Recurrent Neural Network for Estimating Mutual Information of a Gaussian Processes

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

***Abstract*—The estimation of mutual information (MI) is a long-standing problem. The state-of-the-art in this field is currently the Kraskov-Stögbauer-Grassberger (KSG) estimator.**

In this paper, we look at a specific Gaussian process that allows us to estimate MI between two random variables. The MI is used as labels for training a recurrent neural network (RNN) using supervised learning. The training and testing data are generated using the same process. Our suggested model performs an unbiased estimation of MI on this specific data. Furthermore, the model's residuals have a statistically significantly lower variance than that of the KSG. From the test results, we conclude that the model outperforms the KSG on the generated data.

***Index Terms*—Gaussian Process, Kraskov–Stögbauer–Grassberger, 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 [13]. Many innovations and real-world applications have already implemented neural networks in their products, with the majority focusing on feedforward neural networks (FNN) [12]. 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 [12]. Researchers have gained interest in RNNs in recent years since it has been shown that it often outperforms classical statistical models in forecasting applications [13].

Recent studies have examined the effectiveness of estimations of mutual information (MI) using neural networks. Often, MI is difficult to estimate for real-world data [1]. The state-of-the-art method in estimating MI, is the Kraskov-Stögbauer-Grassberger (KSG) estimator [1], [3], which utilizes the k-nearest neighbor classifier. The KSG estimator has been shown to be asymptotically unbiased. However, an impractically large number of samples of discrete data is needed, for this estimator to be unbiased [3], [4].

In the paper [14] 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 estimator.

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 the RNN, and validate the model. Hereafter, the results will be presented in Section III, where the data used to train the model is visualized. Then the residuals of the model 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

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

where I_d is the identity matrix with dimension d. At each iteration a pair $(X_i,\ Y_i)$ is generated independently. For each choice of (σ_x, σ_y) we generate n multiple pairs $\{(x_{i,j},\ y_{i,j})\}_{j=1}^{n}$ [5].

For this generative model, it is the case that

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

which will be used to label the generated data [5]. The aim is to create a versatile RNN model and hence it is desired to generate (X_i, Y_i) such that the MIs are approximately uniformly distributed. Hence, we choose (σ_x, σ_y) according to the following method. Let $\alpha, \beta, \gamma > 0$ be scalars where $\alpha < \beta < \gamma$. 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

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

independently of each other. After generating each pair (σ_x, σ_y) 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 uniformly distributed on $[I_{min}, \tilde{I}_{max}]$.

B. The Network

We have designed a many-to-one two-layered stacked RNN [6], [11] with the hidden dimension equal to the input size for the network and the output size is one. In order to estimate MI, we need a measure of how well a prediction is compared to the exact MI. The loss function used for this purpose is the L^1 norm. The gradient-based optimizer used for the network is the Adam algorithm [6].

C. The KSG Estimator

An implementation of the KSG estimator is used, in order to compare the results obtained from the model:

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

where $\psi = \frac{d}{dx}\ln\Gamma(x)$, is the digamma function, N is the number of samples, and $n_x(i)$ and $n_y(i)$ are counting functions on regions defined by the k-nearest neighbor classifier [7].

D. Statistics 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 [8]. A one-sample students t-test is used to check if a set of samples have a specified mean [9]. A Levene test is an analysis of variances test (ANOVA test), used to check if two sets of samples have equivalent variances [10]. The Levene test is more robust in cases where 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 RNN described in the previous section. Specifications of the model and the data generation algorithm can be found 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}
$(I_{min}; I_{max})$	$(0; 2.5)$
α	0.01
β	2
γ	100

TABLE I: Specifications for the RNN 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 model was implemented using PyTorch's neural network module. Furthermore, we refer to Appendix A for specific versions of Python packages.

A. Data Visualization

Two examples of the generated data can be seen in Figure 1 and Figure 2. Figure 2 shows a higher correlation than

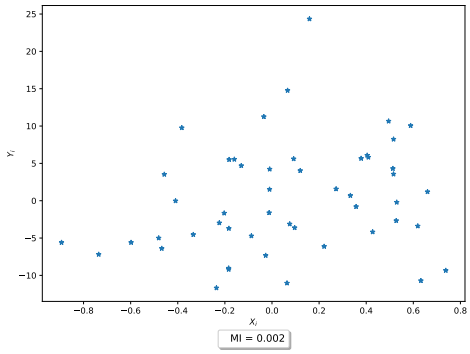

Fig. 1: Plot showing an example of the data, with an MI of approximately 0.002.

