation coefficient $\rho(b(X,Y), b(X))$ of estimators.

As observed in Table 5.8, $\rho(b(X,Y), b(X))$ of the KSG^1 is significantly larger than the 3KL estimator for different samplesize N. Thus, the biases of the three entropies tend to be more cancelled out due to the positive correlations between them, which explains the superior performance of the KSG^1 estimator. In the mean time, $\rho(b(X,Y), b(X))$ of the BI-KSG is slightly larger than the KSG^1, which implies the improvement of bias of the BI-KSG estimator. Notice that Figure 5-7d also shows strong correlation on biases, which explains the better performance of G-knn for highly dependent samples to some extent. However, the G-knn has not been corrected for bias yet. We will discuss more on this and propose an bias-improved G-knn estimator in Chapter 6.

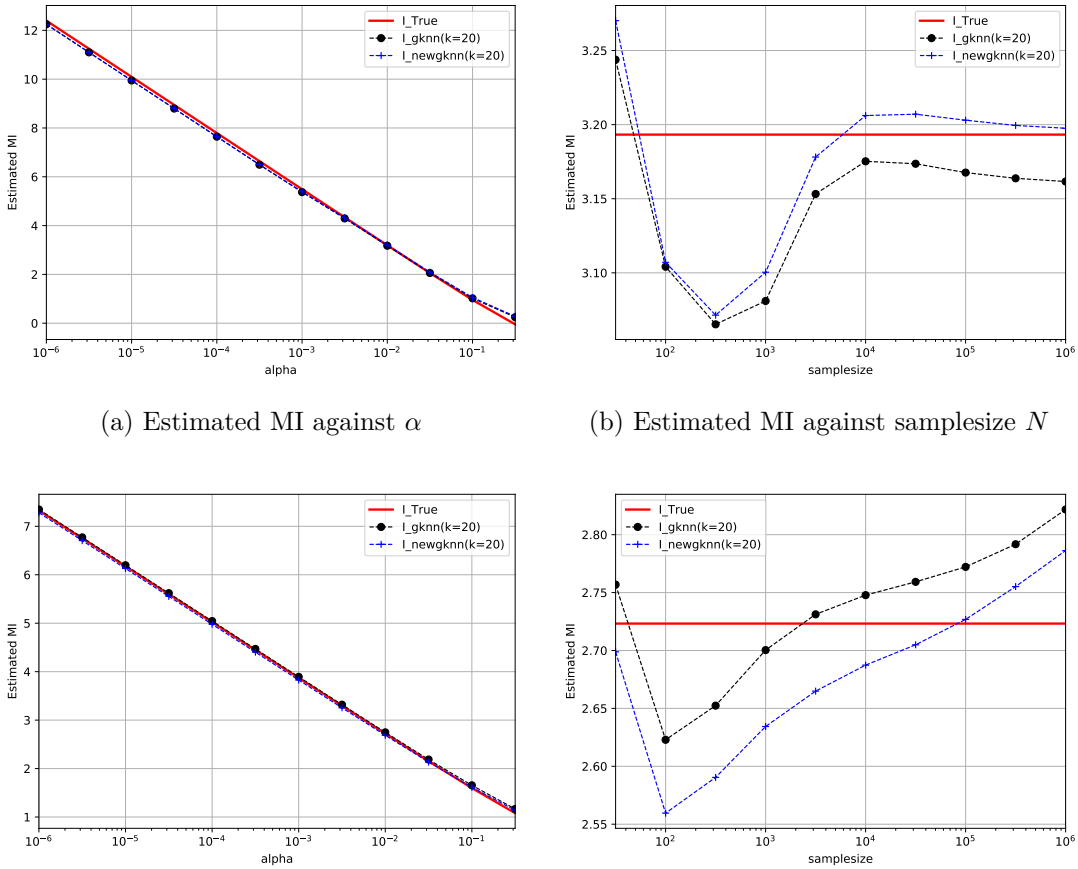
Chapter 6

Original Methods

6.1 Bias-improved G-knn

As we discussed before, the G-knn method raised in [23] is not corrected for bias. Here we propose a bias-improved G-knn and test it on the previous examples. The main idea of the G-knn estimator is using svd of the local data to estimate elliptical local volume elements, which captures the local geometric features. This estimator gives excellent performance for highly dependent samples due to the adaptation of local volume elements. However, the estimation of local density is hard to define in practice. To make sure the neighbourhood contains at least one data point, the G-knn counts all the points inside ellipsoids including the center. In this case, the estimates of local density obtained from Equation (4.10) are actually estimating the conditional local density with the given center point, which introduces bias. Unlike the G-knn, we believe that excluding the center when we are estimating the local densities gives an asymptotically unbiased estimator. When the local volume elements contain no data point excluding the center, we let the estimate of local entropy be 0. This new estimator not only keeps the virtue of G-knn but also corrects the bias. In the rest of this section, we repeat the simulations in Figure 5-3 and Figure 5-4 for the new estimator, and compare it with G-knn estimator. The results are shown in Figure 6-1.

For variant α, the two estimators perform similarly well. However, for increasing N, the BIG-knn estimator outperforms the G-knn. In the first model, the accuracy of BIG-knn is smaller than G-knn when $N \geqslant 10^2$. In the second model, the new estimator performs better than G-knn when $N > 10^4$. Thus, this method is better than G-knn in dealing with large-samplesize and highly dependent samples.



(a) Estimated MI against α

(b) Estimated MI against samplesize N

(c) Estimated MI against α

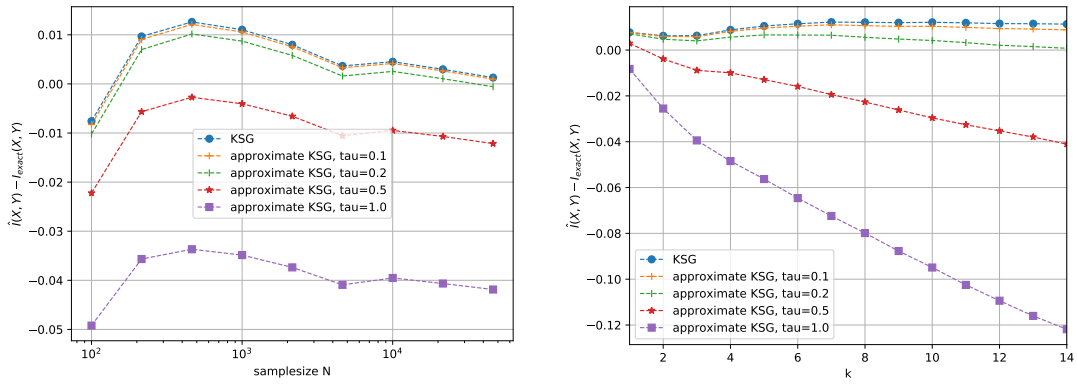
(d) Estimated MI against samplesize N

Figure 6-1: Estimated MI of BIG-knn and G-knn, where N is fixed at 10^4 in Fig (a) and α is fixed at 0.01 in Fig (b).

6.2 Approximate k-NN method

In the previous k-NN estimators, k is fixed for each local estimate. As shown in Section 5.2, algorithms spend most time to find the k-nearest neighbour. For very large samplesize or multiple computations, the efficiency of these algorithms may not satisfy the demands in practice. Thus, we introduce the approximate k-NN method in this section. The main idea of the approximate k-NN method is to find a point x'_k which close to the k^{th} nearest neighbour x_k with distance $\|x'_k - x_k\|_p \leqslant \tau$ in each local estimate, where τ is a fixed tolerance. There are many advanced approximate nearest neighbour search algorithms such as Annoy [6], Efanna [8] and NSG [9], which can be extremely fast in implementations. However, we will not implement them in this report. Instead, we still use k-d tree to simply demonstrate the idea and compare with k-NN estimators in the following experiment. We test this idea on the KSG[1] estimator for

bivariate Gaussian model with $r = 0.9$. the results of the estimated errors compared with the KSG[1] estimator are shown in Figure 6-2a. Since the efficiency improvement is negligible compared with the total time spent in this simulation, we will not plot the time spent of the two estimators.



