Conditional Normalising Flows for Interpretability

Valentyn Melnychuk

Munich 2021

Conditional Normalising Flows for Interpretability

Valentyn Melnychuk

Master Thesis
MSc Data Science
Department of Statistics
Faculty of Mathematics, Informatics and Statistics
at Ludwig-Maximilian University of Munich

Written by
Valentyn Melnychuk
from Kyiv, Ukraine

Munich, 21.03.2021

First supervisor: PhD Candidate Gunnar König
External supervisor: Univ.-Prof. Dr.-Ing. Moritz Grosse-Wentrup
Affiliated supervisor: Prof. Dr. Bernd Bischl
Day of oral examination: 22.04.2021

Declaration of Independence

I hereby confirm that I have written the accompanying thesis

"Conditional Normalising Flows for Interpretability"

by myself, without contributions from any sources other than those cited in the text and acknowledgements. This applies also to all graphics, drawings, maps and images included in the thesis.

	Munich, 21.03.2021
Valentyn Melnychuk	Place & Date

Contents

Abstract

This work aims to fill the gap of two existing interpretability methods for global feature importance (Relative feature importance (RFI) and Shapley additive global importance (SAGE)), by using deep conditional density estimator/sampler – Conditional Normalising Flow. To our knowledge, it is the first usage of Normalising Flows in the scope of feature importance estimation. We argue, that they are superior to other deep and classical density estimators, as they allow to efficiently evaluate likelihood, make a model selection and sample synthetic data, while not overfitting. After utilising noise regularisation, we discovered, that they do not require a rigorous hyperparameter search and could be used in the vanilla setting. We make an extensive empirical evaluation on different synthetic and real datasets with the help of a self-designed evaluation benchmark. Ground-truth feature importances are in general intractable, thus we inherited the concepts of strong and weak feature relevance, which have one-to-one relation to the causal structure of data generating mechanism. By utilising continuous datasets with a known causal graph, we can reason about the validity of estimated importances. Additionally, we provided a use case of RFI for detection of influence of sensitive attributes, when they are included in predictive modelling or completely ignored. In the end, this thesis extends the existing Python library for Relative Feature Importance (https://github.com/gcskoenig/rfi), with Conditional Normalising Flow and Mixture Density Network (an alternative deep learning method), as well as with the synthetic benchmark for future studies.

Acknowledgments

I (Valentyn Melnychuk) express deep gratitude to my supervisors: **Gunnar König** for the fruitful cooperation, weekly meetings and discussions, continuous ideas exchange and communication, for the code for interpretability methods and cross-review of code under RFI library development; **Prof. Moritz Grosse-Wentrup** for succinct comments and recommendations, regarding the directions of research.

I am thankful to Amit Fenn for the LaTeXtemplate [16].

Abbreviations & Notation

Abbreviations:

- CDE – Conditional Density Estimation
- CDF – Cumulative Density Function
- CFI – Conditional Feature Importance
- CNF – Conditional Normalising Flow
- DAG – Directed Acyclic Graph
- GoF – Goodness-Of-Fit
- GP – Gaussian Process
- HD – Hellinger Distance
- i.i.d. – Identically Independently Distributed
- JSD – Jensen-Shanon Divergence
- KLD – Kullback–Leibler Divergence
- MAE – Mean Absolute Error
- MB – Markov Blanket
- MC – Monte Carlo
- MDN – Mixture Density Network
- ML – Machine Learning
- MLE – Maximum Likelihood Estimation
- MSE – Mean Squared Error
- NF – Normalising Flow
- NLL – Negative Log-Likelihood
- NN – Neural Network
- PDF – Probability Density Function
- PFI – Permutation Feature Importance
- RFI – Relative Feature Importance
- SAGE – Shapley Additive Global importancE
- SCM – Structural Causal Model
- SHAP – SHapley Additive exPlanations

Notation:

- X, Y, Z – random variables (capital letters)
- x, y, z – realisation of random variable / datapoint
- X_i, Y_j – index of random variable, feature (subscript)
- $x^{(i)}, y^{(i)}$ – index of datapoint (superscript)
- $p, p(x), p(x/y), p_Z(z)$ – distribution / probability density function (depending on context)
- $X_i \sim p(x)$ – sampling from the distribution of X
- $\mathrm{X}, \mathrm{X}^*, \mathrm{y}$ – design matrices and target vector for Multiple Linear Regression
- $f_\theta(x)$ - parametric probabilistic model with parameters θ

Section 1

Introduction

Machine learning (ML) interpretability methods are designed to explain predictive models. While there exist intrinsically interpretable methods, like decision trees or generalised linear models, more advanced "black-box" models lack an intuitive understanding of their individual predictions or how features contribute to their performance. Many methods were designed for specific ML models (Random Forest Explainer [22; 36], Class Activation Mapping [63] etc.), but in this thesis we focus on **post-hoc model-agnostic** methods for interpretability. Post-hoc means, that an interpretability method is applied to an already fitted model - on post-processing step, and model-agnosticism implies that a method could be applied to any predictive model.

Other important division in ML interpretability hierarchy is between **global** and **local** methods. A local method tries to explain feature contributions of a single prediction, while a global tells features importance for the overall model's performance. One should also distinguish **feature importance**, which scores how much feature contributes to performance/prediction variance, and **feature effect** visualise the quantity of the relationship between features and target. Here, we specifically pay attention to global methods, which estimate feature importance, but some work can also be extended to local ones (e.g. from SAGE to SHAP, see Section 3.1.2).

Many recent ML interpretability methods rely on the crucial step of sampling from the conditional distribution of some features, conditioning on others. This adds an extra issue of conditional density estimation. Originally, many simplifying assumptions were made to simplify explanation generation, like the gaussianity of data or conditional independence of features. But recent papers discover that the misspecification of a density estimator could lead to misleading and false explanations of interpretability methods. This is regarded as the **off-manifold explanations** problem (see Section 2).

This work tries to fill the gap of conditional density estimation (CDE) for interpretability with means of **normalising flows**, a relatively new branch of generative deep learning models. They allow for general and flexible conditional density estimation, with the possibility to easily sample from complex distributions and have a tractable density, which simplifies the task of the model selection among estimators. It has some other great properties, like a possibility to control the support of distribution or access to cumulative density function or quantile function. We provide an extensive comparison of existing density estimators in Section 3.3.

This work extends the Relative Feature Importance (RFI) [27] (a global interpretability method) to deal with non-Gaussian data. Also, we extend Shapley additive global importance (SAGE) [12] to work correctly with conditionally dependent features. In Section 4 we evaluate the performance of different CDE estimators on different synthetic, pseudo-real and real datasets on our own developed benchmark, both in terms of goodness-of-fit and validity of explanations, induced by different estimators. Furthermore, we study the influence of the CDE misspecification in the case of sensitive attributes influence.

We extend the basic functionality of **nflows** library [15] (a comprehensive collection of normalizing flows using PyTorch), by introducing conditional transformations, where parameters are taken from contextual neural networks outputs. The final models (MDN or CNF) are then trained in an end-to-end fashion. Finally, we incorporate different conditional density estimators to the RFI Python library to boost further usage of this interpretability method.

Section 2

Related work

Many interpretability methods, while generating synthetic data, are using simplified assumptions, which in general do not hold. For example Relative Feature Importance [27] experiments only with multivariate Gaussian data with linear regression.

Authors of Shapely-based explanations (SHAP) [30] or SAGE [12] also unrealistically assumed independence of features and used sampling from marginal distribution. By violation of this assumption, synthetic data sampler can produce completely unrealistic combinations of features – **off-manifold data**, which results in misleading explanations. Many descending papers addressed this issue. [17] proposes to use variational autoencoder, with the additional masked context encoder. As VAE has an intractable density of generated data points, it is hard to make a samplers selection. [1] employs different simple density estimators, such as conditional Gaussian distribution, Gaussian copula or kernel estimate. [32] uses a special similarity function for existing train data to estimate SHAP.

Section 3

Methods

This chapter has a three-fold structure: first, we discuss two interpretability methods, which require the generation of synthetic data, sampled from conditional distribution. In the second part, we introduce the task of conditional density estimation (CDE) and sampling. At last, we overview different state-of-the-art deep-learning approaches for CDE.

3.1 Interpretability Methods

As it was already mentioned in the Introduction 1, interpretability methods fall into two categories: global and local. In this thesis, we solely concentrate on two methods for global model-agnostic feature importances: Relative Feature Importance (RFI) and Shapley additive global importance (SAGE).

Starting with a common notation for both methods, let's assume Y is a target random variable and variables X_R are used as training features (here R is an indexing set). We use a training data, sampled i.i.d. from a joint distribution of (Y, X_R): $\mathcal{D} = \{(y^{(1)}, x_R^{(1)}), ..., (y^{(n)}, x_R^{(n)})\}$. We then fit a predictive model by minimising an empirical risk on the $\mathcal{D}$, defined by some inner loss $l(\cdot, \cdot)$:

$$h^* = \underset{h \in \mathcal{H}}{\operatorname{argmin}} \mathcal{R}^{\mathcal{D}}_{\text{emp}}(Y, h(X_R)) \qquad \mathcal{R}^{\mathcal{D}}_{\text{emp}} = \frac{1}{n} \sum_{i=1}^{n} l\left[y^{(i)}, h(x_R^{(i)})\right] \tag{3.1}$$

where $\{h \in \mathcal{H}\}$ is a predicitive models class. The main object of global interpretability is a **generalisation risk** $\mathcal{R}$ of model g^*, meaning the expected value of some loss for all possible values (Y, X_R):

$$\mathcal{R} = \mathbb{E}_{(Y, X_R)}\left[l\big(Y, h^*(X_R)\big)\right] \tag{3.2}$$

On practice, true generalisation risk is usually estimated with empirical risk (e.g. on the hold-out test subset: $\mathcal{D}^* = \{(y^{(1)}, x_R^{(1)}), ..., (y^{(n^*)}, x_R^{(n^*)})\}$).

The interpretability research poses a question: how can we measure the individual or collective contributions of features to the generalisation risk. While RFI estimates individual importance, conditioning on some context, SAGE tries to derive additive individual importances, which approximately restore collective contributions.

3.1.1 Relative Feature Importance

Let's first introduce a slightly different notation (see Figure 3.1), as originally proposed in [27]. Three methods (Permutation Feature Importance (PFI) [8], Conditional Feature Importance (CFI) [50] and Relative feature importance (RFI) [27]) check the individual importance of some feature X_j in different contexts. The main principle here is to introduce some kind of feature perturbation, one calls it a **replacement variable** $\tilde{X}_j$, which is completely independent of the target. Thus, it aims to destroy an existing relationship to Y. We can define the risk of permuted feature as:

$$\tilde{\mathcal{R}}^j = \mathbb{E}_{(Y,X_{-j},\tilde{X}_j)}\left[l\big(Y, h^*(\tilde{X}_j, X_{-j})\big)\right] \tag{3.3}$$

We define a feature importance as the difference of risk for perturbed data and original risk:

$$\mathrm{FI}_j = \tilde{\mathcal{R}}^j - \mathcal{R} \tag{3.4}$$

PFI evaluates the importance of feature $j \in R$ after applying a random permutation of a respective feature. By independent shuffling of feature values, X_j one wants to destroy the relationship between target sample Y and X_j. Permutation is actually equivalent to the sampling from the marginal distribution: $\tilde{X}_j \sim p(x_j)$.

Simple permutation can bring a bias – in cases when the feature is dependent on other train features. This has been referred as **off-manifold** problem (more detailed in Section 2). Imagine a dummy example with a medical dataset, when we have dependent features '*Sex*' and '*Number of pregnancies*'. After permuting the '*Sex*' feature we could generate completely unrealistic datapoints. Also, PFI can overestimate the importance of correlated features.

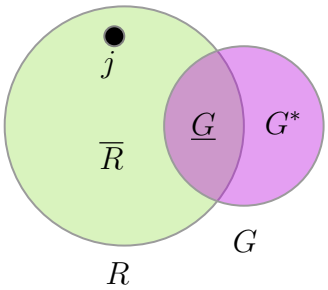


Figure 3.1: Venn diagram for PFI/RFI/CFI notation. $j \in R$ is a feature of interest, R is a set of training features, G is conditioning set for RFI. We mark the set of all predictive features but not j with $-j = R \setminus j$.

On the other hand, CFI uses a sample from the conditional distribution, given all the other training features: $\tilde{X}_j \sim p(x_j/x_{-j})$. Here, the context is taken as from the actual test sample: $x_{-j} = x_{-j}^{(i)*}$. The interpretation of conditional feature importance is the ultimate individual contribution of feature if we know all the other variables values. So CFI for the perfectly correlating features will be zero, as conditional sampling is changed to a deterministic assignment and no actual perturbation is introduced to X_j.

Relative feature importance [27] was introduced as a "spectral" continuation between PFI and RFI. We also want to calculate conditional importance, but now a set of conditioning variables G can range from an empty set (same as PFI) to a full $-j$ (=CFI). We can even add some features, which were missing or omitted during predictive model training – G^*.

$$\mathrm{RFI}_j^G = \tilde{\mathcal{R}}^{j/G} - \mathcal{R} \qquad \tilde{\mathcal{R}}^{j/G} = \mathbb{E}_{(Y,X_{-j},\tilde{X}_j^G)}\left[l\big(Y, h^*(\tilde{X}_j^G, X_{-j})\big)\right]$$

where $\tilde{X}_j^G$ is a replacement variable, sampled from a $\tilde{X}_j^G \sim p(x_j/x_G)$, independently from target Y and all the other variables $\overline{R}$.

In practice, we can only estimate ground-truth RFI with its Monte-Carlo estimate on the test subset: $\widehat{\text{RFI}}_j^G$. We can infer RFI analytically only under a very restricted model class – Multiple Linear regression, and specific risk, defined by a mean squared error (MSE).

Proposition 1. *Let the risk 3.2 be defined via MSE-loss* $l(y, g(x_R)) = (y - x_R)^2$. *Target* Y *is defined via a conditional linear regression formula:* $Y/X_R = \beta^T X_R + \varepsilon$, *where* $\varepsilon \sim N(0, \sigma^2)$. *Let's denote a given train* $\mathcal{D}$ *and test data* $\mathcal{D}^*$ *as design matrices:*

$$\begin{aligned} &\mathrm{X}, \mathrm{X}^* \sim p(X_R) \quad \mathrm{X} \in Mat(n \times r) \quad \mathrm{X}^* \in Mat(n^* \times r) \\ &\mathrm{y} = \mathrm{X}\beta + \varepsilon \quad \mathrm{y}^* = \mathrm{X}^*\beta + \varepsilon \qquad \varepsilon \sim N(0, \sigma^2 I) \end{aligned}$$

Then, under a mild condition: $\frac{X^{*T}X^*}{n^*} \approx \frac{X^TX}{n} = \widehat{Cov}(X_R)$ *and by increasing* n, *the RFI for the model with MLE-estimated parameters:* $h_{\hat{\beta}}(X_R) = \hat{\beta}^T X_R, \hat{\beta} = (\mathrm{X}^T\mathrm{X})^{-1}\mathrm{X}^T\mathrm{y}$:

$$\text{RFI}_j^G \approx 2\beta_i^2(\text{Var}(X_j) - \text{Cov}(\tilde{X}_j^G, X_j)) \tag{3.5}$$

where $\tilde{X}_j^G$ *is sampled from some conditional distribution* $\tilde{X}_j^G \sim p(x_j/x_G)$.

In the case of X_R *having a multivariate Gaussian distribution* $X_R \sim N(\mu_{X_R}, \Sigma_{X_R})$, $\tilde{X}_j^G$ *being sampled from true conditional distribution and* $G \subseteq R$:

$$\text{RFI}_j^G \approx 2\beta_i \left(\Sigma_{jj} - \Sigma_{jG}\Sigma_{GG}^{-1}\Sigma_{Gj}\right) \tag{3.6}$$

where $\Sigma_{jj}, \Sigma_{jG}, \Sigma_{Gj}, \Sigma_{GG}$ *is a partitioning of matrix* Σ_{X_R}.

Proof. See the Appendix A.1. □

Immediately, we can see how this result relates to the PFI and CFI. For PFI replacement variable is sampled independently of original data sample, so $\text{Cov}(\tilde{X}_j^G, X_j) = 0$ and feature importance reaches its highest value: $\text{PFI}_j \approx 2\beta_i^2(\text{Var}(X_j))$. Conversely, in case of feature j being a linear combination of features from G: $\text{Cov}(\tilde{X}_j^G, X_j) = \text{Var}(X_j)$ and importance tends to zero.

Unfortunately, as we practically never know the true conditional distribution, we replace it with the estimated sampler. Original RFI paper [27] experimented only with the multivariate Gaussian data and used model-X knockoffs [9] to estimate samplers parameters (which is equivalent to estimation and sampling from the conditional Gaussian distribution in case of RFI). Thus, the research gap arises – which is the main aim of the thesis.

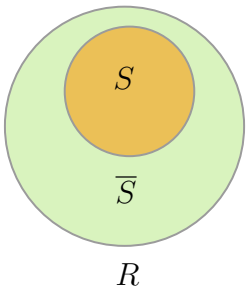


Figure 3.2: Venn diagram for SAGE notation. $S \subseteq R$ is a set of features of interest, R is a set of training features, $\overline{S} = R \setminus S$.

3.1.2 Shapley Additive Global Importance

Shapley Additive Global Importance (SAGE) was introduced by [12], as a global version of the local method **SHapley Additive exPlanations** (SHAP) [30]. SHAP and SAGE answer a different question: what is a collective importance of feature subsets. Unlike perturbation-based methods, where we change the individual features in the original

dataset to destroy relations with a target variable, SAGE deals with marginalising predictions and compares it with the full model. Authors introduced a concept of **reduced model**. Let's say, we are interested in the importance of subset $S \subseteq R$ of features and $\overline{S} = R \setminus S$. One defines a model, restricted on S, as:

$$h^*_S(x_S) = \mathbb{E}_{X_{\overline{S}} \sim p(x_{\overline{S}}/x_S)}\, h^*(x_S, X_{\overline{S}}) \tag{3.7}$$

which is in fact the mean prediction of the model after marginalisation of $X_{\overline{S}}$, conditionally on X_S. So intrinsically, we are also conditionally "perturbing" features in $X_{\overline{S}}$ and using them to calculate the mean prediction of the model. In this regard, SAGE is similar to the CFI (the only difference is that CFI averages generalisation risks of the original model on original and perturbed data and SAGE does it for original and restricted models).

Now we want to compare the risk between the average prediction (= model reduced on empty set, which e.g. outputs the mean of Y) and model, reduced on S:

$$v_{h^*}(S) = \mathbb{E}_{(Y,X_R)}\left[l\big(Y, h^*_\varnothing(X_\varnothing)\big)\right] - \mathbb{E}_{(Y,X_R)}\left[l\big(Y, h^*_S(X_S)\big)\right] = \tilde{\mathcal{R}}^\varnothing - \tilde{\mathcal{R}}^S \tag{3.8}$$

we call this **reduction in risk** over the mean prediction.

As there is an exponential number of different possible subsets w.r.t. number of features, SAGE and SHAP proposed to look for some kind of additive importance of individual features, which should approximate ground truth $v_{h^*}(S)$:

$$u_{h^*}(S) = \phi_0 + \sum_{i \in S} \phi_i \tag{3.9}$$

where ϕ_i are the individual assignments of feature importances. This importance can also be viewed from game theoretic perspective, where we want to find an optimal (in some sense) credit allocation scheme for the game with several players. In terms of interpretability, players are features and the game is reduction in risk, defined via 3.8. The **Shapely values** are such a unique credit allocation, which satisfies 5 optimality properties (efficiency, symmetry, dummy, monotonicity and linearity) and can be expressed in a closed form expression:

$$\phi_i = \frac{1}{r} \sum_{S \subseteq R \setminus \{i\}} \binom{r-1}{|S|}^{-1} \Big(v_{h^*}(S \cup \{i\}) - v_{h^*}(S) \Big) \tag{3.10}$$

Shapely values provide interesting and intuitive properties of feature sets importances (more detailed in original paper [12]).

The inference of exact Shapely values was studied for SHAP in [56]. The authors showed that computation of SHAP is intractable even for simple predictive models, such as logistic regression.

The original work on SAGE proposed a sampling-based approximation algorithm, which internally used sampling from $p(x_{\overline{S}}/x_S)$. Authors made a simplifying assumption, that $p(x_{\overline{S}}/x_S) = p(x_{\overline{S}})$, similarly as in SHAP paper. As this was already addressed in 2, this could lead to the off-manifold problem. Our work tries to eliminate this drawback and sample from properly estimated $p(x_{\overline{S}}/x_S)$.

3.2 Density Estimation

Learning a probabilistic model of data to create a realistic synthetic sample can always be framed as the task of density estimation. In this section, we discuss the existing methods for parametric density estimation, which also allow us to sample new data from estimated models. First, we introduce the general task of density estimation for unconditional and conditional distributions, then we discuss possible goodness-of-fit measures and their interpretations. Furthermore, we list possible solutions to overcome the overfitting problem.

3.2.1 General Task

Unconditional distribution. Let the X be a random variable of dimensionality d_x and $p(x)$ – a probability density function (PDF) of X, defined over domain $\mathcal{X} \subseteq \mathbb{R}^{d_x}$ (continuous) or $\mathcal{X} \subseteq \mathbb{N}^{d_x}$ (discrete). One calls a set $\text{supp}(p) = \{x \in \mathcal{X} : p(x) > 0\}$ – the support of $p(x)$. Given an i.i.d. sample from unknown $p(x)$: $\mathcal{D} = \{x^{(1)}, ..., x^{(n)}\}$ (can be also marked as $X^{(i)} \sim p(x)$), the goal is to find a good estimate $\hat{f}(x)$ (model, estimator). In deep learning context, one often calls it a generative model. Depending on the downstream task, one can also ask for possibility of sampling from model $\tilde{X} \sim \hat{f}(x)$ or the computability of cumulative density function (CDF): $\hat{F}_X(x)$.

