CMI Estimation

Project B

Authors:

Julia Girtler

Helena Wałachowska

January 21, 2025

Contents

1 Introduction

The goal of our project was to understand the functioning of two CMI (Conditional Mutual Information) estimators used for determining conditional dependence in continuous data and to implement them. We chose an estimator based on the kNN method described in the article *"Conditional independence testing based on a nearest-neighbor estimator of conditional mutual information"* by Jakob Runge. The second estimation method uses the chain rule.

We then tested the performance of the implemented estimators on three types of data, each in two variants, comparing the obtained results with the estimator available in the Sklearn library. We also introduced two metrics to evaluate the accuracy of feature selection based on the estimated CMI values.

2 Estimation Methodology

2.1 Estimator 1 – kNN method by Runge

The first method for estimating $I(X, Y|Z)$ is based on creating a sample X_{perm} derived from X, but constructed such that $I(X_{perm}, Y|Z) = 0$, meaning there is no dependence between X_{perm} and Y. However, we aim to preserve the dependencies between X_{perm} and Y, as well as between X_{perm} and Z, so that the joint distributions (X_{perm}, Y) and (X_{perm}, Z) remain equal to the distributions (X, Y) and (X, Z), respectively. The algorithm for generating such an X_{perm} with these properties is described in the article by J. Runge mentioned in the introduction, which enabled us to implement it. In subsequent calculations, we used this algorithm with the parameter $k_{perm} = 5$ (the number of nearest neighbors considered).

In the next step, using the Keras library, we constructed a neural network that, based on the joint samples (X, Y, Z) (from the distribution $p(x, y, z)$) and (X_{perm}, Y, Z) (from the distribution $p(x|z)p(y|z)p(z)$), estimates the value of $I(X, Y|Z)$ using the Donsker-Varadhan loss function or the Nguyen-Wainwright-Jordan loss function. This method estimates the Kullback-Leibler divergence between the distributions derived from the previously prepared joint samples.

The implemented neural network consists of two dense layers with 128 neurons each and one output layer with a single neuron. For the activation of the neurons, we used the Rectified Linear Unit (ReLU) function. Additionally, to stabilize the training process when using a high learning rate, we applied Batch Normalization, as well as Dropout regularization to prevent overfitting. As the optimizer, we used the Adam algorithm with $learning\_rate = 10^{-4}$ and $clipnorm = 1.0$ (to prevent large parameter weight changes). To further improve the model's results, we applied EarlyStopping and ReduceLROnPlateau techniques. The exact parameters used in our neural network can be found in the code attached to the project.

The model is trained on data split into training and test sets in a ratio of 0.2, with the parametersh $epochs = 200$ and $batch\_size = 64$.

2.2 Estimator 2 – Chain Rule

The second estimation method is based on the formula $I(X, Y|Z) = I((X, Z)|Y) - I((X)|Y)$. Using the neural network described in the previous method, we estimate the value of $I((X, Z)|Y)$ as the KL divergence between the distributions of the samples (X, Z, Y) and (X, Z, Y_{perm}), and the value of $I((X)|Y)$ as the KL divergence between the distributions of (X, Y) and (X, Y_{perm}). The vector Y_{perm} is obtained using the random.permutation function from the NumPy library.

After calculating the corresponding mutual information values, we subtract them, which allows us to estimate the value of the conditional mutual information.

2.3 Estimator 3 – Sklearn Estimator

To compare the performance of our implemented methods for estimating CMI, we used the mutual_info_regression(X, Y, discrete_features=False, n_neighbors=5) function from the Sklearn package. This function computes mutual information using kernel density approximation of probability distributions with kNN-based estimators. In this case, we also calculated the value of CMI using the chain rule: $I(X, Y|Z) = I((X, Z)|Y) - I((X)|Y)$.

3 Data

3.1 Data 1 – for which the formula for CMI is known and exact CMI can be computed (on a grid of parameters)

For data from normal distribution, there exists an exact formula for CMI. To calculate it, we implement a function that returns the exact CMI based on:

$$\text{CMI} = \frac{1}{2} \log \left(\frac{\det(\Sigma_{XYZ})}{\det(\Sigma_{XZ}) \cdot \det(\Sigma_{YZ})} \right)$$

3.2 Data 2 – for a simulation example of ranking variable importance based on our estimators

The dataset consists of 10000 samples and 20 features, generated using the `make_regression` function to simulate a regression task with added noise. Additionally, three features have been manually manipulated to have strong relationships with others, which will significantly influence the target variable Y.