(a) Errors against N

(b) Errors against k

Figure 6-2: The performance of the approximate KSG estimator compared with the KSG[1].

The raw concept of this method is to lose some accuracy of estimation for the improvement of efficiency especially for large samplesize. As we observed in the above figures, larger τ introduces more bias. However, the bias is slightly improved if we set the tolerance τ to be an appropriate small number, which may be due to the average of local estimates. To prove that the improvement of bias is independent of k, we plot the errors against k in Figure 6-2b. As shown in the results, for appropriate small τ, the bias is improved regardless of k.

Chapter 7

Applications

In this chapter, we firstly develop the estimators to deal with complex samples and apply them into MIMO communication channels. In the first section, we test the different estimators on complex Gaussian channel and compare the estimates with the true mutual information. In the second part, we test the estimators which deal with discrete-continuous samples on the QAM simulations. Then, we apply the estimators on a feature selection example. We also apply Pearson correlation coefficient on the same example and compare the reliability of the results.

7.1 MIMO communication channels

Complex Gaussian channel

In this section, we test the estimators on the complex Gaussian channel model described in Section 2.5. The model is

$$\boldsymbol{y} = H\boldsymbol{x} + \boldsymbol{n}, \tag{7.1}$$

where $x \in \mathbb{C}^2$ is a complex Gaussian random variable with a random Hermitian non-negative definite covariance matrix $Q \in \mathbb{C}^{2\times 2}$ and $\boldsymbol{n}$ is zero-mean complex Gaussian noise with $\mathbb{E}\left[\boldsymbol{n}\boldsymbol{n}^\dagger\right] = I_2$. We let

$$H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + e^{\pi j} \begin{bmatrix} 0 & 0.01 \\ 0.01 & 0 \end{bmatrix}, \tag{7.2}$$

and plot the estimated mutual information against samplesize N. the results are shown in Figure 7-1.

By Equation (2.21), the n-dimensional complex samples can be transformed into $2n$-dimensional real samples. Thus, the estimate of mutual information in complex

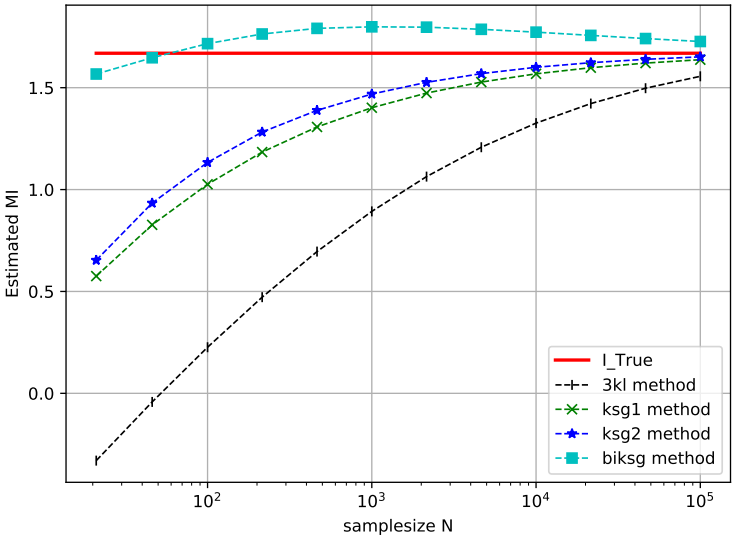


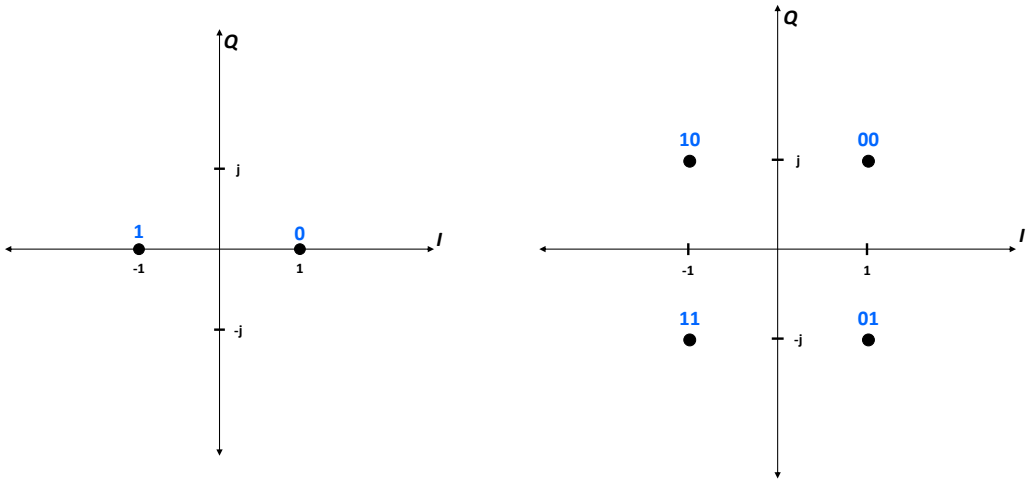
Figure 7-1: The estimated MI of complex Gaussian channel, where k is fixed at 3.

Gaussian channel is actually the estimate of mutual information for corresponding real samples with twice the dimension. All of these estimators are designed to deal with multi-dimensional samples, but the curse of dimensionality is unavoidable. In this example, all the four estimators have serious bias due to the dimensionality. For $N \to \infty$, the KSG^1 and the KSG^2 estimator outperforms the other estimators. For other cases, the only estimator that approaches true solution is the BI-KSG.

Quadrature amplitude modulation (QAM) examples

In this section, we consider the applications of the discrete-continuous estimators for QAM examples. Here we choose three models. The first model is BPSK with the constellation diagram given in Figure 7-2a. The second model is 4-QAM with the constellation diagram given in Figure 7-2b. We choose two different signal power σ_s by scaling the constellation to test the two estimators in the second model. In the simulation, we integrate the true solution of mutual information for the first two models. The third model is 8-QAM with four different constellations given in Figure 7-3. The noises of the three models are generated from zero-mean complex Gaussian distribution with independent, equal variance real and imaginary parts, where we fix the variance to be 0.1. We plot the results in Figure 7-4.

For BPSK example, the Gao's estimator outperforms the Multi-KL estimator for small samplesize. However, the Multi-KL is asymptotically unbiased, which gives better performance for large samplesize. For 4-QAM example, we can observe from the