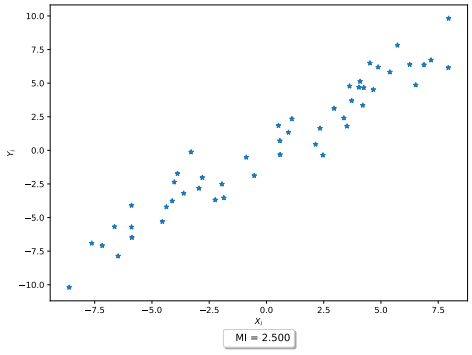

Fig. 2: Plot showing an example of the data, with an MI of approximately 2.5

Figure 1. It is also clear that a better estimate of $y_{i,j}$ can be gained from $x_{i,j}$ in Figure 2 than in Figure 1, meaning that the former figure shows a data-set with higher MI.

B. 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

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

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

where $\hat{I}_{KSG}(X_i;Y_i)$ is the estimate of MI made by the KSG, and $\hat{I}_{RNN}(X_i;Y_i)$ is the MI estimate made by the model. These residuals are shown in Figure 3. The sample mean

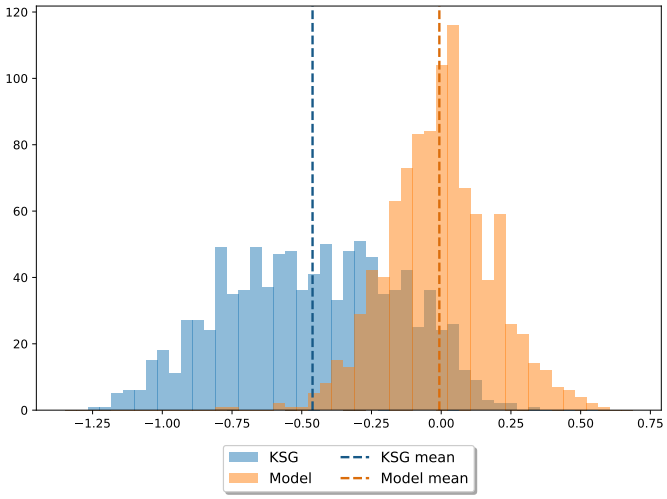

Fig. 3: A histogram of the residuals for the KSG and the model.

of the residuals of the KSG is approximately -0.5, which is expected since the KSG is a biased estimator. The sample mean of the residuals of the model seems to be approximately 0, indicating that the model might be an unbiased estimator. The distribution of the residuals of the model 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 KSG and the model can be seen in Figure 5 and Figure 4, respectively. The figures contain a 95% confidence region, showing how well the residuals fit a Gaussian distribution.

Figure 4 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 model.

In Figure 5 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 of the confidence region.

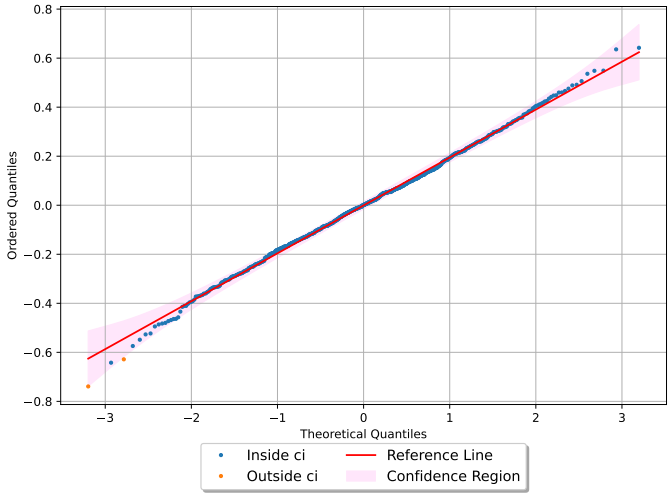

Fig. 4: A q-q plot of the residuals of the model with a significance level $\alpha = 0.05$ and the distribution used for the theoretical quantiles is the normal distribution.

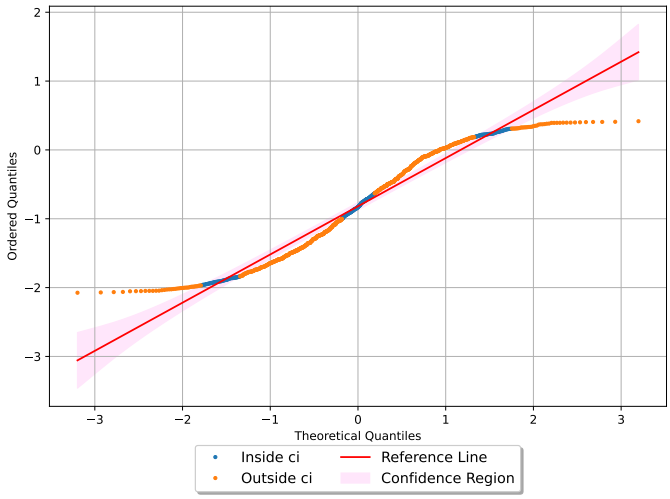

Fig. 5: A q-q plot of the residuals of the KSG with a significance level $\alpha = 0.05$ and the distribution used for the theoretical quantiles is the normal distribution.

