Recurrent Neural Network for Estimating Mutual Information of a Gaussian Process

Anders Lauridsen, Isabella Quillo, Jacob Mørk, Jakob Olsen, and Martin Sørensen
7th Semester of Mathematical Engineering at Aalborg University
{ahll19, iquill, jmark18, jols19, msar18}@student.aau.dk

***Abstract*—Estimating mutual information is a long-standing problem in statistics and machine learning. The current state-of-the-art method for estimating mutual information is the KSG. In this paper, we propose using a recurrent neural network to estimate mutual information. The data used to train and test the network is created by a Gaussian generative model, which allows us to calculate the mutual information between two random variables.**

We evaluate the network on simulated data and find that it is an unbiased estimator of mutual information, for this specific data. Furthermore, we find that the residuals of the network have a statistically significantly lower variance than those of the KSG. From the test results, we conclude that the network outperforms the KSG on the generated data.

This paper suggests that using a recurrent neural network for the estimation of mutual information might be the next development in getting a better estimator than the current state-of-art KSG.

***Index Terms*—Kraskov–Stögbauer–Grassberger, Gaussian Process, Mutual Information, Recurrent Neural Network.**

I. Introduction

NEURAL networks are an important concept and have provided significant improvements in estimation and classification problems across multiple fields [1]. Many innovations and real-world applications have already implemented neural networks in their products, with the majority focusing on feedforward neural networks (FNN) [2]. There are existing applications where sequence processing is needed. The recurrent neural network (RNN) more naturally expresses the dynamics of time-dependent data, making it preferable in these applications [2]. Researchers have gained interest in RNNs, in recent years, since it has been shown that they often outperform classical statistical models in forecasting applications [1].

Recent studies have examined the effectiveness of estimations of mutual information (MI) using neural networks. MI is an important tool for quantifying the dependence between two random variables, and it has useful applications in fields such as information theory, statistics, and machine learning. Often, MI is difficult to estimate for real-world data [3]. The state-of-the-art method in estimating MI is the Kraskov-Stögbauer-Grassberger (KSG) [4], which utilizes the k-nearest neighbor classifier. The KSG is an asymptotically unbiased estimator on sufficiently regular densities. However, the KSG is biased in most practical applications [4].

In the paper [5], the author presents an MI neural estimator (MINE), which estimates the lower upper bound of MI. The results, they obtained for MINE, provide significant improvements compared to the KSG.

In this paper, another approach to the estimation of MI will be considered, where an RNN is implemented. This paper will in Section II present the methods used to create the data, train, and test the network. Hereafter, the results will be presented in Section III, where the data used to train the network is visualized. Then the residuals of the network will be presented and analyzed using statistical methods. The results will be discussed alongside their presentation. Finally, we conclude the paper in Section IV.

II. Methods

A. The Generative Model

We generate the data according to the following generative model,

$$X_i \sim \mathcal{N}\left(0,\ \sigma_x^2 I_d\right), \tag{1}$$
$$Y_i \sim \mathcal{N}\left(X_i,\ \sigma_y^2 I_d\right), \tag{2}$$

where I_d is the identity matrix with dimension d. This generative model is presented in [6]. At each iteration, a pair $(X_i,\ Y_i)$ is generated independently. For each choice of (σ_x^2, σ_y^2), we generate n multiple pairs $\{(x_{i,j},\ y_{i,j})\}_{j=1}^{n}$.

For this generative model, it is the case that,

$$I(X_i; Y_i) = \frac{d}{2}\log_2\left(1 + \frac{\sigma_x^2}{\sigma_y^2}\right), \tag{3}$$

which will be used to label the generated data [6].

We aim to generate data with a uniform distribution of labels. Hence, we choose (σ_x^2, σ_y^2) according to the following method. Let $0 < \alpha < \beta < \gamma$, and then choose

$$a_x, a_y \sim \text{Unif}(\alpha, \beta), \tag{4}$$

independently of each other. Then choose

$$b_x \sim \text{Unif}(a_x + \alpha, a_x + \gamma), \tag{5}$$
$$b_y \sim \text{Unif}(a_y + \alpha, a_y + \gamma), \tag{6}$$

independently of each other. The variances are then chosen

$$\sigma_x^2 \sim \text{Unif}(a_x, b_x), \tag{7}$$
$$\sigma_y^2 \sim \text{Unif}(a_y, b_y), \tag{8}$$

independently of each other. After generating each pair (σ_x^2, σ_y^2), the value of $I(X;Y)$ is calculated.