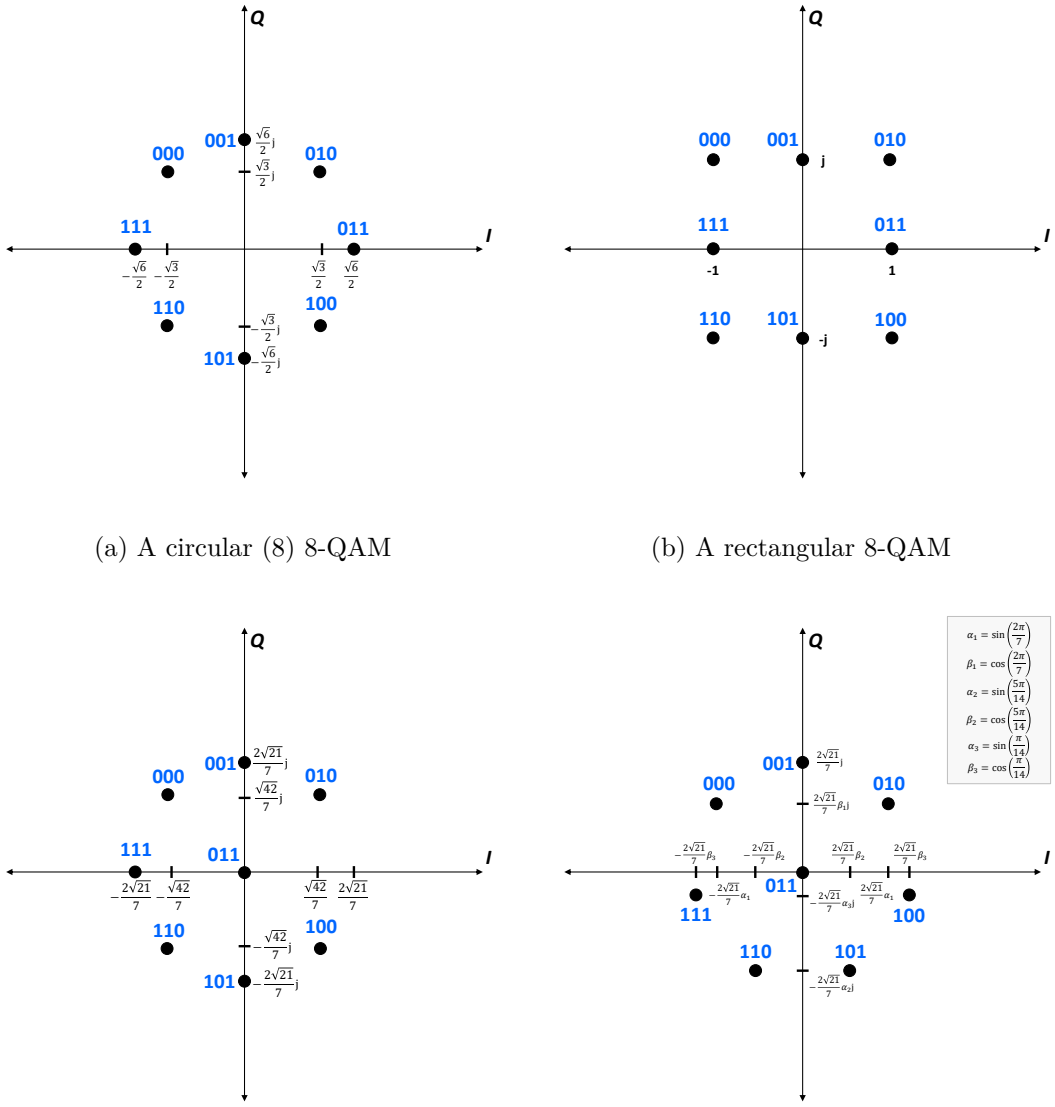
(a) BPSK constellation (b) The 4-QAM constellation

Figure 7-2: The constellations of BPSK and 4-QAM.

true solutions that higher signal power gives higher mutual information. The Gao's estimator gives similar results for two cases, and the results are only close to the first true solution. This may be caused by the sensitivity of k in the Gao's estimator, which makes it inapplicable in practice. In the meanwhile, the Multi-KL performs well on both of them. This superior performance of Multi-KL estimator gives possibility to compare the capacities of communication channel accurately and efficiently under different modulation methods. Thus, we use the Multi-KL estimator in the 8-QAM example to compare channel capacities under the same signal power $\sigma_s^2 = \frac{12}{7}$. As shown in Figure 7-4c, the mutual information of the circular (7,1) 8-QAM is the largest in the four constellations, which indicates larger channel capacity than the other three constellations.

7.2 Feature selection

Feature selection is one of the most important methods of data preprocessing in statistics and machine learning. The main idea is selecting a subset of relevant features to reduce dimensionality and over-fitting. There are many ways to do feature selection. We focus on filter method using an evaluation measure, which is mutual information in our context, to score feature subsets. In this section, we use a simple example to test the performance of our estimators in feature selection. Assuming 100 random variables from independent Gaussian distributions with zero-mean and variance $\sigma = 1$.



(a) A circular (8) 8-QAM

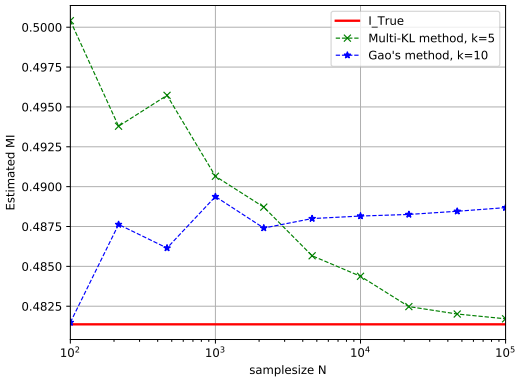
(b) A rectangular 8-QAM

(c) A heterogeneous 8-QAM

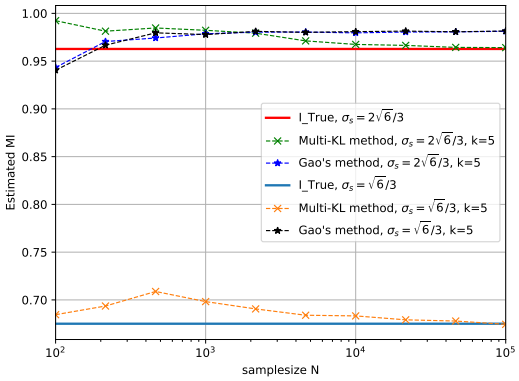
(d) A circular (7,1) 8-QAM

Figure 7-3: The four constellation diagrams of 8-QAM.

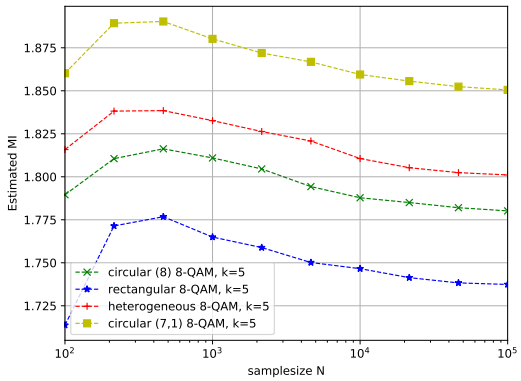
We choose 20 random variables from the 100 random variables and denote them as $\{X_1, X_2, \cdots, X_{20}\}$. Since mutual information, as an evaluation measure, is sensitive to dependence for both linear and non-linear relationships between variables, we create the three dependent variables $Y_1 = \sum_{i=1}^{20} X_i$, $Y_2 = \sum_{i=1}^{10} X_i + \sum_{i=11}^{20} X_i^2$ and $Y_3 = \sum_{i=1}^{20} X_i^2$. We estimate mutual information between the dependent variables and each of the 100 random variables, and rank them by the estimated mutual information. After that, we select features with top-20 highest mutual information. We plot the precision (also



(a) Estimated MI of BPSK



(b) Estimated MI of 4-QAM

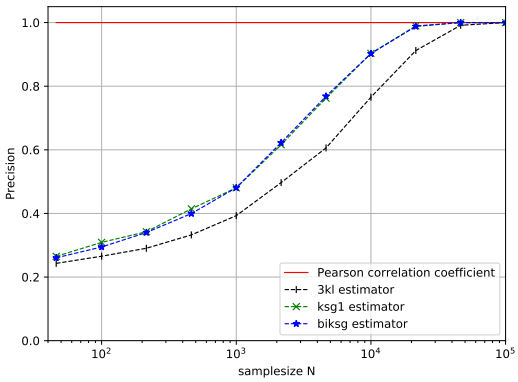


(c) Estimated MI of 8-QAM

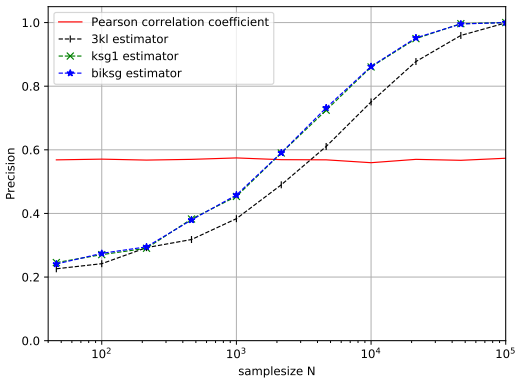
Figure 7-4: The estimated mutual information of QAM examples against samplesize N.

called as positive predictive value) of different estimators against samplesize N and compare with the results obtained by using Pearson correlation coefficient. The results are shown in Figure 7-5. Since the number of features is usually unknown in practice, we select features with top-r highest mutual information, where $r \in \{1, 2, \cdots, 100\}$. After that, we choose y_2 to be the dependent variable and plot the receiver operating characteristic (ROC) curves for different estimators in Figure 7-6.