In **parametric estimation** $\hat{f}(x)$ comes from some parametric family: $\mathcal{F} = \{f_\theta(x) : \theta \in \Theta\}$. The optimal $\hat{\theta}$ is usually selected via **Maximum likelihood estimation** (MLE), which tries to maximize the likelihood of data $\mathcal{D}$:

$$\hat{\theta}_{\text{MLE}} = \underset{\theta \in \Theta}{\text{argmax}} \sum_{i=1}^{n} \log f_\theta(x^{(i)}) \tag{3.11}$$

It can be shown, that the maximisation objective in 3.11 is equivalent to the minimization of Monte Carlo estimate of Kullback–Leibler (KL) divergence between the data generating PDF and model's PDF:

$$\begin{aligned}
\underset{\theta \in \Theta}{\text{argmin}}\, \text{KL}(p||f_\theta) &= \underset{\theta \in \Theta}{\text{argmin}}\, \mathbb{E}_{X \sim p(x)} \left[\log \frac{p(X)}{f_\theta(X)} \right] = \\
&= \underset{\theta \in \Theta}{\text{argmin}}\, \mathbb{E}_{X \sim p(x)} \big[\log p(X) \big] - \mathbb{E}_{X \sim p(x)} \big[\log f_\theta(X) \big] = \\
= \underset{\theta \in \Theta}{\text{argmax}}\, \mathbb{E}_{X \sim p(x)} \big[\log f_\theta(X) \big] &\approx \big[\text{MC estimate, using } \mathcal{D}\big] = \underset{\theta \in \Theta}{\text{argmax}} \frac{1}{n} \sum_{i=1}^{n} \log f_\theta(x^{(i)})
\end{aligned} \tag{3.12}$$

where $\mathbb{E}_{X \sim p(x)} \big[\log(p(X)) \big] = H(X)$ is a constant (differential) entropy and thus ignored while minimisation and $\log p(x)$ is by convention 0, if $p(x) = 0$.

By choosing the optimal $\hat{f} = f_{\hat{\theta}}$ with KL-divergence minimisation objective 3.12, we actually find a projection of data PDF on the selected model's class $\mathcal{F}$. This is also referred as moment projection or M-projection [35].

Conditional distribution. Let (X, Y) be the pair of possibly multivariate random variables (continuous or discrete) with the dimensionality d_x and d_y respectively. Let $p(y/x) = p(y, x)/p(x)$ be the conditional density of Y. The task of unconditional distribution estimation could be simply generalised to conditional case: having a training data from joint distribution $p(y, x)$ – $\mathcal{D} = \{(y^{(1)}, x^{(1)}), ..., (y^{(n)}, x^{(n)})\}$,

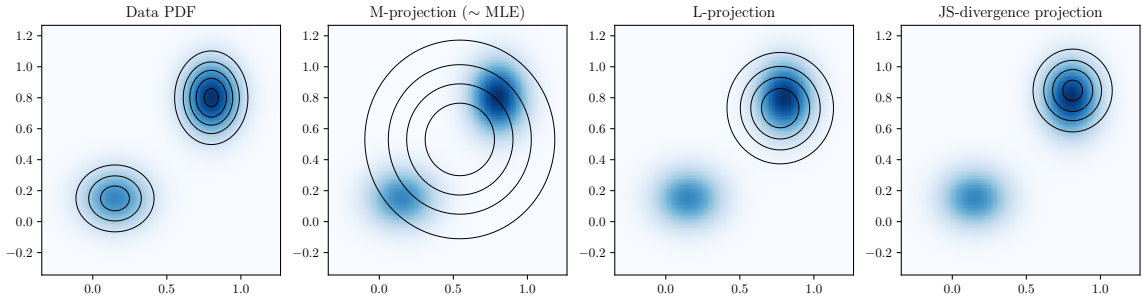


Figure 3.3: Fitting isotropic 2-d Gaussian distribution $f_\theta(\cdot) \sim N((\mu_0, \mu_1)^T, \sigma I)$ to non-Gaussian data PDF. Original PDF is a mixture of two Gaussian densities (leftmost image). One could observe, how different objectives (KL-divergence, reverse KL-divergence and JS-divergence) and respective projections (M-, L- and JSD) favour different fit properties.

the aim is to find a conditional estimator $\hat{f}(y/x)$, without the direct estimation of marginal distribution $p(x)$. Typically, X is called a conditioning variable or context and Y – dependent (input) variable.

In terms of parametric estimation, the context usually influences specific parameter values, which then define the density of Y: $f_\theta(y/x) = f_\theta(y/\theta = \theta(x))$. This allows to use the same models, as for unconditional estimation – we simply substitute an optimal $\hat{\theta}$ with values, dependent on context: $\hat{\theta} = \hat{\theta}(x)$. For the specific model designs, refer to Section 3.3.

One formulates MLE objective in the same manner to 3.11:

$$\hat{\theta}_{\text{MLE}} = \underset{\theta \in \Theta}{\text{argmax}} \sum_{i=1}^{n} \log f_\theta(y^{(i)}/x^{(i)}) \tag{3.13}$$

.

One can also show, that this MLE is equivalent to the minimisation of mean conditional KL-divergence:

$$\mathbb{E}_{X \sim p(x)} \, \text{KL}\left(p(y/X) || f_\theta(y/X)\right) = \mathbb{E}_{X \sim p(x)} \, \mathbb{E}_{Y \sim p(y/X)} \left[\log \frac{p(Y/X)}{f_\theta(Y/X)}\right] \tag{3.14}$$

3.2.2 Model Selection

Training objective. The choice of MLE for parameter estimation brings a huge disadvantage, especially during a models' class misspecification. As it was shown in the previous subsection 3.2.1, the MLE estimate is equivalent to the M-projection of data density on the models' class. Due to the asymmetrical nature of KL-divergence wrt. its arguments, M-projection tends to over-estimate the support of data PDF. Other objectives, like reversed KL-divergence (or L-projection = $\text{argmin}_{\theta \in \Theta} \text{KL}(\text{f}_\theta || \text{p})$) or Jensen-Shannon divergence can provide somewhat better estimate, which will result in more realistic sample from model's distribution. Unfortunately, they not only require the knowledge of true data PDF but also could under-estimate the support and "concentrate" around some mode of data distribution. This issue, as mentioned by [52], is visually explained on Figure 3.3: mean of isotropic Gaussian distribution, fitted on multi-modal data with MLE, lays far away from high-density regions. Thus, resampled data will be unnatural.

For all the models, introduced in this thesis, we solely use MLE (specifically a regularised version of it) as the main training objective, as deep estimators' class is flexible enough to mimic complex non-gaussian PDFs.

Regularisation techniques. For high-dimensional parameter spaces or in low-data regimes MLE estimate 3.11 could have a poor generalisation or overfitting problem. While numerous approaches can be inherited from predictive modelling, such as L_1-/L_2-penalisation, weight decay, dropout or variational inference with Bayesian priors, it is unclear, what kind of inductive biases can be useful for (conditional) density estimation.

Several recent works proposed use variational inference [54] or **noise regularisation** [42] to tackle this issue. We found the last one to work well on all of the benchmarks, as variational estimation requires a selection of reasonable prior parameters. Following [42], we change the original datapoints $(y^{(i)}, x^{(i)})$ with the their noised version:

$$\tilde{y}^{(i)} = y^{(i)} + \xi_y, \quad \tilde{x}^{(i)} = x^{(i)} + \xi_x, \quad \xi_x \sim K_x(\xi_x) \quad \xi_y \sim K_y(\xi_y), \tag{3.15}$$

where ξ_x and ξ_y are noise variables, i.i.d. sampled from zero centered and uncorrelated distributions K_x and K_y:

$$\mathbb{E}\,\xi = 0 \quad \mathrm{Cov}(\xi, \xi) = \sigma^2 I \tag{3.16}$$

Here σ denotes the standard deviation of noise and should be fine-tuned for both K_y and K_x.

In the context of MLE estimation based on mini- or full-batch gradient descent, the regularising noise is sampled for every epoch / datastep (see Appendix A.2 for the algorithm description).

Authors of [42] showed that noise regularisation in the context of density estimation is equivalent to a smoothness constraint. By adding noise, we in fact add concavity penalisation terms to the log-likelihood objective, which favour more smooth densities: both in terms of dependent variable smoothness and contextual dependency smoothness. This is nicely demonstrated on Figure 3.4.

Surprisingly, Gaussian noise regularisation was also useful in the case of discrete conditioning variables, present as one-hot encoded context, even though it was not directly studied by original paper [42]. Previous literature on adding noise to discrete conditioning variables X_i, e.g. [31], only aims at predictive modelling.

Goodness-of-fit metrics. Numerous goodness-of-fit (GoF) metrics exist to evaluate unconditional generative models and some of them can be used for conditional case. Two main groups can be distinguished: sample-based and density-based metrics. The first one uses two (or more) samples X_i and $\tilde{X}_i$, drawn from p and $\hat{f}$ respectively, and does not require the explicit knowledge of density estimate $\hat{f}$. Examples include Maximum mean discrepancy (MMD) [19], Energy distance [41] or Wasserstein distance [4]. Second density-based group explicitely uses $\hat{f}$ (or estimated CDF $\hat{F}_X(x)$), e.g. Minimum distance estimation (Chi-squared [38], Kolmogorov-Smirnov [37] distances), Hellinger distance [20] or Maximum likelihood estimation (negative log-likelihood). We make a comparison of popular existing conditional/unconditional GoF metrics in Table 3.1 in terms of scalability to high-dimensional variables and value bounds.

Out of all the metrics, we utilised Hellinger distance, Kullback-Leibler and Jensen-Shanon divergences for synthetic datasets (see Section 4.1.1) and negative log-likelihood (NLL) – for all synthetic and real benchmarks. To provide an unbiased estimate of theoretical GoF, the easiest practical choice is an empirical estimate

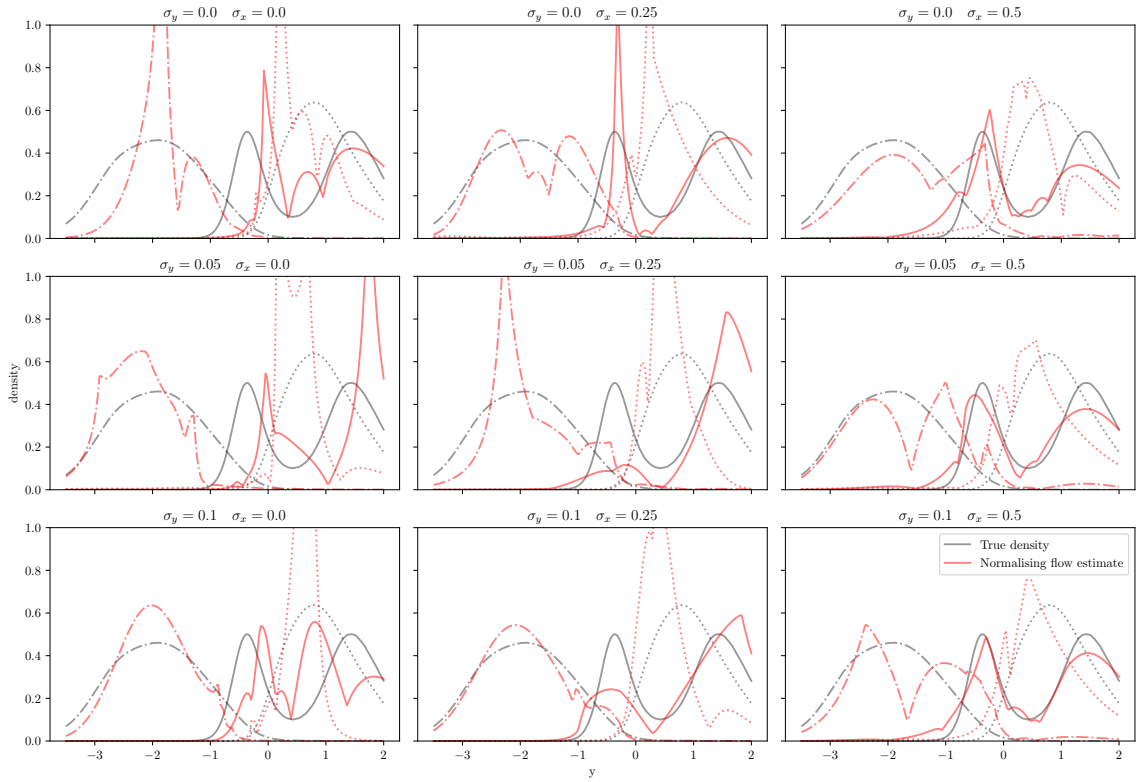


Figure 3.4: Effect on noise regularisation. 10-layers Conditional normalising flow (CNF) estimator $f_\theta(y/x_i)$ (red) and true densities $p(y/x_i)$ (grey) for three test contexts $i \in \{1, 2, 3\}$ (solid, dotted and dash-dotted lines). Both context and input noises ($\sigma_x \in \{0.0, 0.25, 0.5\}$ and $\sigma_y \in \{0.0, 0.05, 0.1\}$) contribute to a smoothness of estimated density. CNF was fitted on 250 samples of *RandomGP-GaussianNoise* benchmark (see Section 4.1.1) with one input Y and 4 contextual (X_1, X_2, X_3, X_4) variables.

GoF Metric	**Conditional**	**High-dim**	**Values bounds**	**Issues**
		Sample-based		
Maximum mean discrepancy	+ (density-based) [23]	+	$[0, +\infty)$	Bandwidth and kernel selection
Wasserstein distance	+ [49]	~ (Gaussian / dual approximations)	$[0, +\infty)$	Not suitable for conditional GoF
Energy distance	–	+	Can be normalised to $[0, 1)$	Not suitable for conditional GoF
		Density-based		
Minimum distance estimation metrics	+ [3; 60]	–	$[0, +\infty)$	Bandwidth selection
Negative log-likelihood	+ [48; 54]	+	$(-\infty, +\infty)$	Lower bound defined by unknown entropy
Hellinger distance	+ [43]	~ (MC-estimate)	$[0, 1]$	Requires $p(x)$ or $p(y/x)$
Kullback-Leibler divergence	+	~ (MC-estimate)	$[0, +\infty)$	Requires $p(x)$ or $p(y/x)$
Jensen-Shanon divergence	+	~ (MC-estimate)	$[0, \log 2]$	Requires $p(x)$ or $p(y/x)$

Table 3.1: A small overview of existing goodness-of-fit metrics. Columns legend: Conditional – suitability for conditional density estimation, High-dim – computational viability of metric wrt. high-dimensionality of dependent variable (~ means that computation requires additional approximations or assumptions), Values bounds – metric values span, Issue – limitations or disadvantages of metric.

based on hold-out test subset: $\mathcal{D}^* = \{(y^{(1)}, x^{(1)}), ..., (y^{(n^*)}, x^{(n^*)})\}$. Here n^* is the

size of the test set. Until further notice, we shift to the conditional estimation formulation in the following sections.

The simplest measure is the average test **negative log-likelihood (NLL)**, which uses the same training objective 3.13:

$$\text{NLL} = -\frac{1}{n^*}\sum_{i=1}^{n^*} \log f_\theta(y^{(i)}/x^{(i)}) \tag{3.17}$$

Test log-likelihood is one the most popular tools to estimate GoF of different generative models: in language modelling [29], structured output learning [48] or image/audio generation [11]. Unfortunately, as it was mentioned in [52], it is hard to tell, what absolute values of log-likelihood produce plausible "natural-looking" samples, especially in high-dimensional scenarios. Conversely, a high-quality sample, drawn from the estimated model, could have poor log-likelihood (e.g. if the model is a look-up-table).

In synthetic benchmarks, we can use more plausible analytic GoF metrics, which require the knowledge of true data generating PDF: conditional **Hellinger distance** (HD), conditional **Kullback-Leibler** (KL) divergence and conditional **Jensen-Shanon** (JS) divergence.

$$\text{HD}\left(p(y/x)||f_\theta(y/x)\right) = \mathbb{E}_{X\sim p(x)} \sqrt{\frac{1}{2}\int_{\mathcal{Y}} \left(\sqrt{p(y/X)} - \sqrt{f_\theta(y/X)}\right)^2 dy} \tag{3.18a}$$

$$\text{KL}\left(p(y/x)||f_\theta(y/x)\right) = \mathbb{E}_{X\sim p(x)}\, \text{KL}\left(p(y/X)||f_\theta(y/X)\right) \tag{3.18b}$$

$$\begin{aligned}\text{JS}\left(p(y/x)||f_\theta(y/x)\right) =& \mathbb{E}_{X\sim p(x)}\, \frac{1}{2}\,\text{KL}\left(p(y/X)\middle|\middle|\frac{1}{2}(p(y/X) + f_\theta(y/X))\right) + \\ &+ \frac{1}{2}\,\text{KL}\left(f_\theta(y/X)\middle|\middle|\frac{1}{2}(p(y/X) + f_\theta(y/X))\right)\end{aligned} \tag{3.18c}$$

On practice, they are approximated with MC-estimates based on test subset:

$$\widehat{\text{HD}} = \frac{1}{n^*}\sum_{i=1}^{n^*} \sqrt{\frac{1}{2}\int_{\mathcal{Y}} \left(\sqrt{p(y/x^{(i)})} - \sqrt{f_\theta(y/x^{(i)})}\right)^2 dy} \tag{3.19a}$$

$$\widehat{\text{KL}} = \frac{1}{n^*}\sum_{i=1}^{n^*} \text{KL}\left(p(y/x^{(i)})||f_\theta(y/x^{(i)})\right) \tag{3.19b}$$

$$\begin{aligned}\widehat{\text{JS}} =& \frac{1}{n^*}\sum_{i=1}^{n^*} \frac{1}{2}\,\text{KL}\left(p(y/x^{(i)})\middle|\middle|\frac{1}{2}(p(y/x^{(i)}) + f_\theta(y/x^{(i)}))\right) + \\ &+ \frac{1}{2}\,\text{KL}\left(f_\theta(y/x^{(i)})\middle|\middle|\frac{1}{2}(p(y/x^{(i)}) + f_\theta(y/x^{(i)}))\right)\end{aligned} \tag{3.19c}$$

The apparent advantage of these GoF metrics (3.18a, 3.18b, 3.18c) is that they are lower-bounded by 0 – for models which perfectly fit the data. Negative log-likelihood (3.17) is, on other hand, lower-bounded by the (differential) entropy, which is unknown together with the true data PDF. Thus, although NLL is useful to compare different models or model classes, one can never know, how far is the chosen model class from true distribution in "absolute" values.

Two metrics (3.18a, 3.18c) additionally have upper-bounds, which could be reached for example for two distributions, having different non-overlapping supports (KL-divergence and NLL for them will be infinite).

3.3 Deep Conditional Density Estimators

While there has been a pleiad of conditional generative models in deep learning, many of them solely concentrate on high-dimensional sample generation, predictive modelling or uncertainty estimation (e.g. Bayesian neural networks [7], Conditional Generative adversarial network [33] or Conditional Variational Autoencoder [48]). For the sake of ML interpretability, we have the following requirements for a generative model:

- Tractable conditional density. Although in the scope of feature importance estimation, the only aim of a conditional estimator is to draw a synthetic sample, a tractable density allows for the model selection (e.g. with negative log-likelihood).
- Easy and exact sampling. Even by having the tractable estimated density, one still needs to effectively draw samples from the model. Omitting the importance or rejection sampling is preferable.
- Generalisation in low data regimes. Deep density estimator should outperform traditional density estimation on tabular low-dimensional data (e.g. UCI Machine Learning repository [13]).

It turned out, that not many neural-network-based methods fit all these requirements – see Table 3.2.

Parametric model	**Tractable density** $f_\theta(y/x)$	**Exact Sampling** $\tilde{Y} \sim f_\theta(y/x)$	**Tractable CDF** $F_\theta(y/x)$	**Tractable quantile function** $F_\theta^{-1}(q/x)$
Latent variable NNs (cVAE [48], cGAN [33])	–	+	–	–
Bayesian NNs [7]	–	+	–	–
Mixture Density Networks [6]	+	+	+	–
Conditional Normalising Flow [54; 62]	+	+	+	+

Table 3.2: Comparison of deep parametric conditional generative models.

The results of studies [42; 43] on different synthetic and real low-dimensional datasets motivated us to employ two models for interpretability research: **Conditional Normalising Flows** (CNFs), as a main method of the thesis, and **Mixture Density Networks** (MDNs), as a concurrent model for comparison. We successfully reproduced the reported performance of both estimators: see Appendix A.7 and found that these neural estimators are flexible enough to capture different non-linearities and heteroscedasticity. At the same time, they don't require tedious hyperparameter selection and can be often used in a vanilla setting. Additionally, before-mentioned papers experiment only with one-dimensional density estimation, while our work extends it to multi-dimensional, e.g. in the context of SAGE calculation.

3.3.1 Conditional Normalising Flows

Normalising flows (NFs) [51] are a flexible tool to model complex densities, which can capture such distribution properties as heavy tails, multimodality or

heteroscedasticity. They were introduced to deep learning in the scope of deep variational inference to model approximate posterior distributions [40]. By having a **tractable density** and easy sampling mechanism, they can also be used in numerous downstream tasks of synthetic data generation, anomaly detection or data imputation.

A normalising flow describes the change of density of continuous random variable after applying a sequence of invertable mappings. Given a random variable Z with some known PDF $Z \sim p_Z(z)$ (e.g. Gaussian or uniform), we define a transformed variable X:

$$X = t(Z) \quad Z \sim p_Z(z) \tag{3.20}$$

Here $t(\cdot) : Z \to X$ denotes a forward transformation and $t^{-1}(\cdot) : X \to Z$ – inverse. Note, that the transformation is defined between spaces of same dimensionality: $d_Z = d_X$. After applying a multivariate **change of variables formula**, we can find a distribution of X:

$$p_X(x) = p_Z(z)\left|\det \frac{dZ}{dX}\right| = p_Z(t^{-1}(x))\left|\det \frac{dt^{-1}}{dX}(x)\right| \tag{3.21}$$

where $\det \frac{dt^{-1}}{dX}$ is a determinant of Jacobian of inverse transformation $t^{-1}(\cdot)$. Due to **inverse function theorem**:

$$\frac{dt^{-1}}{dX} = \left(\frac{dt}{dZ}\right)^{-1}$$

so the Jacobian of inverse transformation can be substituted with the inverse Jacobian of forward transformation. Then 3.21 can be simplified:

$$p_X(x) = p_Z(z)\left|\det\left(\frac{dt}{dZ}(z)\right)^{-1}\right| = \left[\text{Property of determinant; } z = t^{-1}(x)\right] =$$
$$= p_Z(t^{-1}(x))\left|\det \frac{dt}{dZ}(t^{-1}(x))\right|^{-1} \tag{3.22}$$

The name "normalising" comes from the fact, that any complex continuous distribution X could be transformed to a normal Z with a specific $t^{-1}(\cdot)$. In univariate case, this is a solution of partial differential equation 3.21:

$$t^{-1}(X) = \pm F_Z^{-1}(F_X(X)) \tag{3.23}$$

where $F_X(x)$ is unknown data CDF of X and $F_Z^{-1}(q)$ is quantile function of Z. This was referred as Gaussianisation in [10].