3.3 Data 3 – a dataset example of ranking variable importance

The California Housing dataset consists of various features such as geographic location, median income, and housing characteristics (X), with the target variable being the median house value (Y). For efficiency, the dataset has been limited to 20,000 samples.

4 Experiments

In this chapter, we present the results of experiments verifying the quality of our estimators based on the previously described datasets. We introduce quality measures to evaluate the effectiveness and consistency of different feature ranking methods. Specifically:

1. **Count of Inversions:** Measures the dissimilarity between two rankings by counting inversions. Fewer inversions mean more consistency between rankings.
2. **Top-k Agreement:** Quantifies the overlap in the top k features between two rankings. Higher agreement indicates better consistency in identifying important features.

During the experiment, we used a neural network with the DV loss function, and Data 2 was generated with the generator seed set to 42.

4.1 Data 1

For the first variant of the data, where no conditional dependence is present, all estimators correctly determined CMI = 0.

For the second variant of the data, we present a comparison between the true CMI value and the values estimated using the three previously described estimators. The first and second estimators are presented in two variants — using the DV loss function and the NWJ loss function. The lack of comparative values for large ρ values is due to numerical limitations in calculations involving covariance matrices with such values (matrix becomes non-invertible).

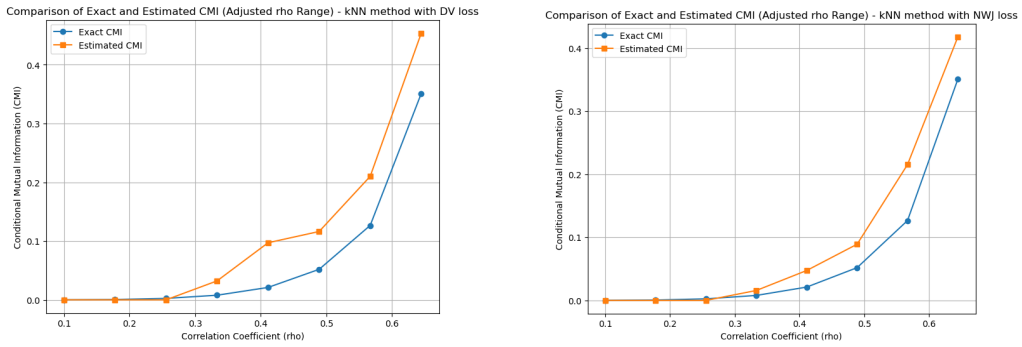

Figure 1: Results of first experiment (Estimator 1 with DV and NWJ loss functions).

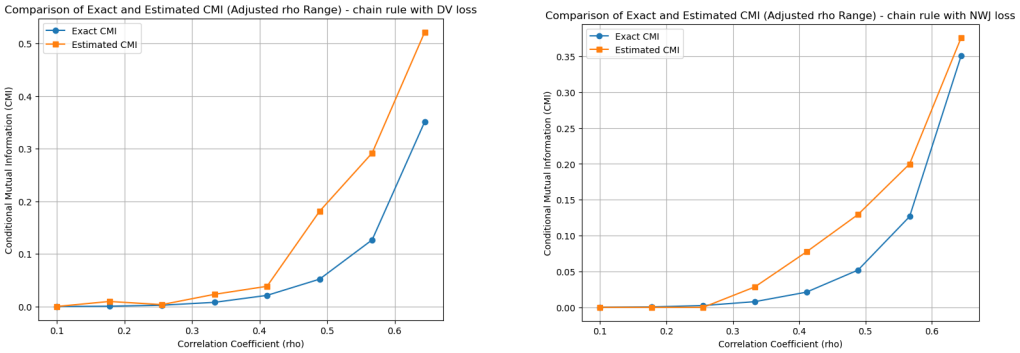

Figure 2: Results of second experiment (Estimator 2 with DV and NWJ loss functions).