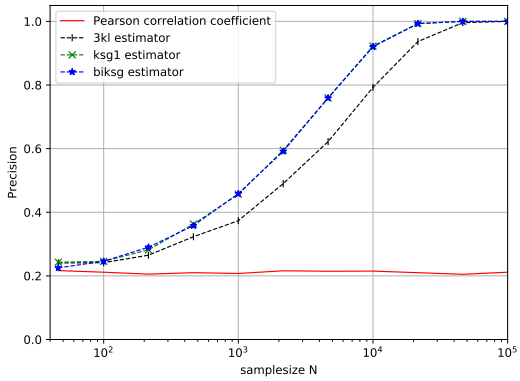
From the results, mutual information estimators give similarly stable performance for the three kinds of dependent variable, while the precision of Pearson correlation coefficient decreases when there are more non-linear relationships added in the simulation. In the ROC curves, it is clear that mutual information estimators give better results than Pearson correlation coefficient. Among the three estimators, the KSG^1 and the BI-KSG outperform the 3KL estimator, since the bias of the KSG^1 and the BI-KSG are smaller than the 3KL.



(a) The precision of estimators for Y_1



(b) The precision of estimators for Y_2



(c) The precision of estimators for Y_3

Figure 7-5: The precision of estimators for different dependent variables, where k is fixed at 3.

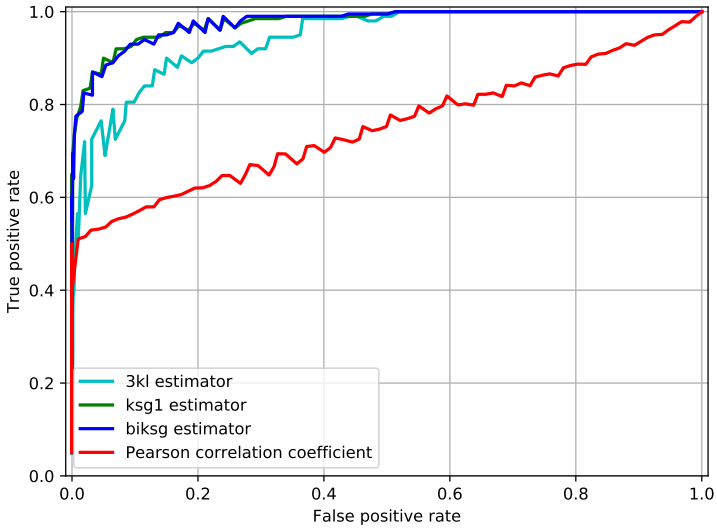


Figure 7-6: The ROC curve of different estimators, where N is fixed at 10^5 and k is fixed at 3.

Chapter 8

Conclusions

This report investigated some numerical estimators of mutual information, proposed new estimators and applied them in applications. The main focus of our research is the k-NN estimators.

We implemented the 3KL, the KSG, the BI-KSG and the G-knn estimators for continuous random variables and tested them on examples. All the four estimators have their own merits. The 3KL estimator is simple and runs fast for variant k, samplesize and dimensionality. However, it has more bias than the KSG and the BI-KSG. The KSG and the BI-KSG perform similarly in different examples. Due to the correlation boosting effect, the BI-KSG has less bias than the KSG estimator theoretically. In numerical experiments, the BI-KSG does outperform the KSG for medium samplesize, but it gives worse results for small samplesize. Since the G-knn estimator is not corrected for bias, we did not compare it with other estimators in bivariate Gaussian, Pareto and logistic examples. For other examples, it shows the superior performance in small samplesize and highly dependent samples, where the other estimators have large bias in these cases. Interestingly, for Pareto and logistic distributions, the 3KL, the KSG and the BI-KSG only converge when samplesize are fairly large, which may be caused by the tail behaviours of the distributions. In efficiency experiments, we observed the time spent of estimators against k, samplesize and dimensionality. As shown in the results, the relationship between time spent and samplesize are close to linear. In the mean time, increasing k does not change much on the time spent of estimators. In all cases, especially for increasing dimensionality, the G-knn is less efficient than others, which may be because of the svd of local datasets.

We proposed the bias-improved G-knn and the approximate k-NN method and tested them on examples. Since the G-knn estimator is not asymptotically unbiased, we considered a slightly different way to estimate local density, which is to exclude the

center when counting points in the neighbourhood. We tested this new estimator on the same examples of G-knn. The results shows that this estimator not only keeps the advantage of G-knn, which is the superior performance on highly dependent samples, but also gives less bias when samplesize is large enough. Inspired by approximate k-NN search algorithms, we proposed the approximate k-NN method. Instead of searching the exact k-nearest neighbour, this method searches a point close to k-nearest neighbour to replace it. We tested this method on the KSG estimator and compared the results. The idea of this estimator is to improve the efficiency by reducing the accuracy. In the simulations, we observed that the accuracy of this new method is also improved under appropriate conditions, which gives the possibility of further studies in this method.

We applied the estimators in MIMO communication channels and feature selection. In QAM examples, we implemented the Gao's estimator and proposed the Multi-KL estimators to deal with discrete-continuous samples. In complex Gaussian channel example, the estimators are actually dealing with high-dimension samples. In this case, the BI-KSG gives close estimates for different samplesize. The KSG still converges to true solution for $N \to \infty$, but it only outperforms the BI-KSG when the samplesize are large enough. In QAM examples, we first tested the two discrete-continuous estimators on the BPSK dataset. As shown in the results, the Gao's estimator outperforms the Multi-KL for small samplesize and the Multi-KL outperforms the Gao's estimator for large samplesize. Then, we tested them on two 4-QAM datasets with different signal power. In this experiment, the Gao's estimator can not identify the two different datasets. Meanwhile, the estimated MI of Multi-KL stay close to true solutions, which indicates the application value of this new estimator. After that, we tested the Multi-KL on 8-QAM with four different constellations to compare the channel capacities. The results showed that the circular (7,1) 8-QAM has largest channel capacity between the four constellations. Finally, we applied the estimators in feature selection. The results shows the stable performance of estimators in given examples compared with Pearson correlation coefficient when the relationship between features and dependent variable is not linear. Among the three estimators, the BI-KSG and the KSG give similar results, but the BI-KSG is slightly more accurate than the KSG.

There are still some unexplored questions. For example, the effects of tail behaviour on the estimates and the theoretical basis of accuracy improvement in the approximate k-NN method are not clear. Future research could discuss more on theoretical parts of them. Another possible research direction is the bias-improved G-knn. Notice that this estimator is still based on 3H-principle, where the bias is not vanished between marginal and joint entropies. The future work could have further improvement of bias in a way similar to the KSG. Additionally, we hope that our new estimators could

spur further interests in applications. For example, the Multi-KL estimator could be applied in gene detection and feature selection, and the approximate k-NN method with advanced approximate k-NN search algorithms may be suitable for problems with large samplesize or high dimensionality.