The range of possible MI is given as $[I_{min}, I_{max}]$ where

$$I_{min} = \frac{1}{2}\log_2\left(1 + \frac{\alpha}{\beta+\gamma}\right), \tag{9}$$

$$I_{max} = \frac{1}{2}\log_2\left(1 + \frac{\beta+\gamma}{\alpha}\right). \tag{10}$$

We further restrict the range of labels to $[I_{min}, \tilde{I}_{max}]$, where $\tilde{I}_{max} < I_{max}$. This interval is then split into N_I evenly sized intervals, and a number, of labels n_I, is specified. After each pair (X_i, Y_i) is generated, we check what interval its label fits into. If that interval has less than n_I pairs associated with it, we associate this new pair with that interval. This leaves us with a total number of labeled pairs equal to $n_I N_I$, with labels approximately uniformly distributed on $[I_{min}, \tilde{I}_{max}]$.

Two examples of the generated data can be seen in Figure 1 and Figure 2, where Figure 1 shows a lower MI than Figure 2.

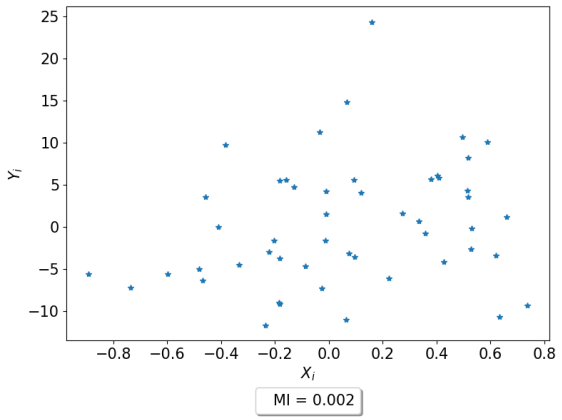

Fig. 1: Plot showing an example of the data, with an MI of approximately 0.002.

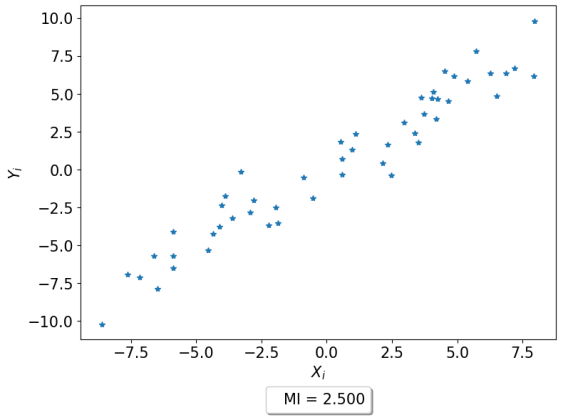

Fig. 2: Plot showing an example of the data, with an MI of approximately 2.5.

From Figure 1 and Figure 2 we can see that when we have high MI, the two series X_i and Y_i are very correlated, and when the MI is low the two series are not very correlated.

B. The Network

We have designed a many-to-one two-layer stacked RNN [7] with the hidden dimension equal to the input size for the network and the output size is one. This is illustrated in Figure 3

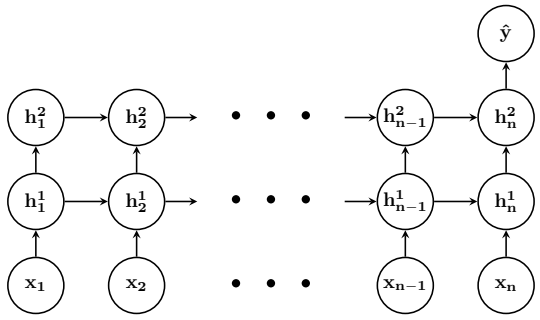

Fig. 3: An illustration of a many-to-one two-layer stacked RNN. The input layer is made up of nodes x_i, where i is the index of the node. The hidden layers are made up of nodes h_i^k, where i is the index of the node, similarly to the input layer, and k refers to the layer of the RNN. Finally $\hat{y}$ is the output.