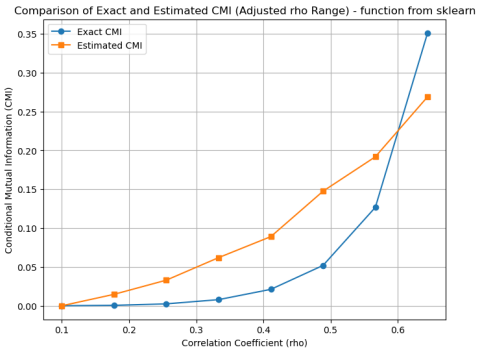

Figure 3: Results of third experiment (Sklearn Estimator).

4.2 Data 2

In the following step, we use the LassoCV model from scikit-learn to compute the importance of each feature (variable) in the dataset.

4.2.1 Example 1

Estimator 1

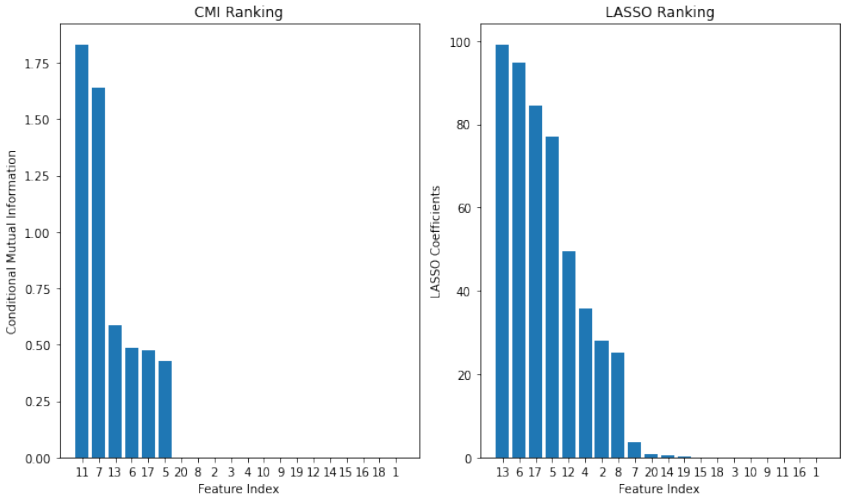

Figure 4: Feature importance for Estimator 1.

Top 10 variables based on CMI: $[11, 7, 13, 6, 17, 5, 20, 8, 2, 3]$ and LASSO: $[13, 6, 17, 5, 12, 4, 2, 8, 7, 20]$.

Number of inversions: 78

Top 10 agreement score: 0.8

Estimator 2

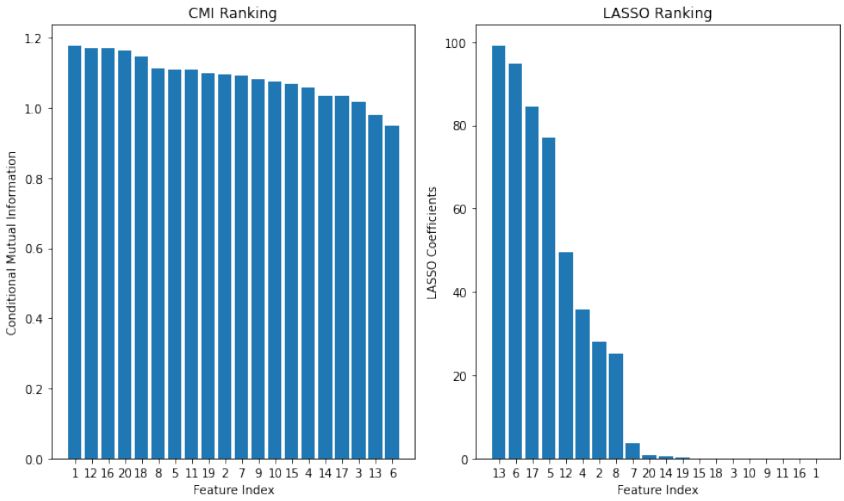

Figure 5: Feature importance for Estimator 2.

Top 10 variables based on CMI: $[1, 12, 16, 20, 18, 8, 5, 11, 19, 2]$ and LASSO: $[13, 6, 17, 5, 12, 4, 2, 8, 7, 20]$.