Appendix A

Proofs

Proof of Theorem 2.2.4

For homeomorphisms $X' = F(X)$ and $Y' = G(Y)$, the Jacobi determinants are $J_X = \|\partial X/\partial X'\|$ and $J_Y = \|\partial Y/\partial Y'\|$. Then, we have $\mu'(x') = J_X(x')\mu(x)$ and $\mu'(x', y') = J_X(x')J_Y(y')\mu(x, y)$. The marginal entropy of X' is

$$\begin{aligned} H(X') &= -\int dx' \mu'_x(x') \log(\mu'_x(x')) \\ &= -\int dx \mu_x(x) \log(J_X(x')\mu_x(x)) \\ &= H(X) - \int dx \mu_x(x) \log(J_X(x')) \\ &= H(X) - \log(J_X(x')). \end{aligned} \tag{A.1}$$

The joint entropy of (X', Y') is

$$\begin{aligned} H(X', Y') &= -\int\int dx' dy' \mu'(x', y') \log(\mu'(x', y')) \\ &= -\int\int dx dy \mu(x, y) \log(J_X(x')J_Y(y')\mu(x, y)) \\ &= H(X, Y) - \int\int dx dy \mu(x, y) \log(J_X(x')J_Y(y')) \\ &= H(X, Y) - \log(J_X(x')J_Y(y')). \end{aligned} \tag{A.2}$$

The mutual information $I(X', Y')$ is derived by Definition 2.2.1:

$$\begin{aligned} I(X', Y') &= H(X') + H(Y') - H(X', Y') \\ &= H(X) - \log(J_X(x')) + H(Y) - \log(J_Y(y')) - H(X, Y) - \log(J_X(x')J_Y(y')) \\ &= H(X) + H(Y) - H(X, Y) \\ &= I(X, Y). \end{aligned} \tag{A.3}$$

Thus, mutual information is invariant under reparametizations.

Proof of Lemma 2.4.2

Consider the range of a random variable X with density $f(x)$ is divided into n small intervals of length Δ_1. Define the quantized random variable $X^{\Delta_1} = x_i,\ if\ i\Delta_1 \leqslant X < (i+1)\Delta_1$, we can obtain that:

$$\begin{aligned} H(X^{\Delta_1}, Y^{\Delta_2}) &= -\sum_i \sum_j p(x_i, y_j) \log p(x_i, y_j) \\ &= -\sum_i \sum_j p_y(y_j) f_j(x_i) \Delta_1 \log p_y(y_j) f_j(x_i) \Delta_1 \\ &= -\sum_i \sum_j p_y(y_j) f_j(x_i) \Delta_1 \log p_y(y_j) f_j(x_i) \\ &\quad -\sum_i \sum_j p_y(y_j) f_j(x_i) \Delta_1 \log \Delta_1. \end{aligned} \tag{A.4}$$

Since

$$\sum_j \sum_i p_y(y_j) f_j(x_i) \Delta_1 = 1, \tag{A.5}$$

we have

$$\begin{aligned} &\lim_{\Delta_1 \to 0} -\sum_i \sum_j p_y(y_j) f_j(x_i) \Delta_1 \log p_y(y_j) f_j(x_i) \Delta_1 \\ &\quad = -\sum_j \int_x p_y(y_j) f_j(x) \log p_y(y_j) f_j(x) dx - \log \Delta_1 \\ &\quad =: H(X, Y^{\Delta_2}) - \log \Delta_1. \end{aligned} \tag{A.6}$$

Proof of Lemma 2.5.4

For Hermitian non-negative definite $Q \in \mathbb{C}^{n\times n}$ and $\boldsymbol{x} \neq 0$, we have $\boldsymbol{x}^\dagger Q\boldsymbol{x} \geqslant 0$. If $Q\boldsymbol{x} = \lambda\boldsymbol{x}$, then

$$\boldsymbol{x}^\dagger Q\boldsymbol{x} = \boldsymbol{x}^\dagger \lambda\boldsymbol{x} = \lambda\boldsymbol{x}^\dagger\boldsymbol{x}. \tag{A.7}$$

Thus, we obtain

$$\lambda = \frac{\boldsymbol{x}^\dagger Q\boldsymbol{x}}{\boldsymbol{x}^\dagger\boldsymbol{x}} \geqslant 0 \tag{A.8}$$

and

$$\det(Q) = \prod_i^n \lambda_i \geqslant 0. \tag{A.9}$$

Proof of Lemma 2.5.6

For Hermitian non-negative definite $Q \in \mathbb{C}^{n\times n}$ and $\boldsymbol{x} \neq 0$, Q can be decomposed into $B^\dagger B$, where $B \in \mathbb{C}^{n\times n}$. Next, we have

$$\begin{aligned} \boldsymbol{x}^\dagger AQA^\dagger\boldsymbol{x} &= (A^\dagger\boldsymbol{x})^\dagger QA^\dagger\boldsymbol{x} \\ &= (A^\dagger\boldsymbol{x})^\dagger B^\dagger BA^\dagger\boldsymbol{x} \\ &= (BA^\dagger\boldsymbol{x})^\dagger(BA^\dagger\boldsymbol{x}) \geqslant 0 \end{aligned} \tag{A.10}$$

and

$$\det(AQA^\dagger) \geqslant 0. \tag{A.11}$$

Thus, $AQA^\dagger$ is also a Hermitian non-negative definite matrix.

Proof of Equation (5.2)

Assuming $X \sim \mathcal{N}(\mu_x, \Sigma_X)$, we can obtain the entropy of X [2]:

$$\begin{aligned} H(X) &= -\int dx f(x) \log f(x) \\ &= -\int f(x) \left(-\frac{1}{2}(x-\mu_x)^T \Sigma_X^{-1}(x-\mu_x) - \frac{1}{2}\log\left((2\pi)^{d_x}|\Sigma_X|\right)\right) dx \\ &= \frac{1}{2}\mathbb{E}\Big[(x-\mu_x)^T \Sigma_X^{-1}(x-\mu_x)\Big] + \frac{1}{2}\log(2\pi)^{d_x}|\Sigma_X| \\ &= \frac{1}{2}\mathbb{E}\Big[(x-\mu_x)^T (x-\mu_x)\Big]\Sigma_X^{-1} + \frac{1}{2}\log(2\pi)^{d_x}|\Sigma_X| \\ &= \frac{1}{2}\Sigma_X\Sigma_X^{-1} + \frac{1}{2}\log(2\pi)^{d_x}|\Sigma_X| \\ &= \frac{d_x}{2} + \frac{1}{2}\log(2\pi)^{d_x}|\Sigma_X| = \frac{1}{2}\log(2\pi e)^{d_x}|\Sigma_X|. \end{aligned} \tag{A.12}$$

Assuming $(X, Y) \sim \mathcal{N}(\mu, \Sigma)$, similarly we have $H(X,Y) = \frac{1}{2}\log(2\pi e)^{d_{x,y}}|\Sigma|$. Thus, the exact mutual information of (X, Y) is given as:

$$I(X,Y) = H(X) + H(Y) - H(X,Y) = \frac{1}{2}\log\left(\frac{|\Sigma_X||\Sigma_Y|}{|\Sigma|}\right). \tag{A.13}$$

Appendix B

Computer code

Methods collection

This python file collect all the estimators in this report.