On practice, we can construct arbitrarily complex densities by applying a composition of K transformations $t_1, t_2, ..., t_K$:

$$X = Z_K = t_K(Z_{K-1}) = t_K(t_{K-1}(Z_{K-2})) = ... = t_K \circ ... \circ t_1(Z_0) \tag{3.24}$$

where Z_0 has known base distribution. One calls this chain of transformations a **flow** (Figure 3.5). Now, density of X can be recursively found as:

$$p_{Z_K}(z_K) = p_{Z_{K-1}}(z_{K-1})\left|\det \frac{dt_K}{dZ_{K-1}}(z_{K-1})\right|^{-1} = p_{Z_0}(z_0)\prod_{k=1}^{K}\left|\det \frac{dt_k}{dZ_{k-1}}(z_{k-1})\right|^{-1} \tag{3.25}$$

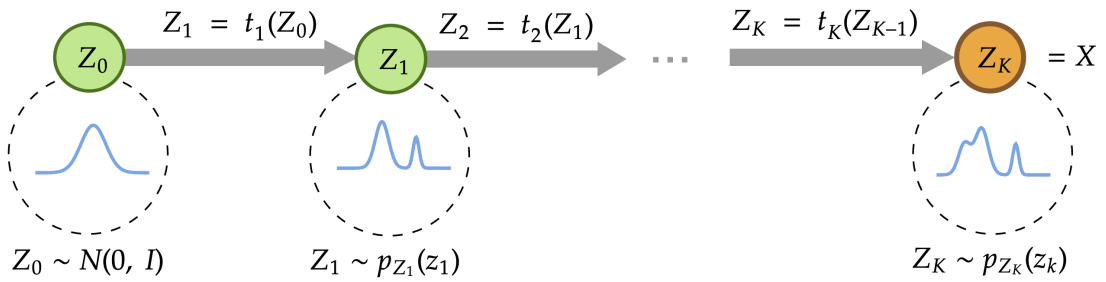


Figure 3.5: Illustration [61] of unconditional normalising flow, which transforms base normal distribution $p_{Z_0} = N(0, I)$ to complex one p_{Z_K}.

where $z_0, z_1, ..., z_K$ are found via 3.24.

Also, log-probability is now simply:

$$\log p_{Z_K}(z_K) = \log p_{Z_0}(z_0) - \sum_{k=1}^{K} \log \left| \det \frac{dt_k}{dZ_{k-1}}(z_{k-1}) \right| \tag{3.26}$$

Now, we can simply calculate the log-likehood of the observation $x^{(i)} = z_K^{(i)}$: one needs to push it through inverse flow to original base distribution space and memorise all intermediate values $z_{K-1}^{(i)}, z_{K-2}^{(i)}, ..., z_0^{(i)}$. Then, we just plug in those values to Jacobians of forward transformations in 3.26 and calculate log-likehood of $z_0^{(i)}$.

If we introduce a parametrisation θ of transformations $t_1, t_2, ..., t_K$, we can directly apply gradient descent methods to optimise MLE objective 3.11:

$$\hat{\theta}_{\text{MLE}} = \underset{\theta \in \Theta}{\text{argmax}} \sum_{i=1}^{n} \log f_\theta(x^{(i)}) = \underset{\theta \in \Theta}{\text{argmin}} \left[- \sum_{i=1}^{n} \log p_{Z_K}(x^{(i)}) \right] \tag{3.27}$$

This loss is different from traditional losses like mean squared error or cross-entropy, simply because the number of terms increases with the depth of the flow. Notably, the only limitation in the choice of transformations, is that we need to know an analytic formula for the determinant of Jacobian and inverse transformation, and both should take a linear computation time. In principle, transformations could be themselves an invertible neural network (e.g. [5]), but then the calculation of Jacobian becomes computationally expensive.

We can understand forward flow as a series of transformations, which squish or expand base density p_{Z_0} to form the complex multimodal distribution: see Figure 3.6.

Sampling from normalising flows is performed via drawing a sample from the base distribution $\tilde{Z}_0 \sim p_{Z_0}(z_0)$ and then pushing it through the forward flow [1] 3.24.

Conditional normalising flows (CNFs) [54; 62] are a direct extension of unconditional – they employ an addintional neural network to produce a context-dependent parameters of transformations.

[1] In practice, forward flows are used to compute log-likelihood and fit the model with 3.27 and reverse transformations – to perform sampling. Reverse transformation is typically more computationally expensive than forward.

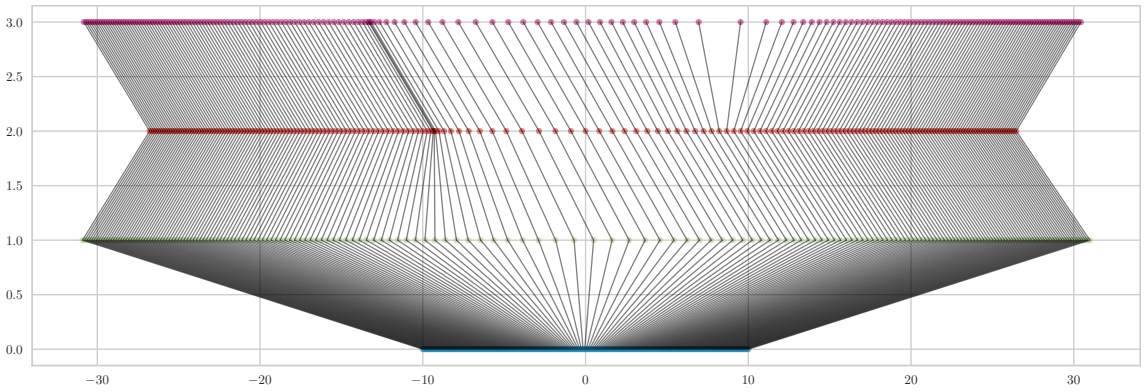


Figure 3.6: 200 equidistant points (row 0.0) on interval $[-10.0, 10.0]$, transformed with 3 layers of invertable radial flow (rows 1.0, 2.0 and 3.0 respectivelly). We can observe, how radial tranformations squish or expand initial uniform space.

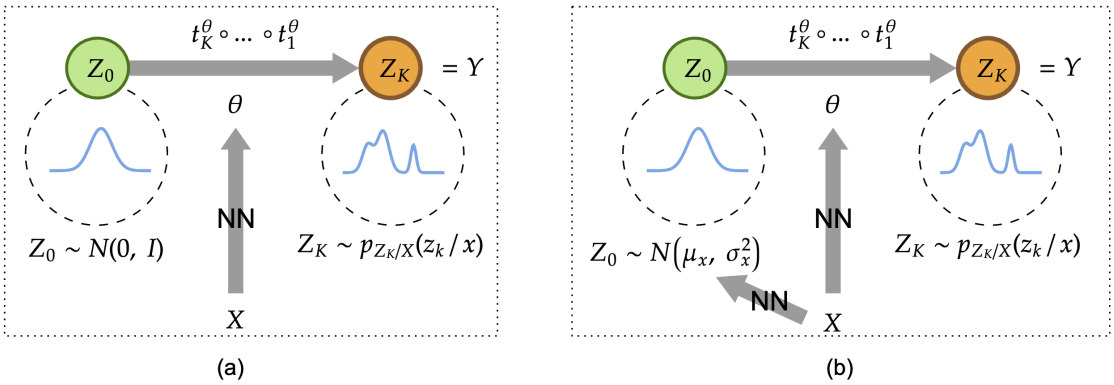


Figure 3.7: Two possible designs for normalising flows conditional estimation [54; 62]. In (a) one uses a contextual variable X to generate parameters of normalising flow $\theta = \mathrm{NN}_{\tilde{\theta}}(x)$, in (b) – X defines both flow parameters and base distribution parameters: $\theta = \mathrm{NN}_{\tilde{\theta}_1}(x)$ and $\{\mu_x, \sigma_x^2\} = \mathrm{NN}_{\tilde{\theta}_2}(x)$.

In this thesis, we inherit the design of [54], where the contextual variable X influences only transformations parameters (see Figure 3.7(a)), but not the base distribution itself. Both architectures are in fact universal density estimators and thus the simpler design (b) is chosen:

$$\theta = \mathrm{NN}_{\tilde{\theta}}(x) \tag{3.28}$$

where NN is a neural network with parameters $\tilde{\theta}$. Similarly to 3.27, we aim to minimise a negative conditional log-likehood:

$$\hat{\tilde{\theta}}_{\mathrm{MLE}} = \underset{\tilde{\theta} \in \tilde{\Theta}}{\operatorname{argmin}} \left[-\sum_{i=1}^{n} \log p_{Z_K}(y^{(i)} / \theta = \mathrm{NN}_{\tilde{\theta}}(x^{(i)})) \right] \tag{3.29}$$

Originally, authors of [54] used variational inference to prevent overfitting of log-likelihood, by putting Gaussian priors on neural network parameters $\tilde{\theta}$ as well as on latent features θ. On the contrary, we were not able to reproduce their results and thus used a noise regularisation, introduced in Section 3.2.2.

In the experiments, we employ two types of transformations: **affine** (linear) and **radial** (non-linear). It also possible to use other non-linear transformations, such as planar [40] or cubic splines [14], but the first two were proved to work well on many real and synthetic datasets.

Affine transformation. Let $a, b \in \mathbb{R}^{d_Z}$ be a log-scale and shift parameters. A point-wise affine transformation is defined as:

$$t(Z) = \exp(a) \odot Z + b \qquad t^{-1}(Z) = (Z - b)/\exp(a) \tag{3.30}$$

where $\odot, /$and exp are pointwise multiplication, division and exponentiation. Then, Jacobian and its log absolute determinant are:

$$\frac{dt}{dZ} = \text{diag}(\exp(a))$$
$$\log \left| \det \frac{dt}{dZ} \right| = \log \left| \det \text{diag}(\exp(a)) \right| = \log \prod_{i=1}^{d_Z} \exp(a_i) = \sum_{i=1}^{d_Z} a_i$$

Exponentiation of scale allows to omit zero division problem while optimisation and inference. Applying this transformation to a Gaussian distribution still leaves it Gaussian, thus we need to use some kind of nonlinearity.

Radial transformation. Let $\alpha \in \mathbb{R}^+$, $\beta \in \mathbb{R}$ be compress and expand parameters and $\gamma \in \mathbb{R}^{d_Z}$ be a reference point. Following the original parametrisation of [54], one defines a radial transformation as:

$$t(Z) = Z + \frac{\alpha\beta(Z - \gamma)}{\alpha + ||Z - \gamma||_2} \tag{3.31}$$

where $||\cdot||_2$ is L_2-norm of a vector. To understand the meaning of parameters, let's set $R = (Z - \gamma)$ and write down a Jacobian (see the derivation in Appendix A.3):

$$\frac{dt}{dZ} = (1 + \frac{\alpha\beta}{\alpha + ||R||_2})I - \frac{\alpha\beta}{||R||_2(\alpha + ||R||_2)^2} RR^T$$
$$\log \left| \det \frac{dt}{dZ} \right| = (d_Z - 1) \log \left| 1 + \frac{\alpha\beta}{\alpha + ||R||_2} \right| + \log \left| 1 + \frac{\alpha^2\beta}{(\alpha + ||R||_2)^2} \right|$$

Radial transformation applies expansion or contraction around a reference point γ. The first shape parameter α controls how fast $\frac{dt}{dZ}$ decays to identity transformation from the reference point. As $\alpha \to 0$, transformation colapses to identity and $\alpha < 0$ makes it non-monotonic (and thus, non-invertable). Second parameter β influences magnitude and direction of distortion, by either pushing points to γ, when $\beta > 0$, or expanding them from γ, when $\beta \in (-1, 0)$. Analogically, we can ensure monotonicity with $\beta > -1$.

Constraints $\alpha > 0$ and $\beta > -1$ can be achived via handy reparametrisation with $\hat{\alpha}, \hat{\beta} \in \mathbb{R}$ [54]:

$$\alpha = \log\big(\exp(\hat{\alpha}) + 1\big) \qquad \beta = \exp(\hat{\beta}) - 1 \tag{3.32}$$

According to [54], $\hat{\beta}$ has an interpretation of a maximum change in log-density of points in the base distribution.

Radial transformation is a very appealing choice to model complex distributions, as it has only $K \times (d_Z + 2)$ parameters. For sake of sampling from radial flow, after reparametrisation 3.32 we also derived the formula for the inverse transformation $t^{-1}(Z)$ (see Appendix A.3, 6.15). **To our knowledge, it is the first application of radial flows to perform sampling**. Although inverse transformation is more computationally complex, it serves well in the scope of interpretability methods.

Both affine and radial transformations should function for multi-dimensional conditional estimation. Although, originally authors [54] proposed to use several parallel flows, where the outputs of ones are fed as contextual features to others. In this way, they factorised a multivariate joint density as a product of conditionals. On the other hand, we employed a simplified approach by using multivariate transformations.

3.3.2 Mixture Density Networks

As a concurent neural density estimation approach we choose Mixture density networks (MDNs), proposed by [6]. Specifically, we use a multivariate Gaussian mixture with diagonal covariance matrix and C mixing components:

$$f_\theta(y/x) = \sum_{i=1}^{C} \pi_x^{[i]} p_{N_x^{[i]}}(y) \tag{3.33}$$

where $p_{N_x^{[i]}}(y)$ are PDFs of multivariate normal distributions $N(\mu_x^{[i]}, \Sigma_x^{[i]})$ with mean $\mu_x^{[i]}$ and diagonal covariance $\Sigma_x^{[i]} = \text{diag}\{{\sigma_x^{[i]}}^2\}$; $\pi_x^{[i]}$ are mixing weights ($\sum_i \pi_x^{[i]} = 1$). All the parameters of mixture are produced by two neural networks:

$$\{\mu_x^{[i]}, \Sigma_x^{[i]}\} = \text{NN}_{\tilde{\theta}_1}(x) \qquad \{\pi_x^{[i]}\} = \text{NN}_{\tilde{\theta}_2}(x) \tag{3.34}$$

where $\tilde{\theta} = \{\tilde{\theta}_1, \tilde{\theta}_2\}$ are parameters of neural networks. To ensure, weights are summing to 1, we apply a softmax nonlinearity, as the last layer of $\text{NN}_{\tilde{\theta}_2}$. Also $\Sigma_x^{[i]}$ needs to be positively defined, so we additionally exponentiate the values of neurons, leading to the variance parameters in $\text{NN}_{\tilde{\theta}_1}$. No restrictions are put on means $\mu_x^{[i]}$ and thus no non-linearity is applied to corresponding neurons. MDNs are trained with the MLE objective, as discussed previously in Section 3.11.

In order to sample from MDN, one needs to first select the mixing component $\tilde{c}$, by sampling it from categorical distribution $\tilde{c} \sim \text{Cat}(\pi_x^{[i]})$. Then we draw the sample from corresponding Gaussian distribution $\tilde{Y} \sim N(\mu_x^{[\tilde{c}]}, \Sigma_x^{[\tilde{c}]})$.

3.3.3 Categorical & Mixed Estimation

Let's say we want to model a Categorical distribution with M classes. Modelling finite discrete data (categorical) with a continuous estimator could lead to degenerate solutions and achieve extreme values of test log-likelihood by locating density spikes around discrete values [55].

There exist discrete normalising flows [53], which transform discrete distributions, or one can use variational dequantisation to transform discrete distribution into contionious [21]. But we can apply a much simpler model, by producing a Categorical distribution parameters with a neural network (similarly to MDNs 3.3.2):

$$\{\pi_x^{[i]}\} = \text{NN}_{\tilde{\theta}}(x) \qquad \sum_{i=1}^{M} \pi_x^{[i]} = 1 \tag{3.35}$$

Note, that the last layer outputs of NN should also be transformed with softmax non-linearity.

Recently [24] proposed Copula Flows, which can tackle multivariate mixed-variable density estimation, by employing the distributional transform [44] (similar to probability integral transform for continuous variables). They introduce a universal density estimator, by modelling a joint copula density and univariate marginals with spline-based normalising flows.

Section 4

Evaluation and Results

In this section, we compare different conditional density estimators and study their effect on feature importance evaluation on different synthetic, semi-synthetic and real datasets. Additionally, we study, how deep density estimators can help to discover the influence of sensitive features.

While designing a benchmark and planning experiments, we follow the guidelines, defined by [34]. Mainly, one needs to report MC-estimates of different estimands (mean and standard deviation/median and confidence intervals), by using several random seeds.

The benchmarking strategy consists of two main stages: density estimation and feature importance evaluation.

Density estimation. In the majority of experiments we employ the following models with the basic setting, without performing a dataset-specific hyperparameter tuning:

- *CondGauss*. Assuming, that the data $(X^{(i)}, Y^{(i)})$ comes from joint Gaussian distribution, we can use MLE estimates of joint mean and covariance $(\hat{\mu}, \hat{\Sigma})$ to model conditional distribution:

$$
\begin{aligned}
& f_{\hat{\theta}}(y/x) = p_{N_{\hat{\theta}}}(y/x) \\
& N_{\hat{\theta}} = N\big(\hat{\mu}_y + \hat{\Sigma}_{yx}\hat{\Sigma}_{xx}^{-1}(x - \hat{\mu}_x), \quad \hat{\Sigma}_{yy} - \hat{\Sigma}_{yx}\hat{\Sigma}_{xx}^{-1}\hat{\Sigma}_{yx}\big) \\
& \hat{\mu} = \begin{pmatrix} \hat{\mu}_y \\ \hat{\mu}_x \end{pmatrix} \quad \hat{\Sigma} = \begin{pmatrix} \hat{\Sigma}_{yy} & \hat{\Sigma}_{yx} \\ \hat{\Sigma}_{xy} & \hat{\Sigma}_{xx} \end{pmatrix}
\end{aligned}
$$

- *VanillaCNF*. Conditional normalising flow (see Section 3.3.1) with one affine transformation followed by two radial transformations ($K = 3$) and standard Gaussian distribution $N(0, 1)$ as the base distribution. The contextual network has one hidden layer with 16 nodes and a hyperbolic tangent as a hidden non-linearity.

- *VanillaMDN*. Mixture density network (see Section 3.3.2) with $C = 5$ components. Mixing components networks both have one hidden layer with the number of nodes equal to the dimensionality of context and use Exponential Linear Unit (ELU) as hidden non-linearity. Additionally, heavy-tailed distributions sometimes caused mixture components parameters to go to infinity,

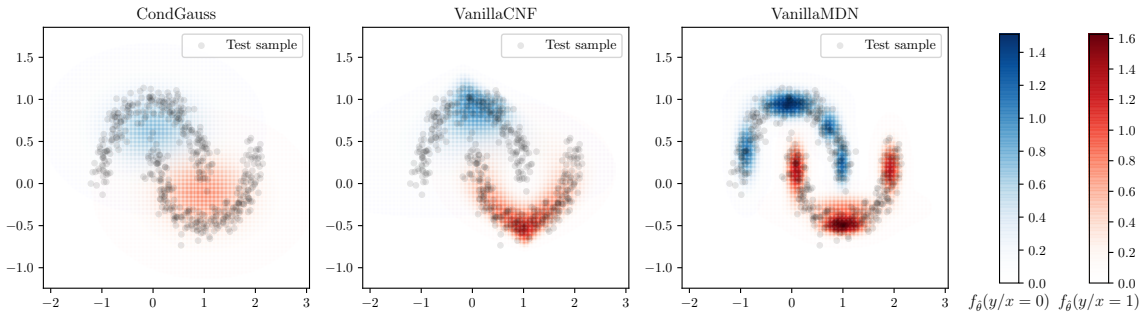


Figure 4.1: *CondGauss*, *VanillaCNF* and *VanillaMDN* fitted on moons toy dataset (two interleaving half circles), based on 2000 training datapoints. 2-dimensional distribution is conditioned on binary variable (upper half-circle $X = 0$ and lower: $X = 1$). Note, that *CondGauss* is a misspesified model, because (Y_1, Y_2, X) are not normally distributed.

so for SynTReN generator, we employed weight decay of 10^{-3} (0.05 for Sachs-2005) together with early stopping, based on undefined loss values. Also, $C = 4$ for Sachs-2005.

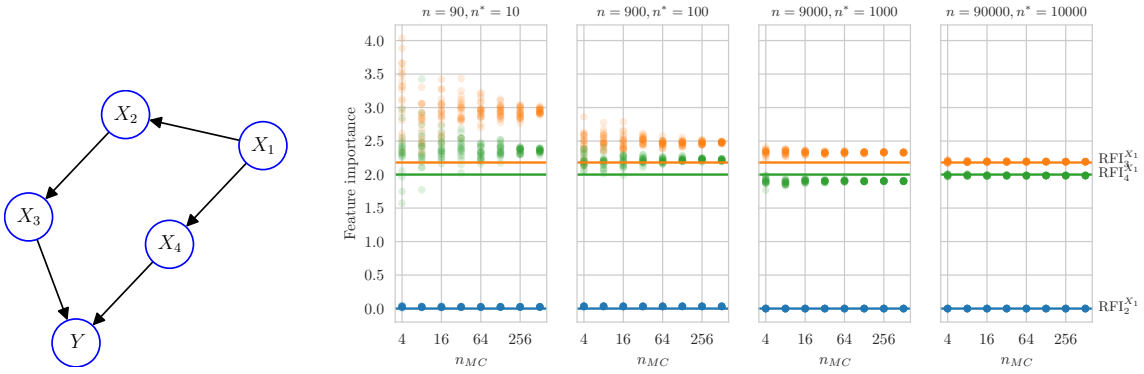
Both *VanillaCNF* and *VanillaMDN* are fitted with full-batch gradient descent with noise regularisation (see Appendix A.2), using Adam optimiser [26] with parameters $\beta_1 = 0.9, \beta_2 = 0.99$ and learning rate $\alpha = 0.001$. Number of epochs is $N_e = 1000$ and noise intensities are $\sigma_x = 0.1, \sigma_y = 0.05$. Even though these two models have very few parameters, they are still capable to fit complex distributions, e.g. moons dataset (Figure 4.1).

Additionally, for Sachs-2005 dataset (see Section 4.1.2) we do a study of hyperparameter tuning for CNFs and MDNs and other possible enhancements, such as different batch sizes or support transformations for CNF.

Feature importance evaluation. It is hard to access ground-truth feature importance values. As we saw in Section 3.1.1 - they could be easily derived for the simple linear regression with MSE risk.

In principle, it is possible to make an MC-estimate for feature importance, but we need to have access to ground-truth conditional distribution. It turns out, a very small amount of multivariate synthetic generating mechanisms allow to perform so-called **conditional queries** – inferring conditional probability distributions for the arbitrary sets of dependent and contextual variables (see Section for details 4.1.1). Thus, in the majority of cases, we can only provide the MC-estimate of the feature importance, based on the estimated sampler. Let's note n_{MC} as the number of times the replacement variable was resampled.

To see, how it influences the estimation, let's review the toy example introduced in [27] (see Figure 4.2). There are four regressors X_1, X_2, X_3, X_4 and a target variable Y, defined by the linear Gaussian structural causal model (SCM) (see Figure 4.2 (a)), with coefficients equal to 1 and additive noise $\sigma_1 = \sigma_2 = \sigma_4 = 1, \sigma_3 = 0.3$ and $\sigma_y = 0.5$. We select ordinary least-squares linear regression to estimate parameters and *CondGauss* to estimate the conditional sampler parameters (as indeed (X_1, X_2, X_3, X_4) have a joint Gaussian distribution). Notably, the reproduced RFI values were on average higher, than reported in [27], as there is a difference between samplers. [27] used equi-correlated model-X knockoffs [9], which additionally required the swap property and thus resampled variables were more correlated with



(a) Causal graph of synthetic example.