C. Statistical Tests

In order to check whether there is a statistically significant bias in the model we perform a one-sample students t-test on the residuals of the model.

We choose the hypotheses

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

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 model 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 model and the KSG yields

$$\hat{\sigma}_{\text{RNN}} \approx 0.038, \tag{14}$$

$$\hat{\sigma}_{\text{KSG}} \approx 0.093, \tag{15}$$

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 [10]. We perform the Levene test since the KSG is not normally distributed and the Levene is robust in cases where samples are not normally distributed.

For the Levene test, we have the hypotheses

$$\mathcal{H}_0 : \sigma_{\text{RNN}} = \sigma_{\text{KSG}}$$
$$\mathcal{H}_1 : \sigma_{\text{RNN}} \neq \sigma_{\text{KSG}}.$$

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 model 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 $(X,\ Y)$ and calculated their exact MI. We have trained an RNN model 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 model outperforms the KSG on this specific type of data, in the sense that the model is unbiased, and has higher accuracy in its estimates.

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

We have shown that a model can outperform the KSG estimator on this specific type of data. Investigating the efficacy of the model on more general data, such as other distributions, processes, or real-world data, might be of interest in future work.

References

[1] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," Physical Review E, vol. 69, no. 6, Jun. 2004, doi: 10.1103/physreve.69.066138.

[2] J. Runge, "Conditional independence testing based on a nearest-neighbor estimator of conditional mutual information," arXiv:1709.01447 [cs, math, stat], Sep. 2017, Accessed: Nov. 09, 2022. [Online]. Available: https://arxiv.org/abs/1709.01447

[3] W. Gao, S. Oh, and P. Viswanath, "Demystifying Fixed k-Nearest Neighbor Information Estimators." Accessed: Nov. 09, 2022. [Online]. Available: https://arxiv.org/pdf/1604.03006.pdf

[4] C. M. Holmes and I. Nemenman, "Estimation of mutual information for real-valued data with error bars and controlled bias," Physical Review E, vol. 100, no. 2, Aug. 2019, doi: 10.1103/physreve.100.022404.

[5] S. Molavipour, "Statistical Inference of Information in Networks Causality and Directed Information Graphs kth royal institute of technology." Accessed: Nov. 09, 2022. [Online]. Available: http://www.diva-portal.org/smash/get/diva2:1599246/FULLTEXT01.pdf

[6] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016.

[7] A. Tsimpiris and I. Vlachos and D. Kugiumtzis, "Nearest neighbor estimate of conditional mutual information in feature selection." Accessed: Nov. 21, 2022. [Online]. Available: http://ict.ihu.gr/wp-content/uploads/2019/10/CMINN.pdf

[8] John I. Marden, "Positions and QQ Plots" Accessed: Nov. 23, 2022. [online]. Available: https://www-jstor-org.zorac.aub.aau.dk/stable/4144431?seq=1

[9] Prabhaker Mishra, Uttam Singh, Chandra M Pandey, Priyadarshni Mishra, Gaurav Pandey, "Application of Student's t-test, Analysis of Variance, and Covariance" Accessed: Nov. 23, 2022. [online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6813708/pdf/ACA-22-407.pdf

[10] Gene V. Glass, "Testing Homogeneity of variance" Accessed: Nov. 23, 2022. [online]. Available: https://www-jstor-org.zorac.aub.aau.dk/stable/pdf/1161802

[11] J. Lambert, "Stacked RNNs for Encoder-Decoder Networks: Accurate Machine Understanding of Images", Department of Computer Science, Stanford University

[12] Salem, F.M. "Recurrent Neural Networks: From Simple to Gated Architectures", Springer International Publishing

[13] Amit K. Tyagi, Ajith Abraham "Recurrent Neural Networks: Concepts and Applications", CRC Press

[14] Mohamed I. Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, R. D. Hjelm "MINE: Mutual Information Neural Estimation", Accessed: Nov. 25, 2022. [online]. Available: https://arxiv.org/pdf/1801.04062.pdf

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.