MI/Methods_Collection.py

```
# Copyright (C) 2018 Haoran Ni & Keith Briggs

'''Collect␣all␣the␣methods␣in␣this␣module'''

import numpy as np
from scipy.spatial import cKDTree,KDTree
from scipy.special import gamma,digamma
from math import log,sqrt
from scipy.special import psi

# KL entropy
def KL_h(samples, n,d, k, norm):
  if norm == np.inf:
    log_c_d = 0
  elif norm == 2:
    log_c_d = (d/2.) * np.log(np.pi) -np.log(gamma(d/2. +1))
  elif norm == 1:
    raise NotImplementedError
  else:
    raise NotImplementedError("Variable␣'norm'␣either␣1,␣2␣or␣np.inf")
  kdtree = cKDTree(samples)
  distances, idx = kdtree.query(samples, k + 1, eps=0, p=norm)
  return -digamma(k) + digamma(n) + log_c_d + (d / n) * np.sum(np.log(2*
      distances[:, -1]))

# 3KL estimator
```

```
def ThreeKL_MI(samples,k,norm):
  n,d=samples.shape
  return KL_h(samples[:,:d//2],n,d//2,  k=k, norm=norm)+KL_h(samples[:,d
      //2:],n,d//2,  k=k, norm=norm)-KL_h(samples,n,d, k=k, norm=norm)

# KSG1 estimator
def KSG1_MI(samples,k,norm):
  n,d=samples.shape
  x_tree  = cKDTree(samples[:,:d//2])
  y_tree  = cKDTree(samples[:,d//2:])
  xy_tree = cKDTree(samples)
  dist, idx = xy_tree.query(samples, k=k+1, p=norm)
  epsilon = dist[:, -1]-(1e-15)
  nx = np.empty(n, dtype=np.int)
  ny = np.empty(n, dtype=np.int)
  for ii in range(n):
    nx[ii] = len(x_tree.query_ball_point(x_tree.data[ii], r=epsilon[ii],
        p=norm)) - 1
    ny[ii] = len(y_tree.query_ball_point(y_tree.data[ii], r=epsilon[ii],
        p=norm)) - 1
  return digamma(k) - np.mean(digamma(nx+1) + digamma(ny+1)) + digamma(n
      )  # Alg.1

# KSG2 estimator
def KSG2_MI(samples,k,norm):
  n,d=samples.shape
  x_tree  = cKDTree(samples[:,:d//2])
  y_tree  = cKDTree(samples[:,d//2:])
  xy_tree = cKDTree(samples)
  dist, idx = xy_tree.query(samples, k=k+1, p=norm)
  epsilon1=np.linalg.norm((samples[:,:d//2]-samples[idx[:,k],:d//2]),
      norm,axis=1)
  epsilon2=np.linalg.norm((samples[:,d//2:]-samples[idx[:,k],d//2:]),
      norm,axis=1)
  nx = np.empty(n, dtype=np.int)
  ny = np.empty(n, dtype=np.int)
  for ii in range(n):
    nx[ii] = len(x_tree.query_ball_point(x_tree.data[ii], r=epsilon1[ii
        ], p=norm)) - 1
    ny[ii] = len(y_tree.query_ball_point(y_tree.data[ii], r=epsilon2[ii
        ], p=norm)) - 1
  return digamma(k) -1.0 -np.mean(digamma(nx) + digamma(ny)) + digamma(n
      ) # Alg.2

# Bias-improved KSG estimator
def BIKSG_MI(samples,k,norm):
```

```
  n,d=samples.shape
  LogVol=np.log(gamma(d/2+1)/(gamma(d/4+1)*gamma(d/4+1)))
  x_tree  = cKDTree(samples[:,:d//2])
  y_tree  = cKDTree(samples[:,d//2:])
  xy_tree = cKDTree(samples)
  dist, idx = xy_tree.query(samples, k=k+1, p=norm)
  epsilon = dist[:, -1]
  nx = np.empty(n, dtype=np.int)
  ny = np.empty(n, dtype=np.int)
  for ii in range(n):
    nx[ii] = len(x_tree.query_ball_point(x_tree.data[ii], r=epsilon[ii],
        p=norm)) - 1
    ny[ii] = len(y_tree.query_ball_point(y_tree.data[ii], r=epsilon[ii],
        p=norm)) - 1
  return digamma(k)+np.log(n)+LogVol-np.mean(np.log(nx)+np.log(ny))

# Gknn entropy
def Gknn_h(samples,n,d,k):
  x_tree=cKDTree(samples)
  dist,idx=x_tree.query(samples,k=k+1,p=2)
  term4=0.0
  nx=np.empty(n,dtype=np.int)
  for ii in range(n):
    samplex=samples[idx[ii,:],:]
    u, s, vh = np.linalg.svd(samplex-np.mean(samplex,axis=0),
       full_matrices=True)
    s=s/s[0]*dist[ii, -1]
    term4+=np.sum(np.log(s))
    samplex=(samples[idx[ii,:],:]-samples[ii,:])
    ellip=np.einsum('ij,jm,km->ik',np.diag(1.0/s),vh,samplex)
    ellip=np.einsum('ji,ji->i',ellip,ellip)
    nx[ii]=len(ellip[ellip<=1])
  return np.log(n)+np.log(np.pi**(d/2.0)/gamma(d/2.0+1.0))-np.mean(np.
     log(nx))+term4/n

# Gknn estimator
def Gknn_MI(samples,k):
  n,d=samples.shape
  return Gknn_h(samples[:,:d//2],n,d//2,  k=k)+Gknn_h(samples[:,d//2:],n
     ,d//2,  k=k)-Gknn_h(samples,n,d, k=k)

# Bias-improved Gknn entropy
def BIGknn_h(samples,n,d,k):
  x_tree=cKDTree(samples)
  dist,idx=x_tree.query(samples,k=k+1,p=2)
  term4=0.0
```

```
  nx=np.empty(n,dtype=np.int)
  for ii in range(n):
    samplex=samples[idx[ii,:],:]
    u, s, vh = np.linalg.svd(samplex-np.mean(samplex,axis=0),
        full_matrices=True)
    s=s/s[0]*dist[ii, -1]
    term4+=np.sum(np.log(s))
    samplex=(samples[idx[ii,:],:]-samples[ii,:])
    ellip=np.einsum('ij,jm,km->ik',np.diag(1.0/s),vh,samplex)
    ellip=np.einsum('ji,ji->i',ellip,ellip)
    nx[ii]=len(ellip[ellip<=1])-1
    if nx[ii]==0:
      term4-=0.0
    else:
      term4-=np.log(nx[ii])
  return np.log(n)+np.log(np.pi**(d/2.0)/gamma(d/2.0+1.0))+term4/n

# Bias-improved Gknn estimator
def BIGknn_MI(samples,k):
  n,d=samples.shape
  return Gknnnew_h(samples[:,:d//2],n,d//2,  k=k)+Gknnnew_h(samples[:,d
      //2:],n,d//2,  k=k)-Gknnnew_h(samples,n,d, k=k)

# Approximate KSG1 estimator
def ApproximateKSG1_MI(samples,k,norm,eps=0.1):
  n,d=samples.shape
  x_tree  = cKDTree(samples[:,:d//2])
  y_tree  = cKDTree(samples[:,d//2:])
  xy_tree = cKDTree(samples)
  dist, idx = xy_tree.query(samples, k=k+1,eps=eps, p=norm)
  epsilon = dist[:, -1]-(1e-15)
  nx = np.empty(n, dtype=np.int)
  ny = np.empty(n, dtype=np.int)
  for ii in range(n):
    nx[ii] = len(x_tree.query_ball_point(x_tree.data[ii], r=epsilon[ii],
        p=norm)) - 1
    ny[ii] = len(y_tree.query_ball_point(y_tree.data[ii], r=epsilon[ii],
        p=norm)) - 1
  return digamma(k) - np.mean(digamma(nx+1) + digamma(ny+1)) + digamma(n
      )  # Alg.1

# Multi-KL estimator
def Multi_KL(samples,possiblex,k=20):
  n,d=samples.shape
  dcmi=KL_h(samples[:,d//2:],n,d//2,k,norm=np.inf)
  nnx=possiblex.shape[0]
```

```
  for jj in range(nnx):
    samplex=samples[np.all((samples[:,:d//2]==possiblex[jj,:]),axis=1)]
    nx=samplex.shape[0]
    p=nx/n
    dcmi-=p*KL_h(samplex[:,d//2:],nx,d//2,k,norm=np.inf)
  return dcmi

# Gao's estimator
def Gao_Mixed_MI(samples,k):
  n,d=samples.shape
  xy_tree=cKDTree(samples)
  x_tree=cKDTree(samples[:,:d//2])
  y_tree=cKDTree(samples[:,d//2:])
  num=np.empty(n,dtype=np.int)
  numx=np.empty(n,dtype=np.int)
  numy=np.empty(n,dtype=np.int)
  for ii in range(n):
    dx,ix=x_tree.query(x_tree.data[ii],k=k+1)
    dy,iy=y_tree.query(y_tree.data[ii],k=k+1)
    l=max(dx[-1],dy[-1])
    if l==0.0:
      num[ii] = len(xy_tree.query_ball_point(xy_tree.data[ii], r=0.0))
    else:
      num[ii]=k
    numx[ii] = len(x_tree.query_ball_point(x_tree.data[ii], r=l))
    numy[ii] = len(y_tree.query_ball_point(y_tree.data[ii], r=l))
  return np.mean(digamma(num))+np.log(n)-np.mean(np.log(numx)+np.log(
     numy))
```

Exact mutual information

This python file collect the calculation of exact mutual information in this report.