(b) RFIs for $G = \{X_1\}$ and features X_2, X_3 and X_4 with respect to dataset sizes and n_{MC}.

Figure 4.2: Convergence of estimated RFI values to the theoretic value. Horizontal lines on (b) are analytically calculated RFIs for multivariate Gaussian distribution (via 3.6) and scatter points are MC-estimates based on estimated conditional Gaussian distribution: 20 random seeds per n_{MC}. Notably, the convergence of $\text{RFI}_2^{X_2}$ to zero is achieved even for small dataset sizes, as the respective estimated coefficient $\hat{\beta}_2$ was always close to zero.

original. This resulted in more narrow perturbations and feature importance were lower. We also discovered the convergence to the theoretic result, as we increase both the number of train/test data (n/n^*) and the number of MC-samples n_{MC}. Interestingly, in low data regimes MC-estimates were converging to the biased values, which mirrored the uncertainty of model/sampler parameters estimation. On Figure 4.2 (b) we plot the MC-estimated RFI values for $G = \{X_1\}$ and features of interest X_2, X_3, X_4. We vary the joined dataset sizes from 10^2 to 10^5, with test subset being 10%, different $n_{\text{MC}} \in \{4, 8, 16, 32, 64, 128, 256, 512\}$ with 20 random seeds for each MC size.

This experiment motivated the trade-off choice of test subset size $n^* = 1000$ for synthetic benchmarks and the choice of $n_{\text{MC}} = 50$ for all the experiments with RFI.

For all the following datasets, we study RFI, its relation to ground-truth or approximate causal graph and the different properties of conditional distributions. For Sachs-2005 we also provide the empirical results for SAGE (see Section 4.1.2).

In a nutshell, aims of the following benchmark study are:

- check goodness-of-fit of estimators in different non-Gaussian scenarios
- evaluate, how goodness-of-fit contributes to feature importance validity
- study the influence of sensitive attributes / confounders

4.1 Datasets with Known Causal Graph

By knowing the ground-truth causal generating mechanism of data in the form of **Structural causal model** (SCM), it is possible to sample arbitrary large datasets. SCM contains a series of conditional functional assignments (conditional distributions of child nodes wrt. their parents), ordered topologically according to the respective causal directed acyclic graph (DAG). In principle, it is possible to evaluate RFIs for all the possible combinations of target - a feature of interest - contextual variables, but it requires an exponential number of conditional density estimations. The same is needed for the exact evaluation of SAGE. But, it is hard to understand the validity of these values, without access to the underlying true conditional

distributions.

We propose to decrease the number of estimations by using the following concepts from feature selection theory, which have an interesting relation to causal SCMs. If we assume some regularity conditions between causal DAG and related joint distribution (Causal Markov condition and faithfulness), we can assess conditional dependence/independence statements with the help of visual criterium on DAG – d-separation. Thus, three following feature selection concepts [39]: **strong relevance**, **weak relevance** and **irrelevance** have the connection to DAG too.

Definition 4.1.1 (Strong relevance)**.** A feature X_i is strongly relevant to the target Y if

$$p(y/x_{-i}) \neq p(y/x_{-i}, x_i)$$

This means that the variable X_i carries the information about Y, that no other variable has. So the target will be always dependent on X_i, conditioning on all possible contexts.

Definition 4.1.2 (Weak relevance)**.** A feature X_i is weakly relevant to the target Y if it is not strongly relevant and

$$\exists S \subseteq -i : p(y/x_S) \neq p(y/x_S, x_i)$$

Weak relevance defines other variables in DAG, which can also carry information about the target, but not in all the contexts.

Definition 4.1.3 (Irrelevance)**.** A feature X_i is irrelevant to the target Y, if it is neither strongly relevant nor weakly and

$$\forall S \subseteq -i : p(y/x_S) = p(y/x_S, x_i)$$

meaning, that irrelevant features are always independent to the target.

These three definitions split all the features in R on mutually exclusive groups. In the context of SCMs, strong relevance coincide with the **Markov blanket** of nodes [2]:

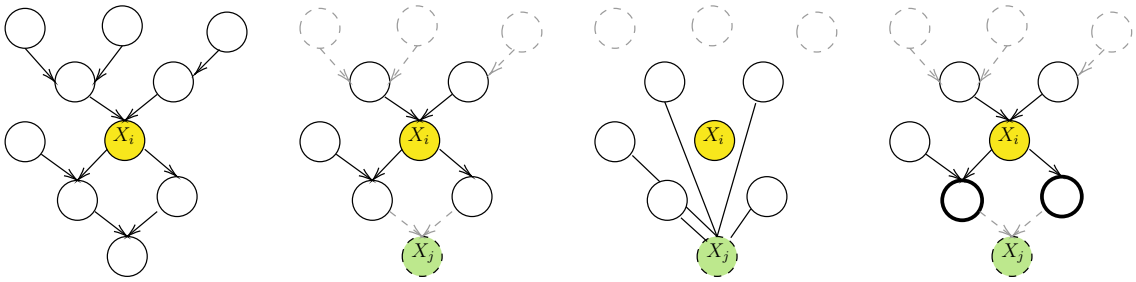
Definition 4.1.4 (Markov Blanket)**.** A Markov Blanket of node X_i is the set of parents Pa_{X_i}, children Ch_{X_i} and parents of children (spouses) $\mathrm{Pa}_{\mathrm{Ch}_{X_i}}$ in a causal DAG $\mathcal{G}$:

$$\mathrm{MB}(X_i) = \mathrm{Pa}_{X_i} \cup \mathrm{Ch}_{X_i} \cup \mathrm{Pa}_{\mathrm{Ch}_{X_i}}$$

Property 4.1.1 (Total conditioning)**.** For the faithfull causal graph $\mathcal{G}$ with set of nodes, indexed with R:

$$\forall i, j \in R : \left(X_i \in \mathrm{MB}(X_j) \Longleftrightarrow p(x_j/x_{R\setminus\{i,j\}}) \neq p(x_j/x_{R\setminus\{i,j\}}, x_i) \right) \tag{4.1}$$

This property bounds the strongly relevant features with the Markov blanket of a target. In the same way, we can say that **non-Markov blanket** (all the nodes of graph outside the Markov blanket) are the set of weakly and irrelevant features. This property motivates to study only specific feature importances:



(a) Initial causal DAG $\mathcal{G}$ and target variable X_i (yellow).

(b) Weakly relevant feature of interest X_j (green) and $\mathrm{MB}(X_i)$ - not hatched nodes.

(c) Fitting conditional distribution $X_j/\mathrm{MB}(X_i)$.

(d) Checking, if $\mathrm{MB}(X_j) \subseteq \mathrm{MB}(X_i)$, $\mathrm{MB}(X_j)$ marked with bold – needed for analytic GoF metrics.

Figure 4.3: Simplified scheme of evaluation steps while calculating $\mathrm{RFI}_j^{\mathrm{MB}(X_i)}$.

1. RFIs, conditioned on strongly relevant features, should be close to 0 (for the perfect predictor and conditional sampler). We mark the first group of RFIs as **MB-RFIs** ($\mathrm{RFI}_j^{\mathrm{MB}(X_i)}$).

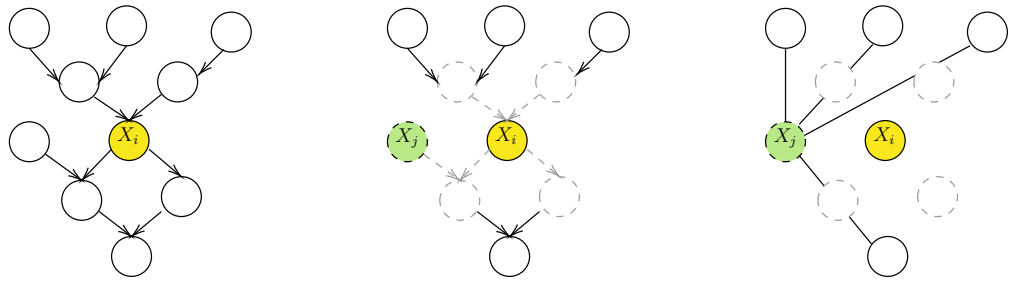
2. RFIs, conditioned on the set of weakly and irrelevant features, should be non zero and depend on the level of noise of structural assignments. Also, they should be on average higher, than MB-RFIs. We mark this group of importances with **non-MB-RFIs** ($\mathrm{RFI}_j^{\text{non-MB}(X_i)}$).

3. SAGE values should be zero for irrelevant features and non-zero for strongly relevant (for correct predictor and correct sampler).

4. SAGE values of weakly relevant features can range from zero to values of strongly relevant features, as Shapely values are only approximations of true collective contributions.

Full RFI evaluation protocol for datasets with known causal DAGs is listed in the Appendix A.4. In a nutshell, we iteratively select a target variable from graph nodes, fit three regression models (Linear Regression, Random Forest [8] and Light Gradient Boosting Machine regression model [25] (LightGBMRegressor)) and evaluate test risks (mean squared error (MSE) and mean absolute error (MAE)). Afterwards, we fit conditional samplers with the conditioning sets equal to $\mathrm{MB}(X_i)$ in the first setting and to non-$\mathrm{MB}(X_i)$ – in second. We report conditional goodness-of-fit metrics: NLL for all the experiments and advanced metrics (3.18a, 3.18b, 3.18c) for special cases of synthetic benchmarks (see Section 4.1.1). Then we evaluate $\mathrm{RFI}_j^{\mathrm{MB}(X_i)}$ and $\mathrm{RFI}_j^{\text{non-MB}(X_i)}$ for all the training variables. Schematically, this evaluation can be seen on Figures 4.3 and 4.4. For the design of SAGE evaluation, we refer to Section 4.1.4.

We found the benchmark for causal structure learning fits the needs of RFI and SAGE evaluation. Specifically, we use some synthetic, semi-synthetic and real datasets with continuous features from [28]. Not many benchmarks provide continuous structural causal models (e.g. a popular bayesian network structure learning library has only discrete SEMs [47]).



(a) Initial causal DAG $\mathcal{G}$ and target variable X_i (yellow).

(b) Strongly relevant feature of interest X_j (green) and non-MB(X_i) - not hatched nodes.

(c) Fitting conditional distribution X_j/non-MB(X_i).

Figure 4.4: Simplified scheme of evaluation steps while calculating $\text{RFI}_j^{\text{non-MB}(X_i)}$.

4.1.1 Structural Causal Models Datasets

We use four synthetic SCMs, proposed by [28]. First, we sample a random DAG according to a Erdős–Rényi model $\mathcal{G} \sim \mathcal{G}(p, m)$: the resulting random DAG will have exactly p nodes and at maximum m edges. Then we define following functional assignments:

1. *LinearGaussianNoise.* SCM, which uses a linear combination of parent nodes and additive Gaussian noise: $X_j/\text{Pa}_{X_j} \sim w_j^T \text{Pa}_{X_j} + 0.2N(0, \sigma_j^2)$, where linear combination weights and standard deviations are sampled uniformly $\sigma_j^2 \sim U[1, 2]$, $w_{ij} \sim U[0, 1]$. As all the variables from $\mathcal{G}$ form a joint Gaussian distribution with parameters $N(0, W^{-1} \operatorname{diag}(0.04 \cdot \sigma^2) W^{-1T})$, we can infer all the conditional distributions (see *CondGauss* 4). Here W is composed with the adjacency matrix of $\mathcal{G}$ multiplied on respective weights w_j.

2. *RandomGPGaussianNoise.* SCM, which samples a non-linear function from a Gaussian process with unit-bandwidth RBF-kernel $k(X, X')$ and adds the Gaussian noise: $X_j/\text{Pa}_{X_j} \sim f_j(\text{Pa}_{X_j}) + 0.2N(0, \sigma_j^2)$; $f_j \sim \mathcal{GP}(0, k(X, X'))$. Standard deviations are sampled uniformly $\sigma_j^2 \sim U[1, 2]$.

3. *PostNonLinearLaplace*[1]. SCM with a non-linear function from a Gaussian process with unit-bandwidth RBF-kernel $k(X, X')$, additive Laplace noise and sigmoid as post non-linearity: $X_j/\text{Pa}_{X_j} \sim \sigma(f_j(\text{Pa}_{X_j}) + \text{Laplace}(0, l_j))$; $f_j \sim \mathcal{GP}(0, k(X, X'))$. Scales of Laplace noises are uniform: $l_j \sim U[0, 1]$. Thus, the support of all marginal and conditional distributions is limited to (0, 1) interval.

4. *PostNonLinearMultiplicativeHalfNormal.* Multiplicative SCM with a Half-Normal noise defined as: $X_j/\text{Pa}_{X_j} \sim \exp\left(\log\left(\sum \text{Pa}_{X_j}\right) + |N(0, \sigma_j^2)|\right)$; $\sigma_j^2 \sim U[0, 1]$. The distribution has heavy-tails and positive support.

[1]We encountered several computational issues while sampling from random Gaussian processes, in cases when data size was larger than 50K or when we numerically calculated integrals for GoF metrics. This is due to the fact, that any new function evaluation takes longer time than previous for GPs. Thus, we used a linear interpolation and zero extrapolation, after we sampled a train subset, which substantially decreased calculation time.

The example of the random synthetic sample from all four SCMs with the fixed random DAG can be found on Figure 6.1 in Appendix A.6, where one can observe different distributional properties: non-linear conditional mean, long tails or bounded support.

Unfortunately, in general, we can infer ground-truth conditional distributions in some limited cases, e.g. when the conditioning set contains only parents – so simply using structural assignments formulas with the change of variables theorem 3.21 (both sigmoid in *PostNonLinearLaplace* and exponentiation in *PostNonLinearMultiplicativeHalfNormal* are invertible transformations) and additivity of parameters of noise distributions (Normal, Half-Normal or Laplace).

The problem of probabilistic inference in Bayesian Networks for the arbitrary conditional query was overviewed in [46]. Authors pinpoint, that exact inference is possible only for specific types of SCMs, like *LinearGaussianNoise* or models with conjugate noises (e.g. from exponential family and assignments of a special form). Even an approximate inference is a huge area of research. This was recently named as **expressiveness versus tractability** issue [58]. Authors proposed an alternative to SCMs – so-called probabilistic circuits, a unified framework for tractable probabilistic modelling.

As we saw with the total conditioning property 4.1.1, conditioning on all the variables in DAG is equivalent to conditioning on Markov Blanket. Luckily, a true MB-conditional distribution for every node, which is used for evaluation of 3.18a, 3.18b, 3.18c, can be exactly inferred or approximated via parents-conditional distributions with the Bayes rule (see Appendix A.5 for a derivation):

$$p(x_i/x_{-i}) = p(x_i/\mathrm{MB}(x_i)) = \frac{p(x_i/\mathrm{Pa}_{x_i}) \prod_{x_j \in \mathrm{Ch}_{x_i}} p(x_j/\mathrm{Pa}_{x_j}, x_i)}{\int_{\mathcal{X}_i} p(x_i/\mathrm{Pa}_{x_i}) \prod_{x_j \in \mathrm{Ch}_{x_i}} p(x_j/\mathrm{Pa}_{x_j}, x_i)\, dx_i} \tag{4.2}$$

On practice, a normalisation constant in the denominator of 4.2 is estimated with the Monte-Carlo integration by sampling $\tilde{X}_i^{(k)} \sim p(x_i/\mathrm{Pa}_{x_i})$:

$$\frac{1}{K} \sum_k \prod_{x_j \in \mathrm{Ch}_{x_i}} p(x_j/\mathrm{Pa}_{x_j}, x_i = \tilde{x_i}^{(k)}) \tag{4.3}$$

where K is the sample size (for all the synthetic experiments we set $K = 5000$). Furthermore, to calculate GoF metrics 3.18a, 3.18b, 3.18c, we employed the numerical integration with Scipy quad method. We calculated integrals for all the contexts in test subset and then averaged values, which didn't surpass bounds for GoF metrics 3.1 (it could happen due to bad quadrature calculation).

As we found no existing open-source tool for continuous non-linear / non-Gaussian SCMs, we made our own code, built over PyTorch distributions tools. In the experiments, we vary train subset size $n \in \{1000, 5000, 10000\}$, number of nodes $p \in \{5, 10, 20\}$ and number of edges $m \in \{p, 2p\}$. We set test subset size to $n^* = 1000$.

4.1.2 Semi-synthetic and Real Datasets

SynTReN generator is a semi-synthetic dataset generator [57], with a known DAG, but black-box assignments. It creates synthetic transcriptional regulatory

networks and produces simulated gene expression data, that approximates experimental data. We use 10 different subgraphs with 20 variables in each[2]. Number of edges ranges from 19 to 34 and each dataset is split into 400 train and 100 test datapoints with 3 different randomly shuffled splits. After examining pairwise scatter matrix of data (see Figure 6.2 (b)), we noticed a huge number of multi-modal (sometimes nearly uniform) and asymmetrical marginals. Similarly, we applied the standard mean-std normalisation.

Sachs-2005 dataset is a real sample from a known causal graph, provided by [45]. It measures the expression level of different proteins and phospholipids in human cells. DAG (see Figure 6.3 (a)) consists of 11 variables and 17 edges and we made 5 random splits with 682 train and 171 test datapoints. Notably, DAG has in fact two disjoint subgraphs. The marginal distributions tend to have heavy tails (see Figure 6.3 (b)). Among other preprocessing steps, we normalised features with a standard mean-std normalisation. The same random splits were used for enhancements study and SAGE experiments.

4.1.3 RFI Results

After performing the experiments, described in the previous section (according to the protocol A.4), we want to check the following intuitive properties of samplers and RFI values:

- Goodness-of-fit and its relation to conditioning size, train size and possible sampler enhancements (for Sachs-2005).
- MB-RFIs should be close to zero.
- Assessing the empirical correlation between MB-RFIs and test risk, so that lower generalisation risk will mean lower importance of weakly relevant features.
- Non-MB RFIs should be larger than MB-RFIs.
- Influence of samplers' enhancements on RFI values (for Sachs-2005).

Goodness-of-fit and conditioning size. On the Figure 4.5 we report conditional goodness-of-fit metrics with respect to increasing size of Markov blanket – from 1 to 8, in the setting of MB-RFIs. Expectedly, *CondGauss* outperforms by a margin on *LinearGaussianNoise* benchmark for all the MB sizes and all the metrics. On *RandomGPGaussianNoise*, one sees no clear preference for any specific sampler (*CondGauss* also performs well in this setting, as the amount of present noise can "mute" random non-linearity and conditional distribution will be similar to Gaussian, e.g. see pairplot 6.1 (b)). On *PostNonLinearLaplace*, both deep estimators on avarage outperform *CondGauss* (but it can be hard to see with the negative log-likelihood). Regarding the last heavy-tailed benchmark *PostNonLinearMultiplicativeHalfNormal*, *VanillaMDN* and *VanillaCNF* are better for all the sizes – it could even be noticed with the NLL.

For the second non-MB-conditional setting, we can only evaluate the analytic GoF metrics for *LinearGaussianNoise* (see Figure 4.6, top-row), as for the other datasets Markov Blanket for features of interest contains the target variable, which

[2] https://github.com/kurowasan/GraN-DAG/blob/master/data/syntren_p20.zip

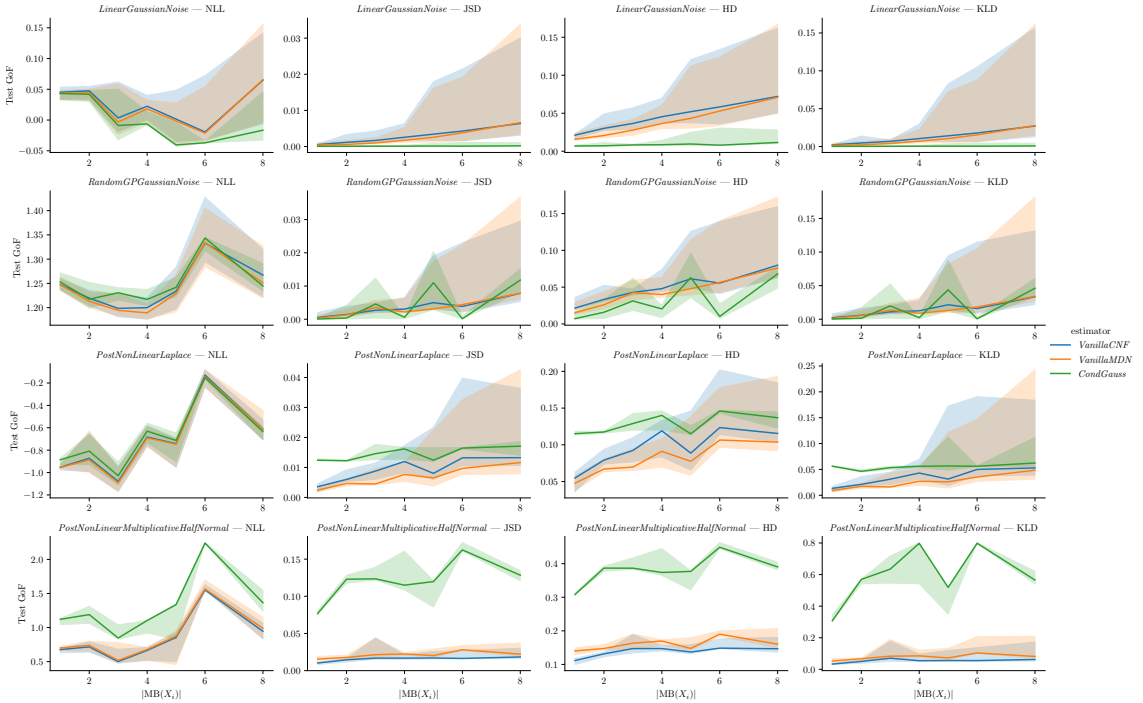


Figure 4.5: Median test goodness-of-fit values and 95% CI with respect to different sizes of conditioning MB sets for three density estimators (the lower the better). Column-wise we report NLL, JSD, HD and KLD and row-wise we differ synthetic SCM types. Note, that y-axes are not shared on the figure.

must be excluded while samplers fit. Thus, for the three remaining benchmarks, we report solely NLL (see Figure 4.6, bottom row). Now, conditioning size ranges from 1 to 18. Similarly, *CondGauss* is the best choice for *LinearGaussianNoise* SCM. For non-linear ones, we also notice that NLL is almost indistinctive for *PostNonLinearLaplace* and *RandomGPGaussianNoise* meaning that all models perform roughly the same. Only for *PostNonLinearMultiplicativeHalfNormal*[3], we see a clear preference of deep estimators.

The general trend in both MB- and non-MB-conditional settings is that the variance of deep density estimators increases with the conditioning set size, for all the SCMs.