The loss function used to train the network is the L^1-norm. The activation function used on the hidden layers is $\tanh$ and no activation function is applied on the output. The gradient-based optimizer used for the network is the Adam optimizer.

C. The KSG

In this paper, we compare the network to the KSG, which is defined as

$$\hat{I}_{KSG}(X_i;Y_i) = \psi(K) + \psi(N) - \frac{1}{N}\sum_{j=1}^{N}\psi(n_x(j)) + \psi(n_y(j)), \tag{11}$$

where $\psi = \frac{d}{dx}\ln\Gamma(x)$, is the digamma function, N is the number of samples, and $n_x(j)$ and $n_y(j)$ are counting functions on regions defined by the k-nearest neighbor classifier [8].

D. Statistical Methods

When analyzing the results, we implement several statistical methods.

A quantile-quantile (q-q) plot is used to check if a set of samples follow a certain distribution [9], which determines which statistical tests are viable for the data. A one-sample Student's t-test is used to check if a set of samples have a specified mean [10].

A Levene test is an analysis of variances test (ANOVA test), used to check if two sets of samples have equivalent variances [11]. The Levene test is more robust in cases where some samples do not follow a Gaussian distribution. Specifically, there are no assumptions made on the underlying distribution of the samples.

III. Results

The tests were carried out using the network described in the previous section. Specifications of the generative model and network can be seen in Table I.

Specifications	
Size of Dataset	10,000
Number of Realisation in a Data-point (n)	50
Train/Test split	90%/10%
Dimension (d)	1
Epochs	20
Batch Size	100
Learning Rate	10^{-4}
$(I_{min}; \tilde{I}_{max})$	$(0; 2.5)$
α	0.01
β	2
γ	100

TABLE I: Specifications for the network and the generative model.

All tests and calculations were carried out in Python 3.10.8. The KSG estimate was calculated using `Sklearn`'s feature selection module, and the network was implemented using `PyTorch`'s neural network module. Furthermore, we refer to Appendix A for specific versions of Python packages.

A. Residuals Distribution

We estimate the MI of each entry in the testing data and compare it with the label of that given entry. We define the residuals

$$r_{KSG} = \hat{I}_{KSG}(X_i; Y_i) - I(X_i; Y_i), \tag{12}$$

$$r_{network} = \hat{I}_{network}(X_i; Y_i) - I(X_i; Y_i), \tag{13}$$

where $\hat{I}_{KSG}(X_i; Y_i)$ is the estimate of MI made by the KSG, and $\hat{I}_{network}(X_i; Y_i)$ is the MI estimate made by the network. A histogram of the residuals is shown in Figure 4. The sample

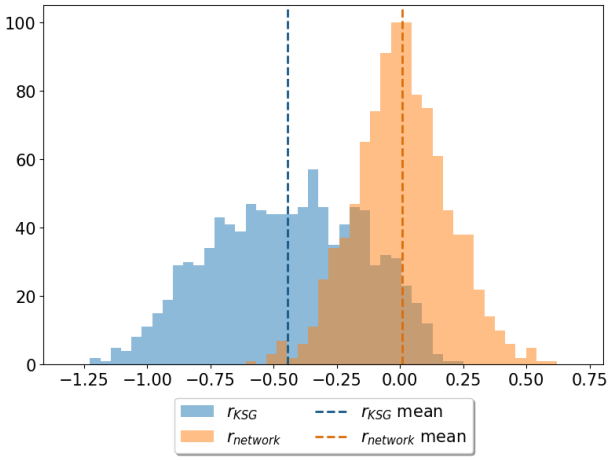

Fig. 4: A histogram of the residuals for the KSG and the network.

mean of the residuals of the KSG is approximately -0.5. A residual mean different from zero is expected since the KSG is a biased estimator. The sample mean of the residuals of the network is approximately 0, indicating that the network might be an unbiased estimator. The distribution of the residuals of the network seems to follow a Gaussian distribution, in contrast to the residuals of the KSG. To further explore the distribution of the respective residuals, two q-q plots are created. The q-q plot of the residuals of the network and the KSG can be seen in Figure 5 and Figure 6, respectively. The figures contain a 95% confidence region, showing how well the residuals fit a Gaussian distribution.