Number of inversions: 95

Top 10 agreement score: 0.5

Estimator 3

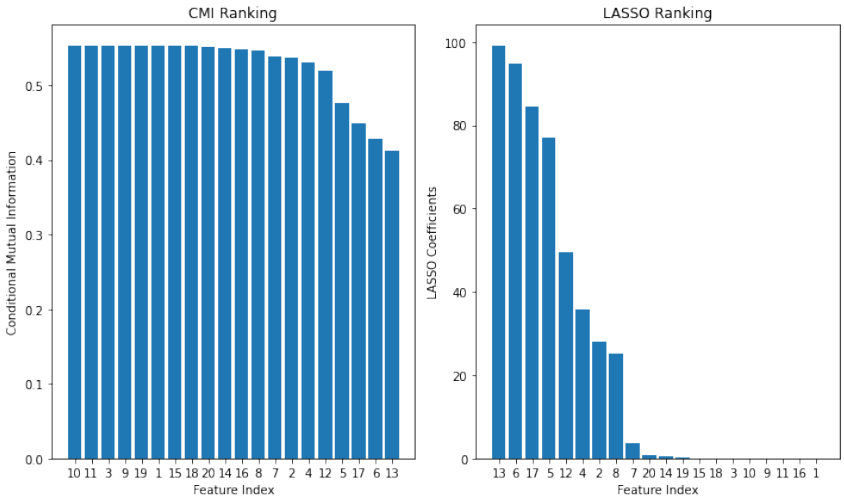

Figure 6: Feature importance for Estimator 3.

Top 10 variables based on CMI: $[10, 11, 3, 9, 19, 1, 15, 18, 20, 14]$ and LASSO: $[13, 6, 17, 5, 12, 4, 2, 8, 7, 20]$.

Number of inversions: 106

Top 10 agreement score: 0.1

4.2.2 Example 2

Estimator 1

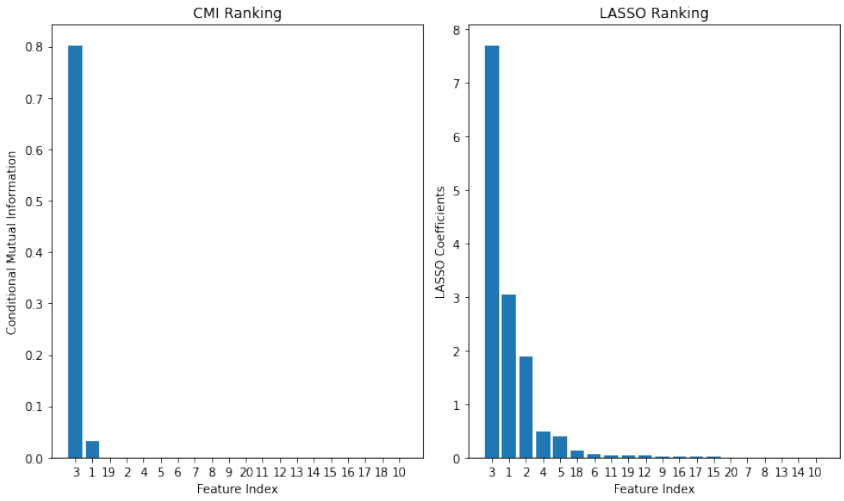

Figure 7: Feature importance for Estimator 1.

Top 10 variables based on CMI: $[3, 1, 19, 2, 4, 5, 6, 7, 8, 9]$ and LASSO: $[3, 1, 2, 4, 5, 18, 6, 11, 19, 12]$.