MI/Exact_Solutions.py

```
# Copyright (C) 2018 Haoran Ni & Keith Briggs

'''Collect␣all␣the␣exact␣solutions␣in␣this␣module'''

import scipy.integrate as integrate
import scipy.special as special
import numpy as np
from scipy.special import digamma
from scipy.stats import multivariate_normal
```

```
# Class for collecting exact solution functions
class Exact:
  def __init__(s, value=None):
    s.value=value

# Compute the MI of uniform with noise model
  def unif_noise(s,alpha):
    yxx= lambda y:special.ndtr(1-y/alpha)-special.ndtr(-y/alpha)
    fxx=integrate.quad(lambda y:-yxx(y)*np.log(yxx(y)),-alpha,alpha)
    return -np.log(alpha)+fxx[0]-0.5*np.log(2.0*np.pi*np.e)

# Compute the MI of multivariate Gaussian model
  def Multi_Gaussian(s,cov):
    n=cov.shape[0]
    return 0.5*np.log(np.linalg.det(cov[0:n//2,0:n//2])*np.linalg.det(
        cov[n//2:n,n//2:n])/np.linalg.det(cov))

# Compute the MI of the 2d pareto and logistic distribution
  def Pareto_logistic_2d(s,a):
    return np.log(1.0+1.0/a)-1.0/(a+1.0)

# Compute the MI of QAM examples
  # BPSK model
  def BPSK(s,mu,sigma,pb):
    n=sigma.shape[0]
    d=sigma.shape[2]
    dets=np.array([np.linalg.det(sigma[i,:,:]) for i in range(n)])
    entropy1=np.sum(pb*0.5*np.log((2*np.pi*np.e)**d*dets))
    func=lambda y,x: -integrand(x,y,mu,sigma,pb,n)*np.log(integrand(x,y,
        mu,sigma,pb,n))
    evals=np.vstack([np.linalg.eigvals(sigma[0]),np.linalg.eigvals(sigma
        [1])])
    a,b=-10*np.max(np.abs(evals)),10*np.max(np.abs(evals))
    entropy2=integrate.dblquad(func,a,b,lambda x: a, lambda x: b)
    return entropy2[0]-entropy1,entropy2[1]

  # 4-QAM model
  def dc_4QAM(s,mu,sigma,pb):
    n=sigma.shape[0]
    d=sigma.shape[2]
    dets=np.array([np.linalg.det(sigma[i,:,:]) for i in range(n)])
    entropy1=np.sum(pb*0.5*np.log((2*np.pi*np.e)**d*dets))
    func=lambda y,x: -integrand(x,y,mu,sigma,pb,n)*np.log(integrand(x,y,
        mu,sigma,pb,n))
    evals=np.vstack([np.linalg.eigvals(sigma[0]),np.linalg.eigvals(sigma
        [1]),np.linalg.eigvals(sigm[2]),np.linalg.eigvals(sigma[3])])
```

```
    a,b=-10*np.max(np.abs(evals)),10*np.max(np.abs(evals))
    entropy2=integrate.dblquad(func,a,b,lambda x: a, lambda x: b)
    return entropy2[0]-entropy1,entropy2[1]

# Compute the MI of Complex gaussian channel
  def cpx_gaussian(s,H,Q):
    I_t=np.zeros(Q.shape)
    for ii in range(len(H[0])):
      I_t[ii,ii]=1.0
    return np.log(np.linalg.det(I_t+np.einsum('ij,kj,km->im',Q,np.conj(H
        ),H))).real

# Integrating function
def integrand(x,y,mu,sigma,pb,n):
  integrand=0.0
  for ii in range(n):
    integrand+=pb[ii]*multivariate_normal.pdf([x,y],mean=mu[ii,:], cov=
        sigma[ii,:,:])
    return integrand
```

Data Generator

This python file collect the all data generators in this report.

MI/Data_Generator.py

```
# Copyright (C) 2018 Haoran Ni & Keith Briggs

'''Collect␣all␣data␣generators␣in␣this␣module'''

import numpy as np

class Data_Generator:
  '␣Generate␣a␣sample␣from␣any␣distribution␣'
  def __init__(s,distribution,**kwargs):
    s.distribution=distribution
    s.kwargs=kwargs
  def sample(s,size):
    return s.distribution(size=size,**s.kwargs)

# Sum of uniform and normal distributions sample generator
def sum_of_unif_stdnorm(size,alpha):
  sample=np.random.random(size)
  return np.c_[sample,np.dot(np.vstack((sample,np.random.random(size),np
      .random.standard_normal(size))).T,np.concatenate(([1],alpha), axis
```

```
      =0))]

# Pareto 2d sample generator
def rpareto_inv(size,theta,a):
  return theta/np.random.random(size)**(1.0/a)

def rpareto_cond_inv(x2,theta,a):
  u=np.random.random(len(x2))
  return theta[0]+theta[0]/theta[1]*x2*(1.0/(u**(1.0/(a+1.0)))-1.0)

def pareto_2d_sample(size,theta,a):
  x2=rpareto_inv(size,theta[1],a)
  x1=rpareto_cond_inv(x2,theta,a)
  return np.vstack((x1,x2)).T

def logistic_2d_sample(size,theta,mu,a):
  u=np.random.random(size)**(-1.0/a)
  x1=mu[0]-np.log(u-1.0)*theta[0]
  x2=mu[1]-np.log(np.random.random(size)**(-0.5/a)*u-u)*theta[1]
  return np.vstack((x1,x2)).T

# Logistic 2d sample generator
def burr_2d_sample(size,d,c,a):
  u=np.random.random(size)**(-1.0/a)
  x1=((u-1.0)/d[0])**(1.0/c[0])
  x2=((np.random.random(size)**(-1.0/a)*u-u)/d[1])**(1.0/c[1])
  return np.vstack((x1,x2)).T

# Complex Gaussian data generator
def mapping_vec(z):
  return np.c_[z.real,z.imag]

def mapping_mat(A):
  return np.c_[np.r_[A.real,A.imag],np.r_[-A.imag,A.real]]

def remapping(z):
  return z[:,:len(z[0])//2]+z[:,len(z[0])//2:]*1j

def complex_gaussian(size,mean,H,Q):
  H_hat=mapping_mat(H)
  mean_hat=np.r_[mean.real,mean.imag]
  cov=0.5*mapping_mat(Q)
  X=np.random.multivariate_normal(size=size,mean=mean_hat,cov=cov)
  I_r=np.eye(len(H),dtype='float')
  cov_n=0.5*mapping_mat(I_r)
  mean_n=np.zeros(len(H_hat),dtype='float')
```

```
  n=np.random.multivariate_normal(size=size,mean=mean_n,cov=cov_n)
  Y=np.empty(X.shape)
  for ii in range(size):
    Y[ii,:]=np.einsum('ij,j->i',H_hat,X[ii,:])+n[ii,:]
  return np.c_[X,Y]

# QAM samples generator
def sampling_QAM(size,pb,possiblex,sigma):
  samples=np.empty((size,4),dtype='float')
  n=possiblex.shape[0]
  x1=np.random.choice(n, size, p=pb)
  for ii in range(size):
    for jj in range(n):
      if x1[ii]==jj:
        samples[ii,:2]=possiblex[jj,:]
        samples[ii,2:]=np.random.multivariate_normal(possiblex[jj,:],
            sigma[jj,:,:],1)
  return samples

# Feature selection sample generator
def feature_selection(size,numx,num_feature,mu,sigma):
  x=np.empty((size,numx))
  y=np.zeros((1,size))
  for ii in range(numx):
    x[:,ii]=np.random.normal(loc=mu,scale=sigma,size=size)
  for jj in range(num_feature//2):
    y+=x[:,5*jj]**2
  for jj in range(num_feature//2):
    y+=x[:,5*(jj+num_feature//2)]**2
```

Bibliography

[1] J. L. Bentley. "Multidimensional Binary Search Trees Used for Associative Searching". In: 18.9 (1975), pp. 509–517.