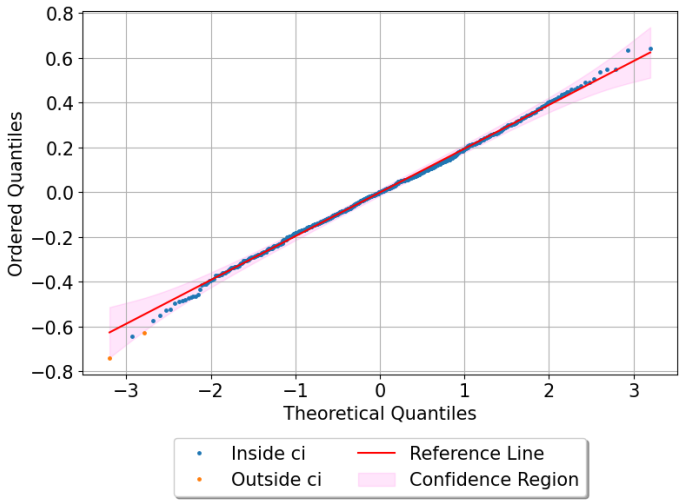

Fig. 5: A q-q plot of the residuals of the network with a significance level $\alpha = 0.05$ and the distribution used for the theoretical quantiles is the normal distribution.

Figure 5 shows that only 2 out of 1000 points are not within the confidence region. This suggests that a Gaussian distribution is a good fit for the residuals of the network.

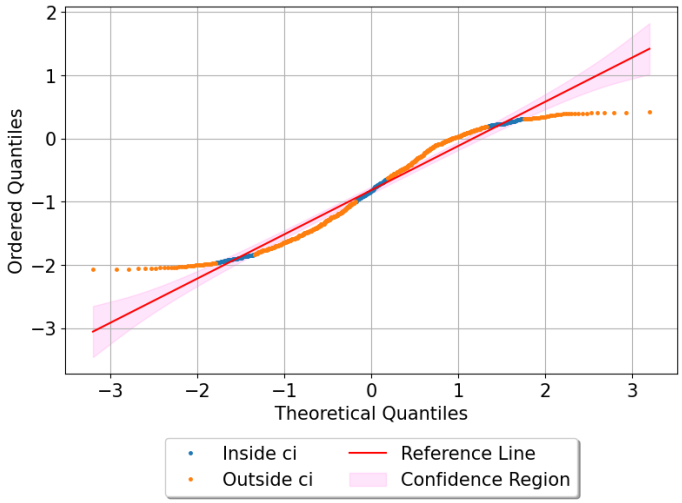

Fig. 6: A q-q plot of the residuals of the KSG with a significance level $\alpha = 0.05$ and the distribution used for the theoretical quantiles is the normal distribution.

In Figure 6, it can be seen that many points are not within the confidence region, indicating that the residuals of the KSG do not follow a Gaussian distribution. Specifically, 747 out of 1000 points fall outside the confidence region.

B. Statistical Tests

In order to check whether there is a statistically significant bias in the network, we perform a one-sample Student's t-test on the residuals of the network.

We choose the hypotheses

$$\mathcal{H}_0 : \mu = 0, \tag{14}$$
$$\mathcal{H}_1 : \mu \neq 0. \tag{15}$$

Running the test yields a p-value of approximately 0.905, meaning we cannot reject the null hypothesis. Calculating a 95% confidence interval results in an interval approximately equal to $[-0.01,\ 0.01]$. We can then say with high confidence that the network is an unbiased estimator, with regard to the data we tested it on.