We observe an average superiority of deep density estimators on SynTReN and Sachs-2005, for all the conditioning sizes and both settings (see Figure 4.7). This is noticeable on the second dataset, which has long-tailed marginal distributions. The larger variance of SynTReN originates due to differences in causal DAGs, whereas Sachs-2005 results are based on a single graph.

Goodness-of-fit and train size. After inspecting the Figure 4.8, we observe that goodness-of-fit is getting better for deep density estimators and stays relatively the same for *CondGauss*. The uncertainty of estimation decreases, too, which is seen from shrinking confidence intervals. *CondGaussian* can outperform deep estimators on *RandomGPGaussianNoise* or *PostNonLinearLaplace* SCMs, but only in low

[3]Some experiments for *VanillaMDNs* on *PostNonLinearMultiplicativeHalfNormal* failed due to the to high values for some mixture components, so we refitted them with the weight decay $= 10^{-3}$.

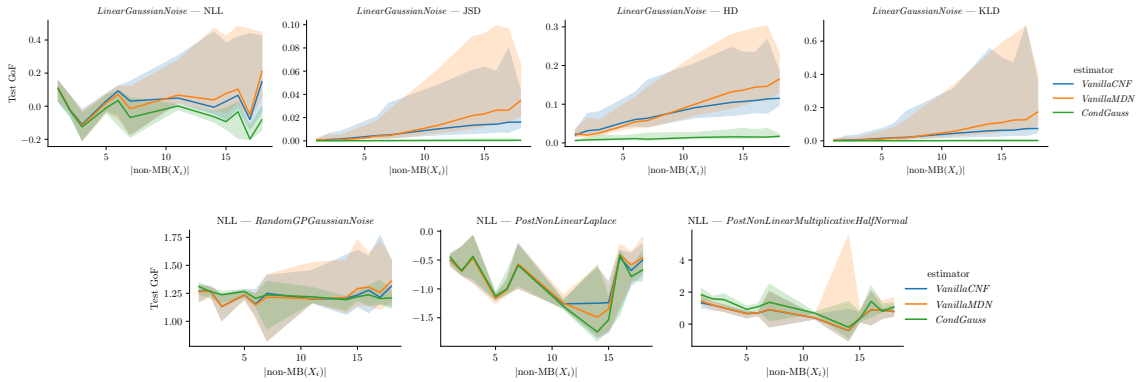


Figure 4.6: Median test goodness-of-fit values and 95% CI with respect to different sizes of conditioning non-MB sets for three density estimators (the lower the better). Top-row: column-wise we report NLL, JSD, HD and KLD for *LinearGaussianNoise*. Bottom-row: column-wise we differ three remaining non-linear SCMs and report NLL.

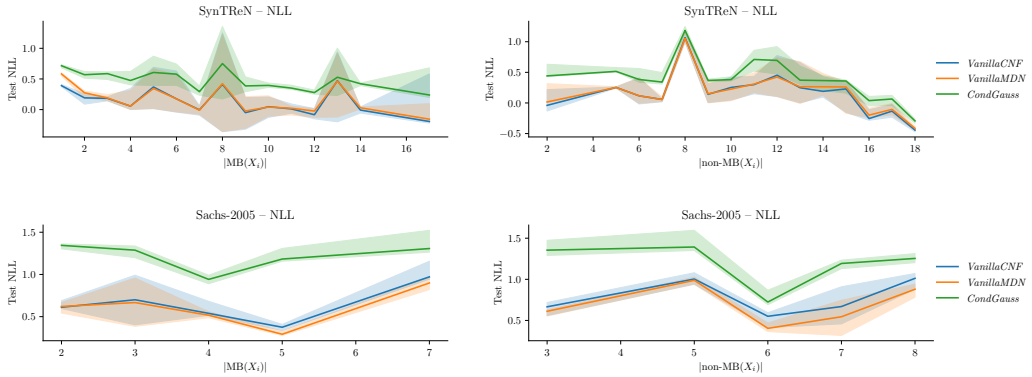


Figure 4.7: Median test negative log-likelihood and 95% CI with respect to different sizes of conditioning MB sets for three density estimators (the lower the better). First and second rows show results on SynTReN and Sachs-2005 respectivelly, and columns stand for MB / non-MB conditional settings.

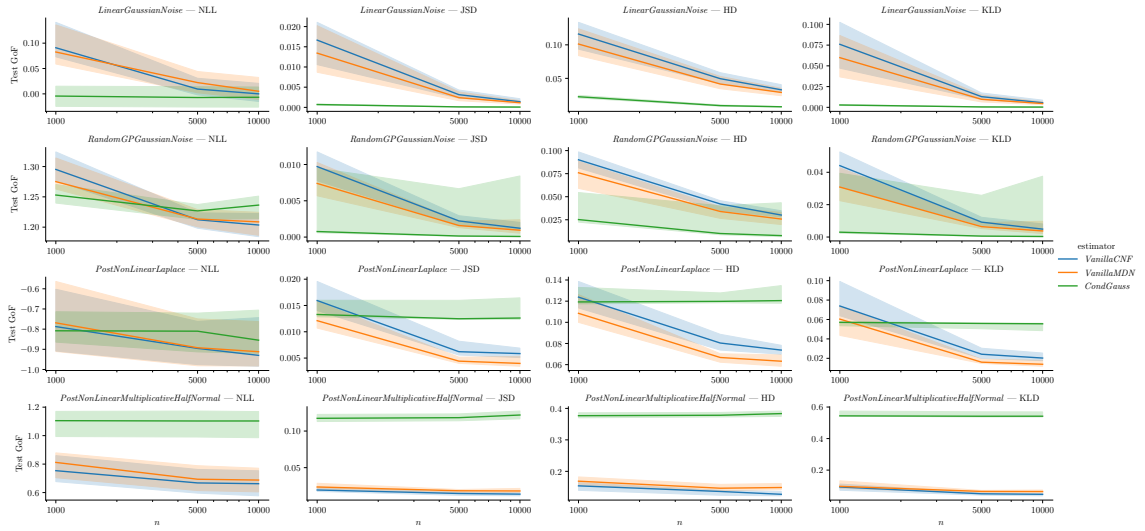


Figure 4.8: Median test goodness-of-fit values and 95% CI with respect to different train sizes $n \in \{1000, 5000, 10000\}$ for three density estimators, averaged over MB and non-MB conditional settings (the lower the better). Column-wise we report NLL, JSD, HD and KLD and row-wise we differ synthetic SCM types. Note, that y-axes are not shared on the figure.

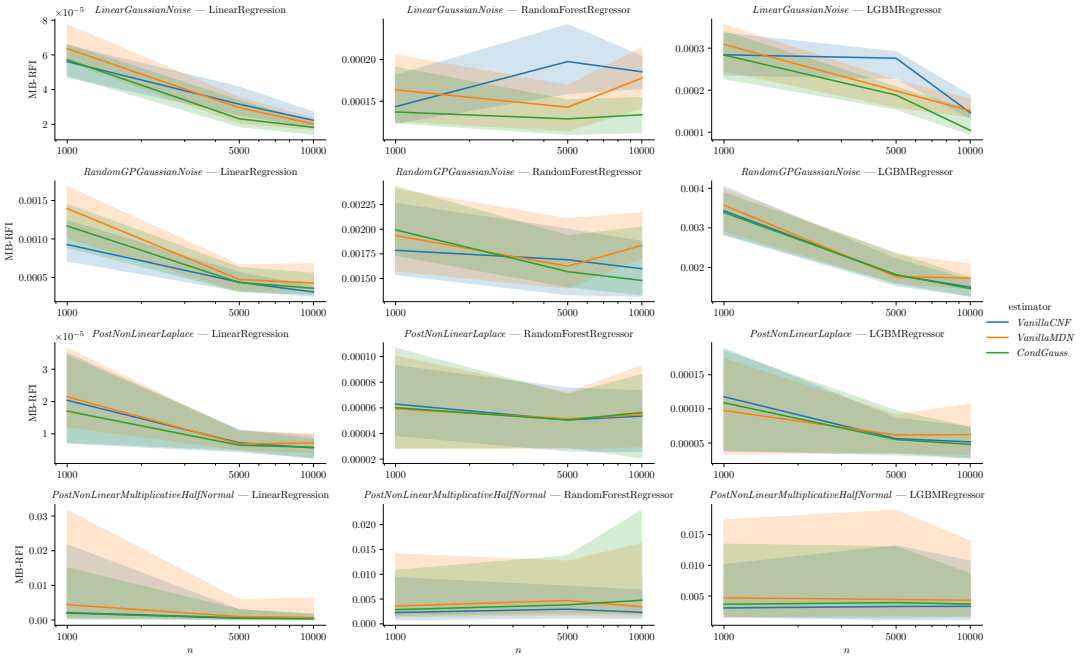


Figure 4.9: Median MB-RFIs for MSE risk and 95% CI with respect to different train sizes $n \in \{1000, 5000, 10000\}$ for three predictive models (the lower the better). Columns list different predictive models and rows – different SCMs. Note, that y-axes are not shared across the figure.

data-regimes. On contrary, for heavy-tailed *PostNonLinearMultiplicativeHalfNormal* deep estimation is always preferred. In general, no single universally best deep estimator can be chosen across SCMs (no free lunch).

RFI values. MB-RFI values mirror the collective performance of the predictive model and estimated sampler. What will happen with RFI-values if we increase train data size? For the correct predictive model class selection, MB-conditional RFIs should tend to zero. Otherwise – under wrong or biased predictive model selection, RFIs estimate some possibly non-zero importance, present in the weakly relevant or irrelevant features. To illustrate this with synthetic benchmark datasets, we plot the averaged MB-RFI values for MSE risk with respect to different train data sizes n, predictive models and data generating SCMs on Figure 4.9 (MAE risk shows the same tendencies). For example, under correctly specified model (*LinearGaussianNoise* – LinearRegression) or for flexible enough LGBMRegressor RFI values indeed decrease. On the other hand, the values for a limited model (RandomForestRegressor had a fixed maximum number of splits) stays relatively the same (see middle column).

In Figure 4.10 we could analyse non-MB conditional RFIs, by looking at the differences between non-MB-RFIs and MB-RFIs. The higher difference will mean more reliance on the strongly relevant features, than on weakly or irrelevant. We observe the differences stay almost the same for all the predictive model types and samplers (a small increase is observed on the *RandomGPGaussianNoise*).

We show the estimated MB- and non-MB RFIs for SynTReN generator on Figure 4.11. *VanillaCNF* has the lowest average MB-RFI values and *VanillaMDN* seems to overestimate them, even though they have nearly similar test NLL (see Figure 4.7). This means that NLL can be a misleading GoF, considering the low amount

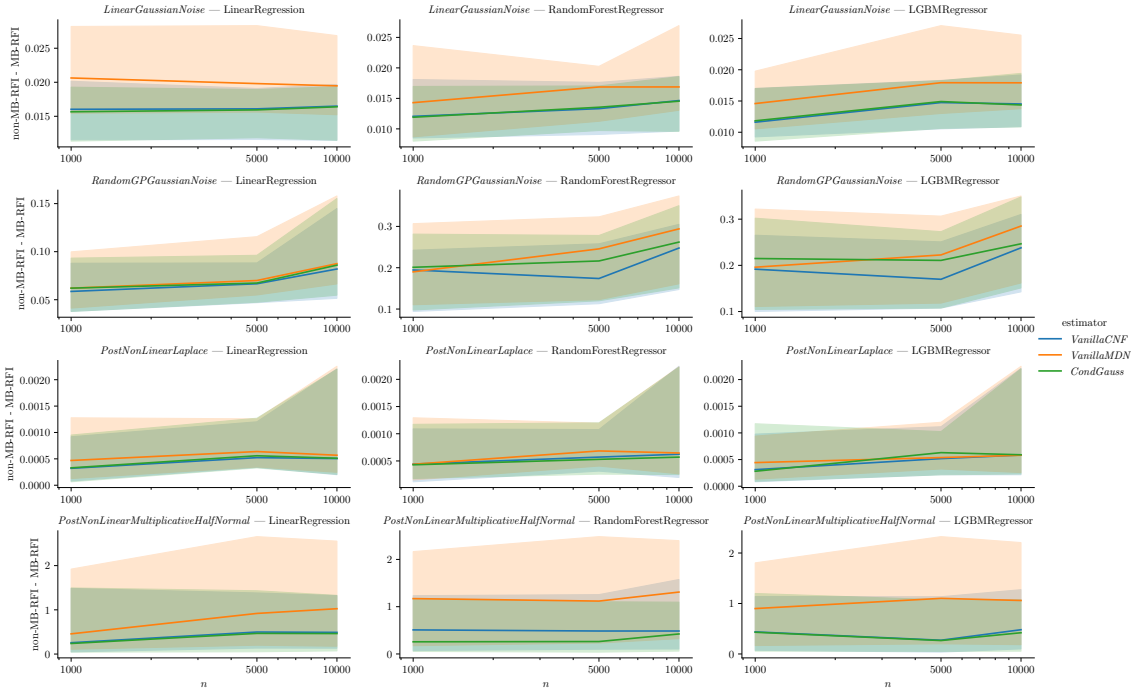


Figure 4.10: Median differences between non-MB-RFIs and MB-RFIs for MSE risk and 95% CI with respect to different train sizes $n \in \{1000, 5000, 10000\}$ for three predictive models (the higher the better). Columns list different predictive models and rows – different SCMs. Note, that y-axes are not shared across the figure.

of training data, present for SynTReN.

Now, we also want to check the empirical correlation between MB-RFIs and test risk. We also use hue to visualise different goodness-of-fit values (see Figure 4.12) and styles to mark three estimators. It easy to notice, how high is the empirical correlation between test MSE loss and MB-RFI values for the synthetic benchmark, meaning that predictive models with lower test risk, tend to have lower RFI values for weakly relevant features, conditioning on strongly relevant. Additionally, we don't see noticeable patterns with respect to the different test NNL values for estimated samplers, meaning that low NLL is not necessarily defining for RFI estimation. We also observe the same trend across other GoF metrics.

Similar pattern can be observed for scatter plot for SynTReN dataset (Figure 4.13) – high correlation for *VanillaCNF* and *CondGauss*. However, now we can notice an additional relation of test NLL and MB-RFI values, so that a good GoF coincides with lower RFIs (but not necessarily lower test risk), for example in case of combination *VanillaCNF* and LinearRegression. Notably, that low values of test loss do not necessarily correspond to low values of MB-RFIs, as predictive models can rely on weakly relevant features and still perform well, e.g. in cases of RandomForestRegressor and LGBMRegressor.

Study of enhancements. Based on Sachs-2005, we evaluated the following possible changes in the fitting process or architectures of vanilla methods:

1. We varied training mini-batch size $B = 128$ or $B = 256$. Note, that the vanilla setting used all 682 train datapoints as one batch. Reducing batch size could have a regularising effect on the optimiser.
2. Fine-tuning for both deep density estimators was performed with the help of

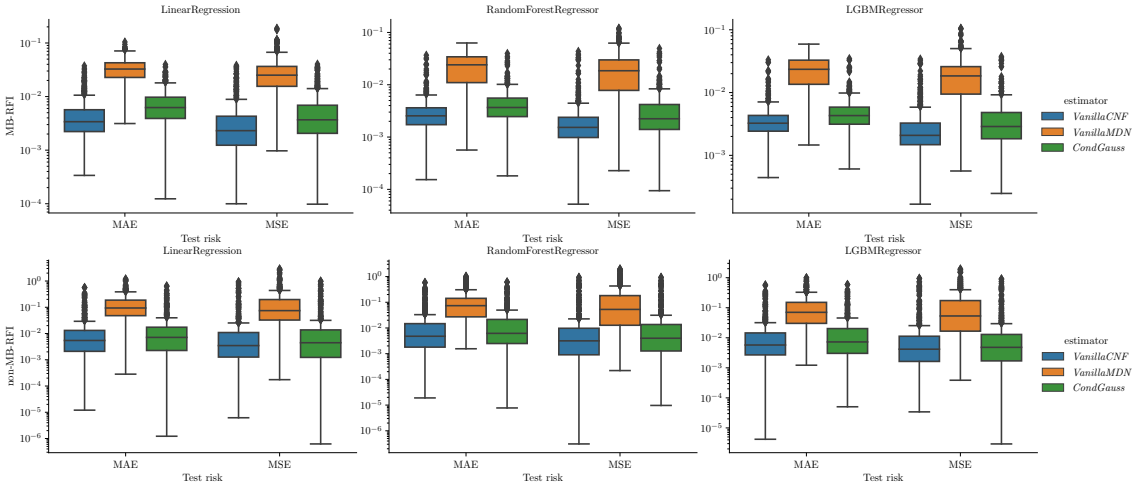


Figure 4.11: Box plots with MB-RFIs (first row, the lower the better) and non-MB-RFIs (second row, the higher the better) for MSE / MAE risks for three predictive models on SynTReN datasets. Note, that y-axes are logarithmically scaled and not shared across the figure.

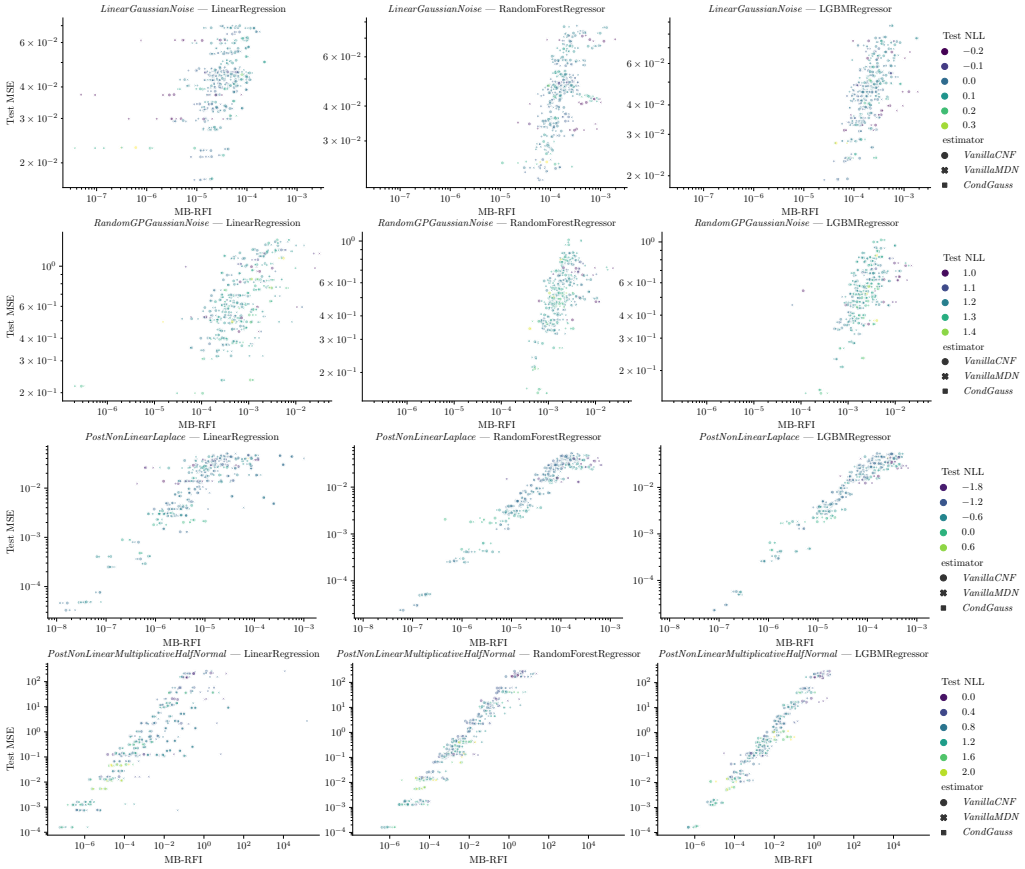


Figure 4.12: Scatter plot of MB-RFI values (averaged for each target) and test MSE risk for the synthetic benchmark. Columns list different predictive models and rows – different SCMs. We can spot a high level of correlation, between the predictive models' performances and MB-RFIs. Note, that both y-axes and x-axes are logarithmically scaled and not shared across the rows.

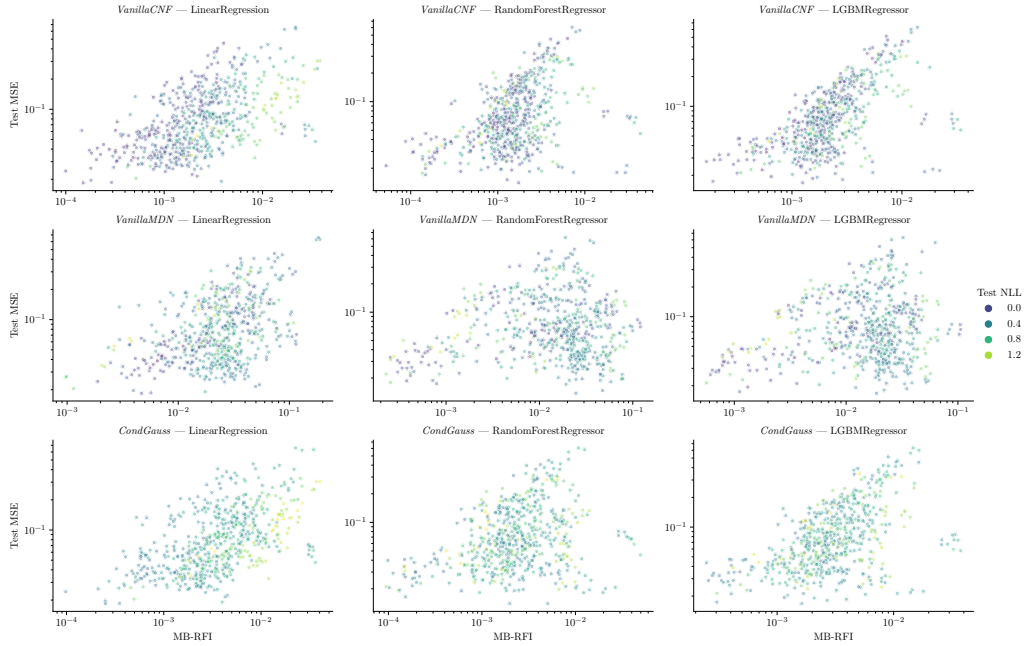


Figure 4.13: Scatter plot of MB-RFI values (averaged for each target) and test MSE risk for SynTReN dataset. Columns list different predictive models and rows – different density estimators. Note, that both y-axes and x-axes are logarithmically scaled and not shared across the rows.

5-fold cross-validation with the NLL for every fitted feature of interest and context. Hyper-parameter grid search encompassed 96 runs with varying amount of noise regularisation, weight decay, number of epochs N_e and number of layers (for CNFs) / number of components (for MDN).