Number of inversions: 75

Top 10 agreement score: 0.7

Estimator 2

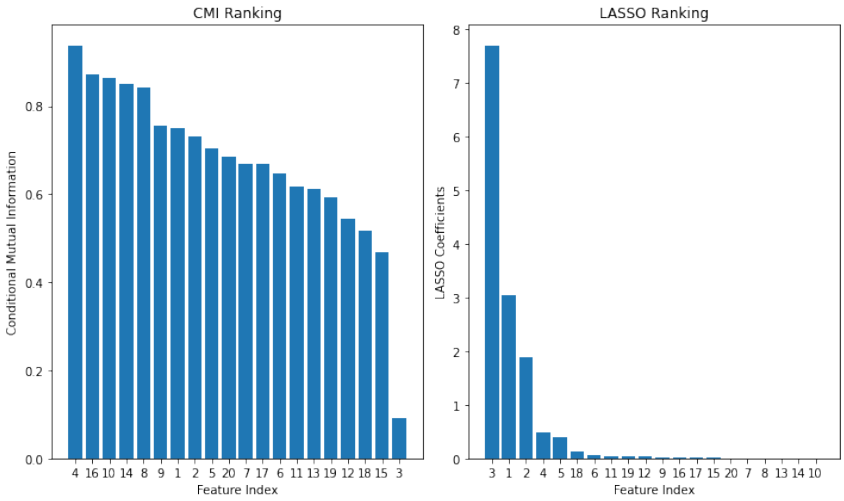

Figure 8: Feature importance for Estimator 2.

Top 10 variables based on CMI: $[4, 16, 10, 14, 8, 9, 1, 2, 5, 20]$ and LASSO: $[3, 1, 2, 4, 5, 18, 6, 11, 19, 12]$.

Number of inversions: 97

Top 10 agreement score: 0.4

Estimator 3

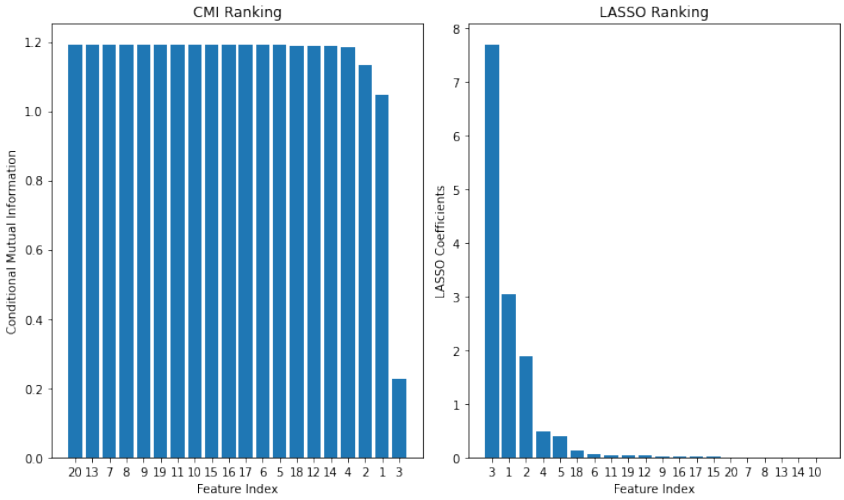

Figure 9: Feature importance for Estimator 3.

Top 10 variables based on CMI: $[20, 13, 7, 8, 9, 19, 11, 10, 15, 16]$ and LASSO: $[3, 1, 2, 4, 5, 18, 6, 11, 19, 12]$.

Number of inversions: 95

Top 10 agreement score: 0.2

4.3 Data 3

4.3.1 Example 1

Estimator 1

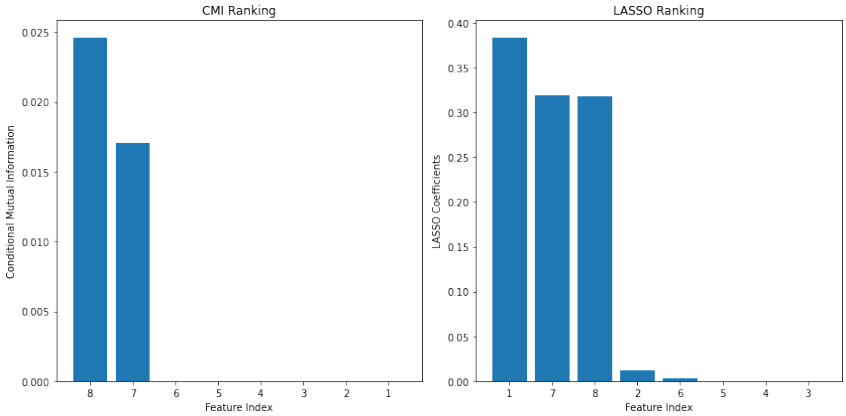

Figure 10: Feature importance for Estimator 1.