[2] T. Cover and J. Thomas. *Elements of information theory*. 2nd ed. Hoboken N. J.: Wiley-Interscience, 2006.

[3] L. Cui, D. Pi, and C. Wang. "Topic Discovery Algorithm Based on Mutual Information and Label Clustering under Dynamic Social Networks". In: *International Journal of Database Theory and Application* 9 (2016), pp. 169–180.

[4] G. A. Darbellay and I. Vajda. "Estimation of the Information by an Adaptive Partitioning of the Observation Space". In: *IEEE Transactions on Information Theory* 45.4 (1999).

[5] G. A. Darbellay and I. Vajda. "Entropy Expressions for Multivariate Continuous Distributions". In: *IEEE Transactions on Information Theory* 46.2 (2000).

[6] S. Dasgupta and Y. Freund. "Random Projection Trees and Low Dimensional Manifolds". In: *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*. STOC '08. Victoria, British Columbia, Canada: ACM, 2008, pp. 537–546. DOI: 10.1145/1374376.1374452. URL: http://doi.acm.org/10.1145/1374376.1374452.

[7] P. A. Estevez et al. "Normalized Mutual Information Feature Selection". In: *IEEE Transactions on Neural Networks* 20.2 (2009), pp. 189–201.

[8] C. Fu and D. Cai. "EFANNA: An Extremely Fast Approximate Nearest Neighbor Search Algorithm Based on kNN Graph". In: *CoRR* abs/1609.07228 (2016). arXiv: 1609.07228. URL: http://arxiv.org/abs/1609.07228.

[9] C. Fu, C. Wang, and D. Cai. "Fast Approximate Nearest Neighbor Search With Navigating Spreading-out Graphs". In: *CoRR* abs/1707.00143 (2017).

[10] S. Gao, G. Ver Steeg, and A. Galstyan. "Efficient Estimation of Mutual Information for Strongly Dependent Variables". In: *Artificial intelligence and statistics* (2015), pp. 277–286.

[11] W. Gao, S. Oh, and P. Viswanath. "Demystifying fixed k-nearest neighbor information estimators". In: *IEEE International Symposium on Information Theory (ISIT)* (2017), pp. 1267–1271.

[12] W. Gao et al. "Estimating Mutual Information for Discrete-Continuous Mixtures". In: *Advances in Neural Information Processing Systems 30* (2017), pp. 5986–5997.

[13] I. Grosse et al. "Analysis of symbolic sequences using the Jensen-Shannon divergence". In: *Phys. Rev. E* 65 (2002).

[14] X. Guo, H. Zhang, and T. Tian. "Development of stock correlation networks using mutual information and financial big data". In: *PLOS ONE* 13.4 (Apr. 2018), pp. 1–16. DOI: 10.1371/journal.pone.0195941. URL: https://doi.org/10.1371/journal.pone.0195941.

[15] P. Hausladen et al. "Classical information capacity of a quantum channel". In: *Phys. Rev. A* 54 (3 Sept. 1996), pp. 1869–1876. DOI: 10.1103/PhysRevA.54.1869. URL: https://link.aps.org/doi/10.1103/PhysRevA.54.1869.

[16] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. 5th ed. Hoboken N. J.: John Wiley & Sons, Inc., 2013.

[17] J. B. Kinney and G. S. Atwal. "Equitability, mutual information, and the maximal information coefficient". In: *Proceedings of the National Academy of Sciences of the United States of America* 111.9 (2014), pp. 3354–3359.

[18] L. F. Kozachenko and N. N. Leonenko. "Sample Estimate of the Entropy of a Random Vector". In: *Probl. Peredachi Inf.* 23.2 (1987), pp. 95–101.

[19] A. Kraskov, H. Stögbauer, and P. Grassberger. "Estimating mutual information". In: *Phys. Rev. E* 69.6 (2004).

[20] S. Kullback and R. A. Leibler. "On information and sufficiency". In: *Ann. Math. Statist.* 22.1 (1951), pp. 79–86.

[21] N. Kwak and C. Choi. "Input feature selection by mutual information based on Parzen window". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.12 (2002), pp. 1667–1671.

[22] H. Liu et al. "Feature selection with dynamic mutual information". In: *Pattern Recognition* 42.7 (2009), pp. 1330–1339.

[23] W. M. Lord, J. Sun, and E. M. Bollt. "Geometric k-nearest neighbor estimation of entropy and mutual information". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 28.3 (2018).

[24] M. Madiman. "On the entropy of sums". In: *IEEE Information Theory Workshop* (2008), pp. 303–307.

[25] Y. I. Moon, B. Rajagopalan, and U. Lall. "Estimation of mutual information using kernel density estimators". In: *Phys. Rev. E* 52.3 (1995).

[26] H. Peng, F. Long, and C. Ding. "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.8 (2005), pp. 1226–1238.

[27] F. Pérez-Cruz. "Kullback-Leibler Divergence Estimation of Continuous Distributions". In: *ISIT* (2008).

[28] B. C. Ross. "Mutual Information between Discrete and Continuous Data Sets". In: *PLOS ONE* 9.2 (Feb. 2014), pp. 1–5. DOI: 10.1371/journal.pone.0087357. URL: https://doi.org/10.1371/journal.pone.0087357.

[29] H. L. Royden. *Real analysis.* 4th ed. Boston, Mass.: Prentice Hall, 2010, pp. 381–385.

[30] H. Shakibian and N. M. Charkari. "Mutual information model for link prediction in hetergeneous complex networks". In: *Scientific Reports* 9.5 (2017), pp. 169–180.

[31] C. E. Shannon. "A Mathematical Theory of Communication". In: *The Bell System Technical Journal* 27 (1948), pp. 379–423, 623–656.

[32] R. Steuer et al. "The mutual information: Detecting and evaluating dependencies between variables". In: *BIOINFORMATICS* 18.Suppl. 2 (2002), S231–S240.

[33] I. E. Telatar and D. N. C. Tse. "Capacity and Mutual Information of Wideband Multipath Fading Channels". In: *IEEE TRANSACTIONS ON INFORMATION THEORY* 46.4 (2000), pp. 1384–1400.

[34] I. Emre Telatar. "Capacity of multi-antenna Gaussian channels". In: *European Transactions on Telecommunications* 10 (1999), pp. 585–595.

[35] Q. Wang, S. A. KUlkarni, and S. Verdú. "Divergence Estimation of Continuous Distributions Based on Data-Dependent Partitions". In: *IEEE Transactions on Information Theory* 51.9 (2005).

[36] X. Wang and X. Hui. “Mutual Information Based Analysis for the Distribution of Financial Contagion in Stock Markets”. In: *Discrete Dynamics in Nature and Society* 2017.3218042 (2017), pp. 1–13.