3. As Sachs-2005 has only positive datapoints, we experiment with adding an extra log-transformation as the first transformation of normalising flow[4]. This aimed to restrict the support of flow transformations to only positive real values.

We present the results of possible enhancements in Table 4.1, where we average the test performances of all the samplers for both MB and non-MB conditional settings. We notice, that only fine-tuning can give a solid improvement for *VanillaCNF*, whereas batch size decrease have a steady effect on the performance of *VanillaMDN*. Generally, MDNs reached the best GoF with an average NLL equal to 0.6547.

Setting	V	V & $B = 128$	V & $B = 256$	Fine-tuning	V & Log-transform
CondGauss	1.2569 ± 0.174	–	–	–	–
VanillaCNF	0.7619 ± 0.305	0.7836 ± 0.284	0.7210 ± 0.264	**0.6824 ± 0.256**	0.7482 ± 0.269
VanillaMDN	0.6704 ± 0.251	**0.6547 ± 0.252**	0.6602 ± 0.251	0.6573 ± 0.254	–

Table 4.1: Test negative log-likelihoods of different samplers, averaged between MB and non-MB conditional settings for Sachs-2005 dataset. We report mean values and standard deviations. V stays for Vanilla, B is a train batch size. Lower is better.

To see, how the advances in goodness-of-fit contribute to the RFI values, we depict them on Box plot 4.14. No large difference was detected between different versions of deep estimators and thus no practical need to fine-tune samplers.

[4]Also, in this case, one-sided additive inputs noise was employed, by using additional ReLU transformation for Gaussian noise.

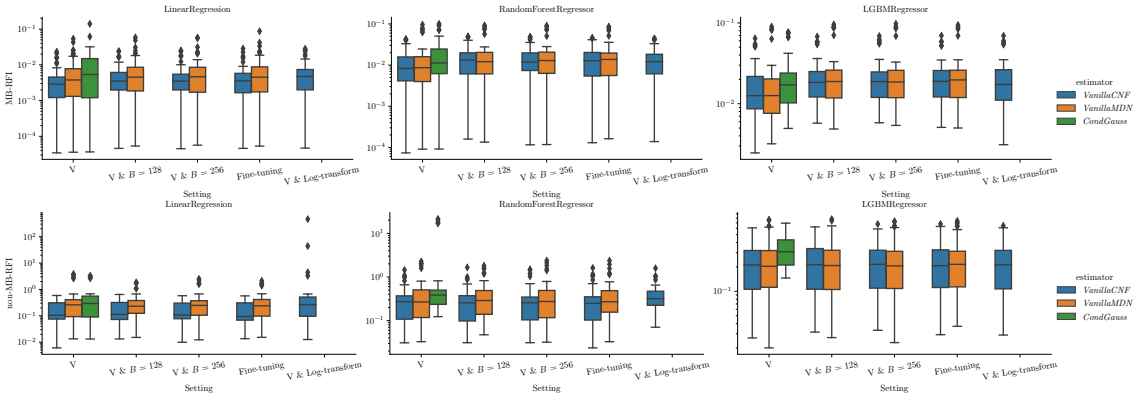


Figure 4.14: Box plots with MB-RFIs (first row, the lower the better) and non-MB-RFIs (second row, the higher the better) for MSE risk for three predictive models on Sachs-2005 dataset. X-axes vary different types of samplers enhancements. Note, that y-axes are logarithmically scaled and not shared across the figure.

CondGauss, on the other hand, seems to have a bias from other samplers (in view of much worse GoF – see Figure 4.7 and Table 4.1).

4.1.4 SAGE Results

To prove the viability of the deep density estimators for the other interpretability method – SAGE (introduced in Section 3.1.2), we utilise a permutation-sampling-based algorithm [12]. It randomly selects n_P permutations of train features and then perturbs joint sets of features n_{MC} times, conditionally on other sets of features. It sequentially adds features from dependent set to a context, in the order of a permutation. Then, the Shapely importance estimate of feature j is simply the average over all the risk differences, after this feature was moved from input to context (see original paper for details). In all the experiments, we set number of permutations $n_P = 20$ and number of MC-samples $n_{\text{MC}} = 100$.

The experiments were conducted similarly to RFI: we iterate over all the nodes of DAG, select a target, fit predictive models with all the other nodes as training features and then compute SAGE values for all the training features. The same set of predictive models (Linear Regression, Random Forest and LightGBMRegressor) and density estimators (*CondGauss*, *VanillaCNF* and *VanillaMDN*) were evaluated. We report test NLL and estimated SAGE values, averaged across 5 random seeds, which influence train/test split and selected random permutations.

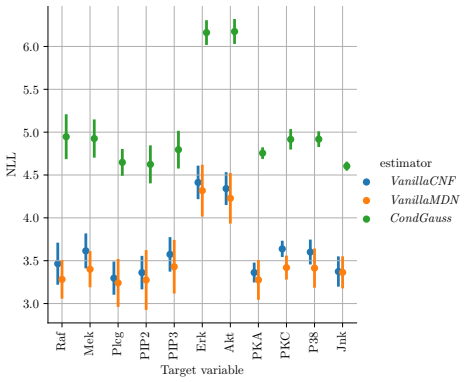


Figure 4.15: Mean and std NLL, averaged across $n_p = 20$ permutations, of density estimators, used for SAGE estimation on Sachs-2005 dataset. Lower is better.

Test NLL are provided on Figure 4.15. Clearly, both deep estimators are preferable for this dataset, as in the case of RFI estimation. We additionally saw that values of NNL decrease linearly with the decrease of the number of dependent variables d_Y (a known property of likelihood), so that averaged values are near 5.0 for *CondGauss*.

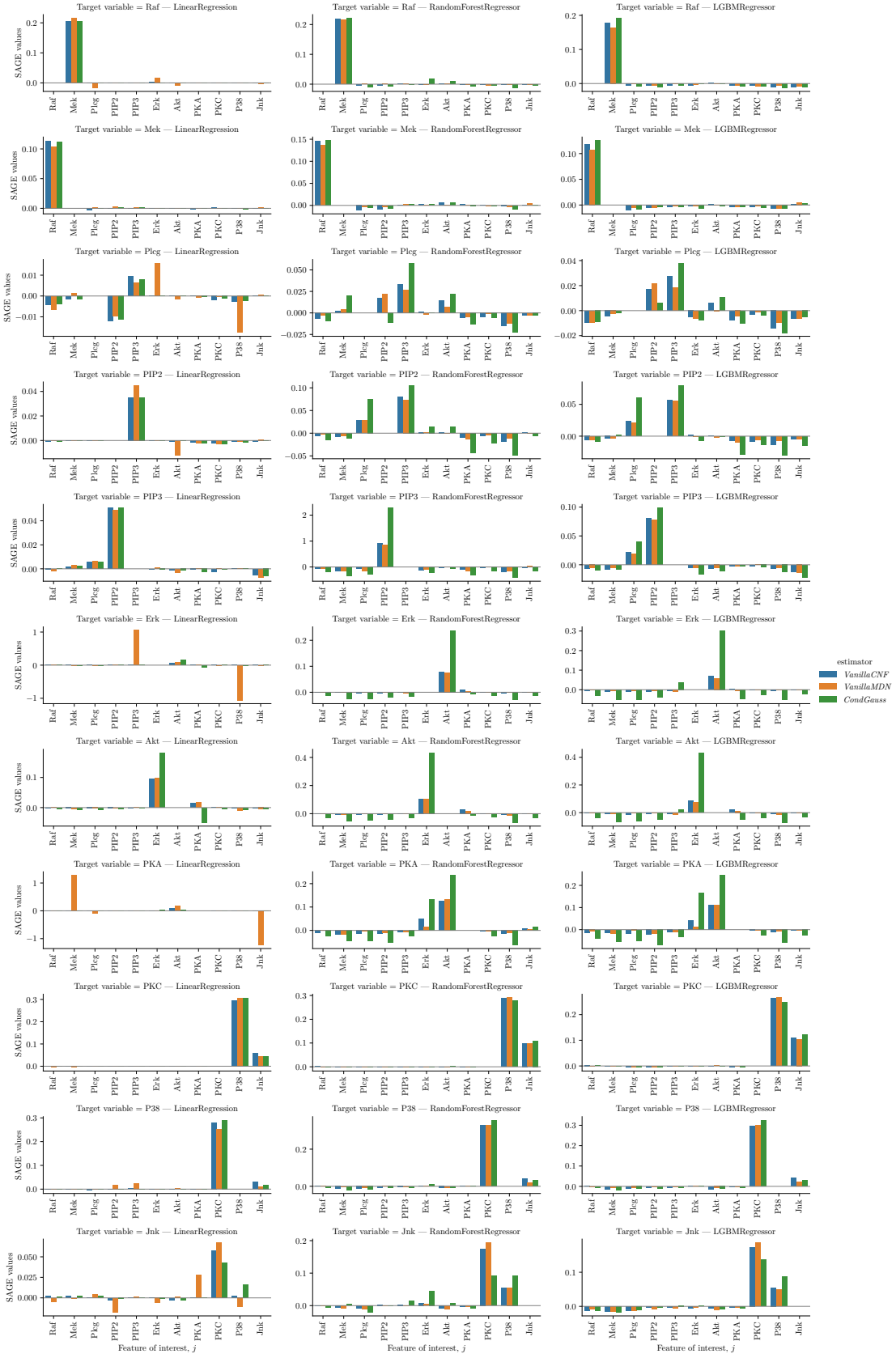


Figure 4.16: Bar plots with estimated SAGE values for MAE risk for three predictive models on Sachs-2005 dataset. Rows are different targets and columns – different predictive models. Note, that y-axes are not shared across the figure.

After estimating SAGE values for MAE risk (see Figure 4.16), we observe, how **estimated importances mirror the ground-truth connections of data-generating DAG** from Figure 6.3 (a), so that high SAGE values are assigned to strongly relevant to the target features. Also, features with the highest SAGE are the same for the majority of predictive models, while secondary features could differ, for example, see SAGE values for target '*Plcg*'. Notably, values of weakly-/irrelevant features are slightly below zero, and for *VanillaMDN* they are sometimes as far as positive values (e.g. for targets '*PKA*' or '*Erk*' in case of Linear Regression). We hypothesise, that the main reason is that sampling from hugely multivariate distribution can underestimate the real support of distribution and sampled values will lay in a smaller range, in comparison to sampling from marginalised distributions. So when one estimates the feature contribution, test risk can be even increased, when comparing original and marginalised samplers perturbations. That is why the choice of a good sampler is crucial for the correct estimation of SAGE values.

Additionally, although *VanillaMDN* had on average better GoF (see Figure 4.15), it was still capable of generating false values. For example, feature '*PIP3*' had large SAGE importance for target variable '*Erk*' and LinearRegression, even though this feature is completely irrelevant to the target. The main reason for that could be a numerical instability of MDNs when parameters of mixture tend to infinity.

4.2 Sensetive Attributes

Census Income Dataset from UCI library [13] was used in several studies on SHAP interpretability [17; 18; 59]. The task for the dataset is to predict, whether a person had an income, larger than 50K$, based on census data. It contains 12 train features: 8 categorical ('*Age*', '*Country*', '*Marital Status*', '*Occupation*', '*Race*', '*Relationship*', '*Sex*', '*Workclass*') and 4 continuous ('*Capital Gain*', '*Capital Loss*', '*Education-Num*', '*Hours per week*'). Train/test split consists of 26048 and 6513 datapoints respectively. [59] proposed a possible causal graph, where '*Age*', '*Race*', '*Sex*' and '*Country*' are root nodes, connected to target, and all the other variables are mediators between roots and target.

For Census Income dataset, we had to deal with categorical features as both contextual and input variables. To use a categorical feature as a contextual variable, we used one-hot encoding. Notably, that Gaussian noise regularisation was still applicable in practice for such a context. To model conditional categorical distributions, we utilised *CatEstimator* from Section 3.3.3. The underlying neural network is defined in the same way, as for *VanillaMDN*. We similarly used noise regularisation with $\sigma_x = 0.1$ and $N_e = 1000$, but slightly higher learning rate $\alpha = 0.005$. Among other preprocessing steps, we normalised continuous features with a standard mean-std normalisation.

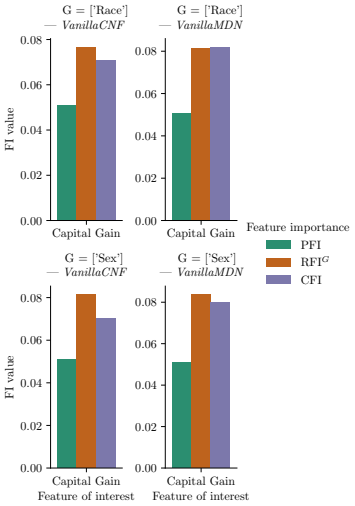


Figure 4.17: Wrong RFI values for mixed-type '*Capital Gain*' feature: RFI and CFI are higher, than PFI. Rows are different excluded sensitive features, columns – different CDE estimators for '*Capital Gain*'.

After the first feature importance evaluation, we encountered the issue with '*Capital Gain*' variable, as PFI for this feature was much lower than CFI and RFIs (see Figure 4.17). The main reason for that was '*Capital Gain*' having de-facto a mixed-type distribution with a non-zero point mass at $x = 0.0$. We also noticed unnaturally low values of test NLL while fitting conditional samplers, which supports the idea of a wrong modelling approach (already mentioned in 3.3.3). Thus, we used a 50-bins discretizer and treated this feature as categorical.

As the main classification model, we used LightGBM classifier, which achieved 87.18% test accuracy (0.923 ROC-AUC), by using all the features. After excluding some sensitive features, like '*Age*', '*Race*' or '*Sex*', the performance dropped at most to 86.38% (0.917 ROC-AUC, after exclusion of '*Age*').

4.2.1 RFI Results

We are interested in checking, whether sensitive attributes have a contribution to the classifier, if they are included or excluded as the training features. We define three sets of sensitive features: $G_1 = \{$'*Age*'$\}$, $G_1 = \{$'*Race*'$\}$ and $G_3 = \{$'*Sex*'$\}$.

Now, we aim to evaluate:

- difference between PFI and RFI^{G_i} for train features of models, which used G_i while training. Sensitive features could contribute either independently or as interactions with other features.

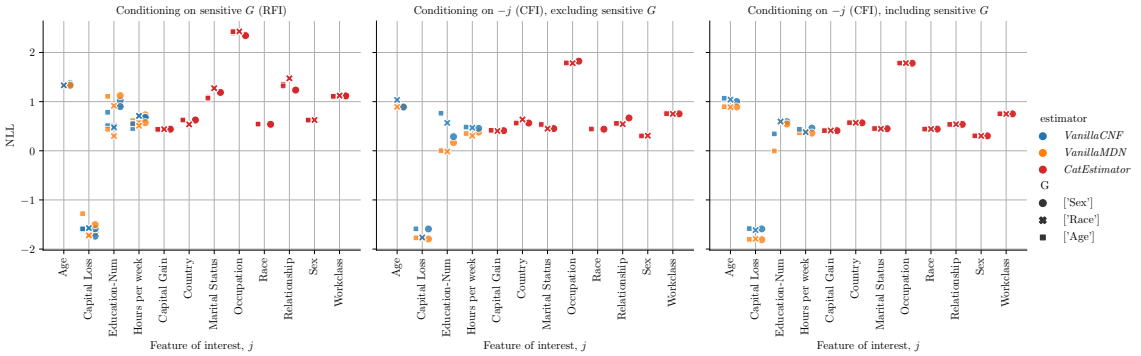


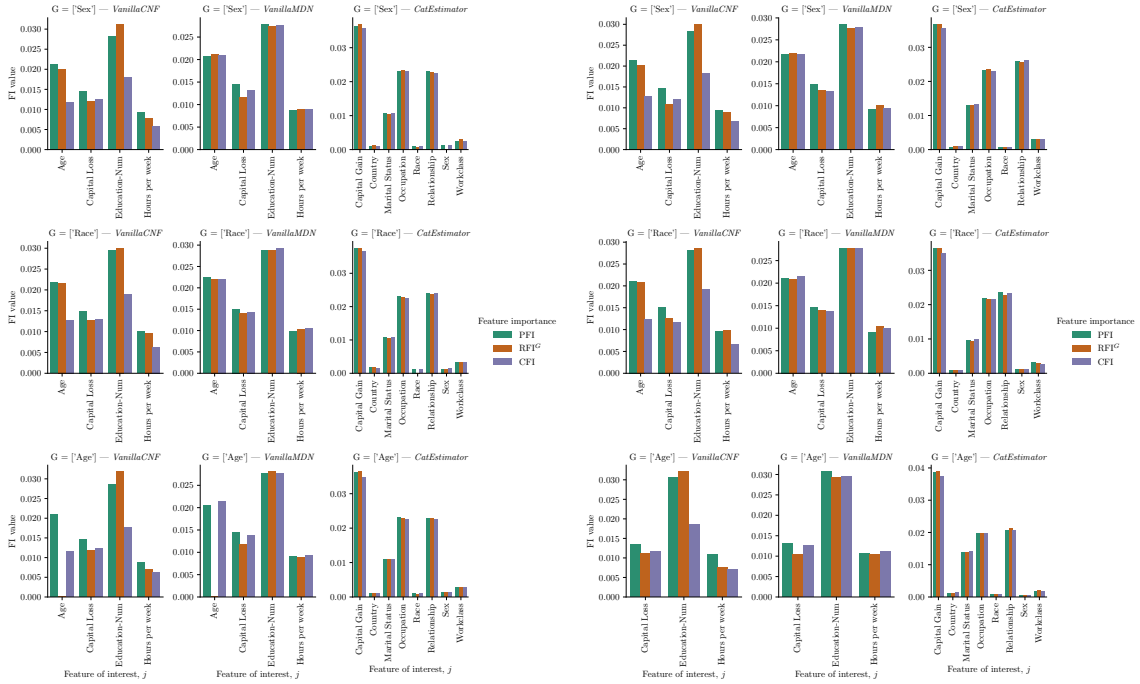
Figure 4.18: Test NLL of estimators, which are to be used for RFI (left), CFI without sensitive attributes in context (center) and CFI with sensitive attributes in context. Note, that '*Capital Gain*' feature is discretised. Lower is better.

- difference between PFI and RFI^{G_i} for train features of models, which excluded G_i. Even after features ignorance, we seek for any possible sensitive information leakage through other features.
- CFIs in both before-mentioned cases, by conditioning on all respective training features (for reference).

First, we present the goodness-of-fit results for three categories of samplers: used for RFIs, used for CFIs without sensitive features and used for CFIs with sensitive features (Figure 4.18). PFI simply shuffled test datapoints and thus no samplers were fitted. We observe, that *VanillaMDN* outperformed *VanillaCNF* in the majority of cases and thus, feature importances, estimated via *VanillaMDN* should, in general, be more trusted.

Overall, feature importances between the classifier, which used sensitive attributes, and the one, which ignored them, were relatively the same. This was mainly due to the fact, that PFIs of sensitive features '*Sex*' and '*Race*' were almost zero and PFI of '*Age*' was just 2% (see Figure 4.19(a)). After exclusion of '*Age*', test accuracy dropped on 0.8%, which makes the leakage of age information through other features very unlikely.

After examining the RFI and PFI differences for e.g. test accuracy (see Figure 4.19(a-b)), we have found a slight difference between RFI values, generated by *VanillaMDN* and *VanillaCNF*, for example for '*Hours per week*' feature of interest if $G = \{$'*Age*'$\}$. However, this discrepancy accounts only for nearly 0.5% of accuracy and could be neglected. After all, one would trust the RFI values of a sampler with a better GoF (see Figure 4.18(centre-right)), which is *VanillaMDN* in the case of '*Hours per week*'. Also, '*Capital Loss*' seems to have the difference between PFIs and RFIs for all the contexts, but as we already discovered, it could also be a mixed-type variable and should be excluded from analysis (or modelled via discretisation). Thus, **we do not observe any leakage of sensitive attributes via other features if we include or even exclude them from training**.



(a) Sensitive features are included while predictive modelling.

(b) Sensitive features are ignored while predictive modelling.

Figure 4.19: Bar plots of feature importances for Census Income dataset and LightGBM classifier with respect to the test accuracy in two settings (with and without sensitive features as training features, subfigures (a) and (b) respectively). Rows correspond to different sensitive sets ($G_1 =$ {'Age'} , $G_1 =$ {'Race'} and $G_3 =$ {'Sex'}) and columns – to different types of samplers.

Section 5

Conclusion & Discussion

Both Mixture Density Networks and Conditional Normalising Flows provide a flexible synthetic data generation tool, applicable in many real-world scenarios. Containing a few parameters, they don't suffer from overfitting (with the help of noise regularisation) and do not require too much fine-tuning. We evaluated their usage for two feature importance measures: RFI and SAGE.

We have discovered, that deep density estimators should be always preferred over Gaussian conditional distribution in cases of heavy-tailed, multimodal or heteroscedastic distributions. One viable approach to detect such properties can be a simple inspection of the pairwise scatter matrix of datapoints. The presence of heavy-tails could also lead to wrong negative SAGE estimates, as the underestimation of the real support is increasing with dimensionality increase.

Intuitively, one always aims to utilise RFI/SAGE values of the sampler with the best goodness-of-fit value. But often, the difference between feature importances of rigorously fine-tuned and vanilla models is negligible, meaning that all the samplers produce roughly the same importances. We empirically evaluated it on Sachs-2005 dataset. Consequently, in terms of resources allocation – there is no need to spend too much computational power to fit samplers and can concentrate on actual predictive modelling.

There was no prevalence of one deep estimator over the other, we can not define one universally best sampler. Speaking about numerical stability although, CNFs were performing better, as MDNs required some tuning of weight decay, otherwise the optimisation led to the infinitely large mixture parameters – for example for datasets with the heavy tails.

We provided a use case of both deep density estimators for detecting the influence of sensitive attributes on UCI Census Income Dataset. In the case of discrete dependent or contextual variables, conditional Gaussian distribution can not be used. Also, we found, how modelling mixed-type distribution with a continuous estimator could lead to wrong feature importances. Mixed-variable density estimation is still an open research question and could be studied under future work.

References

[1] Aas, K., Jullum, M., & Løland, A. (2019). Explaining individual predictions when features are dependent: More accurate approximations to shapley values. *arXiv preprint arXiv:1903.10464*.

[2] Aliferis, C. F., Tsamardinos, I., & Statnikov, A. (2003). Hiton: a novel markov blanket algorithm for optimal variable selection. In *Amia annual symposium proceedings* (Vol. 2003, p. 21).

[3] Andrews, D. W. (1997). A conditional kolmogorov test. *Econometrica: Journal of the Econometric Society*, 1097–1128.

[4] Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein generative adversarial networks. In *International conference on machine learning* (pp. 214–223).

[5] Baird, L., Smalenberger, D., & Ingkiriwang, S. (2005). One-step neural network inversion with pdf learning and emulation. In *Proceedings. 2005 ieee international joint conference on neural networks, 2005.* (Vol. 2, pp. 966–971).