Top 10 variables based on CMI: $[8, 7, 6, 5, 4, 3, 2, 1]$ and LASSO: $[1, 7, 8, 2, 6, 5, 4, 3]$.

Number of inversions: 12

Top 5 agreement score: 0.6

Estimator 2

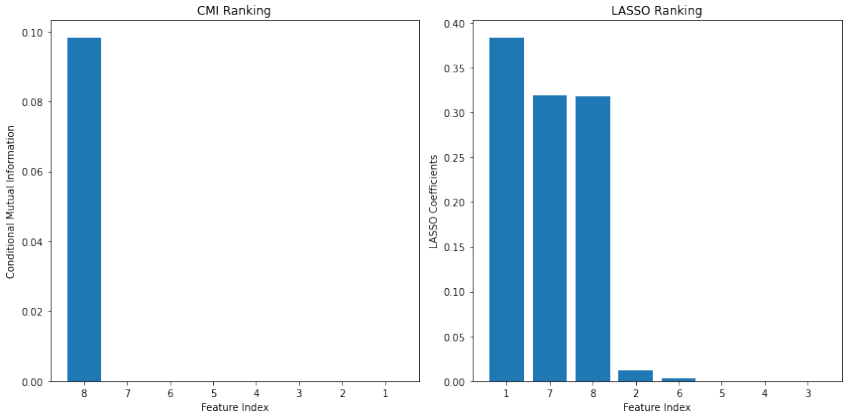

Figure 11: Feature importance for Estimator 2.

Top 10 variables based on CMI: $[8, 7, 6, 5, 4, 3, 2, 1]$ and LASSO: $[1, 7, 8, 2, 6, 5, 4, 3]$.

Number of inversions: 12

Top 5 agreement score: 0.6

Estimator 3

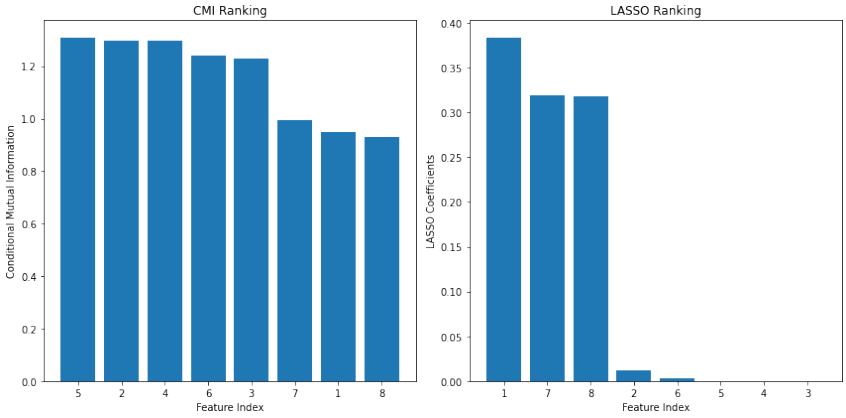

Figure 12: Feature importance for Estimator 3.

Top 10 variables based on CMI: $[5, 2, 4, 6, 3, 7, 1, 8]$ and LASSO: $[1, 7, 8, 2, 6, 5, 4, 3]$.

Number of inversions: 17

Top 5 agreement score: 0.4

5 Summary

1. In the first experiment, it was verified that all proposed estimators perform well in detecting the absence of conditional dependence in the data.
2. Based on the data for which CMI can be determined numerically, we observe that both Estimator 1 and 2 perform well in this scenario. In both cases, when using the DV loss function and the NWJ method, there is some instability in the results. Estimator 3 slightly overestimates the results.
3. In the case of artificially generated data with the problem of selecting the 10 most significant features, Estimator 1 performed the best, achieving 70-80% agreement with the LASSO method. Estimator 2 performed slightly worse, achieving 40-50% agreement. Estimator 3 made a decisively different selection, achieving only 10-20% agreement with LASSO.
4. In the case of selecting the 5 most significant features for a real dataset, both estimators 1 and 2 performed similarly, achieving 60% agreement with the features selected by LASSO. Estimator 3 performed slightly worse, achieving 40% agreement.