Calculating the maximum likelihood estimation of the variances of the residuals of the network and the KSG yields

$$\hat{\sigma}^2_{\text{network}} \approx 0.038, \tag{16}$$
$$\hat{\sigma}^2_{\text{KSG}} \approx 0.093, \tag{17}$$

respectively. The variances serve as a dispersion measure to show how far off a typical estimation of MI is. To test if these variances are statistically significantly different, we perform a Levene test. We perform the Levene test since the residuals of the KSG are not normally distributed and the Levene is robust in cases where samples are not normally distributed.

For the Levene test, we choose the hypotheses

$$\mathcal{H}_0 : \sigma^2_{\text{network}} = \sigma^2_{\text{KSG}}, \tag{18}$$
$$\mathcal{H}_1 : \sigma^2_{\text{network}} \neq \sigma^2_{\text{KSG}}. \tag{19}$$

Performing the test yields a p-value approximately equal to $4 \cdot 10^{-54}$, meaning these variances are statistically significantly different.

The results of our testing show that the network is more precise and accurate than the KSG. Precise meaning the network has a lower variance when estimating MI for a data point. Accurate meaning that the network is unbiased for this specific type of data.

IV. Conclusion

In this paper, we have used synthetic Gaussian data to generate stochastic variables $(X_i,\ Y_i)$ and calculated their exact MI. We have trained a network to estimate the MI between these variables and compared these estimates to the estimates of the KSG. The results of this comparison show that the network outperforms the KSG, on this specific type of data.

The data used to train the network, and test both the network and the KSG, does not represent a typical real-world process. This restricts the application of the network, and does not reflect its ability to estimate MI between more general stochastic variables.

We have shown that a network can outperform the KSG on this specific type of data. Investigating the efficacy of the network on more general data, such as other distributions, processes, or real-world data, might be of interest in future work. Furthermore, it could be of interest to train a network to estimate conditional mutual information.

References

[1] K. T. Amit and A. Ajith, "Recurrent Neural Networks: Concepts and Applications", Taylor Francis Group, 2022, p. ix.

[2] M. S. Fathi, "Recurrent Neural Networks: From Simple to Gated Architectures", Springer, 2022, p. xvii.

[3] K. Alexander, S. Harald and G. Peter, "Estimating mutual information", 2008, p. 1, [Online], Available: https://arxiv.org/abs/cond-mat/0305641.

[4] M. H. Caroline and N. Ilya, "Estimation of mutual information for real-valued data with error bars and controlled bias", 2019, pp. 3-4, [Online], Available: https://arxiv.org/abs/1903.09280.

[5] I. B. Mohamed, B. Aristide, R. Sai, O. Sherjil, B. Yoshua, C. Aaron and D. H. R, "MINE: Mutual Information Neural Estimation", 2021, p. 4, [online], Available: https://arxiv.org/abs/1801.04062.

[6] M. Sina, "Statistical Inference of Information in Networks: Causality and Directed Information Graphs", KTH Royal Institute of Technology, 2021, p. 51.

[7] L. John, "Stacked RNNs for Encoder-Decoder Networks: Accurate Machine Understanding of Images", 2016, p. 2, [Online].

[8] T. Alkiviadis, V. Ioannis and K. Dimitris, "Nearest neighbor estimate of conditional mutual information in feature selection", 2022, pp. 12698-12699, [Online].

[9] I. M. John, "Positions and QQ Plots", 2004, p. 606, [online].

[10] M. Prabhaker, S. Uttam, M. P. Chandra, M. Priyadarshni and P. Gaurav, "Application of Student's t-test, Analysis of Variance, and Covariance", 2019, p. 1, [online].

[11] V. G. Gene, "Testing Homogeneity of variance", 1965, p. 1, [online].

Appendix
Software Specifications

Package	Version
`torch`	1.12.1
`torchviz`	0.0.2
`scikit_learn`	1.0.2
`scipy`	1.9.2
`matplotlib`	3.6.2
`numpy`	1.23.3

TABLE II: Python packages and their versions.

Table II shows the Python packages, and their version number, used in the calculations of this article.