[6] Bishop, C. M. (1994). Mixture density networks.

[7] Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). Weight uncertainty in neural network. In *International conference on machine learning* (pp. 1613–1622).

[8] Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5–32.

[9] Candes, E., Fan, Y., Janson, L., & Lv, J. (2016). Panning for gold: Model-x knockoffs for high-dimensional controlled variable selection. *arXiv preprint arXiv:1610.02351*.

[10] Chen, S. S., & Gopinath, R. A. (2000). Gaussianization. In *Proceedings of the 13th international conference on neural information processing systems* (pp. 402–408).

[11] Child, R., Gray, S., Radford, A., & Sutskever, I. (2019). Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

[12] Covert, I., Lundberg, S., & Lee, S.-I. (2020). *Understanding global feature contributions with additive importance measures.*

[13] Dua, D., & Graff, C. (2017). *UCI machine learning repository.* Retrieved from http://archive.ics.uci.edu/ml

[14] Durkan, C., Bekasov, A., Murray, I., & Papamakarios, G. (2019). Cubic-spline flows. *arXiv preprint arXiv:1906.02145*.

[15] Durkan, C., Bekasov, A., Murray, I., & Papamakarios, G. (2020, November). *nflows: normalizing flows in PyTorch.* Zenodo. Retrieved from https://doi.org/10.5281/zenodo.4296287 doi: 10.5281/zenodo.4296287

[16] Fenn, A. (2018). *Lmu-master-thesis-latex-template.* https://github.com/amitfenn/LMU-Master-Thesis-LaTeX-Template. GitHub.

[17] Frye, C., de Mijolla, D., Begley, T., Cowton, L., Stanley, M., & Feige, I. (2020). Shapley explainability on the data manifold. *arXiv preprint arXiv:2006.01272*.

[18] Frye, C., Feige, I., & Rowat, C. (2019). Asymmetric shapley values: incorporating causal knowledge into model-agnostic explainability. *arXiv preprint arXiv:1910.06358*.

[19] Gretton, A., Borgwardt, K., Rasch, M. J., Scholkopf, B., & Smola, A. J. (2008). A kernel method for the two-sample problem. *arXiv preprint arXiv:0805.2368*.

[20] Hellinger, E. (1909). Neue begründung der theorie quadratischer formen von unendlichvielen veränderlichen. *Journal für die reine und angewandte Mathematik*, *1909*(136), 210–271.

[21] Ho, J., Chen, X., Srinivas, A., Duan, Y., & Abbeel, P. (2019). Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International conference on machine learning* (pp. 2722–2730).

[22] Ishwaran, H., Kogalur, U. B., Gorodeski, E. Z., Minn, A. J., & Lauer, M. S. (2010). High-dimensional variable selection for survival data. *Journal of the American Statistical Association*, *105*(489), 205-217. Retrieved from https://doi.org/10.1198/jasa.2009.tm08622 doi: 10.1198/jasa.2009.tm08622

[23] Jitkrittum, W., Kanagawa, H., & Schölkopf, B. (2020). Testing goodness of fit of conditional density models with kernels. In *Conference on uncertainty in artificial intelligence* (pp. 221–230).

[24] Kamthe, S., Assefa, S., & Deisenroth, M. (2021). Copula flows for synthetic data generation. *arXiv preprint arXiv:2101.00598*.

[25] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., ... Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, *30*, 3146–3154.

[26] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[27] König, G., Molnar, C., Bischl, B., & Grosse-Wentrup, M. (2020). Relative feature importance. *arXiv preprint arXiv:2007.08283*.

[28] Lachapelle, S., Brouillard, P., Deleu, T., & Lacoste-Julien, S. (2019). Gradient-based neural dag learning. *arXiv preprint arXiv:1906.02226*.

[29] Lu, S., Zhu, Y., Zhang, W., Wang, J., & Yu, Y. (2018). Neural text generation: Past, present and beyond. *arXiv preprint arXiv:1803.07133*.

[30] Lundberg, S., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*.

[31] Maaten, L., Chen, M., Tyree, S., & Weinberger, K. (2013). Learning with marginalized corrupted features. In *International conference on machine learning* (pp. 410–418).

[32] Mase, M., Owen, A. B., & Seiler, B. (2019). Explaining black box decisions by shapley cohort refinement. *arXiv preprint arXiv:1911.00467*.

[33] Mirza, M., & Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

[34] Morris, T. P., White, I. R., & Crowther, M. J. (2019). Using simulation studies to evaluate statistical methods. *Statistics in medicine*, *38*(11), 2074–2102.

[35] Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

[36] Paluszynska, A., Biecek, P., & Jiang, Y. (2020). randomforestexplainer: Explaining and visualizing random forests in terms of variable importance

[Computer software manual]. Retrieved from https://CRAN.R-project.org/package=randomForestExplainer (R_j package version 0.10.1)

[37] Parr, W. C., & Schucany, W. R. (1980). Minimum distance and robust estimation. *Journal of the American Statistical Association*, *75*(371), 616–624.

[38] Pearson, K. (1900). X. on the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, *50*(302), 157–175.

[39] Pellet, J.-P., & Elisseeff, A. (2008). Using markov blankets for causal structure learning. *Journal of Machine Learning Research*, *9*(7).

[40] Rezende, D., & Mohamed, S. (2015). Variational inference with normalizing flows. In *International conference on machine learning* (pp. 1530–1538).

[41] Rizzo, M. L., & Székely, G. J. (2016). Energy distance. *wiley interdisciplinary reviews: Computational statistics*, *8*(1), 27–38.

[42] Rothfuss, J., Ferreira, F., Boehm, S., Walther, S., Ulrich, M., Asfour, T., & Krause, A. (2019). Noise regularization for conditional density estimation. *arXiv preprint arXiv:1907.08982*.

[43] Rothfuss, J., Ferreira, F., Walther, S., & Ulrich, M. (2019). Conditional density estimation with neural networks: Best practices and benchmarks. *arXiv preprint arXiv:1903.00954*.

[44] Rüschendorf, L. (2009). On the distributional transform, sklar's theorem, and the empirical copula process. *Journal of statistical planning and inference*, *139*(11), 3921–3927.

[45] Sachs, K., Perez, O., Pe'er, D., Lauffenburger, D. A., & Nolan, G. P. (2005). Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, *308*(5721), 523–529.

[46] Salmerón, A., Rumí, R., Langseth, H., Nielsen, T. D., & Madsen, A. L. (2018). A review of inference algorithms for hybrid bayesian networks. *Journal of Artificial Intelligence Research*, *62*, 799–828.

[47] Scutari, M. (2010). Learning bayesian networks with the bnlearn R package. *Journal of Statistical Software*, *35*(3), 1–22. Retrieved from http://www.jstatsoft.org/v35/i03/

[48] Sohn, K., Lee, H., & Yan, X. (2015). Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, *28*, 3483–3491.

[49] Strittmatter, W. (1989). Measures of dependence for processes in metric spaces. *Stochastics: An International Journal of Probability and Stochastic Processes*, *27*(1), 33–50.

[50] Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T., & Zeileis, A. (2008). Conditional variable importance for random forests. *BMC bioinformatics*, *9*(1), 1–11.

[51] Tabak, E. G., Vanden-Eijnden, E., et al. (2010). Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, *8*(1), 217–233.

[52] Theis, L., Oord, A. v. d., & Bethge, M. (2015). A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*.

[53] Tran, D., Vafa, K., Agrawal, K. K., Dinh, L., & Poole, B. (2019). Dis-

crete flows: Invertible generative models of discrete data. *arXiv preprint arXiv:1905.10347*.

[54] Trippe, B. L., & Turner, R. E. (2018). Conditional density estimation with bayesian normalising flows. *arXiv preprint arXiv:1802.04908*.

[55] Uria, B., Murray, I., & Larochelle, H. (2013). Rnade: The real-valued neural autoregressive density-estimator. *arXiv preprint arXiv:1306.0186*.

[56] Van den Broeck, G., Lykov, A., Schleich, M., & Suciu, D. (2021). On the tractability of shap explanations.

[57] Van den Bulcke, T., Van Leemput, K., Naudts, B., van Remortel, P., Ma, H., Verschoren, A., ... Marchal, K. (2006). Syntren: a generator of synthetic gene expression data for design and analysis of structure learning algorithms. *BMC bioinformatics*, *7*(1), 1–12.

[58] Vergari, A., Choi, Y., Peharz, R., & Van den Broeck, G. (2020). Probabilistic circuits: Representations, inference, learning and applications. In *Tutorial at the the 34th aaai conference on artificial intelligence.*

[59] Wang, J., Wiens, J., & Lundberg, S. (2020). Shapley flow: A graph-based approach to interpreting model predictions. *arXiv preprint arXiv:2010.14592*.

[60] Wang, M., Chen, X., & Zhang, H. (2010). Maximal conditional chi-square importance in random forests. *Bioinformatics*, *26*(6), 831–837.

[61] Weng, L. (2018). Flow-based deep generative models. *lilianweng.github.io/lil-log*. Retrieved from http://lilianweng.github.io/lil-log/2018/10/13/flow-based-deep-generative-models.html

[62] Winkler, C., Worrall, D., Hoogeboom, E., & Welling, M. (2019). Learning likelihoods with conditional normalizing flows. *arXiv preprint arXiv:1912.00042*.

[63] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning deep features for discriminative localization. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 2921–2929).

Appendix

A.1 RFI for Multiple Linear Regression

Proof. Generalisation risk for multiple linear regression, defined via MSE, is directly related to a mean squared error of prediction (MSPE):

$$\mathcal{R} = \mathbb{E}_{(Y,X_R)}\left[l\big(Y, h^*(X_R)\big)\right] = \mathbb{E}_{X_R}\,\mathrm{MSPE}_{X_R}$$
$$\mathrm{MSPE}_{X_R} = \mathbb{E}_{Y\sim Y/X_R}\left(Y - h_{\hat\beta}(X_R)\big)\right)^2$$

Note, that estimated parameters are also random: $\hat\beta = \hat\beta(Y, X_R)$

We can also define the MSPE for the whole train and test design matrices $(\mathrm{X}, \mathrm{X}^*)$, which serves as a MC-estimate of generalisation risk:

$$\mathrm{MSPE}_{\mathrm{X},\mathrm{X}^*} = \frac{1}{n^*}\,\mathbb{E}_{\mathrm{y}^*/\mathrm{X}^*,\mathrm{y}/\mathrm{X}}\left\|\mathrm{y}^* - h_{\hat\beta(\mathrm{y},\mathrm{X})}(\mathrm{X}^*)\right\|_2^2 = \big[\text{i.i.d.}\big] = \tag{6.1}$$

$$= \frac{1}{n^*}\sum_{i=1}^{n^*}\mathbb{E}_{Y_i^*\sim Y/x^{*(i)},\mathrm{y}/\mathrm{X}}\left(Y_i^* - h_{\hat\beta(\mathrm{y},\mathrm{X})}(x^{*(i)})\right)^2 \tag{6.2}$$

$$\mathcal{R} = \mathbb{E}_{\mathrm{X},\mathrm{X}^*\sim p(x_R)}\,\mathrm{MSPE}_{\mathrm{X},\mathrm{X}^*} \tag{6.3}$$

where $\|\cdot\|_2$ is L_2-norm of random vector.

Following the lecture notes [1], $\mathrm{MSPE}_{\mathrm{X},\mathrm{X}^*}$ can be calculated for the linear regression:

$$\mathrm{MSPE}_{\mathrm{X},\mathrm{X}^*} = \frac{1}{n^*}\mathrm{tr}(\mathrm{X}^{*T}\mathrm{X}^*\Sigma_{\hat\beta}) + \sigma^2, \quad \text{where } \Sigma_{\hat\beta} = \sigma^2(\mathrm{X}^T\mathrm{X})^{-1} \tag{6.4}$$

It can be shown, that when keeping:

$$\frac{\mathrm{X}^{*T}\mathrm{X}^*}{n^*} \approx \frac{\mathrm{X}^T\mathrm{X}}{n} = \widehat{\mathrm{Cov}}(X_R) \tag{6.5}$$

the first term disappears:

$$\mathrm{MSPE}_{\mathrm{X},\mathrm{X}^*} \xrightarrow[n\to\infty]{} \sigma^2 \tag{6.6}$$

Note, that by increasing n, we also increase n^*.

To calculate an analytic value of RFI_j^G, we need to find a risk with a resampled replacement variable j. Let $\tilde{\mathrm{X}}^*$ be a test design matrix with conditionally resampled variable j. Then Mean squared prediction error of resampled design matrix

[1] http://dept.stat.lsa.umich.edu/~kshedden/Courses/Regression_Notes/prediction.pdf

$\text{MSPE}_{\text{X},\tilde{\text{X}}^*}$ is:

$$\text{MSPE}_{\text{X},\tilde{\text{X}}^*} = \frac{1}{n^*}\,\mathbb{E}\left\|\text{y}^* - \tilde{\text{X}}^*\hat{\beta}\right\|_2^2 = \frac{1}{n^*}\,\mathbb{E}\left\|\text{X}^*\beta - \tilde{\text{X}}^*\hat{\beta}\right\|_2^2 + \frac{1}{n^*}\,\mathbb{E}\,\|\varepsilon\|_2^2 = \tag{6.7}$$

$$= \frac{1}{n^*}\sum_{i=1}^{n^*}\mathbb{E}\left(\beta^T x^{*(i)} - \hat{\beta}^T\tilde{x}^{*(i)}\right)^2 + \sigma^2 \tag{6.8}$$

where $x^{*(i)}$ and $\tilde{x}^{*(i)}$ are original test datapoint and datapoint with replaced variable j. Here, subscripts for conditional expectation are dropped for clarity, but they are the same, as in 6.1. The only random quantity is now the estimated parameters $\hat{\beta}$.

Let's try to break down this expectation, by partitioning the vectors ($R \setminus j$ is marked as $-j$, indices of sample $^{(i)}$ are dropped as well):

$$\mathbb{E}\left(\beta^T x^* - \hat{\beta}^T\tilde{x}^*\right)^2 = \mathbb{E}\big(x_j^*\beta_j + \beta_{-j}^T x_{-j}^* - \tilde{x}_j^*\hat{\beta}_j - \hat{\beta}_{-j}^T x_{-j}^*\big)^2 =$$

$$= \mathbb{E}\left[(x_j^*\beta_j - \tilde{x}_j^*\hat{\beta}_j)^2 + \big(x_{-j}^{*T}(\beta_{-j} - \hat{\beta}_{-j})\big)^2 - 2\big(x_j^*\beta_j - \tilde{x}_j^*\hat{\beta}_j\big)\cdot\big(x_{-j}^{*T}(\beta_{-j} - \hat{\beta}_{-j})\big)\right] =$$

$$= *_1 + *_2 + *_3$$

$$*_1 = \mathbb{E}(x_j^*\beta_j - \tilde{x}_j^*\hat{\beta}_j)^2 = \mathbb{E}(x_j^*\beta_j)^2 + \mathbb{E}(\tilde{x}_j^*\hat{\beta}_j)^2 - 2\mathbb{E}(x_j^*\beta_j)(\tilde{x}_j^*\hat{\beta}_j) =$$

$$= (x_j^*\beta_j)^2 + \tilde{x}_j^{*2}\left(\text{Var}(\hat{\beta}_j) + \beta_j^2\right) - 2x_j^*\tilde{x}_j^*\beta_j^2 =$$

$$= \beta_j^2(x_j^{*2} + \tilde{x}_j^{*2} - 2x_j^*\tilde{x}_j) + \tilde{x}_j^{*2}\,\text{Var}(\hat{\beta}_j) = \beta_j^2(x_j^* - \tilde{x}_j^*)^2 + \tilde{x}_j^{*2}\,\text{Var}(\hat{\beta}_j)$$

$$*_2 = \mathbb{E}(x_{-j}^{*T}\big(\beta_{-j} - \hat{\beta}_{-j}\big)\big)^2 = \big[\text{... the same as } \text{MSPE}_{\text{X}_{:,-\text{j}},\text{X}_{:,-\text{j}}^*} \text{ with excluded feature } j\big]$$

$$*_3 = -2\,\mathbb{E}(x_j^*\beta_j - \tilde{x}_j^*\hat{\beta}_j)(x_{-j}^{*T}(\beta_{-j} - \hat{\beta}_{-j})) =$$

$$= -2(x_j^*\beta_j)\cdot x_{-j}^{*T}\,\mathbb{E}(\beta_{-j} - \hat{\beta}_{-j}) + 2\tilde{x}_j^*\cdot\mathbb{E}\,\hat{\beta}_j(x_{-j}^{*T}(\beta_{-j} - \hat{\beta}_{-j})) = \big[\mathbb{E}(\beta_{-j} - \hat{\beta}_{-j}) = 0\big] =$$

$$= 2\tilde{x}_j^*\cdot\mathbb{E}\,\hat{\beta}_j\sum_{i\neq j}\left(x_i(\beta_i - \hat{\beta}_i)\right) = 2\tilde{x}_j^*\cdot\left(\sum_{i\neq j}\beta_i\beta_j x_i - \sum_{i\neq j}x_i\,\mathbb{E}\,\hat{\beta}_i\hat{\beta}_j\right) =$$

$$= 2\tilde{x}_j^*\cdot\left(\sum_{i\neq j}\beta_i\beta_j x_i - \sum_{i\neq j}x_i\big(\,\text{Cov}(\hat{\beta}_i,\hat{\beta}_j) + \beta_i\beta_j\big)\right) = -2\tilde{x}_j^*\cdot\sum_{i\neq j}x_i\,\text{Cov}(\hat{\beta}_i,\hat{\beta}_j) =$$

$$= -2\tilde{x}_j^*(x_{-j}^{*T}\Sigma_{\hat{\beta}_{j,-j}})$$

Putting everything together to 6.7 and transforming back matrix form:

$$\text{MSPE}_{\text{X},\tilde{\text{X}}^*} = \frac{\beta_j^2}{n^*}\left\|\text{X}_{:,j}^* - \tilde{\text{X}}_{:,j}^*\right\|_2^2 + \frac{\text{Var}(\hat{\beta}_j)}{n^*}\left\|\tilde{\text{X}}_{:,j}^*\right\|_2^2 + \\ + \frac{1}{n^*}\text{tr}(\text{X}_{:,-j}^{*T}\text{X}_{:,-j}^*\Sigma_{\hat{\beta}_{-j,-j}}) - \frac{2}{n^*}\tilde{\text{X}}_{:,j}^{*T}\text{X}_{:,-j}^*\Sigma_{\hat{\beta}_{j,-j}} + \sigma^2 \tag{6.9}$$

When the condition 6.5 is satisfied, it easy to see that for increasing n:

$$\frac{1}{n^*}\text{tr}(\text{X}_{:,-j}^{*T}\text{X}_{:,-j}^*\Sigma_{\hat{\beta}_{-j,-j}}) - \frac{2}{n^*}\tilde{\text{X}}_{:,j}^{*T}\text{X}_{:,-j}^*\Sigma_{\hat{\beta}_{j,-j}} \xrightarrow[n\to\infty]{} 0 \tag{6.10}$$

Let's now compute the expectation of $\text{MSPE}_{\text{X},\tilde{\text{X}}^*}$, if we know, that X is the i.i.d. sample from $p(x_R)$ and $\tilde{\text{X}}^*$ – sequentially sampled from $p(x_R)$ and then column j is

replaced with $p(x_j/x_G = \mathrm{X}^*_{:,\mathrm{G}})$ respectively:

$$\begin{aligned}
\mathcal{R}^{j/G} &= \mathbb{E}_{\mathrm{X},\tilde{\mathrm{X}}^*\sim p(x_R);\tilde{\mathrm{X}}^*_{:,j}\sim p(x_j/x_G=\mathrm{X}^*_{:,\mathrm{G}})} \mathrm{MSPE}_{\mathrm{X},\tilde{\mathrm{X}}^*} \approx \\
&\approx \frac{\beta_j^2}{n^*}\,\mathbb{E}\left\|\mathrm{X}^*_{:,j} - \tilde{\mathrm{X}}^*_{:,j}\right\|_2^2 + \frac{\mathrm{Var}(\hat{\beta}_j)}{n^*}\,\mathbb{E}\left\|\tilde{\mathrm{X}}^*_{:,j}\right\|_2^2 + \sigma^2 = \left[i.i.d.\right] = \\
&= \beta_j^2\mathbb{E}(\tilde{X}_j - X_j)^2 + \mathrm{Var}(\hat{\beta}_j)\,\mathbb{E}\,\tilde{X}_j + \sigma^2 = \left[\mathrm{Var}(\hat{\beta}_j) \xrightarrow[n\to\infty]{} 0\right] = \\
&= \beta_j^2\left(\mathbb{E}\,\tilde{X}_j^2 - 2\,\mathbb{E}\,X_j\tilde{X}_j + \mathbb{E}\,X_j^2\right) + \sigma^2 = \\
&= \beta_j^2\left(\mathrm{Var}(\tilde{X}_j) + (\mathbb{E}\,\tilde{X}_j)^2 + \mathrm{Var}(X_j) + (\mathbb{E}\,X_j)^2 - 2\,\mathrm{Cov}(\tilde{X}_j, X_j) - 2\,\mathbb{E}\,X_j\,\mathbb{E}\,\tilde{X}_j\right) + \sigma^2 = \\
&= 2\beta_j^2\left(\mathrm{Var}(X_j) - \mathrm{Cov}(\tilde{X}_j, X_j)\right) + \sigma^2
\end{aligned}$$

Thus, incorporating 6.6:

$$\mathrm{RFI}_j^G = \tilde{\mathcal{R}}^{j/G} - \mathcal{R} \approx 2\beta_j^2\left(\mathrm{Var}(X_j) - \mathrm{Cov}(\mathrm{X}_j, X_j)\right) \tag{6.11}$$

In the case of X_R having a multivariate Gaussian distribution $X_R \sim N(\mu_{X_R}, \Sigma_{X_R})$, the conditional distribution of replacement variable has the form:

$$\tilde{X}_j/X_G = \varepsilon * \sqrt{\Sigma_{jj} - \Sigma_{jG}\Sigma_{GG}^{-1}\Sigma_{Gj}} + \Sigma_{jG}\Sigma_{GG}^{-1}\Sigma_G, \quad \varepsilon \sim N(0,1) \tag{6.12}$$

where $\Sigma_{jj}, \Sigma_{jG}, \Sigma_{Gj}, \Sigma_{GG}$ is a partitioning of matrix Σ_{X_R}. With a true conditional sampler in the reparametrised form as in 6.12, it is easy to calculate:

$$\begin{aligned}
&\mathrm{Cov}(\tilde{X}_j, X_j) = \mathrm{Cov}(\Sigma_{jG}\Sigma_{GG}^{-1}X_G, X_j) = \Sigma_{jG}\Sigma_{GG}^{-1}\Sigma_{Gj} \\
&\mathrm{Var}(X_j) = \Sigma_{jj}
\end{aligned}$$

So, the RFI will have the formula:

$$\mathrm{RFI}_j^G \approx 2\beta_i\left(\Sigma_{jj} - \Sigma_{jG}\Sigma_{GG}^{-1}\Sigma_{Gj}\right) \tag{6.13}$$

□

A.2 Gradient-descent MLE with Noise Regularisation

The algorithm is used in the thesis as the main regularisation technique for Conditional density estimation with neural networks. Gaussian distribution is used to

generate noise in all the experiments.

Algorithm 1: Mini-batch gradient-descent MLE with Normal Noise Regularisation

Input : Train data $\mathcal{D} = \{(y^{(1)}, x^{(1)}), ..., (y^{(n)}, x^{(n)})\}$; number of epochs N_e; mini-batch size B; noise standard deviations σ_y, σ_x; learning rate α

Output: Fitted parameters $\hat{\theta}$

Initialize θ;

for *epoch* $= 1$ **to** N_e **do**

  for *batch* $= 1$ **to** n div B **do**

    Sample minibatch $\{(y^{(1)}, x^{(1)}), ..., (y^{(B)}, x^{(B)})\} \subset \mathcal{D}$;

    for $j = 1$ **to** B **do**

      Sample noise $\xi_y \sim N(0, \sigma_y^2 I)$ and $\xi_x \sim N(0, \sigma_x^2 I)$;

      Perturb batch: $\tilde{y}^{(j)} = y^{(j)} + \xi_y \quad \tilde{x}^{(j)} = x^{(j)} + \xi_x$

    end

    Gradient descent step: $\theta \leftarrow \theta + \alpha \nabla_\theta \frac{1}{b} \sum_{j=1}^{b} \log f_\theta(\tilde{y}^{(j)}/\tilde{x}^{(j)})$

  end

end

$\hat{\theta} \leftarrow \theta$

A.3 Radial flow

Jacobian and determinant of Jacobian. Using the same $R = (Z - \gamma)$:

$$\frac{d}{dZ}\Big(Z + \frac{\alpha\beta R}{\alpha + \|R\|_2}\Big) = I + \frac{d}{dZ}\Big(\frac{\alpha\beta R}{\alpha + \|R\|_2}\Big) = \Big[\text{Chain rule}\Big] =$$

$$= I + \frac{\alpha\beta}{\alpha + \|R\|_2} I + \alpha\beta R \frac{d}{dZ}\Big(\frac{1}{\alpha + \|R\|_2}\Big) = \Big[\text{Jacobian of } L_2\text{-norm: } \frac{d\|X\|_2}{dX} = \frac{X}{\|X\|_2}\Big] =$$

$$= \Big(1 + \frac{\alpha\beta}{\alpha + \|R\|_2}\Big) I - \frac{\alpha\beta}{\|R\|_2(\alpha + \|R\|_2)^2} R R^T$$

$$\det \frac{dt}{dZ} = \Big[\text{Matrix determinant lemma}\Big] =$$

$$= \left(1 - \frac{\alpha\beta}{\|R\|_2(\alpha + \|R\|_2)^2} \cdot \Big(1 + \frac{\alpha\beta}{\alpha + \|R\|_2}\Big)^{-1} R^T R\right) \cdot \det\Big(1 + \frac{\alpha\beta}{\alpha + \|R\|_2}\Big) I =$$

$$= \left(1 + \frac{\alpha\beta}{\alpha + \|R\|_2}\right)^{d_Z - 1} \cdot \left(1 + \frac{\alpha\beta}{\alpha + \|R\|_2} - \frac{\alpha\beta}{\|R\|_2(\alpha + \|R\|_2)^2} R^T R\right) =$$

$$= \Big[R^T R = \|R\|_2^2\Big] = \left(1 + \frac{\alpha\beta}{\alpha + \|R\|_2}\right)^{d_Z - 1} \cdot \left(1 + \frac{\alpha\beta}{\alpha + \|R\|_2} - \frac{\alpha\beta\|R\|_2}{(\alpha + \|R\|_2)^2}\right) =$$

$$= \left(1 + \frac{\alpha\beta}{\alpha + \|R\|_2}\right)^{d_Z - 1} \cdot \left(1 + \frac{\alpha^2\beta}{(\alpha + \|R\|_2)^2}\right)$$

Inverse transformation. Assuming α and β satisfy contraints: $\alpha > 0, \beta > -1$, one wants to solve the following equation with respect to Z:

$$t(Z) = Z + \frac{\alpha\beta(Z-\gamma)}{\alpha + \|Z-\gamma\|_2}$$

Let's use $R = Z - \gamma$, then:

$$\begin{aligned} & t(Z) = R + \gamma + \frac{\alpha\beta R}{\alpha + \|R\|_2} \quad \iff \\ \iff \quad & t(Z) - \gamma = R\left(1 + \frac{\alpha\beta}{\alpha + \|R\|_2}\right) \end{aligned}$$

As $\alpha > 0$ and $\beta > -1$: $\left(1 + \frac{\alpha\beta}{\alpha+\|R\|_2}\right) > 0$. Then, we can write R as:

$$R = \left(1 + \frac{\alpha\beta}{\alpha + \|R\|_2}\right)^{-1}(t(Z) - \gamma) \tag{6.14}$$

After applying L_2-norm to both parts we can solve a quadratic equation with respect to $\|R\|_2$:

$$\begin{aligned} & \|R\|_2 = \left(1 + \frac{\alpha\beta}{\alpha + \|R\|_2}\right)^{-1}\|t(Z) - \gamma\|_2 \quad \iff \\ \iff \quad & \|R\|_2^2 + \|R\|_2(\alpha + \alpha\beta - \|t(Z)-\gamma\|_2) - \alpha\|t(Z)-\gamma\|_2 = 0 \quad \iff \\ \iff \quad & \begin{cases} \|R\|_2 = -\frac{1}{2}(\alpha + \alpha\beta - \|t(Z)-\gamma\|_2) + \frac{1}{2}\sqrt{D} \\ D = (\alpha + \alpha\beta - \|t(Z)-\gamma\|_2)^2 + 4\alpha\|t(Z)-\gamma\|_2 \end{cases} \end{aligned}$$

Thus, we can evaluate R from 6.14 and consequently Z:

$$Z = \left(1 + \frac{\alpha\beta}{\alpha - \frac{1}{2}(\alpha + \alpha\beta - \|t(Z)-\gamma\|_2) + \frac{1}{2}\sqrt{D}}\right)^{-1}(t(Z) - \gamma) + \gamma \tag{6.15}$$

A.4 Evaluation protocol

Algorithm 2: RFI Evaluation protocol for data with a known causal DAG

Input : Causal DAG $\mathcal{G}$ (p nodes and m edges); Train data $\mathcal{D}$, test data $\mathcal{D}^*$.
Output: Test risks, test log-likelihood, test HD, KL and JS (if SCM is known), averaged $RFI_j^{\text{MB}(X_i)}$ and $RFI_j^{\text{non-MB}(X_i)}$ across j.

for $X_i \in \mathcal{G}$ **do**
  Fit predictive models with $\mathcal{D}$ for target X_i and features X_{-i};
  Report test risks for each predictor based on $\mathcal{D}^*$;
  for $X_j \in$ non-MB(X_i) **do**
    Fit conditional sampler for input X_j and context MB(X_i) on $\mathcal{D}$: $f_{\hat{\theta}}(x_j / \text{MB}(x_i))$;
    Report test log-likelihood of sampler;
    if *SCM is known and* MB$(X_j) \subseteq$ MB(X_i) *or SCM is LinearGaussianNoise* **then**
      Infere approximate / exact $\tilde{p}(x_j / \text{MB}(x_j))$;
      Report test HD, KL and JS between $f_{\hat{\theta}}(x_j / \text{MB}(x_i))$ and $p(x_j / \text{MB}(x_i)) = \tilde{p}(x_j / \text{MB}(x_j))$;
    end
    Use sampler to estimate $RFI_j^{\text{MB}(X_i)}$ for each predictor and risk;
  end
  Report averaged $RFI_j^{\text{MB}(X_i)}$ across j;
  for $X_j \in$ MB(X_i) **do**
    Fit conditional sampler for input X_j and context non-MB(X_i) on $\mathcal{D}$: $f_{\hat{\theta}}(x_j / \text{non-MB}(x_i))$;
    Report test log-likelihood of sampler;
    if *SCM is LinearGaussianNoise* **then**
      Infere exact $p(x_j / \text{non-MB}(x_i))$;
      Report test HD, KL and JS between $f_{\hat{\theta}}(x_j / \text{non-MB}(x_i))$ and $p(x_j / \text{non-MB}(x_j))$;
    end
    Use sampler to estimate $RFI_j^{\text{non-MB}(X_i)}$ for each predictor and risk;
  end
  Report averaged $RFI_j^{\text{non-MB}(X_i)}$ across j;
end

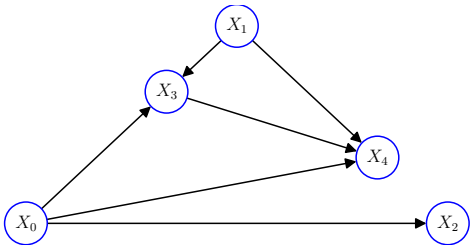
A.5 Markov-Blanket conditional distribution

Credits to the discussion[2], by applying Bayes' rule to children of X_i and using d-separation properties:

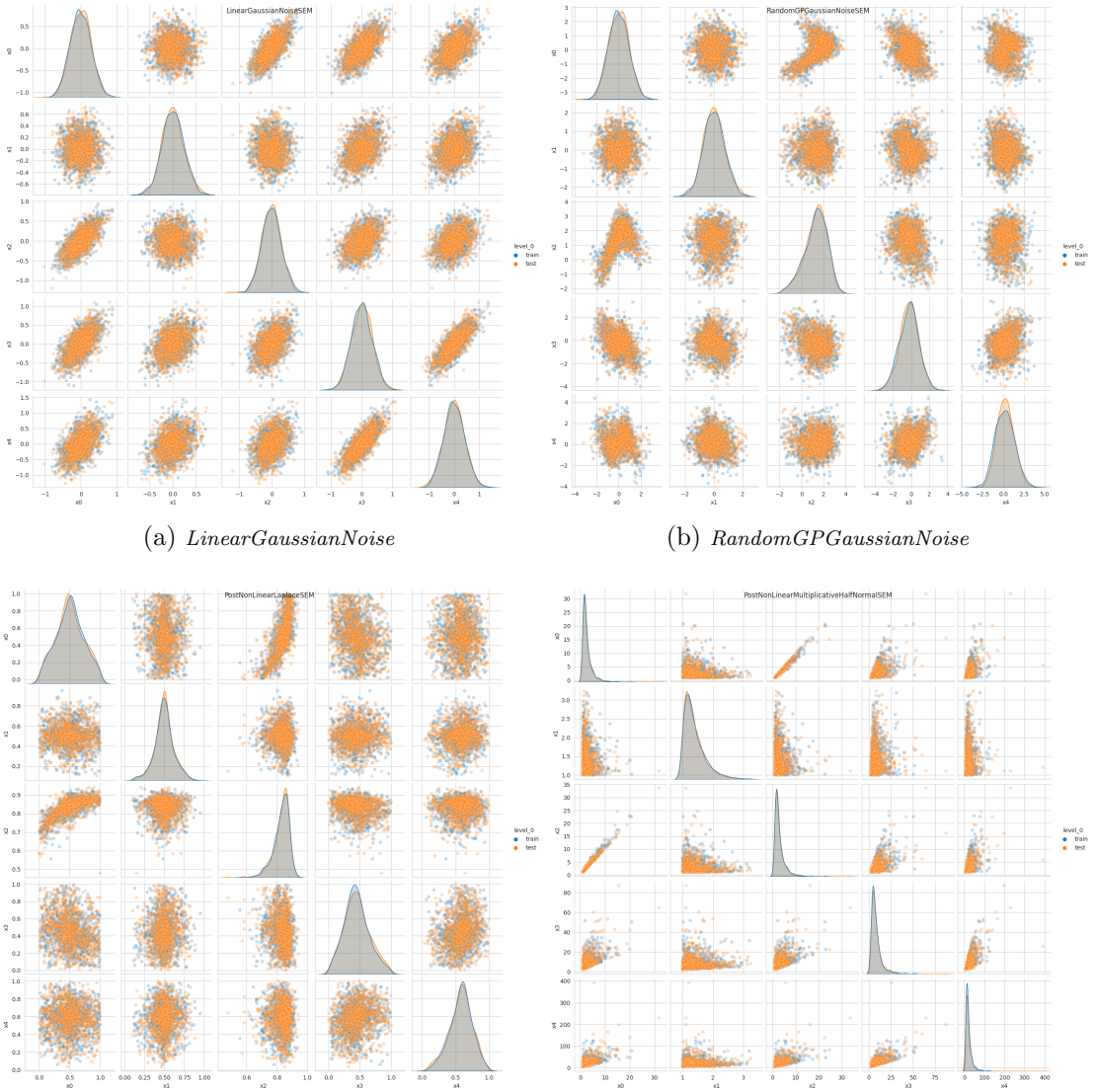
$$
\begin{aligned}
p(x_i/\text{MB}(x_i)) &= p(x_i/\text{Pa}_{x_i}, \text{Ch}_{x_i}, \text{Pa}_{\text{Ch}_{x_i}}) = \\
&= \frac{p(\text{Ch}_{x_i}/x_i, \text{Pa}_{x_i}, \text{Pa}_{\text{Ch}_{x_i}}) \cdot p(x_i/\text{Pa}_{x_i}, \text{Pa}_{\text{Ch}_{x_i}})}{\int_{\mathcal{X}_i} p(\text{Ch}_{x_i}/x_i, \text{Pa}_{x_i}, \text{Pa}_{\text{Ch}_{x_i}}) \cdot p(x_i/\text{Pa}_{x_i}, \text{Pa}_{\text{Ch}_{x_i}})\, dx_i} = \\
&= \frac{p(\text{Ch}_{x_i}/\text{Pa}_{\text{Ch}_{x_i}}, x_i) \cdot p(x_i/\text{Pa}_{x_i})}{\int_{\mathcal{X}_i} p(\text{Ch}_{x_i}/\text{Pa}_{\text{Ch}_{x_i}}, x_i) \cdot p(x_i/\text{Pa}_{x_i})\, dx_i} = \\
&= \frac{p(x_i/\text{Pa}_{x_i}) \cdot \prod_{x_j \in \text{Ch}_{x_i}} p(x_j/\text{Pa}_{x_j}, x_i)}{\int_{\mathcal{X}_i} p(x/\text{Pa}_x) \cdot \prod_{x_j \in \text{Ch}_x} p(x_j/\text{Pa}_{x_j}, x_i)\, dx_i}
\end{aligned}
$$

[2] https://stats.stackexchange.com/questions/108790/markov-blanket-conditional-distribution-derivation

A.6 Pairwise scatter-matrices for datasets with known causal DAG



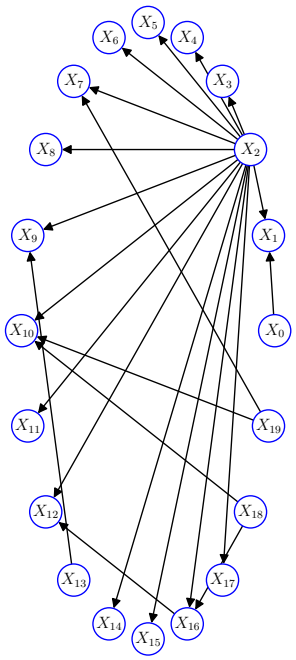
Random Causal DAG $\mathcal{G}$, sampled from $\mathcal{G}(5, 10)$. Common for all underlying SCMs.



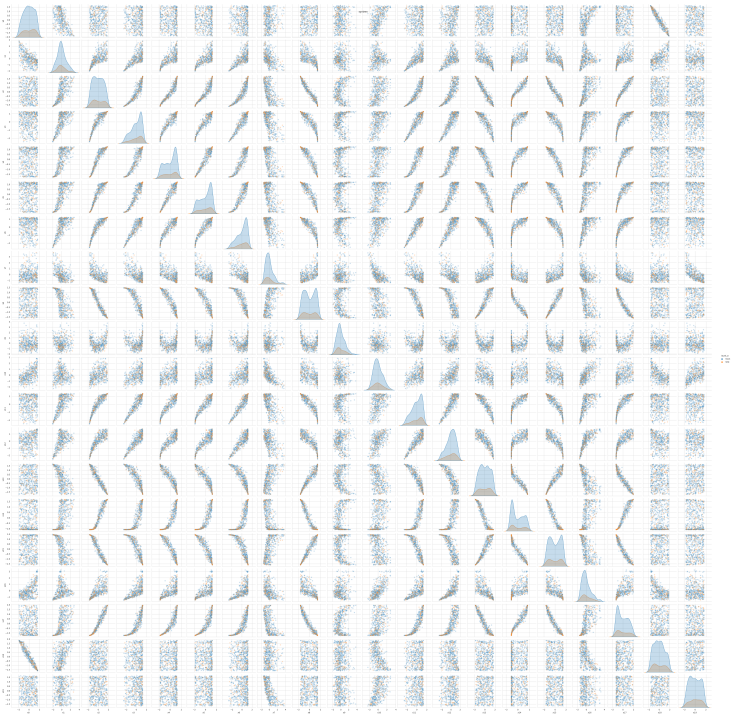
(a) *LinearGaussianNoise* (b) *RandomGPGaussianNoise*

(c) *PostNonLinearLaplace* (d) *PostNonLinearMultiplicativeHalfNormal*

Figure 6.1: Pairwise scatter-matrix of synthetic data, sampled from four structural causal models. Orange scatter points (test subset, $n^* = 1000$) are overlaid over blue (train, $n = 1000$). Rows and columns are ordered alphabetically.

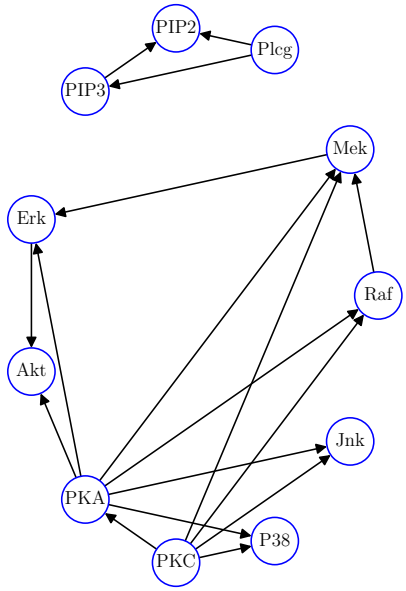


(a) One out of ten ground-truth causal DAGs.

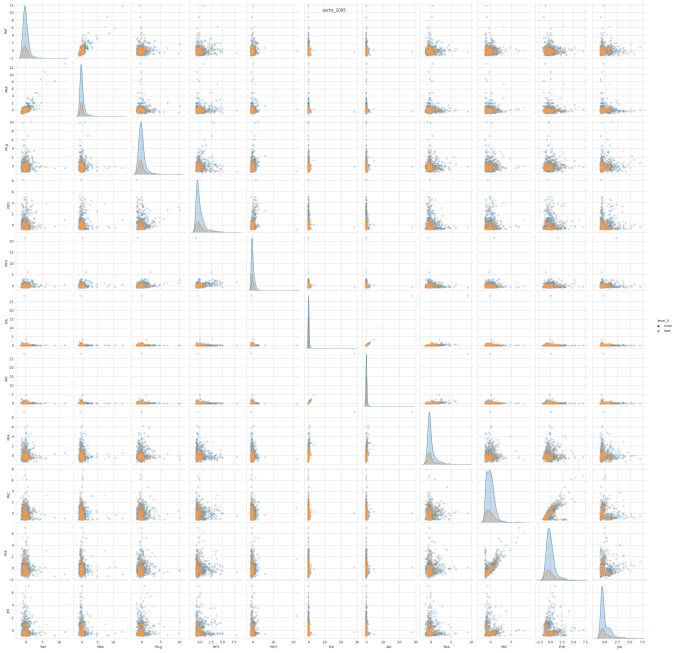


(b) Pairwise scatter-matrix of semi-synthetic data, sampled from SynTReN generator (one of ten DAGs). Orange scatter points (test subset, $n^* = 100$) are overlaid over blue (train, $n = 400$). Rows and columns are ordered alphabetically.

Figure 6.2: SynTReN dataset visual characteristics.



(a) Ground-truth causal DAG of Sachs-2005 dataset.



(b) Pairwise scatter-matrix of Sachs-2005 dataset. Orange scatter points (test subset, $n^* = 171$) are overlaid over blue (train, $n = 682$).

Figure 6.3: Sachs-2005 visual characteristics.

A.7 CNF and MDN results reproduction for UCI benchmark

We successfully reproduced the results of the paper on noise regularisation for CNF and MDN estimators [42] on three UCI benchmarks [13]: Boston ($d_X = 13, d_Y = 1, n = 405$), Concrete ($d_X = 8, d_Y = 1, n = 824$) and Energy ($d_X = 9, d_Y = 1, n = 615$).

We report mean and standard deviation for log-likelihood (see Figure 6.4), calculated on three 20% test hold-out subsets and five random seed initialisations of weights. For CNF and MDN, we performed a small 5-fold hyperparameter selection, based on train subset, and then retrained the best configuration with different initialisation seeds. Hyperparameters grid for both CNF and MDN encompassed 324 runs with varying inputs and context noise deviation, number of epochs, number of transformations/mixture components, number of hidden units and weight decay. Other parameters, such as learning rate or optimiser, are kept the same (see *VanillaCNF* and *VanillaMDN*, Section 4). As a sanity check, we add a performance of *CondGauss* model (introduced in Section 4) on 15 20% test hold-out datasets.

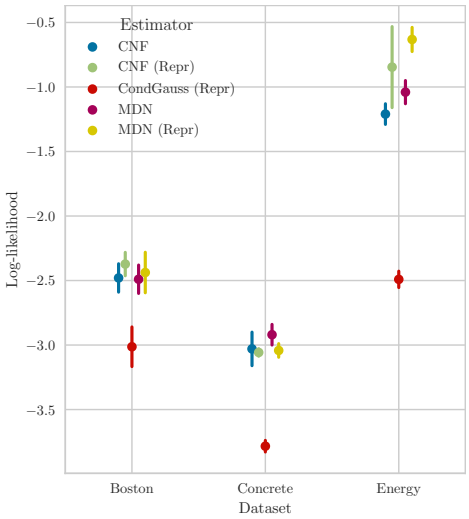


Figure 6.4: Reproduction of results from [42]: mean and std of test log-likelihood for Boston, Concrete and Energy datasets (higher is better).

We observe, that (1) both neural estimators outperformed the simple *CondGauss* model on all the datasets and (2) we achieved similar performance with a smaller grid search, compared to the original paper.